Basi di dati II — Esame — 1 luglio 2015 — Compito A

Staccare questo foglio all'inizio del compito per poter consultare quanto scritto sul retro.
Questo foglio non va consegnato

Basi di dati II — Esame — 1 luglio 2015 — Compito A

Tempo a disposizione: un'ora e quindici minuti per la prova breve e due ore e trenta minuti per la prova completa.

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

Domanda 1 (15% per la prova completa, 40% per la prova breve) Come visto a lezione, molti DBMS utilizzano, oltre a quelli basati sui B$^+$-tree, anche indici basati su una tecnica detta "bitmap." Si ricorda che un indice bitmap, su un attributo A con a valori diversi, per una relazione con N_R ennuple, è costituito da a vettori (uno per ciascun valore di A) di N_R bit ciascuno (un bit per ogni ennupla della relazione): l'i-esimo bit del vettore associato al valore a_j è 1 se il valore della i-esima ennupla sull'attributo A è a_j e 0 se il valore è diverso. L'accesso ai vettori è organizzato per mezzo di un albero. In sostanza, rispetto ad un B$^+$-tree, abbiamo una struttura di accesso simile, ma con foglie diverse: nel B$^+$-tree, per ogni valore dell'attributo abbiamo una lista di indirizzi, mentre nell'indice bitmap abbiamo un vettore di bit. Esiste poi una struttura (o un metodo) per associare gli indirizzi delle ennuple ai numeri da 1 a N_R. Sulla base degli elementi forniti, con riferimento ad un sistema con blocchi di dimensione $B = 4$ KB e indirizzi dei record di lunghezza $b = 8$ byte e ad una relazione R con N_R =1.600.000 ennuple e (fra gli altri) quattro attributi, A, C, E, F ciascuno con $a = 100$ valori diversi e una lunghezza $c = 8$ byte

1. Valutare lo spazio SB (in blocchi) necessario per le foglie di un indice bitmap su A; per confronto, calcolare lo spazio ST necessario per le foglie di un B$^+$-tree (con riferimenti ai record e una coppia chiave-valore per ciascun record) per un indice su A (N.B.: si noterà che le dimensioni dei bitmap come descritte non sono poi così convenienti; in pratica si usano tecniche di compressione, che omettiamo).

Poi, valutare (sia nel caso di indici bitmap, sia nel caso di B$^+$-tree) il costo (in termini di numero di accessi a memoria secondaria, supponendo che tutti i nodi intermedi di tutti gli indici siano nei buffer; basta quindi contare il numero di foglie da visitare e l'accesso alle ennuple; anche per quest'ultimo si può considerare il solo accesso al record, trascurando il costo di accesso alla struttura di supporto) per le seguenti interrogazioni

2. `SELECT * FROM R WHERE A = 5` (supporre che il risultato contenga N_R/a ennuple)
3. `SELECT * FROM R WHERE A = 15 AND C = 14 AND E = 13 AND F = 9` (non potendo sapere quanto i valori siano indipendenti, assumere che il risultato contenga 10 ennuple)

Riportare le risposte nella tabella sottostante, nelle righe da 1 a 3, indicando formula e valore numerico.

	Risposte	
	Indici bitmap	Indici B$^+$-tree
1		
2		
3		

Basi di dati II — 1 luglio 2015 — Compito A

Domanda 2 (25% per la prova completa, 60% per la prova breve) Con riferimento all'esempio discusso nelle esercitazioni ai fini del supporto alla valutazione della didattica e, in particolare, al relativo schema dei dati, fornito per comodità in allegato, mostrare uno schema dimensionale relativo all'analisi delle immatricolazioni.

Basi di dati II — 1 luglio 2015 — Compito A

Domanda 3 (10%) Considerare un sistema con dischi con $N = 500$ 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 $t_B = 10$ μsec

Rispondere alle seguenti domande mostrando formula e valore numerico

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

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

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 $p = 5$ e fan-out (fattore di blocco dell'indice) $f_I = 50$, usato di recente, ma in modo sporadico?

4. Qual è il tempo che si può ipotizzare necessario per eseguire $m = 10.000$ accessi diretti in tempi ravvicinati a record di un file (molto grande) attraverso un indice che abbia profondità $p = 5$, fan-out $f_I = 50$, con disponibilità di circa $P = 4000$ pagine di buffer?

Basi di dati II — 1 luglio 2015 — Compito A

Domanda 4 (15%) Si consideri un DBMS che preveda, in aggiunta alle tradizionali operazioni di lettura e scrittura, anche un'operazione AUMENTA*(x,i)* che modifica il valore di x (intero) incrementandolo di i, senza renderlo disponibile alla transazione (il che richiederebbe un'esplicita operazione di lettura). Per includere questo tipo di operazioni, in aggiunta a quelle di lettura e scrittura, è stato proposto di utilizzare un terzo tipo di lock, detto a-lock, con la seguente regola di compatibilità (in aggiunta a quelle note per r-lock e w-lock): due a-lock su un oggetto sono fra loro compatibili, mentre un a-lock è incompatibile sia con un r-lock sia con un w-lock.

1. Spiegare intuitivamente perché questa tecnica può generare transazioni a maggiore livello di concorrenza (cioè con un throughput maggiore).

2. Mostrare che schedule 2-PL che soddisfino le regole di compatibilità note estese con quelle sopra citate risultano serializzabili.

Basi di dati II — 1 luglio 2015 — Compito A

Domanda 5 (20%)
Considerare le operazioni di raggruppamento, specificate in SQL con clausole `GROUP BY` e funzioni aggregative sui gruppi. Ad esempio:

```
SELECT B, SUM(C) FROM R GROUP BY B
```

Queste operazioni vengono concettualmente eseguite in due passi (supponiamo per semplicità che, come nell'esempio, ci sia un solo attributo di raggruppamento e una sola funzione aggregativa)

1. partizionare le ennuple dell'operando sulla base dei valori dell'attributo di raggruppamento (`B` nell'esempio)
2. per ogni partizione, produrre una ennupla con (i) il valore dell'attributo di raggruppamento e (ii) il valore della funzione aggregativa (`SUM(C)` nell'esempio) applicata alle ennuple della partizione

Una possibile implementazione efficiente può essere ottenuta con una variante del mergesort a più vie, con porzioni ordinate e aggregazioni effettuate esaminando i valori affioranti di ciascuna partizione.

Considerare la relazione sotto schematizzata, sugli attributi `A`, `B` e `C`. Supponendo una disponibilità di buffer abbastanza ampia, ma non sufficiente a caricare in memoria l'intera relazione (supporre ad esempio una disponibilità di 4 buffer, con un fattore di blocco pari a 2 e quindi uno spazio occupato dalla relazione pari a 8 blocchi), considerare l'esecuzione dell'interrogazione sopra mostrata, in due passate (il che è possibile, visto che il numero di buffer è maggiore della radice del numero di blocchi).

Mostrare lo stato delle strutture in memoria centrale e secondaria dopo l'esecuzione di tre chiamate al metodo `next()` sullo scan che implementa l'interrogazione complessiva. In particolare, mostrare i "run" (cioè le porzioni di file ordinate durante prima passata — per comodità mostrare tutti gli attributi) memorizzati su disco e i buffer in memoria centrale, evidenziando per ciascun buffer il record corrente (il primo non ancora considerato). Mostrare anche i record prodotti dalle prime tre chiamate di `next()`.

Run su disco | Buffer | Record prodotti dalle prime tre `next()`

A	B	C
101	c	1
102	d	2
103	d	4
104	c	1
105	f	3
106	d	1
107	e	3
108	f	4
109	c	5
110	d	2
111	c	2
112	f	4
113	c	1
114	f	2
115	e	3
116	d	3

Basi di dati II — 1 luglio 2015 — Compito A

Domanda 6 (15%) Nel commit a due fasi, per ovviare alle conseguenze negative di un guasto del coordinatore, alcuni sistemi prevedono la possibilità di eleggere un nuovo coordinatore, fra i partecipanti rimanenti. Il nuovo coordinatore, a questo punto, contatta i partecipanti e, per ciascuna transazione T cerca di prendere una decisione. Per una data transazione, considerare i casi seguenti e per ciascuno, indicare se si può verificare, spiegare perché e indicare quale decisione può in tal caso prendere il nuovo coordinatore. Assumere che il preesistente coordinatore fosse anche un partecipante.

1. uno o più partecipanti (ma non tutti) hanno il record di ready nel log e nessuno ha il record di commit

Si può verificare (sì o no)?
Perché?

Quale decisione può prendere il coordinatore?

2. un partecipante ha il record di ready nel log e un altro ha il record di abort

Si può verificare (sì o no)?
Perché?

Quale decisione può prendere il coordinatore?

3. tutti i partecipanti hanno il record di ready nel log

Si può verificare (sì o no)?
Perché?

Quale decisione può prendere il coordinatore?

4. uno o più partecipanti hanno il record di abort nel log e uno ha il record di commit

Si può verificare (sì o no)?
Perché?

Quale decisione può prendere il coordinatore?

Basi di dati II — Esame — 1 luglio 2015 — Compito B

Staccare questo foglio all'inizio del compito per poter consultare quanto scritto sul retro.
Questo foglio non va consegnato

Basi di dati II — Esame — 1 luglio 2015 — Compito B

Tempo a disposizione: un'ora e quindici minuti per la prova breve e due ore e trenta minuti per la prova completa.

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

Domanda 1 (15% per la prova completa, 40% per la prova breve) Come visto a lezione, molti DBMS utilizzano, oltre a quelli basati sui B$^+$-tree, anche indici basati su una tecnica detta "bitmap." Si ricorda che un indice bitmap, su un attributo A con v valori diversi, per una relazione con N_R ennuple, è costituito da v vettori (uno per ciascun valore di A) di N_R bit ciascuno (un bit per ogni ennupla della relazione): l'i-esimo bit del vettore associato al valore a_j è 1 se il valore della i-esima ennupla sull'attributo A è a_j e 0 se il valore è diverso. L'accesso ai vettori è organizzato per mezzo di un albero. In sostanza, rispetto ad un B$^+$-tree, abbiamo una struttura di accesso simile, ma con foglie diverse: nel B$^+$-tree, per ogni valore dell'attributo abbiamo una lista di indirizzi, mentre nell'indice bitmap abbiamo un vettore di bit. Esiste poi una struttura (o un metodo) per associare gli indirizzi delle ennuple ai numeri da 1 a N_R. Sulla base degli elementi forniti, con riferimento ad un sistema con blocchi di dimensione $D = 8$ KB e indirizzi dei record di lunghezza $p = 8$ byte e ad una relazione R con N_R =3.200.000 ennuple e (fra gli altri) quattro attributi, A, C, E, F ciascuno con $v = 100$ valori diversi e una lunghezza $a = 8$ byte

1. Valutare lo spazio SB (in blocchi) necessario per le foglie di un indice bitmap su A; per confronto, calcolare lo spazio ST necessario per le foglie di un B$^+$-tree (con riferimenti ai record e una coppia chiave-valore per ciascun record) per un indice su A (N.B.: si noterà che le dimensioni dei bitmap come descritte non sono poi così convenienti; in pratica si usano tecniche di compressione, che omettiamo).

Poi, valutare (sia nel caso di indici bitmap, sia nel caso di B$^+$-tree) il costo (in termini di numero di accessi a memoria secondaria, supponendo che tutti i nodi intermedi di tutti gli indici siano nei buffer; basta quindi contare il numero di foglie da visitare e l'accesso alle ennuple; anche per quest'ultimo si può considerare il solo accesso al record, trascurando il costo di accesso alla struttura di supporto) per le seguenti interrogazioni

2. `SELECT * FROM R WHERE A = 5` (supporre che il risultato contenga N_R/v ennuple)
3. `SELECT * FROM R WHERE A = 15 AND C = 14 AND E = 13 AND F = 9` (non potendo sapere quanto i valori siano indipendenti, assumere che il risultato contenga 10 ennuple)

Riportare le risposte nella tabella sottostante, nelle righe da 1 a 3, indicando formula e valore numerico.

	Risposte	
	Indici bitmap	Indici B$^+$-tree
1		
2		
3		

Basi di dati II — 1 luglio 2015 — Compito B

Domanda 2 (25% per la prova completa, 60% per la prova breve) Con riferimento all'esempio discusso nelle esercitazioni ai fini del supporto alla valutazione della didattica e, in particolare, al relativo schema dei dati, fornito per comodità in allegato, mostrare uno schema dimensionale relativo all'analisi delle immatricolazioni.

Basi di dati II — 1 luglio 2015 — Compito B

Domanda 3 (10%) Considerare un sistema con dischi con $N = 500$ blocchi per traccia

- tempo medio di posizionamento della testina (tempo di seek) $t_S = 4$ msec
- tempo medio di latenza (attesa dovuta alla rotazione) $t_L = 4$ msec
- tempo minimo di lettura di un blocco $t_B = 10$ μsec

Rispondere alle seguenti domande mostrando formula e valore numerico

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

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

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 $p = 5$ e fan-out (fattore di blocco dell'indice) $f_I = 50$, usato di recente, ma in modo sporadico?

4. Qual è il tempo che si può ipotizzare necessario per eseguire $m = 20.000$ accessi diretti in tempi ravvicinati a record di un file (molto grande) attraverso un indice che abbia profondità $p = 5$, fan-out $f_I = 50$, con disponibilità di circa $P = 4000$ pagine di buffer?

Basi di dati II — 1 luglio 2015 — Compito B

Domanda 4 (15%) Si consideri un DBMS che preveda, in aggiunta alle tradizionali operazioni di lettura e scrittura, anche un'operazione PIÙ*(x,i)* che modifica il valore di x (intero) incrementandolo di i, senza renderlo disponibile alla transazione (il che richiederebbe un'esplicita operazione di lettura). Per includere questo tipo di operazioni, in aggiunta a quelle di lettura e scrittura, è stato proposto di utilizzare un terzo tipo di lock, detto p-lock, con la seguente regola di compatibilità (in aggiunta a quelle note per r-lock e w-lock): due p-lock su un oggetto sono fra loro compatibili, mentre un p-lock è incompatibile sia con un r-lock sia con un w-lock.

1. Spiegare intuitivamente perché questa tecnica può generare transazioni a maggiore livello di concorrenza (cioè con un throughput maggiore).

2. Mostrare che schedule 2-PL che soddisfino le regole di compatibilità note estese con quelle sopra citate risultano serializzabili.

Basi di dati II — 1 luglio 2015 — Compito B

Domanda 5 (20%)
Considerare le operazioni di raggruppamento, specificate in SQL con clausole `GROUP BY` e funzioni aggregative sui gruppi. Ad esempio:

```
SELECT B, SUM(C) FROM R GROUP BY B
```

Queste operazioni vengono concettualmente eseguite in due passi (supponiamo per semplicità che, come nell'esempio, ci sia un solo attributo di raggruppamento e una sola funzione aggregativa)

1. partizionare le ennuple dell'operando sulla base dei valori dell'attributo di raggruppamento (`B` nell'esempio)
2. per ogni partizione, produrre una ennupla con (i) il valore dell'attributo di raggruppamento e (ii) il valore della funzione aggregativa (`SUM(C)` nell'esempio) applicata alle ennuple della partizione

Una possibile implementazione efficiente può essere ottenuta con una variante del mergesort a più vie, con porzioni ordinate e aggregazioni effettuate esaminando i valori affioranti di ciascuna partizione.

Considerare la relazione sotto schematizzata, sugli attributi `A`, `B` e `C`. Supponendo una disponibilità di buffer abbastanza ampia, ma non sufficiente a caricare in memoria l'intera relazione (supporre ad esempio una disponibilità di 4 buffer, con un fattore di blocco pari a 2 e quindi uno spazio occupato dalla relazione pari a 8 blocchi), considerare l'esecuzione dell'interrogazione sopra mostrata, in due passate (il che è possibile, visto che il numero di buffer è maggiore della radice del numero di blocchi).

Mostrare lo stato delle strutture in memoria centrale e secondaria dopo l'esecuzione di tre chiamate al metodo `next()` sullo scan che implementa l'interrogazione complessiva. In particolare, mostrare i "run" (cioè le porzioni di file ordinate durante prima passata — per comodità mostrare tutti gli attributi) memorizzati su disco e i buffer in memoria centrale, evidenziando per ciascun buffer il record corrente (il primo non ancora considerato). Mostrare anche i record prodotti dalle prime tre chiamate di `next()`.

Run su disco | Buffer | Record prodotti dalle prime tre `next()`

A	B	C
101	c	2
102	d	2
103	d	4
104	c	2
105	f	3
106	d	1
107	e	3
108	f	4
109	c	5
110	d	2
111	c	2
112	f	4
113	c	1
114	f	2
115	e	3
116	d	3

Basi di dati II — 1 luglio 2015 — Compito B

Domanda 6 (15%) Nel commit a due fasi, per ovviare alle conseguenze negative di un guasto del coordinatore, alcuni sistemi prevedono la possibilità di eleggere un nuovo coordinatore, fra i partecipanti rimanenti. Il nuovo coordinatore, a questo punto, contatta i partecipanti e, per ciascuna transazione T cerca di prendere una decisione. Per una data transazione, considerare i casi seguenti e per ciascuno, indicare se si può verificare, spiegare perché e indicare quale decisione può in tal caso prendere il nuovo coordinatore. Assumere che il preesistente coordinatore fosse anche un partecipante.

1. tutti i partecipanti hanno il record di ready nel log

Si può verificare (sì o no)?
Perché?

Quale decisione può prendere il coordinatore?

2. uno o più partecipanti (ma non tutti) hanno il record di ready nel log e uno ha il record di commit

Si può verificare (sì o no)?
Perché?

Quale decisione può prendere il coordinatore?

3. uno o più partecipanti (ma non tutti) hanno il record di ready nel log e nessuno ha il record di commit

Si può verificare (sì o no)?
Perché?

Quale decisione può prendere il coordinatore?

4. uno o più partecipanti (ma non tutti) hanno il record di commit nel log e gli altri il record di ready

Si può verificare (sì o no)?
Perché?

Quale decisione può prendere il coordinatore?

Basi di dati II — Esame — 1 luglio 2015 — Compito C

Staccare questo foglio all'inizio del compito per poter consultare quanto scritto sul retro.
Questo foglio non va consegnato

Basi di dati II — Esame — 1 luglio 2015 — Compito C

Tempo a disposizione: un'ora e quindici minuti per la prova breve e due ore e trenta minuti per la prova completa.

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

Domanda 1 (15% per la prova completa, 40% per la prova breve) Come visto a lezione, molti DBMS utilizzano, oltre a quelli basati sui B$^+$-tree, anche indici basati su una tecnica detta "bitmap." Si ricorda che un indice bitmap, su un attributo A con v valori diversi, per una relazione con N_R ennuple, è costituito da v vettori (uno per ciascun valore di A) di N_R bit ciascuno (un bit per ogni ennupla della relazione): l'i-esimo bit del vettore associato al valore a_j è 1 se il valore della i-esima ennupla sull'attributo A è a_j e 0 se il valore è diverso. L'accesso ai vettori è organizzato per mezzo di un albero. In sostanza, rispetto ad un B$^+$-tree, abbiamo una struttura di accesso simile, ma con foglie diverse: nel B$^+$-tree, per ogni valore dell'attributo abbiamo una lista di indirizzi, mentre nell'indice bitmap abbiamo un vettore di bit. Esiste poi una struttura (o un metodo) per associare gli indirizzi delle ennuple ai numeri da 1 a N_R. Sulla base degli elementi forniti, con riferimento ad un sistema con blocchi di dimensione $B = 8$ KB e indirizzi dei record di lunghezza $p = 8$ byte e ad una relazione R con N_R =3.200.000 ennuple e (fra gli altri) quattro attributi, A, C, E, F ciascuno con $v = 100$ valori diversi e una lunghezza $a = 8$ byte

1. Valutare lo spazio SB (in blocchi) necessario per le foglie di un indice bitmap su A; per confronto, calcolare lo spazio ST necessario per le foglie di un B$^+$-tree (con riferimenti ai record e una coppia chiave-valore per ciascun record) per un indice su A (N.B.: si noterà che le dimensioni dei bitmap come descritte non sono poi così convenienti; in pratica si usano tecniche di compressione, che omettiamo).

Poi, valutare (sia nel caso di indici bitmap, sia nel caso di B$^+$-tree) il costo (in termini di numero di accessi a memoria secondaria, supponendo che tutti i nodi intermedi di tutti gli indici siano nei buffer; basta quindi contare il numero di foglie da visitare e l'accesso alle ennuple; anche per quest'ultimo si può considerare il solo accesso al record, trascurando il costo di accesso alla struttura di supporto) per le seguenti interrogazioni

2. `SELECT * FROM R WHERE A = 5` (supporre che il risultato contenga N_R/v ennuple)
3. `SELECT * FROM R WHERE A = 15 AND C = 14 AND E = 13 AND F = 9` (non potendo sapere quanto i valori siano indipendenti, assumere che il risultato contenga 10 ennuple)

Riportare le risposte nella tabella sottostante, nelle righe da 1 a 3, indicando formula e valore numerico.

	Risposte	
	Indici bitmap	Indici B$^+$-tree
1		
2		
3		

Basi di dati II — 1 luglio 2015 — Compito C

Domanda 2 (25% per la prova completa, 60% per la prova breve) Con riferimento all'esempio discusso nelle esercitazioni ai fini del supporto alla valutazione della didattica e, in particolare, al relativo schema dei dati, fornito per comodità in allegato, mostrare uno schema dimensionale relativo all'analisi delle immatricolazioni.

Basi di dati II — 1 luglio 2015 — Compito C

Domanda 3 (10%) Considerare un sistema con dischi con $N = 500$ blocchi per traccia

- tempo medio di posizionamento della testina (tempo di seek) $t_S = 4$ msec
- tempo medio di latenza (attesa dovuta alla rotazione) $t_L = 4$ msec
- tempo minimo di lettura di un blocco $t_B = 10$ μsec

Rispondere alle seguenti domande mostrando formula e valore numerico

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

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

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 $p = 5$ e fan-out (fattore di blocco dell'indice) $f_I = 50$, usato di recente, ma in modo sporadico?

4. Qual è il tempo che si può ipotizzare necessario per eseguire $m = 10.000$ accessi diretti in tempi ravvicinati a record di un file (molto grande) attraverso un indice che abbia profondità $p = 5$, fan-out $f_I = 50$, con disponibilità di circa $P = 4000$ pagine di buffer?

Basi di dati II — 1 luglio 2015 — Compito C

Domanda 4 (15%) Si consideri un DBMS che preveda, in aggiunta alle tradizionali operazioni di lettura e scrittura, anche un'operazione AUMENTA*(x,i)* che modifica il valore di x (intero) incrementandolo di i, senza renderlo disponibile alla transazione (il che richiederebbe un'esplicita operazione di lettura). Per includere questo tipo di operazioni, in aggiunta a quelle di lettura e scrittura, è stato proposto di utilizzare un terzo tipo di lock, detto a-lock, con la seguente regola di compatibilità (in aggiunta a quelle note per r-lock e w-lock): due a-lock su un oggetto sono fra loro compatibili, mentre un a-lock è incompatibile sia con un r-lock sia con un w-lock.

1. Spiegare intuitivamente perché questa tecnica può generare transazioni a maggiore livello di concorrenza (cioè con un throughput maggiore).

2. Mostrare che schedule 2-PL che soddisfino le regole di compatibilità note estese con quelle sopra citate risultano serializzabili.

Basi di dati II — 1 luglio 2015 — Compito C

Domanda 5 (20%)
Considerare le operazioni di raggruppamento, specificate in SQL con clausole `GROUP BY` e funzioni aggregative sui gruppi. Ad esempio:

```
SELECT B, SUM(C) FROM R GROUP BY B
```

Queste operazioni vengono concettualmente eseguite in due passi (supponiamo per semplicità che, come nell'esempio, ci sia un solo attributo di raggruppamento e una sola funzione aggregativa)

1. partizionare le ennuple dell'operando sulla base dei valori dell'attributo di raggruppamento (`B` nell'esempio)
2. per ogni partizione, produrre una ennupla con (i) il valore dell'attributo di raggruppamento e (ii) il valore della funzione aggregativa (`SUM(C)` nell'esempio) applicata alle ennuple della partizione

Una possibile implementazione efficiente può essere ottenuta con una variante del mergesort a più vie, con porzioni ordinate e aggregazioni effettuate esaminando i valori affioranti di ciascuna partizione.

Considerare la relazione sotto schematizzata, sugli attributi `A`, `B` e `C`. Supponendo una disponibilità di buffer abbastanza ampia, ma non sufficiente a caricare in memoria l'intera relazione (supporre ad esempio una disponibilità di 4 buffer, con un fattore di blocco pari a 2 e quindi uno spazio occupato dalla relazione pari a 8 blocchi), considerare l'esecuzione dell'interrogazione sopra mostrata, in due passate (il che è possibile, visto che il numero di buffer è maggiore della radice del numero di blocchi).

Mostrare lo stato delle strutture in memoria centrale e secondaria dopo l'esecuzione di tre chiamate al metodo `next()` sullo scan che implementa l'interrogazione complessiva. In particolare, mostrare i "run" (cioè le porzioni di file ordinate durante prima passata — per comodità mostrare tutti gli attributi) memorizzati su disco e i buffer in memoria centrale, evidenziando per ciascun buffer il record corrente (il primo non ancora considerato). Mostrare anche i record prodotti dalle prime tre chiamate di `next()`.

Run su disco | Buffer | Record prodotti dalle prime tre `next()`

A	B	C
101	c	3
102	d	2
103	d	4
104	c	3
105	f	3
106	d	1
107	e	3
108	f	4
109	c	5
110	d	2
111	c	2
112	f	4
113	c	1
114	f	2
115	e	3
116	d	3

Basi di dati II — 1 luglio 2015 — Compito C

Domanda 6 (15%) Nel commit a due fasi, per ovviare alle conseguenze negative di un guasto del coordinatore, alcuni sistemi prevedono la possibilità di eleggere un nuovo coordinatore, fra i partecipanti rimanenti. Il nuovo coordinatore, a questo punto, contatta i partecipanti e, per ciascuna transazione T cerca di prendere una decisione. Per una data transazione, considerare i casi seguenti e per ciascuno, indicare se si può verificare, spiegare perché e indicare quale decisione può in tal caso prendere il nuovo coordinatore. Assumere che il preesistente coordinatore fosse anche un partecipante.

1. uno o più partecipanti hanno il record di abort nel log e uno ha il record di commit

Si può verificare (sì o no)?
Perché?

Quale decisione può prendere il coordinatore?

2. tutti i partecipanti hanno il record di ready nel log

Si può verificare (sì o no)?
Perché?

Quale decisione può prendere il coordinatore?

3. uno o più partecipanti (ma non tutti) hanno il record di ready nel log e nessuno ha il record di commit

Si può verificare (sì o no)?
Perché?

Quale decisione può prendere il coordinatore?

4. un partecipante ha il record di ready nel log e un altro ha il record di abort

Si può verificare (sì o no)?
Perché?

Quale decisione può prendere il coordinatore?

Basi di dati II — Esame — 1 luglio 2015 — Compito D

Staccare questo foglio all'inizio del compito per poter consultare quanto scritto sul retro.
Questo foglio non va consegnato

Basi di dati II — Esame — 1 luglio 2015 — Compito D

Tempo a disposizione: un'ora e quindici minuti per la prova breve e due ore e trenta minuti per la prova completa.

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

Domanda 1 (15% per la prova completa, 40% per la prova breve) Come visto a lezione, molti DBMS utilizzano, oltre a quelli basati sui B$^+$-tree, anche indici basati su una tecnica detta "bitmap." Si ricorda che un indice bitmap, su un attributo A con v valori diversi, per una relazione con N_R ennuple, è costituito da v vettori (uno per ciascun valore di A) di N_R bit ciascuno (un bit per ogni ennupla della relazione): l'i-esimo bit del vettore associato al valore a_j è 1 se il valore della i-esima ennupla sull'attributo A è a_j e 0 se il valore è diverso. L'accesso ai vettori è organizzato per mezzo di un albero. In sostanza, rispetto ad un B$^+$-tree, abbiamo una struttura di accesso simile, ma con foglie diverse: nel B$^+$-tree, per ogni valore dell'attributo abbiamo una lista di indirizzi, mentre nell'indice bitmap abbiamo un vettore di bit. Esiste poi una struttura (o un metodo) per associare gli indirizzi delle ennuple ai numeri da 1 a N_R. Sulla base degli elementi forniti, con riferimento ad un sistema con blocchi di dimensione $D = 4$ KB e indirizzi dei record di lunghezza $p = 8$ byte e ad una relazione R con N_R =1.600.000 ennuple e (fra gli altri) quattro attributi, A, C, E, F ciascuno con $v = 100$ valori diversi e una lunghezza $a = 8$ byte

1. Valutare lo spazio SB (in blocchi) necessario per le foglie di un indice bitmap su A; per confronto, calcolare lo spazio ST necessario per le foglie di un B$^+$-tree (con riferimenti ai record e una coppia chiave-valore per ciascun record) per un indice su A (N.B.: si noterà che le dimensioni dei bitmap come descritte non sono poi così convenienti; in pratica si usano tecniche di compressione, che omettiamo).

Poi, valutare (sia nel caso di indici bitmap, sia nel caso di B$^+$-tree) il costo (in termini di numero di accessi a memoria secondaria, supponendo che tutti i nodi intermedi di tutti gli indici siano nei buffer; basta quindi contare il numero di foglie da visitare e l'accesso alle ennuple; anche per quest'ultimo si può considerare il solo accesso al record, trascurando il costo di accesso alla struttura di supporto) per le seguenti interrogazioni

2. `SELECT * FROM R WHERE A = 5` (supporre che il risultato contenga N_R/v ennuple)
3. `SELECT * FROM R WHERE A = 15 AND C = 14 AND E = 13 AND F = 9` (non potendo sapere quanto i valori siano indipendenti, assumere che il risultato contenga 10 ennuple)

Riportare le risposte nella tabella sottostante, nelle righe da 1 a 3, indicando formula e valore numerico.

	Risposte	
	Indici bitmap	Indici B$^+$-tree
1		
2		
3		

Basi di dati II — 1 luglio 2015 — Compito D

Domanda 2 (25% per la prova completa, 60% per la prova breve) Con riferimento all'esempio discusso nelle esercitazioni ai fini del supporto alla valutazione della didattica e, in particolare, al relativo schema dei dati, fornito per comodità in allegato, mostrare uno schema dimensionale relativo all'analisi delle immatricolazioni.

Basi di dati II — 1 luglio 2015 — Compito D

Domanda 3 (10%) Considerare un sistema con dischi con $N = 500$ 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 $t_B = 10$ μsec

Rispondere alle seguenti domande mostrando formula e valore numerico

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

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

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 $p = 5$ e fan-out (fattore di blocco dell'indice) $f_I = 50$, usato di recente, ma in modo sporadico?

4. Qual è il tempo che si può ipotizzare necessario per eseguire $m = 20.000$ accessi diretti in tempi ravvicinati a record di un file (molto grande) attraverso un indice che abbia profondità $p = 5$, fan-out $f_I = 50$, con disponibilità di circa $P = 4000$ pagine di buffer?

Basi di dati II — 1 luglio 2015 — Compito D

Domanda 4 (15%) Si consideri un DBMS che preveda, in aggiunta alle tradizionali operazioni di lettura e scrittura, anche un'operazione PIÙ*(x,i)* che modifica il valore di x (intero) incrementandolo di i, senza renderlo disponibile alla transazione (il che richiederebbe un'esplicita operazione di lettura). Per includere questo tipo di operazioni, in aggiunta a quelle di lettura e scrittura, è stato proposto di utilizzare un terzo tipo di lock, detto p-lock, con la seguente regola di compatibilità (in aggiunta a quelle note per r-lock e w-lock): due p-lock su un oggetto sono fra loro compatibili, mentre un p-lock è incompatibile sia con un r-lock sia con un w-lock.

1. Spiegare intuitivamente perché questa tecnica può generare transazioni a maggiore livello di concorrenza (cioè con un throughput maggiore).

2. Mostrare che schedule 2-PL che soddisfino le regole di compatibilità note estese con quelle sopra citate risultano serializzabili.

Basi di dati II — 1 luglio 2015 — Compito D

Domanda 5 (20%)
Considerare le operazioni di raggruppamento, specificate in SQL con clausole `GROUP BY` e funzioni aggregative sui gruppi. Ad esempio:

```
SELECT B, SUM(C) FROM R GROUP BY B
```

Queste operazioni vengono concettualmente eseguite in due passi (supponiamo per semplicità che, come nell'esempio, ci sia un solo attributo di raggruppamento e una sola funzione aggregativa)

1. partizionare le ennuple dell'operando sulla base dei valori dell'attributo di raggruppamento (`B` nell'esempio)
2. per ogni partizione, produrre una ennupla con (i) il valore dell'attributo di raggruppamento e (ii) il valore della funzione aggregativa (`SUM(C)` nell'esempio) applicata alle ennuple della partizione

Una possibile implementazione efficiente può essere ottenuta con una variante del mergesort a più vie, con porzioni ordinate e aggregazioni effettuate esaminando i valori affioranti di ciascuna partizione.

Considerare la relazione sotto schematizzata, sugli attributi `A`, `B` e `C`. Supponendo una disponibilità di buffer abbastanza ampia, ma non sufficiente a caricare in memoria l'intera relazione (supporre ad esempio una disponibilità di 4 buffer, con un fattore di blocco pari a 2 e quindi uno spazio occupato dalla relazione pari a 8 blocchi), considerare l'esecuzione dell'interrogazione sopra mostrata, in due passate (il che è possibile, visto che il numero di buffer è maggiore della radice del numero di blocchi).

Mostrare lo stato delle strutture in memoria centrale e secondaria dopo l'esecuzione di tre chiamate al metodo `next()` sullo scan che implementa l'interrogazione complessiva. In particolare, mostrare i "run" (cioè le porzioni di file ordinate durante prima passata — per comodità mostrare tutti gli attributi) memorizzati su disco e i buffer in memoria centrale, evidenziando per ciascun buffer il record corrente (il primo non ancora considerato). Mostrare anche i record prodotti dalle prime tre chiamate di `next()`.

Run su disco | Buffer | Record prodotti dalle prime tre `next()`

A	B	C
101	c	4
102	d	2
103	d	4
104	c	4
105	f	3
106	d	1
107	e	3
108	f	4
109	c	5
110	d	2
111	c	2
112	f	4
113	c	1
114	f	2
115	e	3
116	d	3

Basi di dati II — 1 luglio 2015 — Compito D

Domanda 6 (15%) Nel commit a due fasi, per ovviare alle conseguenze negative di un guasto del coordinatore, alcuni sistemi prevedono la possibilità di eleggere un nuovo coordinatore, fra i partecipanti rimanenti. Il nuovo coordinatore, a questo punto, contatta i partecipanti e, per ciascuna transazione T cerca di prendere una decisione. Per una data transazione, considerare i casi seguenti e per ciascuno, indicare se si può verificare, spiegare perché e indicare quale decisione può in tal caso prendere il nuovo coordinatore. Assumere che il preesistente coordinatore fosse anche un partecipante.

1. uno o più partecipanti (ma non tutti) hanno il record di ready nel log e uno ha il record di commit

Si può verificare (sì o no)?
Perché?

Quale decisione può prendere il coordinatore?

2. uno o più partecipanti (ma non tutti) hanno il record di ready nel log e nessuno ha il record di commit

Si può verificare (sì o no)?
Perché?

Quale decisione può prendere il coordinatore?

3. tutti i partecipanti hanno il record di ready nel log

Si può verificare (sì o no)?
Perché?

Quale decisione può prendere il coordinatore?

4. uno o più partecipanti (ma non tutti) hanno il record di commit nel log e gli altri il record di ready

Si può verificare (sì o no)?
Perché?

Quale decisione può prendere il coordinatore?

Basi di dati II — Esame — 1 luglio 2015 — Compito A

Staccare questo foglio all'inizio del compito per poter consultare quanto scritto sul retro.
Questo foglio non va consegnato

Basi di dati II — Esame — 1 luglio 2015 — Compito A

Cenni sulle soluzioni
(solo Compito A, le varianti del testo sono in rosso)

Tempo a disposizione: un'ora e quindici minuti per la prova breve e due ore e trenta minuti per la prova completa.

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

Domanda 1 (15% per la prova completa, 40% per la prova breve) Come visto a lezione, molti DBMS utilizzano, oltre a quelli basati sui B$^+$-tree, anche indici basati su una tecnica detta "bitmap." Si ricorda che un indice bitmap, su un attributo A con a valori diversi, per una relazione con N_R ennuple, è costituito da a vettori (uno per ciascun valore di A) di N_R bit ciascuno (un bit per ogni ennupla della relazione): l'i-esimo bit del vettore associato al valore a_j è 1 se il valore della i-esima ennupla sull'attributo A è a_j e 0 se il valore è diverso. L'accesso ai vettori è organizzato per mezzo di un albero. In sostanza, rispetto ad un B$^+$-tree, abbiamo una struttura di accesso simile, ma con foglie diverse: nel B$^+$-tree, per ogni valore dell'attributo abbiamo una lista di indirizzi, mentre nell'indice bitmap abbiamo un vettore di bit. Esiste poi una struttura (o un metodo) per associare gli indirizzi delle ennuple ai numeri da 1 a N_R. Sulla base degli elementi forniti, con riferimento ad un sistema con blocchi di dimensione $B = 4$ KB e indirizzi dei record di lunghezza $b = 8$ byte e ad una relazione R con N_R =1.600.000 ennuple e (fra gli altri) quattro attributi, A, C, E, F ciascuno con $a = 100$ valori diversi e una lunghezza $c = 8$ byte

1. Valutare lo spazio SB (in blocchi) necessario per le foglie di un indice bitmap su A; per confronto, calcolare lo spazio ST necessario per le foglie di un B$^+$-tree (con riferimenti ai record e una coppia chiave-valore per ciascun record) per un indice su A (N.B.: si noterà che le dimensioni dei bitmap come descritte non sono poi così convenienti; in pratica si usano tecniche di compressione, che omettiamo).

Poi, valutare (sia nel caso di indici bitmap, sia nel caso di B$^+$-tree) il costo (in termini di numero di accessi a memoria secondaria, supponendo che tutti i nodi intermedi di tutti gli indici siano nei buffer; basta quindi contare il numero di foglie da visitare e l'accesso alle ennuple; anche per quest'ultimo si può considerare il solo accesso al record, trascurando il costo di accesso alla struttura di supporto) per le seguenti interrogazioni

2. `SELECT * FROM R WHERE A = 5` (supporre che il risultato contenga N_R/a ennuple)
3. `SELECT * FROM R WHERE A = 15 AND C = 14 AND E = 13 AND F = 9` (non potendo sapere quanto i valori siano indipendenti, assumere che il risultato contenga 10 ennuple)

Riportare le risposte nella tabella sottostante, nelle righe da 1 a 3, indicando formula e valore numerico.

	Risposte	
	Indici bitmap	Indici B$^+$-tree
1	$\text{SB} = \frac{a \times N_R}{8 \times B} = \frac{100 \times 1.600.000}{8 \times 4000} = 5000$	$\text{ST} = \frac{N_R \times (b+c)}{B \times f} = \frac{1.6M \times 16}{4000 \times 0,66} =\sim 10.000$ (f è il fattore di riempimento del B$^+$-tree assunto pari a 0,66)
2	Basta accedere ad un vettore: $\frac{N_R}{8 \times B} = \frac{1.600.000}{8 \times 4000} = 50$ più l'accesso ai record (attraverso la struttura intermedia), che in questo caso domina $\frac{N_R}{8 \times B} + \frac{N_R}{a} = 50 + \frac{1.600.000}{100} =\sim 16.000$	$\frac{1}{a}\frac{N_R \times (b+c)}{B \times f} = \frac{1.600.000 \times 16}{100 \times 4000 \times 0,66} =\sim 100$ più l'accesso ai record $\ldots + \frac{N_R}{a} = \ldots + \frac{1.600.000}{100} =\sim 16.000$
4	(Si suppone un indice per ciascuno degli attributi) Quattro accessi a vettori, più l'accesso ai record $4 \times \frac{N_R}{8 \times B} + 2 \times 10 = 200 + 2 \times 10 =\sim 200$	(Si suppone un indice per ciascuno degli attributi) Quattro accessi a foglie, più l'accesso ai record $4 \times \frac{1}{a}\frac{N_R \times (b+c)}{B \times f} + 10 =\sim 400$

Basi di dati II — 1 luglio 2015 — Compito A

Domanda 2 (25% per la prova completa, 60% per la prova breve) Con riferimento all'esempio discusso nelle esercitazioni ai fini del supporto alla valutazione della didattica e, in particolare, al relativo schema dei dati, fornito per comodità in allegato, mostrare uno schema dimensionale relativo all'analisi delle immatricolazioni.

Basi di dati II — 1 luglio 2015 — Compito A

Domanda 3 (10%) Considerare un sistema con dischi con $N = 500$ 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 $t_B = 10$ μsec

Rispondere alle seguenti domande mostrando formula e valore numerico

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

$$t_{tot} = t_S + t_L + t_B = 5 + 3 + 0,01 \text{ msec } = \text{ ca } 8 \text{ msec}$$

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

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

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 $p = 5$ e fan-out (fattore di blocco dell'indice) $f_I = 50$, usato di recente, ma in modo sporadico?

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

$$((p - 1) + 1) \times t_{tot} = \text{ ca } 40 \text{ msec}$$

4. Qual è il tempo che si può ipotizzare necessario per eseguire $m = 10.000$ accessi diretti in tempi ravvicinati a record di un file (molto grande) attraverso un indice che abbia profondità $p = 5$, fan-out $f_I = 50$, con disponibilità di circa $P = 4000$ pagine di buffer?

Si può immaginare che la radice e altri due livelli 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 piè volte):

$$(p - 2 + 1) \times m \times t_{tot} = \text{ ca } 320 \text{ sec}$$

Basi di dati II — 1 luglio 2015 — Compito A

Domanda 4 (15%) Si consideri un DBMS che preveda, in aggiunta alle tradizionali operazioni di lettura e scrittura, anche un'operazione AUMENTA*(x,i)* che modifica il valore di x (intero) incrementandolo di i, senza renderlo disponibile alla transazione (il che richiederebbe un'esplicita operazione di lettura). Per includere questo tipo di operazioni, in aggiunta a quelle di lettura e scrittura, è stato proposto di utilizzare un terzo tipo di lock, detto a-lock, con la seguente regola di compatibilità (in aggiunta a quelle note per r-lock e w-lock): due a-lock su un oggetto sono fra loro compatibili, mentre un a-lock è incompatibile sia con un r-lock sia con un w-lock.

1. Spiegare intuitivamente perché questa tecnica può generare transazioni a maggiore livello di concorrenza (cioè con un throughput maggiore).

Con questa tecnica, operazioni di incremento multiple non si rallentano a vicenda, mentre con un approccio tradizionale ognuna di essere sarebbe costituita da una lettura e una scrittura, con conseguenti rallentamenti dovuti ai lock esclusivi.

2. Mostrare che schedule 2-PL che soddisfino le regole di compatibilità note estese con quelle sopra citate risultano serializzabili.

Facendo riferimento alla view serializzabilità, cambiare l'ordine di operazioni di incremento non modifica né i riusultati che vengono forniti all'esterno (perché gli incrementi non acquisiscono valori e quindi non permettono di comunicare niente) né le interazioni fra le transazioni e quindi le modifiche sulla base di dati.
Con riferimento alla conflict-serializzazbilità, di conseguenza, due incrementi non sono in conflitto fra loro.
Le regole sui lock permettono appunto di non forzare l'ordine sugli incrementi, ma fra incrementi e altre operazioni

Basi di dati II — 1 luglio 2015 — Compito A

Domanda 5 (20%)
Considerare le operazioni di raggruppamento, specificate in SQL con clausole `GROUP BY` e funzioni aggregative sui gruppi. Ad esempio:

```
SELECT B, SUM(C) FROM R GROUP BY B
```

Queste operazioni vengono concettualmente eseguite in due passi (supponiamo per semplicità che, come nell'esempio, ci sia un solo attributo di raggruppamento e una sola funzione aggregativa)

1. partizionare le ennuple dell'operando sulla base dei valori dell'attributo di raggruppamento (`B` nell'esempio)
2. per ogni partizione, produrre una ennupla con (i) il valore dell'attributo di raggruppamento e (ii) il valore della funzione aggregativa (`SUM(C)` nell'esempio) applicata alle ennuple della partizione

Una possibile implementazione efficiente può essere ottenuta con una variante del mergesort a più vie, con porzioni ordinate e aggregazioni effettuate esaminando i valori affioranti di ciascuna partizione.

Considerare la relazione sotto schematizzata, sugli attributi `A`, `B` e `C`. Supponendo una disponibilità di buffer abbastanza ampia, ma non sufficiente a caricare in memoria l'intera relazione (supporre ad esempio una disponibilità di 4 buffer, con un fattore di blocco pari a 2 e quindi uno spazio occupato dalla relazione pari a 8 blocchi), considerare l'esecuzione dell'interrogazione sopra mostrata, in due passate (il che è possibile, visto che il numero di buffer è maggiore della radice del numero di blocchi).

Mostrare lo stato delle strutture in memoria centrale e secondaria dopo l'esecuzione di tre chiamate al metodo `next()` sullo scan che implementa l'interrogazione complessiva. In particolare, mostrare i "run" (cioè le porzioni di file ordinate durante prima passata — per comodità mostrare tutti gli attributi) memorizzati su disco e i buffer in memoria centrale, evidenziando per ciascun buffer il record corrente (il primo non ancora considerato). Mostrare anche i record prodotti dalle prime tre chiamate di `next()`.

A	B	C
101	c	1
102	d	2
103	d	4
104	c	1
105	f	3
106	d	1
107	e	3
108	f	4
109	c	5
110	d	2
111	c	2
112	f	4
113	c	1
114	f	2
115	e	3
116	d	3

Run su disco

101	c	1
104	c	1
102	d	2
103	d	4
106	d	1
105	f	3

109	c	5
111	c	2
110	d	2
107	e	3
108	f	4
112	f	4

113	c	1
116	d	3
115	e	3
114	f	2

Buffer

106	d	1	←
105	f	3	

108	f	4	←
112	f	4	

115	e	3	←
114	f	2	

Record prodotti dalle prime 3 `next()`

c	10
d	12
e	6

Il numero ideale di buffer (da utilizzare sia nella prima sia nella seconda passata) è pari alla radice quadrata del numero dei blocchi e quindi a tre.
Nella prima passata, si costruiscono quindi tre run ordinati, ciascuno a partire da tre blocchi del file originario. L'ordinamento di ciascun run può essere effettuato in memoria centrale, usando tre buffer. Poiché il risultato della prima passata viene materializzato, vengono mostrati i run ordinati.
Nella seconda passata, si carica un blocco per ciascuno dei run e si fa calcola l'aggregazione usando nuovamente tre buffer e "consumando" i record via via. La seconda passata viene svolta in pipeline, e quindi i buffer sono mostrati con il contenuto che hanno quando è stato appena prodotto il terzo record.

Basi di dati II — 1 luglio 2015 — Compito A

Domanda 6 (15%) Nel commit a due fasi, per ovviare alle conseguenze negative di un guasto del coordinatore, alcuni sistemi prevedono la possibilità di eleggere un nuovo coordinatore, fra i partecipanti rimanenti. Il nuovo coordinatore, a questo punto, contatta i partecipanti e, per ciascuna transazione T cerca di prendere una decisione. Per una data transazione, considerare i casi seguenti e per ciascuno, indicare se si può verificare, spiegare perché e indicare quale decisione può in tal caso prendere il nuovo coordinatore. Assumere che il preesistente coordinatore fosse anche un partecipante.

1. uno o più partecipanti (ma non tutti) hanno il record di ready nel log e nessuno ha il record di commit

Si può verificare (sì o no)? sì
Perché?
Qualche partecipante non ha ricevuto o ha ricevuto e non accettato e la decisione non è stata presa
Quale decisione può prendere il coordinatore?
Abort (oppure attendere, ma non è opportuno)

2. un partecipante ha il record di ready nel log e un altro ha il record di abort

Si può verificare (sì o no)? sì
Perché?
Sono compatibili
Quale decisione può prendere il coordinatore?
Abort

3. tutti i partecipanti hanno il record di ready nel log

Si può verificare (sì o no)? sì
Perché?
I partecipanti contattati hanno ricevuto tutti e accettato
Quale decisione può prendere il coordinatore?
Nessuna, perché il vecchio coordinatore è anche un partecipante e non sappiamo se ha deciso

4. uno o più partecipanti hanno il record di abort nel log e uno ha il record di commit

Si può verificare (sì o no)? no
Perché?
La decisione è una
Quale decisione può prendere il coordinatore?
—