Basi di dati II — Esame — 28 giugno 2016 — Compito A

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

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

Domanda 1 (30% per la prova breve e 10% per quella lunga)
Considerare un sistema che utilizzi blocchi di lunghezza $D = 8$ KB (approssimabili a 8000 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 $T = 50.000$ 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 50.000 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 undo-only (no-redo)
  - strategia redo-only (no-undo)

Come nel caso precedente, ma con riferimento ad un programma che, per realizzare i 50.000 inserimenti, utilizzi complessivamente $k = 10$ transazioni, ognuna con 5000 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 undo-only (no-redo)

    :
  - strategia redo-only (no-undo)

    :

Basi di dati II — 28 giugno 2016 — Compito A

Domanda 2 (35% per la prova breve e 12,5% per quella lunga)
Considerare il seguente scenario in cui tre 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. 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).

client 1	client 2	client 3
		begin read(x)
begin read(x)		
	begin read(x) x = x + 20 write(x) commit	
x = x + 10 write(x) commit		
		read(x) commit

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 **300**. 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

Si verificano anomalie?

Basi di dati II — 28 giugno 2016 — Compito A

Considerare nuovamente lo scenario della pagina precedente, ripetuto qui sotto per comodità.

client 1	client 2	client 3
		begin read(x)
begin read(x)		
	begin read(x) x = x + 20 write(x) commit	
x = x + 10 write(x) commit		
		read(x) commit

Considerare uno scheduler con controllo di concorrenza ancora basato su **Multversioni** (come in Postgres) ma con livello di isolamento **SERIALIZABLE** sulle prime due transazioni e **READ COMMITTED** sulla terza. Mostrare il comportamento dello scheduler, supponendo che il valore iniziale dell'oggetto x sia ancora **3**.

client 1	client 2	client 3

Si verificano anomalie?

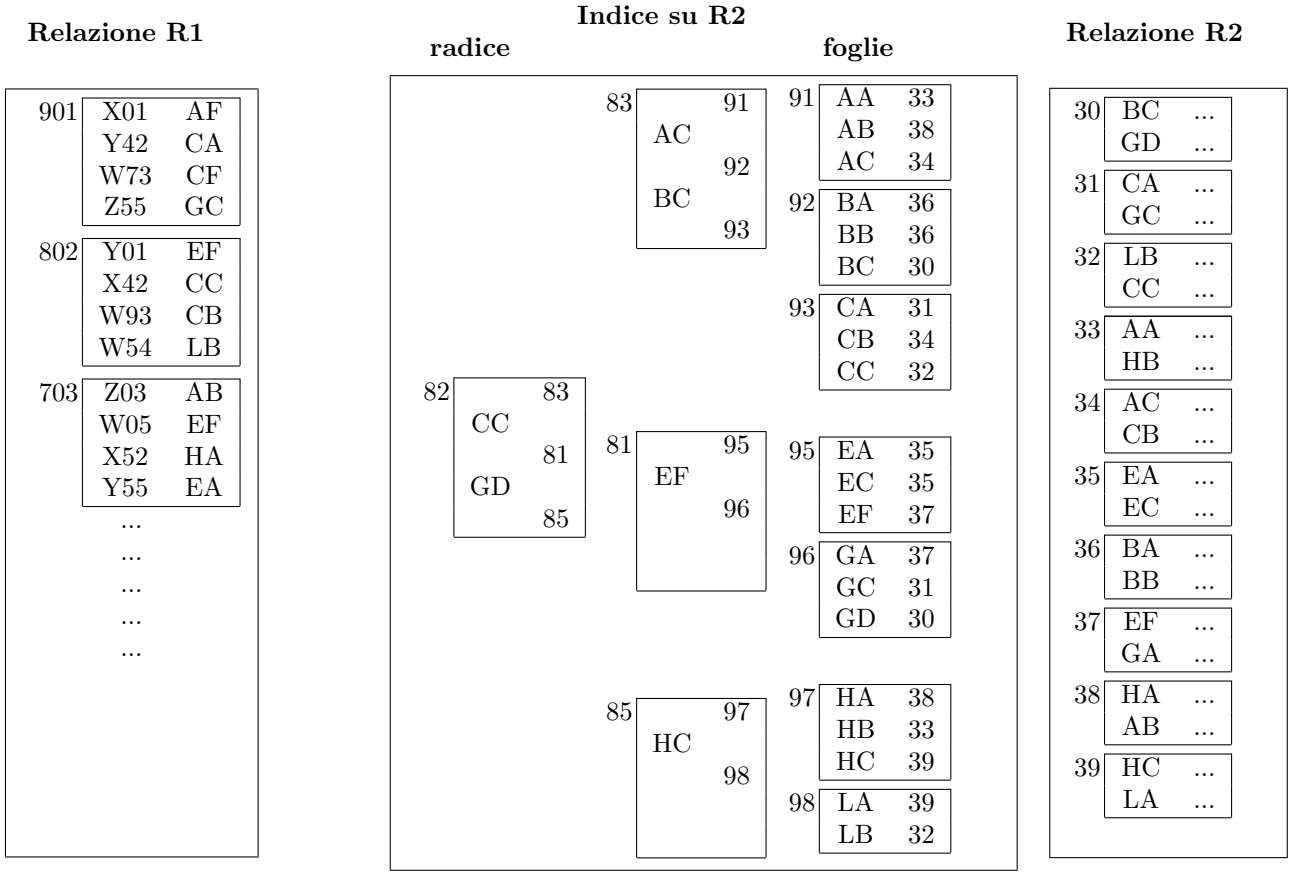
Basi di dati II — 28 giugno 2016 — Compito A

Domanda 3 (35% per la prova breve e 12,5% per quella lunga)
Considerare un sistema distribuito con tre nodi N_1, N_2 e N_3, che eseguono due transazioni T_A e T_B che coinvolgono i tre nodi, in modo diverso. Per la prima transazione N_1 è il coordinatore, mentre per la seconda il coordinatore è N_2. I due coordinatori inviano, come riportato nello schema sottostante, le richieste di `prepare`. Il nodo N_3 va in crash subito dopo aver risposto alla prima richiesta (senza avere il tempo di ricevere il messaggio di `commit`) e prima di ricevere la seconda. Indicare, nello schema sottostante, una possibile sequenza di scritture sui log e invio di messaggi (che includa anche i passi sopra illustrati), supponendo che entrambi i nodi siano ripristinati abbastanza presto (ma che vengano persi alcuni messaggi di risposta, ad esempio inviati a seguito di una decisione). Per i messaggi si usi la notazione *tipo(transaz)→destinatari* (come nell'esempio: `prepare(`T_A`)→`N_2`,`N_3). Supporre che nel log del coordinatore si scrivano solo i record di `prepare`, `commit` e `complete`, con i messaggi gestiti invece in memoria. Indicare ragionevoli istanti per i timeout, che permettano di concludere il protocollo per entrambe le transazioni.

Nodo N_1		Nodo N_2		Nodo N_3	
Log	Messaggi	Log	Messaggi	Log	Messaggi
`prep(`T_A`,`N_2`,`N_3`)`					
	`prep(`T_A`)→`N_2`,`N_3				
				crash	
		`prep(`T_B`,`N_1`,`N_3`)`			
			`prep(`T_B`)→`N_1`,`N_3		
crash					
restart					
				restart	

Basi di dati II — 28 giugno 2016 — Compito A

Domanda 4 (15%) Considerare le relazioni R1 ed R2 e l'indice I2 su R2 schematizzati sotto. I riquadri interni indicano i blocchi e il numero a fianco a ciascun riquadro indica l'indirizzo del blocco. Nell'indice, i valori numerici sono riferimenti ai blocchi (blocchi dell'indice, per la radice e il livello intermedio, e blocchi di R2 per le foglie).



Supponendo di disporre di un buffer di **sei** pagine, considerare l'esecuzione del join di R1 ed R2, sulla base dei valori del secondo attributo di R1 e del primo di R2, con un **nested loop con accesso diretto** tramite l'indice di R2.

Indicare gli indirizzi dei blocchi su cui si eseguono operazioni di pin (o fix) per produrre le prime tre ennuple del risultato.

Assumendo una politica di rimpiazzo *LRU*, indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato.

In tal caso, indicare gli indirizzi dei blocchi che si può presumere si trovino nei buffer nel momento in cui si produce la terza ennupla.

Indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato, con riferimento ad una politica di rimpiazzo *clock*.

Basi di dati II — 28 giugno 2016 — Compito A

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 $L = 2$Byte, molto più piccola della dimensione del blocco pari a $B = 4000$ Byte. Si supponga che le relazioni abbiano entrambe $N = 2.000.000$ 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.

(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.

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

Ripetere la valutazione effettuata alla pagina precedente nel caso in cui le due frequenze sono invece $f_1 = 1000$ e $f_2 = 10$

(i) memorizzazione separata delle due relazioni ...

(ii) memorizzazione in un cluster delle due relazioni ...

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

Basi di dati II — 28 giugno 2016 — Compito A

Domanda 6 (30%) Considerare uno schema dimensionale relativo a presenze e punti di giocatori di pallacanestro, che utilizzi, come tabella dei fatti e come una delle dimensioni, relazioni come quelle qui schematizzate:

KGiocatore	KSquadra	KTorneo	KStagione	Presenze	Punti
301	201	401	21	...	...
301	202	402	28	...	...
302	201	401	30	...	...
302	203	403	22	...	...
...	...	...	...	...	...

KTorneo	Nome	...
401	Serie A	...
402	Coppa dei Campioni	...
403	Coppa Italia	...
...	...	...

Supporre che si presentino le seguenti esigenze di modifica:

- I tornei possono cambiare nome nel tempo: per esempio, il torneo nella seconda ennupla potrebbe ad un certo punto cambiare nome da "Coppa dei Campioni" in "Eurolega"; interessano selezioni e aggregazioni relative alle presenze e ai punti tanto con riferimento al nome del torneo (distinguendo quindi la "Coppa dei Campioni" dalla "Eurolega") quanto alla sua identità ("Coppa dei Campioni" e "Eurolega" in questo senso avrebbero la stessa identità; allo scopo si usa un codice, ma gli analisti talvolta preferiscono usare il nome attuale del torneo, quindi nell'esempio "Eurolega" anche per le vecchie edizioni). Le modifiche sono rare, ma è possibile che ci siano tornei con vari cambiamenti di nome. Mostrare modifiche alle relazioni (una o entrambe) che permettano di soddisfare questa esigenza (mostrare i dati, con riferimento a quelli presenti negli esempi sopra, aggiungendo nuovi dati ragionevoli, che permettano di comprendente le modifiche).

- Per ogni giocatore, interessa rappresentare anche l'allenatore sotto la guida del quale ha giocato (per sommare ad esempio presenze e punti segnati con un certo allenatore); supporre (anche se l'ipotesi non è realistica) che ogni squadra abbia, in ciascuna stagione, un solo allenatore e che siano disponibili, nella staging area, tutti i dati relativi agli allenatori nelle varie stagioni. Anche in questo caso, mostrare le modifiche da apportare a una o a entrambe le relazioni.

- Con riferimento al quesito precedente, considerare invece il caso, più realistico, in cui gli allenatori possono cambiare nel corso della stagione. Indicare quali modifiche sarebbero necessarie allo schema dimensionale e, soprattutto, quali dati sarebbero necessari nella staging area per poter riorganizzare il tutto.

Basi di dati II — Esame — 28 giugno 2016 — Compito B

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

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

Domanda 1 (30% per la prova breve e 10% per quella lunga)
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 undo-only (no-redo)
  - strategia redo-only (no-undo)

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 undo-only (no-redo)

    :
  - strategia redo-only (no-undo)

    :

Basi di dati II — 28 giugno 2016 — Compito B

Domanda 2 (35% per la prova breve e 12,5% per quella lunga)
Considerare il seguente scenario in cui tre 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. 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).

client 1	client 2	client 3
begin		
read(x)		
	begin	
	read(x)	
x = x + 10		
write(x)		
		begin
		read(x)
	x = x + 20	
	write(x)	
	commit	
commit		
		read(x)
		commit

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 **200**. 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

Si verificano anomalie?

Basi di dati II — 28 giugno 2016 — Compito B

Considerare nuovamente lo scenario della pagina precedente, ripetuto qui sotto per comodità.

client 1	client 2	client 3
begin		
read(x)		
	begin	
	read(x)	
x = x + 10		
write(x)		
		begin
		read(x)
	x = x + 20	
	write(x)	
	commit	
commit		
		read(x)
		commit

Considerare uno scheduler con controllo di concorrenza ancora basato su **Multversioni** (come in Postgres) ma con livello di isolamento **SERIALIZABLE** sulle prime due transazioni e **READ COMMITTED** sulla terza. Mostrare il comportamento dello scheduler, supponendo che il valore iniziale dell'oggetto x sia ancora **2**.

client 1	client 2	client 3

Si verificano anomalie?

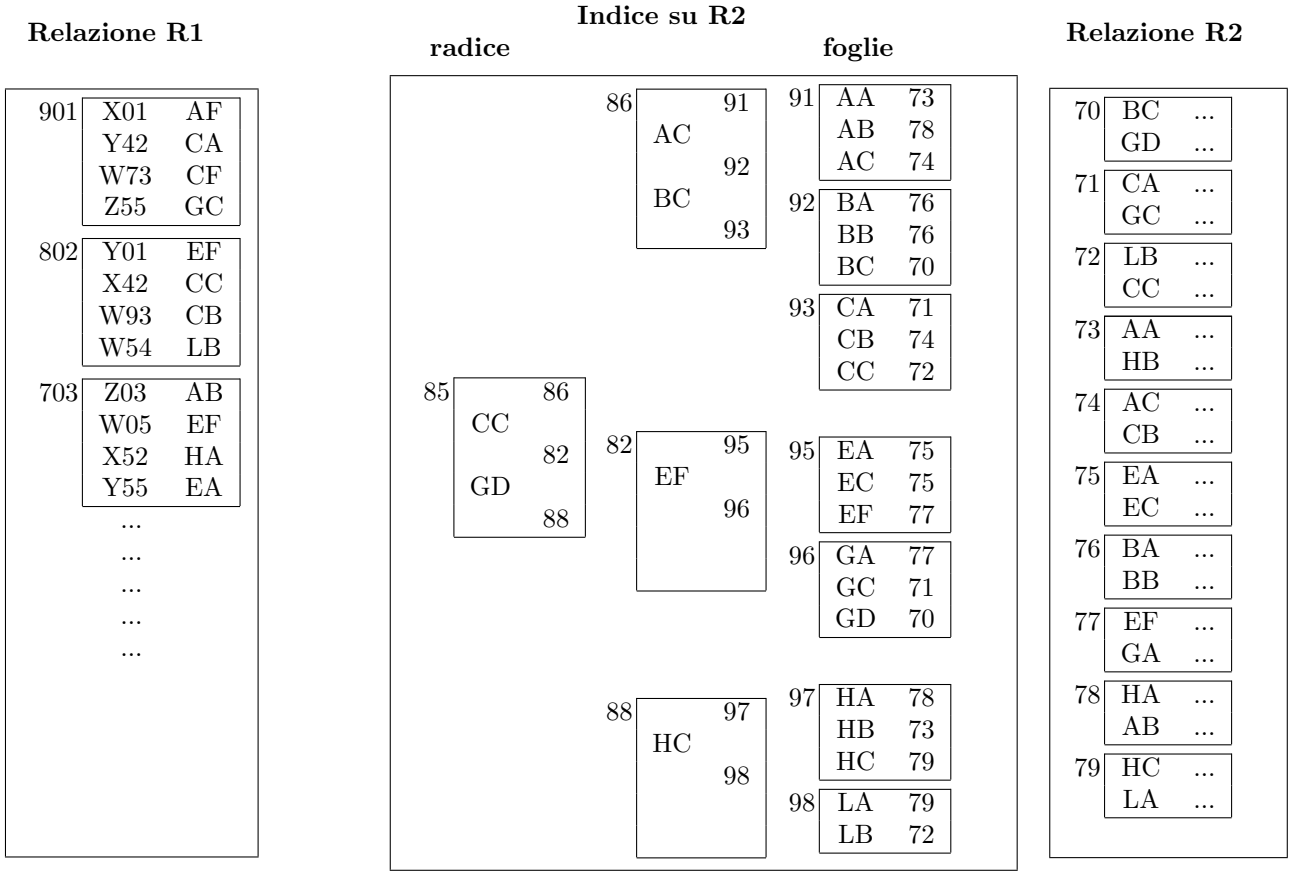
Basi di dati II — 28 giugno 2016 — Compito B

Domanda 3 (35% per la prova breve e 12,5% per quella lunga)
Considerare un sistema distribuito con tre nodi A, B e C, che eseguono due transazioni T_1 e T_2 che coinvolgono i tre nodi, in modo diverso. Per la prima transazione A è il coordinatore, mentre per la seconda il coordinatore è B. I due coordinatori inviano, come riportato nello schema sottostante, le richieste di `prepare`. Il nodo C va in crash subito dopo aver risposto alla prima richiesta (senza avere il tempo di ricevere il messaggio di `commit`) e prima di ricevere la seconda. Indicare, nello schema sottostante, una possibile sequenza di scritture sui log e invio di messaggi (che includa anche i passi sopra illustrati), supponendo che entrambi i nodi siano ripristinati abbastanza presto (ma che vengano persi alcuni messaggi di risposta, ad esempio inviati a seguito di una decisione). Per i messaggi si usi la notazione *tipo(transaz)→destinatari* (come nell'esempio: `prepare(`T_1`)→B,C`). Supporre che nel log del coordinatore si scrivano solo i record di `prepare`, `commit` e `complete`, con i messaggi gestiti invece in memoria. Indicare ragionevoli istanti per i timeout, che permettano di concludere il protocollo per entrambe le transazioni.

Nodo A		Nodo B		Nodo C	
Log	Messaggi	Log	Messaggi	Log	Messaggi
`prep(`T_1`,B,C)`	`prep(`T_1`)→B,C`				
				crash	
		`prep(`T_2`,A,C)`			
			`prep(`T_2`)→A,C`		
crash					
restart					
				restart	

Basi di dati II — 28 giugno 2016 — Compito B

Domanda 4 (15%) Considerare le relazioni R1 ed R2 e l'indice I2 su R2 schematizzati sotto. I riquadri interni indicano i blocchi e il numero a fianco a ciascun riquadro indica l'indirizzo del blocco. Nell'indice, i valori numerici sono riferimenti ai blocchi (blocchi dell'indice, per la radice e il livello intermedio, e blocchi di R2 per le foglie).



Supponendo di disporre di un buffer di **sei** pagine, considerare l'esecuzione del join di R1 ed R2, sulla base dei valori del secondo attributo di R1 e del primo di R2, con un **nested loop con accesso diretto** tramite l'indice di R2.

Indicare gli indirizzi dei blocchi su cui si eseguono operazioni di pin (o fix) per produrre le prime tre ennuple del risultato.

Assumendo una politica di rimpiazzo *LRU*, indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato.

In tal caso, indicare gli indirizzi dei blocchi che si può presumere si trovino nei buffer nel momento in cui si produce la terza ennupla.

Indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato, con riferimento ad una politica di rimpiazzo *clock*.

Basi di dati II — 28 giugno 2016 — Compito B

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 $a = 2$Byte, molto più piccola della dimensione del blocco pari a $B = 4000$ Byte. Si supponga che le relazioni abbiano entrambe $L = 2.000.000$ 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 $f_1 = 1000$;
- o_2 scansione dell'intera relazione R_1, con frequenza $f_2 = 10$

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.

(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.

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

Ripetere la valutazione effettuata alla pagina precedente nel caso in cui le due frequenze sono invece $f_1 = 10.000$ e $f_2 = 1$

(i) memorizzazione separata delle due relazioni ...

(ii) memorizzazione in un cluster delle due relazioni ...

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

Basi di dati II — 28 giugno 2016 — Compito B

Domanda 6 (30%) Considerare uno schema dimensionale relativo a presenze e punti di giocatori di pallacanestro, che utilizzi, come tabella dei fatti e come una delle dimensioni, relazioni come quelle qui schematizzate:

KGiocatore	KSquadra	KTorneo	KStagione	Presenze	Punti
201	401	301	27	...	...
201	402	302	28	...	...
202	401	301	30	...	...
202	403	303	22	...	...
...	...	...	...	...	...

KTorneo	Nome	...
301	Serie A	...
302	Coppa dei Campioni	...
303	Coppa Italia	...
...	...	...

Supporre che si presentino le seguenti esigenze di modifica:

- I tornei possono cambiare nome nel tempo: per esempio, il torneo nella seconda ennupla potrebbe ad un certo punto cambiare nome da "Coppa dei Campioni" in "Eurolega"; interessano selezioni e aggregazioni relative alle presenze e ai punti tanto con riferimento al nome del torneo (distinguendo quindi la "Coppa dei Campioni" dalla "Eurolega") quanto alla sua identità ("Coppa dei Campioni" e "Eurolega" in questo senso avrebbero la stessa identità; allo scopo si usa un codice, ma gli analisti talvolta preferiscono usare il nome attuale del torneo, quindi nell'esempio "Eurolega" anche per le vecchie edizioni). Le modifiche sono rare, ma è possibile che ci siano tornei con vari cambiamenti di nome. Mostrare modifiche alle relazioni (una o entrambe) che permettano di soddisfare questa esigenza (mostrare i dati, con riferimento a quelli presenti negli esempi sopra, aggiungendo nuovi dati ragionevoli, che permettano di comprendente le modifiche).

Basi di dati II — 28 giugno 2016 — Compito B

- Per ogni giocatore, interessa rappresentare anche l'allenatore sotto la guida del quale ha giocato (per sommare ad esempio presenze e punti segnati con un certo allenatore); supporre (anche se l'ipotesi non è realistica) che ogni squadra abbia, in ciascuna stagione, un solo allenatore e che siano disponibili, nella staging area, tutti i dati relativi agli allenatori nelle varie stagioni. Anche in questo caso, mostrare le modifiche da apportare a una o a entrambe le relazioni.

- Con riferimento al quesito precedente, considerare invece il caso, più realistico, in cui gli allenatori possono cambiare nel corso della stagione. Indicare quali modifiche sarebbero necessarie allo schema dimensionale e, soprattutto, quali dati sarebbero necessari nella staging area per poter riorganizzare il tutto.

Basi di dati II — Esame — 28 giugno 2016 — Compito C

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

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

Domanda 1 (30% per la prova breve e 10% per quella lunga)
Considerare un sistema che utilizzi blocchi di lunghezza $D = 8$ KB (approssimabili a 8000 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 $R = 50.000$ 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 50.000 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 undo-only (no-redo)
  - strategia redo-only (no-undo)

Come nel caso precedente, ma con riferimento ad un programma che, per realizzare i 50.000 inserimenti, utilizzi complessivamente $k = 5$ 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 undo-only (no-redo)

    :
  - strategia redo-only (no-undo)

    :

Basi di dati II — 28 giugno 2016 — Compito C

Domanda 2 (35% per la prova breve e 12,5% per quella lunga)
Considerare il seguente scenario in cui tre 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. 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).

client 1	client 2	client 3
		begin read(x)
begin read(x)		
	begin read(x) x = x + 20 write(x) commit	
x = x + 10 write(x) commit		
		read(x) commit

Considerare uno scheduler con controllo di concorrenza basato su **Multversioni** (come in Postgres) e livello di isolamento **SERIALIZABLE** sulle prime due transazioni e **READ COMMITTED** sulla terza. Mostrare il comportamento dello scheduler, supponendo che il valore iniziale dell'oggetto x 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

Si verificano anomalie?

Basi di dati II — 28 giugno 2016 — Compito C

Considerare nuovamente lo scenario della pagina precedente, ripetuto qui sotto per comodità.

client 1	client 2	client 3
		begin read(x)
begin read(x)		
	begin read(x) x = x + 20 write(x) commit	
x = x + 10 write(x) commit		
		read(x) commit

Considerare uno scheduler con controllo di concorrenza ancora basato su **Multversioni** (come in Postgres) ma con 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 ancora **1**.

client 1	client 2	client 3

Si verificano anomalie?

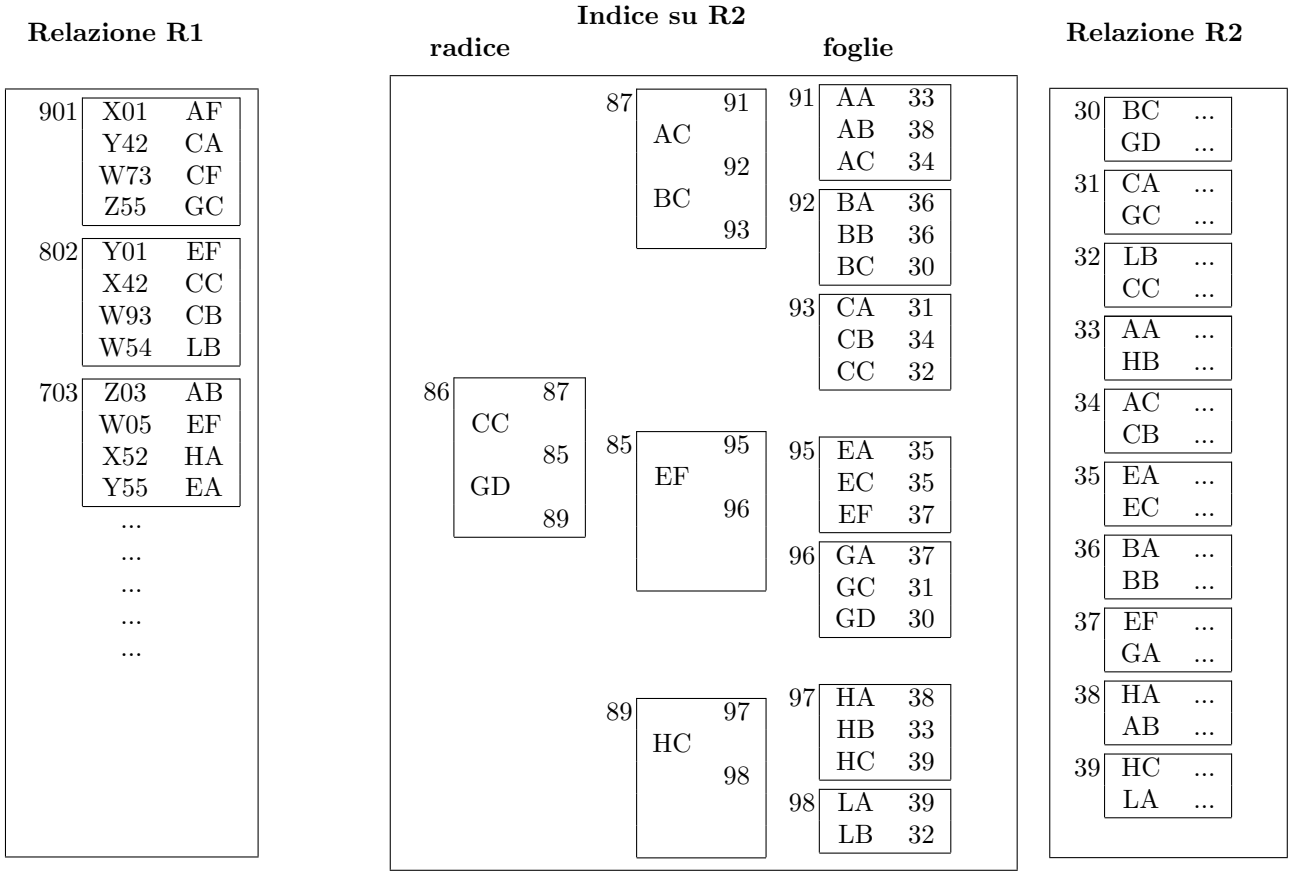
Basi di dati II — 28 giugno 2016 — Compito C

Domanda 3 (35% per la prova breve e 12,5% per quella lunga)
Considerare un sistema distribuito con tre nodi N_1, N_2 e N_3, che eseguono due transazioni T_x e T_y che coinvolgono i tre nodi, in modo diverso. Per la prima transazione N_1 è il coordinatore, mentre per la seconda il coordinatore è N_2. I due coordinatori inviano, come riportato nello schema sottostante, le richieste di `prepare`. Il nodo N_3 va in crash subito dopo aver risposto alla prima richiesta (senza avere il tempo di ricevere il messaggio di `commit`) e prima di ricevere la seconda. Indicare, nello schema sottostante, una possibile sequenza di scritture sui log e invio di messaggi (che includa anche i passi sopra illustrati), supponendo che entrambi i nodi siano ripristinati abbastanza presto (ma che vengano persi alcuni messaggi di risposta, ad esempio inviati a seguito di una decisione). Per i messaggi si usi la notazione *tipo(transaz)→destinatari* (come nell'esempio: `prepare(`T_x`)→`N_2`,`N_3). Supporre che nel log del coordinatore si scrivano solo i record di `prepare`, `commit` e `complete`, con i messaggi gestiti invece in memoria. Indicare ragionevoli istanti per i timeout, che permettano di concludere il protocollo per entrambe le transazioni.

Nodo N_1		Nodo N_2		Nodo N_3	
Log	Messaggi	Log	Messaggi	Log	Messaggi
`prep(`T_x`,`N_2`,`N_3`)`	`prep(`T_x`)→`N_2`,`N_3				
				crash	
		`prep(`T_y`,`N_1`,`N_3`)`			
			`prep(`T_y`)→`N_1`,`N_3		
crash					
restart					
				restart	

Basi di dati II — 28 giugno 2016 — Compito C

Domanda 4 (15%) Considerare le relazioni R1 ed R2 e l'indice I2 su R2 schematizzati sotto. I riquadri interni indicano i blocchi e il numero a fianco a ciascun riquadro indica l'indirizzo del blocco. Nell'indice, i valori numerici sono riferimenti ai blocchi (blocchi dell'indice, per la radice e il livello intermedio, e blocchi di R2 per le foglie).



Supponendo di disporre di un buffer di **sei** pagine, considerare l'esecuzione del join di R1 ed R2, sulla base dei valori del secondo attributo di R1 e del primo di R2, con un **nested loop con accesso diretto** tramite l'indice di R2.

Indicare gli indirizzi dei blocchi su cui si eseguono operazioni di pin (o fix) per produrre le prime tre ennuple del risultato.

Assumendo una politica di rimpiazzo *LRU*, indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato.

In tal caso, indicare gli indirizzi dei blocchi che si può presumere si trovino nei buffer nel momento in cui si produce la terza ennupla.

Indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato, con riferimento ad una politica di rimpiazzo *clock*.

Basi di dati II — 28 giugno 2016 — Compito C

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 $a = 2$Byte, molto più piccola della dimensione del blocco pari a $B = 4000$ Byte. Si supponga che le relazioni abbiano entrambe $L = 2.000.000$ 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.

(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.

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

Ripetere la valutazione effettuata alla pagina precedente nel caso in cui le due frequenze sono invece $f_1 = 1000$ e $f_2 = 10$

(i) memorizzazione separata delle due relazioni ...

(ii) memorizzazione in un cluster delle due relazioni ...

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

Basi di dati II — 28 giugno 2016 — Compito C

Domanda 6 (30%) Considerare uno schema dimensionale relativo a presenze e punti di giocatori di pallacanestro, che utilizzi, come tabella dei fatti e come una delle dimensioni, relazioni come quelle qui schematizzate:

KGiocatore	KSquadra	KTorneo	KStagione	Presenze	Punti
301	201	401	23	...	...
301	202	402	28	...	...
302	201	401	30	...	...
302	203	403	22	...	...
...	...	...	...	...	...

KTorneo	Nome	...
401	Serie A	...
402	Coppa dei Campioni	...
403	Coppa Italia	...
...	...	...

Supporre che si presentino le seguenti esigenze di modifica:

- I tornei possono cambiare nome nel tempo: per esempio, il torneo nella seconda ennupla potrebbe ad un certo punto cambiare nome da "Coppa dei Campioni" in "Eurolega"; interessano selezioni e aggregazioni relative alle presenze e ai punti tanto con riferimento al nome del torneo (distinguendo quindi la "Coppa dei Campioni" dalla "Eurolega") quanto alla sua identità ("Coppa dei Campioni" e "Eurolega" in questo senso avrebbero la stessa identità; allo scopo si usa un codice, ma gli analisti talvolta preferiscono usare il nome attuale del torneo, quindi nell'esempio "Eurolega" anche per le vecchie edizioni). Le modifiche sono rare, ma è possibile che ci siano tornei con vari cambiamenti di nome. Mostrare modifiche alle relazioni (una o entrambe) che permettano di soddisfare questa esigenza (mostrare i dati, con riferimento a quelli presenti negli esempi sopra, aggiungendo nuovi dati ragionevoli, che permettano di comprendente le modifiche).

Basi di dati II — 28 giugno 2016 — Compito C

- Per ogni giocatore, interessa rappresentare anche l'allenatore sotto la guida del quale ha giocato (per sommare ad esempio presenze e punti segnati con un certo allenatore); supporre (anche se l'ipotesi non è realistica) che ogni squadra abbia, in ciascuna stagione, un solo allenatore e che siano disponibili, nella staging area, tutti i dati relativi agli allenatori nelle varie stagioni. Anche in questo caso, mostrare le modifiche da apportare a una o a entrambe le relazioni.

- Con riferimento al quesito precedente, considerare invece il caso, più realistico, in cui gli allenatori possono cambiare nel corso della stagione. Indicare quali modifiche sarebbero necessarie allo schema dimensionale e, soprattutto, quali dati sarebbero necessari nella staging area per poter riorganizzare il tutto.

Basi di dati II — Esame — 28 giugno 2016 — Compito D

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

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

Domanda 1 (30% per la prova breve e 10% per quella lunga)
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 $N = 25.000$ 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 25.000 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 undo-only (no-redo)
  - strategia redo-only (no-undo)

Come nel caso precedente, ma con riferimento ad un programma che, per realizzare i 25.000 inserimenti, utilizzi complessivamente $k = 5$ transazioni, ognuna con 5000 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 undo-only (no-redo)

    :
  - strategia redo-only (no-undo)

    :

Basi di dati II — 28 giugno 2016 — Compito D

Domanda 2 (35% per la prova breve e 12,5% per quella lunga)
Considerare il seguente scenario in cui tre 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. 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).

client 1	client 2	client 3
begin		
read(x)		
	begin	
	read(x)	
x = x + 10		
write(x)		
		begin
		read(x)
	x = x + 20	
	write(x)	
	commit	
commit		
		read(x)
		commit

Considerare uno scheduler con controllo di concorrenza basato su **Multversioni** (come in Postgres) e livello di isolamento **SERIALIZABLE** sulle prime due transazioni e **READ COMMITTED** sulla terza. Mostrare il comportamento dello scheduler, supponendo che il valore iniziale dell'oggetto x sia **400**. 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

Si verificano anomalie?

Basi di dati II — 28 giugno 2016 — Compito D

Considerare nuovamente lo scenario della pagina precedente, ripetuto qui sotto per comodità.

client 1	client 2	client 3
begin		
read(x)		
	begin	
	read(x)	
x = x + 10		
write(x)		
		begin
		read(x)
	x = x + 20	
	write(x)	
	commit	
commit		
		read(x)
		commit

Considerare uno scheduler con controllo di concorrenza ancora basato su **Multversioni** (come in Postgres) ma con 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 ancora **4**.

client 1	client 2	client 3

Si verificano anomalie?

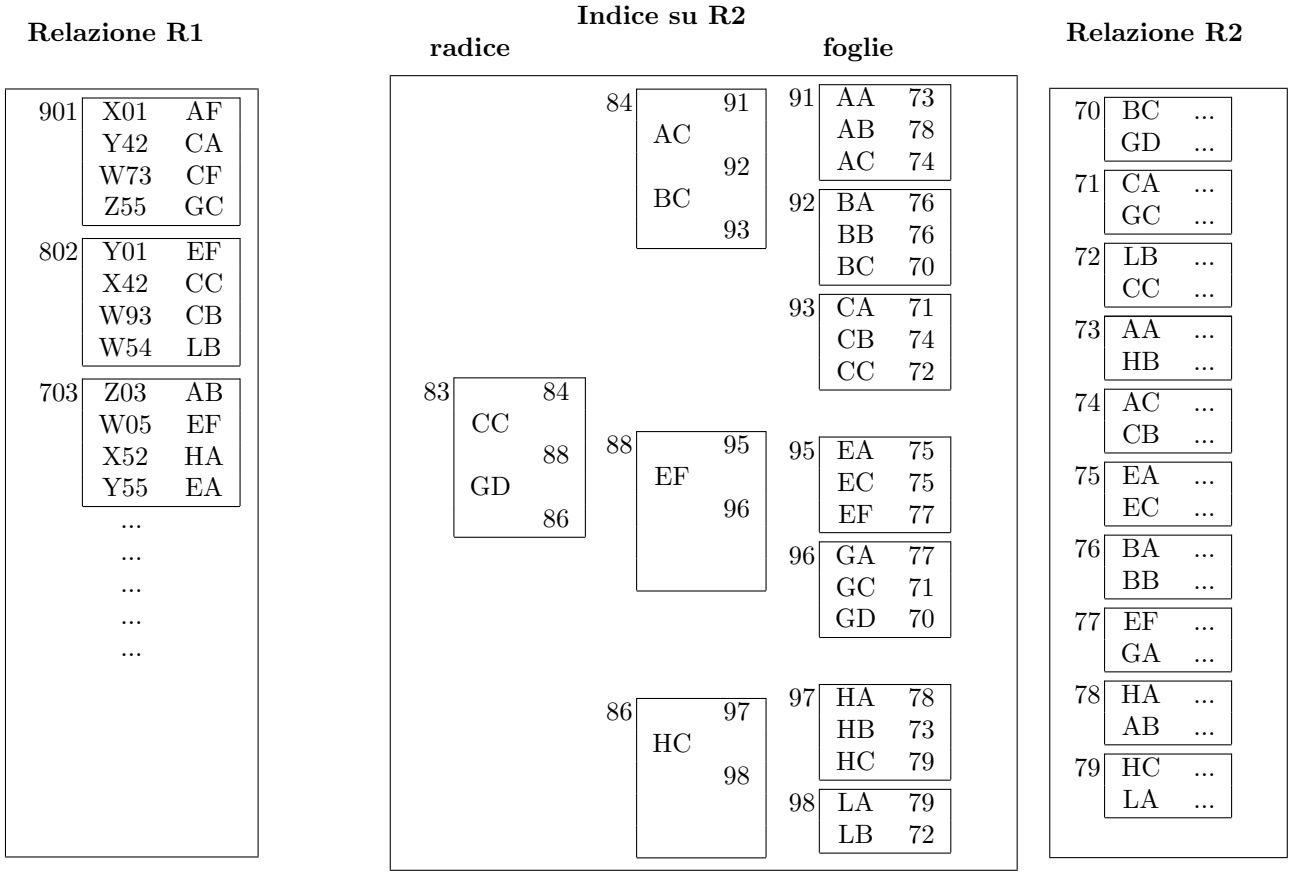
Basi di dati II — 28 giugno 2016 — Compito D

Domanda 3 (35% per la prova breve e 12,5% per quella lunga)
Considerare un sistema distribuito con tre nodi X, Y e Z, che eseguono due transazioni T_1 e T_2 che coinvolgono i tre nodi, in modo diverso. Per la prima transazione X è il coordinatore, mentre per la seconda il coordinatore è Y. I due coordinatori inviano, come riportato nello schema sottostante, le richieste di `prepare`. Il nodo Z va in crash subito dopo aver risposto alla prima richiesta (senza avere il tempo di ricevere il messaggio di `commit`) e prima di ricevere la seconda. Indicare, nello schema sottostante, una possibile sequenza di scritture sui log e invio di messaggi (che includa anche i passi sopra illustrati), supponendo che entrambi i nodi siano ripristinati abbastanza presto (ma che vengano persi alcuni messaggi di risposta, ad esempio inviati a seguito di una decisione). Per i messaggi si usi la notazione *tipo(transaz)→destinatari* (come nell'esempio: `prepare(`T_1`)→Y,Z`). Supporre che nel log del coordinatore si scrivano solo i record di `prepare`, `commit` e `complete`, con i messaggi gestiti invece in memoria. Indicare ragionevoli istanti per i timeout, che permettano di concludere il protocollo per entrambe le transazioni.

Nodo X		Nodo Y		Nodo Z	
Log	Messaggi	Log	Messaggi	Log	Messaggi
`prep(`T_1`,Y,Z)`	`prep(`T_1`)→Y,Z`				
				crash	
		`prep(`T_2`,X,Z)`			
			`prep(`T_2`)→X,Z`		
crash					
restart					
				restart	

Basi di dati II — 28 giugno 2016 — Compito D

Domanda 4 (15%) Considerare le relazioni R1 ed R2 e l'indice I2 su R2 schematizzati sotto. I riquadri interni indicano i blocchi e il numero a fianco a ciascun riquadro indica l'indirizzo del blocco. Nell'indice, i valori numerici sono riferimenti ai blocchi (blocchi dell'indice, per la radice e il livello intermedio, e blocchi di R2 per le foglie).



Supponendo di disporre di un buffer di **sei** pagine, considerare l'esecuzione del join di R1 ed R2, sulla base dei valori del secondo attributo di R1 e del primo di R2, con un **nested loop con accesso diretto** tramite l'indice di R2.

Indicare gli indirizzi dei blocchi su cui si eseguono operazioni di pin (o fix) per produrre le prime tre ennuple del risultato.

Assumendo una politica di rimpiazzo *LRU*, indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato.

In tal caso, indicare gli indirizzi dei blocchi che si può presumere si trovino nei buffer nel momento in cui si produce la terza ennupla.

Indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato, con riferimento ad una politica di rimpiazzo *clock*.

Basi di dati II — 28 giugno 2016 — Compito D

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 $L = 2$Byte, molto più piccola della dimensione del blocco pari a $B = 4000$ Byte. Si supponga che le relazioni abbiano entrambe $N = 2.000.000$ 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 $f_1 = 1000$;
- o_2 scansione dell'intera relazione R_1, con frequenza $f_2 = 10$

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.

(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.

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

Ripetere la valutazione effettuata alla pagina precedente nel caso in cui le due frequenze sono invece $f_1 = 10.000$ e $f_2 = 1$

(i) memorizzazione separata delle due relazioni ...

(ii) memorizzazione in un cluster delle due relazioni ...

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

Basi di dati II — 28 giugno 2016 — Compito D

Domanda 6 (30%) Considerare uno schema dimensionale relativo a presenze e punti di giocatori di pallacanestro, che utilizzi, come tabella dei fatti e come una delle dimensioni, relazioni come quelle qui schematizzate:

KGiocatore	KSquadra	KTorneo	KStagione	Presenze	Punti
401	201	301	27	...	...
401	202	302	28	...	...
402	201	301	30	...	...
402	203	303	22	...	...
...	...	...	...	...	...

KTorneo	Nome	...
301	Serie A	...
302	Coppa dei Campioni	...
303	Coppa Italia	...
...	...	...

Supporre che si presentino le seguenti esigenze di modifica:

- I tornei possono cambiare nome nel tempo: per esempio, il torneo nella seconda ennupla potrebbe ad un certo punto cambiare nome da "Coppa dei Campioni" in "Eurolega"; interessano selezioni e aggregazioni relative alle presenze e ai punti tanto con riferimento al nome del torneo (distinguendo quindi la "Coppa dei Campioni" dalla "Eurolega") quanto alla sua identità ("Coppa dei Campioni" e "Eurolega" in questo senso avrebbero la stessa identità; allo scopo si usa un codice, ma gli analisti talvolta preferiscono usare il nome attuale del torneo, quindi nell'esempio "Eurolega" anche per le vecchie edizioni). Le modifiche sono rare, ma è possibile che ci siano tornei con vari cambiamenti di nome. Mostrare modifiche alle relazioni (una o entrambe) che permettano di soddisfare questa esigenza (mostrare i dati, con riferimento a quelli presenti negli esempi sopra, aggiungendo nuovi dati ragionevoli, che permettano di comprendente le modifiche).

Basi di dati II — 28 giugno 2016 — Compito D

- Per ogni giocatore, interessa rappresentare anche l'allenatore sotto la guida del quale ha giocato (per sommare ad esempio presenze e punti segnati con un certo allenatore); supporre (anche se l'ipotesi non è realistica) che ogni squadra abbia, in ciascuna stagione, un solo allenatore e che siano disponibili, nella staging area, tutti i dati relativi agli allenatori nelle varie stagioni. Anche in questo caso, mostrare le modifiche da apportare a una o a entrambe le relazioni.

- Con riferimento al quesito precedente, considerare invece il caso, più realistico, in cui gli allenatori possono cambiare nel corso della stagione. Indicare quali modifiche sarebbero necessarie allo schema dimensionale e, soprattutto, quali dati sarebbero necessari nella staging area per poter riorganizzare il tutto.

Basi di dati II — Esame — 28 giugno 2016 — Compito A

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

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

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

Domanda 1 (30% per la prova breve e 10% per quella lunga)
Considerare un sistema che utilizzi blocchi di lunghezza $D = 8$ KB (approssimabili a 8000 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 $T = 50.000$ 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 50.000 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 $T = 50.000$, 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, $T = 50.000$; probabilmente di meno, anche solo $T/f = 125$, una per blocco (dove f indica il fattore di blocco $f = D/L = 400$ nei compiti A e C e 200 nei compiti B e D)
  - strategia undo-only (no-redo) : in questo caso certamente una per transazione, $T = 50.000$, perché è l'unico modo per evitare il redo: le modifiche debbono essere scritte prima del commit
  - strategia redo-only (no-undo) : come per la strategia undo-redo

Come nel caso precedente, ma con riferimento ad un programma che, per realizzare i 50.000 inserimenti, utilizzi complessivamente $k = 10$ transazioni, ognuna con 5000 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 T/k ennuple e quindi le relative scritture su log occupano $T/k \times L \times 3 = 300$KB e cioè $T/k \times L \times 3 * 1/D =$ ca.38 blocchi. In totale, per $k = 10$ transazioni, circa $T \times L \times 3 * 1/D =$ ca.380 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 $T/f = 125$, una per blocco
  - strategia undo-only (no-redo) : 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 $T/f = 125$
  - strategia redo-only (no-undo) : anche in questo caso, si può pensare che ogni blocco si scriva una volta sola, quindi ancora $T/f = 125$

Basi di dati II — 28 giugno 2016 — Compito A

Domanda 2 (35% per la prova breve e 12,5% per quella lunga)
Considerare il seguente scenario in cui tre 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. 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).

client 1	client 2	client 3
		begin read(x)
begin read(x)		
	begin read(x) x = x + 20 write(x) commit	
x = x + 10 write(x) commit		
		read(x) commit

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 **300**. 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) *legge 300*
begin read(x) *legge 300*		
	begin read(x) *legge 300* x = x + 20 write(x) *scrive 320* commit	
x = x + 10 write(x) *scrive 310* commit		
		read(x) *legge 300* commit

Si verificano anomalie?
Perdita di aggiornamento fra il client 1 e il client 2

Basi di dati II — 28 giugno 2016 — Compito A

Considerare nuovamente lo scenario della pagina precedente, ripetuto qui sotto per comodità.

client 1	client 2	client 3
		begin read(x)
begin read(x)		
	begin read(x) x = x + 20 write(x) commit	
x = x + 10 write(x) commit		
		read(x) commit

Considerare uno scheduler con controllo di concorrenza ancora basato su **Multversioni** (come in Postgres) ma con livello di isolamento **SERIALIZABLE** sulle prime due transazioni e **READ COMMITTED** sulla terza. Mostrare il comportamento dello scheduler, supponendo che il valore iniziale dell'oggetto x sia ancora **3**.

client 1	client 2	client 3
		begin read(x) *legge 300*
begin read(x) *legge 300*		
	begin read(x) *legge 300* x = x + 20 write(x) *scrive 320* commit	
x = x + 10 write(x) *scrive 310* abort		
begin read(x) *legge 320* x = x + 10 write(x) *scrive 330* commit		
		read(x) *legge 330* commit

Si verificano anomalie?
Lettura inconsistente per il client 3

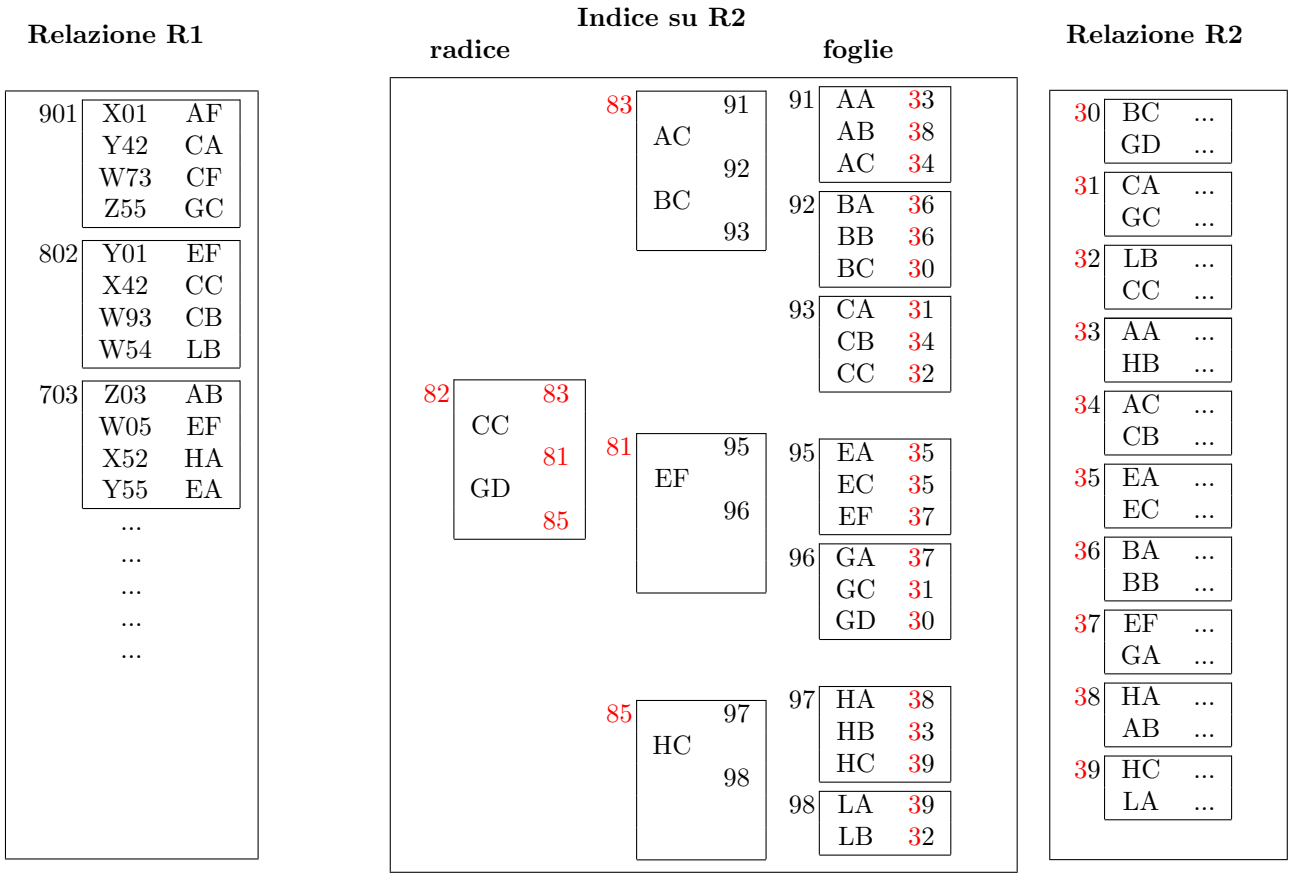
Basi di dati II — 28 giugno 2016 — Compito A

Domanda 3 (35% per la prova breve e 12,5% per quella lunga)
Considerare un sistema distribuito con tre nodi N_1, N_2 e N_3, che eseguono due transazioni T_A e T_B che coinvolgono i tre nodi, in modo diverso. Per la prima transazione N_1 è il coordinatore, mentre per la seconda il coordinatore è N_2. I due coordinatori inviano, come riportato nello schema sottostante, le richieste di `prepare`. Il nodo N_3 va in crash subito dopo aver risposto alla prima richiesta (senza avere il tempo di ricevere il messaggio di `commit`) e prima di ricevere la seconda. Indicare, nello schema sottostante, una possibile sequenza di scritture sui log e invio di messaggi (che includa anche i passi sopra illustrati), supponendo che entrambi i nodi siano ripristinati abbastanza presto (ma che vengano persi alcuni messaggi di risposta, ad esempio inviati a seguito di una decisione). Per i messaggi si usi la notazione *tipo(transaz)*→*destinatari* (come nell'esempio: `prepare`(T_A)→N_2,N_3). Supporre che nel log del coordinatore si scrivano solo i record di `prepare`, `commit` e `complete`, con i messaggi gestiti invece in memoria. Indicare ragionevoli istanti per i timeout, che permettano di concludere il protocollo per entrambe le transazioni.

Nodo N_1		Nodo N_2		Nodo N_3	
Log	Messaggi	Log	Messaggi	Log	Messaggi
prep(T_A,N_2,N_3)	prep(T_A)→N_2,N_3				
		ready(T_A)		ready(T_A)	
			ready(T_A)→N_1		ready(T_A)→N_1
				crash	
commit(T_A)					
	commit(T_A)→N_2,N_3				
		commit(T_A)			
			ack(T_A)→N_1		
		prep(T_B,N_1,N_3)			
			prep(T_B)→N_1,N_3		
ready(T_B)					
	ready(T_B)→N_2				
crash					
		abort(T_B)			
			abort(T_B)→N_1,N_3		
restart					
	commit(T_A)→N_2,N_3				
			ack(T_A)→N_1		
			abort(T_B)→N_1,N_3		
abort(T_B)					
	ack(T_B)→N_2				
				restart	
			abort(T_B)→N_3		
				abort(T_B)	
					ack(T_B)→N_2
		complete(T_B)			
	commit(T_A)→N_3				
				commit(T_A)	
					ack(T_A)→N_1
complete(T_A)					

Basi di dati II — 28 giugno 2016 — Compito A

Domanda 4 (15%) Considerare le relazioni R1 ed R2 e l'indice I2 su R2 schematizzati sotto. I riquadri interni indicano i blocchi e il numero a fianco a ciascun riquadro indica l'indirizzo del blocco. Nell'indice, i valori numerici sono riferimenti ai blocchi (blocchi dell'indice, per la radice e il livello intermedio, e blocchi di R2 per le foglie).



Supponendo di disporre di un buffer di **sei** pagine, considerare l'esecuzione del join di R1 ed R2, sulla base dei valori del secondo attributo di R1 e del primo di R2, con un **nested loop con accesso diretto** tramite l'indice di R2.

Indicare gli indirizzi dei blocchi su cui si eseguono operazioni di pin (o fix) per produrre le prime tre ennuple del risultato.

901, 82, 83, 92,
82, 83, 93, 31,
82, 81, 95,
82, 81, 96, 31,
802, 82, 81, 95, 37

Assumendo una politica di rimpiazzo *LRU*, indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato.

901, 82, 83, 92,
93, 31,
81, 95,
96,
802, 95, 37

In tal caso, indicare gli indirizzi dei blocchi che si può presumere si trovino nei buffer nel momento in cui si produce la terza ennupla.

901, 802, 82, 81, 95, 37

Indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato, con riferimento ad una politica di rimpiazzo *clock*.

901, 82, 83, 92,
93, 31,
81, 95,
83, 96,
802, 37

Basi di dati II — 28 giugno 2016 — Compito A

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 $L = 2$Byte, molto più piccola della dimensione del blocco pari a $B = 4000$ Byte. Si supponga che le relazioni abbiano entrambe $N = 2.000.000$ 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

Ripetere la valutazione effettuata alla pagina precedente nel caso in cui le due frequenze sono invece $f_1 = 1000$ e $f_2 = 10$

(i) memorizzazione separata delle due relazioni ...

cambia solo il costo complessivo:

$$c_1 \times f_1 + c_2 \times f_2 = 6 \times 1000 + 2000 \times 10 = ...$$

(ii) memorizzazione in un cluster delle due relazioni ...

cambia solo il costo complessivo:

$$c_1 \times f_1 + c_2 \times f_2 = 3 \times 1000 + 6000 \times 10 = ...$$

In conclusione, conviene quindi la memorizzazione in un cluster? (Sì o No) Il contrario del caso precedente

Basi di dati II — 28 giugno 2016 — Compito A

Domanda 6 (30%) Considerare uno schema dimensionale relativo a presenze e punti di giocatori di pallacanestro, che utilizzi, come tabella dei fatti e come una delle dimensioni, relazioni come quelle qui schematizzate:

KGiocatore	KSquadra	KTorneo	KStagione	Presenze	Punti
301	201	401	21	...	...
301	202	402	28	...	...
302	201	401	30	...	...
302	203	403	22	...	...
...	...	...	...	...	...

KTorneo	Nome	...
401	Serie A	...
402	Coppa dei Campioni	...
403	Coppa Italia	...
...	...	...

Supporre che si presentino le seguenti esigenze di modifica:

- I tornei possono cambiare nome nel tempo: per esempio, il torneo nella seconda ennupla potrebbe ad un certo punto cambiare nome da "Coppa dei Campioni" in "Eurolega"; interessano selezioni e aggregazioni relative alle presenze e ai punti tanto con riferimento al nome del torneo (distinguendo quindi la "Coppa dei Campioni" dalla "Eurolega") quanto alla sua identità ("Coppa dei Campioni" e "Eurolega" in questo senso avrebbero la stessa identità; allo scopo si usa un codice, ma gli analisti talvolta preferiscono usare il nome attuale del torneo, quindi nell'esempio "Eurolega" anche per le vecchie edizioni). Le modifiche sono rare, ma è possibile che ci siano tornei con vari cambiamenti di nome. Mostrare modifiche alle relazioni (una o entrambe) che permettano di soddisfare questa esigenza (mostrare i dati, con riferimento a quelli presenti negli esempi sopra, aggiungendo nuovi dati ragionevoli, che permettano di comprendente le modifiche).

La dimensione Torneo va gestita come una "slowly changing dimension", con un doppio accorgimento:

- va aggiunta una ennupla per ogni nuova versione di torneo
- vanno aggiunti un attributo con il nome attuale anche per la versione precedente e uno con un codice

KTorneo	Codice	NomeAlMomento	NomeAttuale	...
401	T1	Serie A	Seria A	...
402	T2	Coppa dei Campioni	Eurolega	...
403	T3	Coppa Italia	Coppa Italia	...
...	...	...	...	...
409	T2	Eurolega	Eurolega	...

- Per ogni giocatore, interessa rappresentare anche l'allenatore sotto la guida del quale ha giocato (per sommare ad esempio presenze e punti segnati con un certo allenatore); supporre (anche se l'ipotesi non è realistica) che ogni squadra abbia, in ciascuna stagione, un solo allenatore e che siano disponibili, nella staging area, tutti i dati relativi agli allenatori nelle varie stagioni. Anche in questo caso, mostrare le modifiche da apportare a una o a entrambe le relazioni.

Va aggiunta, per l'allenatore, una dimensione secondaria (o supplementare), cioè una dimensione la cui chiave, nella tabella dei fatti, dipende funzionalmente da altre. Il valore da inserire per ogni ennupla viene univocamente determinato dai valori delle chiavi di squadra e stagione.

KGiocatore	KSquadra	KTorneo	KStagione	KAllenatore	Presenze	Punti
301	201	401	21	...	...	...
301	202	402	28	...	...	...
302	201	401	30	...	...	...
302	203	403	22	...	...	...
...	...	...	...	...	...	...

(Seguendo la consuetudine, KAllenatore è sottolineato, anche se in effetti non sarebbe corretto).

Va anche aggiunta una tabella per la dimensione allenatore.

KAllenatore	Cognome	...	...	...
...	...	...	...	...

- Con riferimento al quesito precedente, considerare invece il caso, più realistico, in cui gli allenatori possono cambiare nel corso della stagione. Indicare quali modifiche sarebbero necessarie allo schema dimensionale e, soprattutto, quali dati sarebbero necessari nella staging area per poter riorganizzare il tutto.

In questo caso l'allenatore non è univocamente individuato dalle altre dimensioni e quindi la relativa dimensione non è secondaria. Lo schema rimane apparentemente lo stesso, con l'unica differenza che KAllenatore diventa parte effettiva della chiave della tabella dei fatti.
La vera differenza è nel contenuto: la grana è diversa, e ci possono essere più ennuple per un giocatore e una stagione. La tabella dei fatti va quindi ricalcolata e l'unica possibilità ragionevole per farlo è la disponibilità, nella staging area, dei dati di tutte le partite (qualunque altra ipotesi, ad esempio sui periodi è poco realistica).