Basi di dati II — Esame — 21 giugno 2018 — Compito A
Cenni sulle soluzioni

Tempo a disposizione: 1h15 per la prova breve e 2h45 per la prova completa.

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

Scrivere in modo ordinato e leggibile, negli spazi a disposizione.

Basi di dati II — 21 giugno 2018 — Compito A

Domanda 1 (15% per la prova completa, 40% per la prova breve)
Si consideri uno schema dimensionale con la seguente tabella dei fatti relativa alle carriere degli studenti presso una università:

- FattiEsami(KInsegnamento, KSessione, KCorsoDiStudio, NumeroEsami, Media)

Con riferimento a tale schema, si supponga che

- complessivamente la tabella contenga 100.000 ennuple
- vi siano 2000 insegnamenti e 10 sessioni e per ogni sessione e per ogni insegnamento ci sia almeno un esame verbalizzato
- vi siano 50 corsi di studio
- ogni insegnamento abbia studenti di mediamente 5 corsi di studio (e che in ogni sessione e per ogni insegnamento ci sia almeno uno studente per ciascuno di tali corsi di studio)
- le operazioni più frequenti siano quelle che producono:
  1. numero esami per un insegnamento e ciascuna sessione (ad esempio, Basi di dati II, in ciascuna delle 10 sessioni; quindi il risultato contiene 10 ennuple) con frequenza $f_1 = 10$
  2. numero esami complessivo per ciascun insegnamento (quindi il risultato contiene 2000 ennuple) con frequenza $f_2 = 100$
  3. numero esami in una sessione (ad esempio, la sessione estiva del 2017) per ciascun corso di studio (quindi vanno prodotte 50 ennuple, una per corso di studio) con frequenza $f_3 = 1$
- sia possibile realizzare una sola vista materializzata
- nelle selezioni, il costo sia pari al numero di ennuple estratte dalla relazione (prima dell'eventuale aggregazione)

Scegliere, fra le tre viste sostanzialmente corrispondenti alle tre interrogazioni, quella che si ritiene supporti meglio il carico applicativo. Indicare lo schema di ciascuna vista.

Indichiamo con $c = 50$ il numero di corsi di studio, $s = 10$ il numero di sessioni, $i = 2000$ il numero di insegnamenti e $N = 100.000$ il numero di ennuple nella tabella dei fatti.
Le tre viste:

1. EsamiInsegnSessione(KInsegnamento, KSessione, NumeroEsami, Media), cardinalità $i \times s = 2000 \times 10 = 20.000$, supporta l'interrogazione 1 e 2
2. EsamiInsegn(KInsegnamento, NumeroEsami,Media), cardinalità $i = 2000$, supporta l'interrogazione 2
3. EsamiSessioneCdS(KSessione, KCorsoDiStudio, NumeroEsami, Media), cardinalità $s \times c = 10 \times 50 = 500$, supporta l'interrogazione 3

Notare che le operazioni 1 e 3 chiedono un sottoinsieme delle ennuple della vista (rispettivamente, quelle relative ad un insegnamento e quelle relative ad una sessione)

Costo unitario per ogni operazione con ciascuna vista e costo complessivo:

	con vista 1	con vista 2	con vista 3
c_1	Si deve accedere alle ennuple della vista relative a un insegnamento, per tutte le sessioni; costo unitario: $s = 10$	Si deve usare la relazione base, accedendo a tutte le ennuple di un insegnamento: $N/i = 100.000/2000 = 50$	Si deve usare la relazione base, accedendo a tutte le ennuple di un insegnamento: $N/i = 100.000/2000 = 50$
c_2	Si deve accedere a tutte le ennuple della vista 1: $i \times s = 2000 \times 10 = 20.000$	Si deve accedere a tutte le ennuple della vista 2: $i = 2000$	Si deve accedere a tutte le ennuple della relazione base: $N = 100.000$
c_3	Si deve usare la relazione base, accedendo a tutte le ennuple di una sessione: $N/s = 100.000/10 = 10.000$	Si deve usare la relazione base, accedendo a tutte le ennuple di una sessione: $N/s = 100.000/10 = 10.000$	Si deve accedere alle ennuple della vista relative alla sessione, una per corso di studio: $c = 50$
$\Sigma_i\ c_i \times f_i$	$10 \times 10 + 20.000 \times 100 + 10.000 \times 1$	$50 \times 10 + 2000 \times 100 + 10.000 \times 1$	$50 \times 10 + 100.000 \times 100 + 50 \times 1$

Conviene quindi scegliere la vista 2

Basi di dati II — 21 giugno 2018 — Compito A

Domanda 2 (20% per la prova completa, 60% per la prova breve)
Si consideri la seguente porzione dello schema dell'archivio delle carriere degli studenti di una università:

- Studenti(Matricola, Cognome, Nome, DataNascita, CorsoDiStudio, AnnoDiCorso)
- Insegnamenti(Codice, Titolo, Crediti, SSD)
- Esami(Studente, Insegnamento, Data, Voto)
- CorsiDiStudio(CodiceCdS, Titolo, Livello, Classe, Dipartimento)
- Dipartimento(CodiceDip, NomeDip)
- Classe(CodiceClasse, NomeClasse)

Progettare uno schema dimensionale che permetta di rispondere, fra le altre, alle seguenti interrogazioni:

- calcolare il numero di studenti (con la relativa media dei voti) che hanno superato l'esame di un certo insegnamento in un certo anno accademico (si supponga che, per la data, l'unico dettaglio rilevante sia l'anno accademico e che esista un modo univoco per associare un anno accademico ad una data di esame)
- come nel caso precedente, ma anche distinto per corso di studio (oppure per dipartimento, oppure per livello, oppure per classe)
- come nel caso precedente, ma anche distinto per anno di iscrizione (ad esempio, primo, secondo, terzo, quarto; si supponga che l'anno sia un numero che può crescere indefinitamente)
- calcolare il numero di crediti complessivamente conseguiti in un anno accademico, in un corso di studio, divisi per SSD (come si vede dallo schema, SSD, che sta per "settore scientifico-disciplinare," è una proprietà di ciascun insegnamento; ad esempio, per Basi di dati II il settore scientifico-disciplinare è ING-INF/05)

Assumere che, per ragioni di privatezza e di compattezza, sia opportuno limitare la cardinalità della tabella dei fatti, utilizzando la minima cardinalità che permetta la risposta alle precedenti interrogazioni.
Mostrare lo schema dimensionale, specificando la grana scelta.

Grana: esami di un certo insegnamento, superati da studenti iscritti ad un certo anno di corso di un certo corso di studio, in un certo anno accademico.

Tabella dei fatti:
FattiEsami(KInsegnamento, KCdS, KAnnoAccademico, KAnnoDiCorso,NumStudenti, VotoMedio, Crediti)

Dimensioni:
Insegnamento(KInsegnamento, TitoloIns, SSD ...)
CorsoDiStudio(KCdS, CodiceCdS, TitoloCdS,Livello, CodiceClasse, NomeClasse, CodiceDip, NomeDip...)
AnnoAccademico(KAnnoAccademico, ...)
AnnoDiCorso(KAnnoDiCorso, ...)

AnnoDiCorso potrebbe essere degenere

Basi di dati II — 21 giugno 2018 — Compito A

Descrivere, informalmente, ma in modo il più possibile strutturato e comprensibile, il processo di ETL che porta alla tabella dei fatti mostrata in risposta alla domanda precedente

Ci si aspettava una risposta con un po' di dettagli, relativi ai seguenti aspetti

- join delle relazioni Esami, Insegnamenti, Studenti
- calcolo dell'anno accademico a partire dalla data
- raggruppamento per insegnamento, corso di studio, anno accademico e anno di corso, con conteggio delle ennuple, media dei voti e somma dei CFU
- sostituzione degli identificatori con le chiavi surrogate delle dimensioni

Basi di dati II — 21 giugno 2018 — Compito A

Domanda 3 (20%, solo per la prova completa)
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	d	11
102	d	2
103	d	4
104	a	11
105	k	3
106	d	1
107	e	3
108	k	4
109	a	5
110	a	2
111	a	2
112	k	4
113	a	1
114	k	2
115	e	3
116	d	3

Run su disco

104	a	11
101	d	11
102	d	2
103	d	4
106	d	1
105	k	3

109	a	5
110	a	2
111	a	2
107	e	3
108	k	4
112	k	4

113	a	1
116	d	3
115	e	3
114	k	2

Buffer

106	d	1	
105	k	3	←

108	k	4	←
112	k	4	

115	e	3	
114	k	2	←

Record prodotti dalle prime 3 `next()`

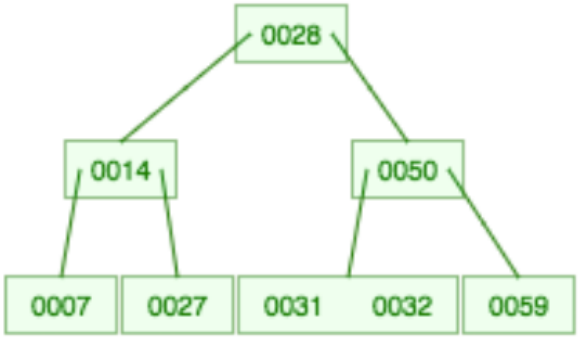
a	21
d	21
e	6

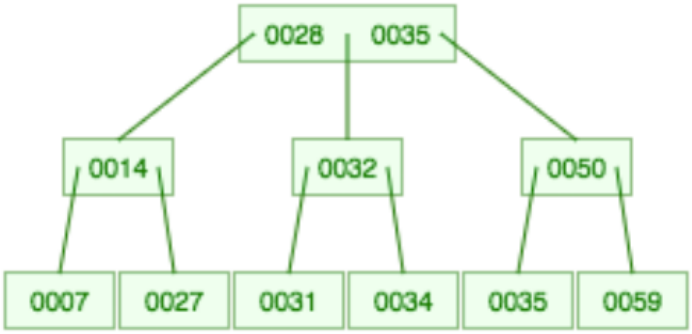
Il numero ideale di buffer (da utilizzare sia nella prima sia nella seconda passata) è pari alla radice quadrata del numero dei blocchi e quindi a tre.
Nella prima passata, si costruiscono quindi tre run ordinati, ciascuno a partire da tre blocchi del file originario. L'ordinamento di ciascun run può 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 — 21 giugno 2018 — Compito A

Domanda 4 (10%, solo per la prova completa)
Si consideri un B-tree con nodi intermedi che contengono due chiavi e tre puntatori e foglie con due chiavi, in cui vengano inserite chiavi (a partire dall'albero vuoto) nel seguente ordine: 50, 59, 7, 32, 14, 27, 28, 31, 34, 35, 36. Mostrare l'albero dopo l'inserimento di cinque chiavi, di otto chiavi e alla fine.

Vengono mostrate le soluzioni per il compito A. Le altre sono analoghe (isomorfe).





Basi di dati II — 21 giugno 2018 — Compito A

Domanda 5 (10%, solo per la prova completa)
Per ciascuno degli schedule sotto riportati, indicare, scrivendo **sì** o **no** nelle varie caselle, a quali classi appartiene: S (seriale, rispetto a letture e scritture, ignorare commit e abort), CSR (conflict-serializzabile), S2PL (cioè generabile da uno scheduler basato su 2PL stretto), MV (cioè generabile da uno scheduler multiversion con controllo di serializzabilità: "a serializable transaction cannot modify or lock rows changed by other transactions after the serializable transaction began"). Negli schedule, s_i indica l'inizio della transazione i e c_i il suo commit.

Soluzioni per il compito A

	S	CSR	S2PL	MV
$s_2, s_1, r_2(x), w_2(x), r_1(x), w_1(x), c_2, c_1$	sì	sì	no	no
$s_2, r_2(x), w_2(x), s_1, c_2, r_1(x), w_1(x), c_1$	sì	sì	sì	no
$s_1, s_2, r_1(x), r_2(x), w_1(x), c_1, w_2(x), c_2$	no	no	no	no
$s_1, s_2, r_1(x), r_2(x), w_2(x), r_2(y), w_2(y), c_2, r_1(y), c_1$	no	no	no	sì

Domanda 6 (10%, solo per prova completa)
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
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
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
Quale decisione può prendere il coordinatore?
Nessuna, perché il vecchio coordinatore è anche un partecipante e non sappiamo come ha deciso (potrebbe avere per qualche motivo abortito)

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
Quale decisione può prendere il coordinatore?
Poiché il caso non si può verificare, non ha senso parlare della decisione che viene presa

Basi di dati II — 21 giugno 2018 — Compito A

Domanda 7 (15% solo per prova completa)
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 — Esame — 21 giugno 2018 — Compito B
Cenni sulle soluzioni

Tempo a disposizione: 1h15 per la prova breve e 2h45 per la prova completa.

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

Scrivere in modo ordinato e leggibile, negli spazi a disposizione.

Basi di dati II — 21 giugno 2018 — Compito B

Domanda 1 (15% per la prova completa, 40% per la prova breve)
Si consideri uno schema dimensionale con la seguente tabella dei fatti relativa alle carriere degli studenti presso una università:

- FattiEsami(KInsegnamento, KSessione, KCorsoDiStudio, NumeroEsami, Media)

Con riferimento a tale schema, si supponga che

- complessivamente la tabella contenga 200.000 ennuple
- vi siano 2000 insegnamenti e 10 sessioni e per ogni sessione e per ogni insegnamento ci sia almeno un esame verbalizzato
- vi siano 100 corsi di studio
- ogni insegnamento abbia studenti di mediamente 10 corsi di studio (e che in ogni sessione e per ogni insegnamento ci sia almeno uno studente per ciascuno di tali corsi di studio)
- le operazioni più frequenti siano quelle che producono:
  1. numero esami per un insegnamento e ciascuna sessione (ad esempio, Basi di dati II, in ciascuna delle 10 sessioni; quindi il risultato contiene 10 ennuple) con frequenza $f_1 = 100$
  2. numero esami complessivo per ciascun insegnamento (quindi il risultato contiene 2000 ennuple) con frequenza $f_2 = 100$
  3. numero esami in una sessione (ad esempio, la sessione estiva del 2017) per ciascun corso di studio (quindi vanno prodotte 100 ennuple, una per corso di studio) con frequenza $f_3 = 1$
- sia possibile realizzare una sola vista materializzata
- nelle selezioni, il costo sia pari al numero di ennuple estratte dalla relazione (prima dell'eventuale aggregazione)

Scegliere, fra le tre viste sostanzialmente corrispondenti alle tre interrogazioni, quella che si ritiene supporti meglio il carico applicativo. Indicare lo schema di ciascuna vista.

Indichiamo con $c = 100$ il numero di corsi di studio, $s = 10$ il numero di sessioni, $i = 2000$ il numero di insegnamenti e $N = 200.000$ il numero di ennuple nella tabella dei fatti.
Le tre viste:

1. EsamiInsegnSessione(KInsegnamento, KSessione, NumeroEsami, Media), cardinalità $i \times s = 2000 \times 10 = 20.000$, supporta l'interrogazione 1 e 2
2. EsamiInsegn(KInsegnamento, NumeroEsami,Media), cardinalità $i = 2000$, supporta l'interrogazione 2
3. EsamiSessioneCdS(KSessione, KCorsoDiStudio, NumeroEsami, Media), cardinalità $s \times c = 10 \times 100 = 1000$, supporta l'interrogazione 3

Notare che le operazioni 1 e 3 chiedono un sottoinsieme delle ennuple della vista (rispettivamente, quelle relative ad un insegnamento e quelle relative ad una sessione)

Costo unitario per ogni operazione con ciascuna vista e costo complessivo:

	con vista 1	con vista 2	con vista 3
c_1	Si deve accedere alle ennuple della vista relative a un insegnamento, per tutte le sessioni; costo unitario: $s = 10$	Si deve usare la relazione base, accedendo a tutte le ennuple di un insegnamento: $N/i = 200.000/2000 = 100$	Si deve usare la relazione base, accedendo a tutte le ennuple di un insegnamento: $N/i = 200.000/2000 = 100$
c_2	Si deve accedere a tutte le ennuple della vista 1: $i \times s = 2000 \times 10 = 20.000$	Si deve accedere a tutte le ennuple della vista 2: $i = 2000$	Si deve accedere a tutte le ennuple della relazione base: $N = 200.000$
c_3	Si deve usare la relazione base, accedendo a tutte le ennuple di una sessione: $N/s = 200.000/10 = 20.000$	Si deve usare la relazione base, accedendo a tutte le ennuple di una sessione: $N/s = 200.000/10 = 20.000$	Si deve accedere alle ennuple della vista relative alla sessione, una per corso di studio: $c = 100$
$\Sigma_i\ c_i \times f_i$	$10 \times 100 + 20.000 \times 100 + 20.000 \times 1$	$100 \times 100 + 2000 \times 100 + 20.000 \times 1$	$100 \times 100 + 200.000 \times 100 + 100 \times 1$

Conviene quindi scegliere la vista 2

Basi di dati II — 21 giugno 2018 — Compito B

Domanda 2 (20% per la prova completa, 60% per la prova breve)
Si consideri la seguente porzione dello schema dell'archivio delle carriere degli studenti di una università:

- Studenti(Matricola, Cognome, Nome, DataNascita, CorsoDiStudio, AnnoDiCorso)
- Insegnamenti(Codice, Titolo, CFU, Settore)
- Esami(Studente, Insegnamento, Data, Voto)
- CorsiDiStudio(CodiceCdS, Titolo, Livello, Classe, Dipartimento)
- Dipartimento(CodiceDip, NomeDip)
- Classe(CodiceClasse, NomeClasse)

Progettare uno schema dimensionale che permetta di rispondere, fra le altre, alle seguenti interrogazioni:

- calcolare il numero di studenti (con la relativa media dei voti) che hanno superato l'esame di un certo insegnamento in un certo anno accademico (si supponga che, per la data, l'unico dettaglio rilevante sia l'anno accademico e che esista un modo univoco per associare un anno accademico ad una data di esame)
- come nel caso precedente, ma anche distinto per corso di studio (oppure per dipartimento, oppure per livello, oppure per classe)
- come nel caso precedente, ma anche distinto per anno di iscrizione (ad esempio, primo, secondo, terzo, quarto; si supponga che l'anno sia un numero che può crescere indefinitamente)
- calcolare il numero di CFU complessivamente conseguiti in un anno accademico, in un corso di studio, divisi per settore (come si vede dallo schema, Settore è una proprietà di ciascun insegnamento; ad esempio, per Basi di dati II il settore è ING-INF/05)

Assumere che, per ragioni di privatezza e di compattezza, sia opportuno limitare la cardinalità della tabella dei fatti, utilizzando la minima cardinalità che permetta la risposta alle precedenti interrogazioni.
Mostrare lo schema dimensionale, specificando la grana scelta.

Grana: esami di un certo insegnamento, superati da studenti iscritti ad un certo anno di corso di un certo corso di studio, in un certo anno accademico.

Tabella dei fatti:
FattiEsami(KInsegnamento, KCdS, KAnnoAccademico, KAnnoDiCorso,NumStudenti, VotoMedio, CFU)

Dimensioni:
Insegnamento(KInsegnamento, TitoloIns, Settore ...)
CorsoDiStudio(KCdS, CodiceCdS, TitoloCdS,Livello, CodiceClasse, NomeClasse, CodiceDip, NomeDip...)
AnnoAccademico(KAnnoAccademico, ...)
AnnoDiCorso(KAnnoDiCorso, ...)

AnnoDiCorso potrebbe essere degenere

Basi di dati II — 21 giugno 2018 — Compito B

Descrivere, informalmente, ma in modo il più possibile strutturato e comprensibile, il processo di ETL che porta alla tabella dei fatti mostrata in risposta alla domanda precedente

Ci si aspettava una risposta con un po' di dettagli, relativi ai seguenti aspetti

- join delle relazioni Esami, Insegnamenti, Studenti
- calcolo dell'anno accademico a partire dalla data
- raggruppamento per insegnamento, corso di studio, anno accademico e anno di corso, con conteggio delle ennuple, media dei voti e somma dei CFU
- sostituzione degli identificatori con le chiavi surrogate delle dimensioni

Basi di dati II — 21 giugno 2018 — Compito B

Domanda 3 (20%, solo per la prova completa)
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	d	12
102	d	2
103	d	4
104	a	12
105	k	3
106	d	1
107	e	3
108	k	4
109	a	5
110	a	2
111	a	2
112	k	4
113	a	1
114	k	2
115	e	3
116	d	3

Run su disco

104	a	12
101	d	12
102	d	2
103	d	4
106	d	1
105	k	3

109	a	5
110	a	2
111	a	2
107	e	3
108	k	4
112	k	4

113	a	1
116	d	3
115	e	3
114	k	2

Buffer

106	d	1	
105	k	3	←

108	k	4	←
112	k	4	

115	e	3	
114	k	2	←

Record prodotti dalle prime 3 `next()`

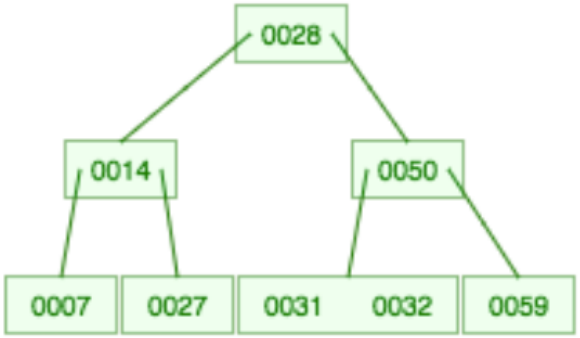
a	22
d	22
e	6

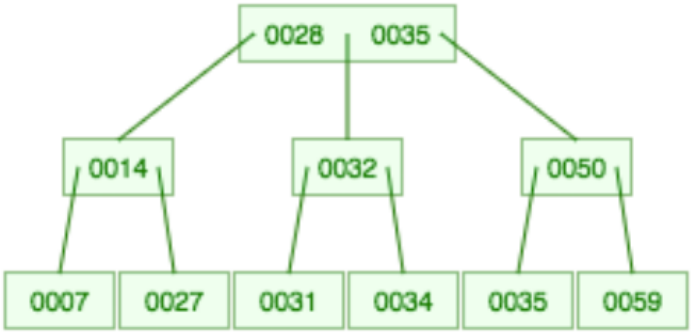
Il numero ideale di buffer (da utilizzare sia nella prima sia nella seconda passata) è pari alla radice quadrata del numero dei blocchi e quindi a tre.
Nella prima passata, si costruiscono quindi tre run ordinati, ciascuno a partire da tre blocchi del file originario. L'ordinamento di ciascun run può 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 — 21 giugno 2018 — Compito B

Domanda 4 (10%, solo per la prova completa)
Si consideri un B-tree con nodi intermedi che contengono due chiavi e tre puntatori e foglie con due chiavi, in cui vengano inserite chiavi (a partire dall'albero vuoto) nel seguente ordine: 38, 51, 8, 32, 21, 25, 28, 31, 34, 35, 36. Mostrare l'albero dopo l'inserimento di cinque chiavi, di otto chiavi e alla fine.

Vengono mostrate le soluzioni per il compito A. Le altre sono analoghe (isomorfe).





Basi di dati II — 21 giugno 2018 — Compito B

Domanda 5 (10%, solo per la prova completa)
Per ciascuno degli schedule sotto riportati, indicare, scrivendo **sì** o **no** nelle varie caselle, a quali classi appartiene: S (seriale, rispetto a letture e scritture, ignorare commit e abort), CSR (conflict-serializzabile), S2PL (cioè generabile da uno scheduler basato su 2PL stretto), MV (cioè generabile da uno scheduler multiversion con controllo di serializzabilità: "a serializable transaction cannot modify or lock rows changed by other transactions after the serializable transaction began"). Negli schedule, s_i indica l'inizio della transazione i e c_i il suo commit.

Soluzioni per il compito A

	S	CSR	S2PL	MV
$s_2, r_2(x), w_2(x), s_1, c_2, r_1(x), w_1(x), c_1$	sì	sì	no	no
$s_1, s_2, r_1(x), r_2(x), w_1(x), c_1, w_2(x), c_2$	sì	sì	sì	no
$s_2, s_1, r_2(x), w_2(x), r_1(x), w_1(x), c_2, c_1$	no	no	no	no
$s_1, s_2, r_1(x), r_2(x), w_2(x), r_2(y), w_2(y), c_2, r_1(y), c_1$	no	no	no	sì

Domanda 6 (10%, solo per prova completa)
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)? sì
Quale decisione può prendere il coordinatore?
Nessuna, perché il vecchio coordinatore è anche un partecipante e non sappiamo come ha deciso (potrebbe avere per qualche motivo abortito)

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)? no
Quale decisione può prendere il coordinatore?
Poiché il caso non si può verificare, non ha senso parlare della decisione che viene presa

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)? sì
Quale decisione può prendere il coordinatore?
Abort (oppure attendere, ma non è opportuno)

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)? sì
Quale decisione può prendere il coordinatore?
Commit

Basi di dati II — 21 giugno 2018 — Compito B

Domanda 7 (15% solo per prova completa)
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(T1)→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				
		ready(T_1)		ready(T_1)	
			ready(T_1)→A		ready(T_1)→A
				crash	
commit(T_1)					
	commit(T_1)→B,C				
		commit(T_1)			
			ack(T_1)→A		
		prep(T_2,A,C)			
			prep(T_2)→A,C		
ready(T_2)					
	ready(T_2)→B				
	crash				
		abort(T_2)			
			abort(T_2)→A,C		
	restart				
	commit(T_1)→B,C				
			ack(T_1)→A		
			abort(T_2)→A,C		
abort(T_2)					
	ack(T_2)→B				
				restart	
			abort(T_2)→C		
				abort(T_2)	
					ack(T_2)→B
		complete(T_2)			
	commit(T_1)→C				
				commit(T_1)	
					ack(T_1)→A
complete(T_1)					

Basi di dati II — Esame — 21 giugno 2018 — Compito C
Cenni sulle soluzioni

Tempo a disposizione: 1h15 per la prova breve e 2h45 per la prova completa.

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

Scrivere in modo ordinato e leggibile, negli spazi a disposizione.

Basi di dati II — 21 giugno 2018 — Compito C

Domanda 1 (15% per la prova completa, 40% per la prova breve)
Si consideri uno schema dimensionale con la seguente tabella dei fatti relativa alle carriere degli studenti presso una università:

- FattiEsami(KInsegnamento, KSessione, KCorsoDiStudio, NumeroEsami, Media)

Con riferimento a tale schema, si supponga che

- complessivamente la tabella contenga 100.000 ennuple
- vi siano 1000 insegnamenti e 10 sessioni e per ogni sessione e per ogni insegnamento ci sia almeno un esame verbalizzato
- vi siano 100 corsi di studio
- ogni insegnamento abbia studenti di mediamente 10 corsi di studio (e che in ogni sessione e per ogni insegnamento ci sia almeno uno studente per ciascuno di tali corsi di studio)
- le operazioni più frequenti siano quelle che producono:
  1. numero esami per un insegnamento e ciascuna sessione (ad esempio, Basi di dati II, in ciascuna delle 10 sessioni; quindi il risultato contiene 10 ennuple) con frequenza $f_1 = 100$
  2. numero esami complessivo per ciascun insegnamento (quindi il risultato contiene 1000 ennuple) con frequenza $f_2 = 1$
  3. numero esami in una sessione (ad esempio, la sessione estiva del 2017) per ciascun corso di studio (quindi vanno prodotte 100 ennuple, una per corso di studio) con frequenza $f_3 = 100$
- sia possibile realizzare una sola vista materializzata
- nelle selezioni, il costo sia pari al numero di ennuple estratte dalla relazione (prima dell'eventuale aggregazione)

Scegliere, fra le tre viste sostanzialmente corrispondenti alle tre interrogazioni, quella che si ritiene supporti meglio il carico applicativo. Indicare lo schema di ciascuna vista.

Indichiamo con $c = 100$ il numero di corsi di studio, $s = 10$ il numero di sessioni, $i = 1000$ il numero di insegnamenti e $N = 100.000$ il numero di ennuple nella tabella dei fatti.
Le tre viste:

1. EsamiInsegnSessione(KInsegnamento, KSessione, NumeroEsami, Media), cardinalità $i \times s = 1000 \times 10 = 10.000$, supporta l'interrogazione 1 e 2
2. EsamiInsegn(KInsegnamento, NumeroEsami,Media), cardinalità $i = 1000$, supporta l'interrogazione 2
3. EsamiSessioneCdS(KSessione, KCorsoDiStudio, NumeroEsami, Media), cardinalità $s \times c = 10 \times 100 = 1000$, supporta l'interrogazione 3

Notare che le operazioni 1 e 3 chiedono un sottoinsieme delle ennuple della vista (rispettivamente, quelle relative ad un insegnamento e quelle relative ad una sessione)

Costo unitario per ogni operazione con ciascuna vista e costo complessivo:

	con vista 1	con vista 2	con vista 3
c_1	Si deve accedere alle ennuple della vista relative a un insegnamento, per tutte le sessioni; costo unitario: $s = 10$	Si deve usare la relazione base, accedendo a tutte le ennuple di un insegnamento: $N/i = 100.000/1000 = 100$	Si deve usare la relazione base, accedendo a tutte le ennuple di un insegnamento: $N/i = 100.000/1000 = 100$
c_2	Si deve accedere a tutte le ennuple della vista 1: $i \times s = 1000 \times 10 = 10.000$	Si deve accedere a tutte le ennuple della vista 2: $i = 1000$	Si deve accedere a tutte le ennuple della relazione base: $N = 100.000$
c_3	Si deve usare la relazione base, accedendo a tutte le ennuple di una sessione: $N/s = 100.000/10 = 10.000$	Si deve usare la relazione base, accedendo a tutte le ennuple di una sessione: $N/s = 100.000/10 = 10.000$	Si deve accedere alle ennuple della vista relative alla sessione, una per corso di studio: $c = 100$
$\Sigma_i\ c_i \times f_i$	$10 \times 100 + 10.000 \times 1 + 10.000 \times 100$	$100 \times 100 + 1000 \times 1 + 10.000 \times 100$	$100 \times 100 + 100.000 \times 1 + 100 \times 100$

Conviene quindi scegliere la vista 3

Basi di dati II — 21 giugno 2018 — Compito C

Domanda 2 (20% per la prova completa, 60% per la prova breve)
Si consideri la seguente porzione dello schema dell'archivio delle carriere degli studenti di una università:

- Studenti(Matricola, Cognome, Nome, DataNascita, CorsoDiStudio, AnnoDiCorso)
- Insegnamenti(Codice, Titolo, Crediti, SSD)
- Esami(Studente, Insegnamento, Data, Voto)
- CorsiDiStudio(CodiceCdS, Titolo, Livello, Classe, Dipartimento)
- Dipartimento(CodiceDip, NomeDip)
- Classe(CodiceClasse, NomeClasse)

Progettare uno schema dimensionale che permetta di rispondere, fra le altre, alle seguenti interrogazioni:

- calcolare il numero di studenti (con la relativa media dei voti) che hanno superato l'esame di un certo insegnamento in un certo anno accademico (si supponga che, per la data, l'unico dettaglio rilevante sia l'anno accademico e che esista un modo univoco per associare un anno accademico ad una data di esame)
- come nel caso precedente, ma anche distinto per corso di studio (oppure per dipartimento, oppure per livello, oppure per classe)
- come nel caso precedente, ma anche distinto per anno di iscrizione (ad esempio, primo, secondo, terzo, quarto; si supponga che l'anno sia un numero che può crescere indefinitamente)
- calcolare il numero di crediti complessivamente conseguiti in un anno accademico, in un corso di studio, divisi per SSD (come si vede dallo schema, SSD, che sta per "settore scientifico-disciplinare," è una proprietà di ciascun insegnamento; ad esempio, per Basi di dati II il settore scientifico-disciplinare è ING-INF/05)

Assumere che, per ragioni di privatezza e di compattezza, sia opportuno limitare la cardinalità della tabella dei fatti, utilizzando la minima cardinalità che permetta la risposta alle precedenti interrogazioni.
Mostrare lo schema dimensionale, specificando la grana scelta.

Grana: esami di un certo insegnamento, superati da studenti iscritti ad un certo anno di corso di un certo corso di studio, in un certo anno accademico.

Tabella dei fatti:
FattiEsami(KInsegnamento, KCdS, KAnnoAccademico, KAnnoDiCorso,NumStudenti, VotoMedio, Crediti)

Dimensioni:
Insegnamento(KInsegnamento, TitoloIns, SSD ...)
CorsoDiStudio(KCdS, CodiceCdS, TitoloCdS,Livello, CodiceClasse, NomeClasse, CodiceDip, NomeDip...)
AnnoAccademico(KAnnoAccademico, ...)
AnnoDiCorso(KAnnoDiCorso, ...)

AnnoDiCorso potrebbe essere degenere

Basi di dati II — 21 giugno 2018 — Compito C

Descrivere, informalmente, ma in modo il più possibile strutturato e comprensibile, il processo di ETL che porta alla tabella dei fatti mostrata in risposta alla domanda precedente

Ci si aspettava una risposta con un po' di dettagli, relativi ai seguenti aspetti

- join delle relazioni Esami, Insegnamenti, Studenti
- calcolo dell'anno accademico a partire dalla data
- raggruppamento per insegnamento, corso di studio, anno accademico e anno di corso, con conteggio delle ennuple, media dei voti e somma dei CFU
- sostituzione degli identificatori con le chiavi surrogate delle dimensioni

Basi di dati II — 21 giugno 2018 — Compito C

Domanda 3 (20%, solo per la prova completa)
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	d	13
102	d	2
103	d	4
104	a	13
105	k	3
106	d	1
107	e	3
108	k	4
109	a	5
110	a	2
111	a	2
112	k	4
113	a	1
114	k	2
115	e	3
116	d	3

Run su disco

104	a	13
101	d	13
102	d	2
103	d	4
106	d	1
105	k	3

109	a	5
110	a	2
111	a	2
107	e	3
108	k	4
112	k	4

113	a	1
116	d	3
115	e	3
114	k	2

Buffer

106	d	1	←
105	k	3	

108	k	4	←
112	k	4	

115	e	3	←
114	k	2	

Record prodotti dalle prime 3 `next()`

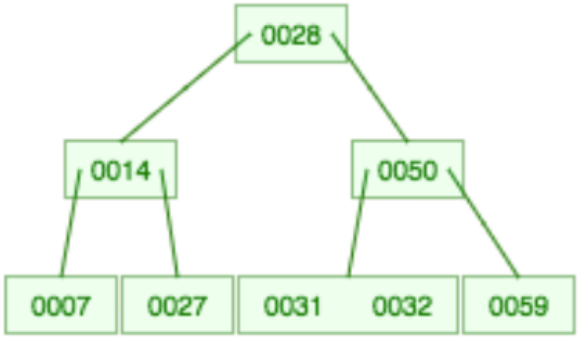
a	23
d	23
e	6

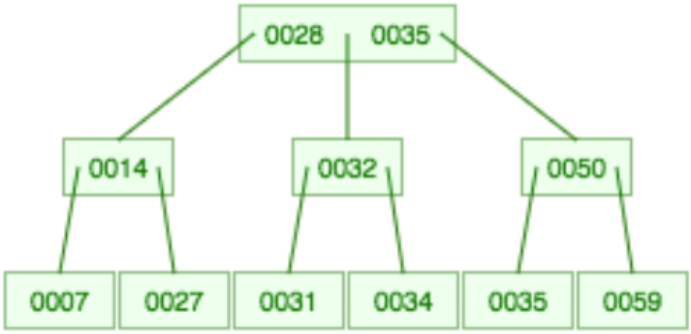
Il numero ideale di buffer (da utilizzare sia nella prima sia nella seconda passata) è pari alla radice quadrata del numero dei blocchi e quindi a tre.
Nella prima passata, si costruiscono quindi tre run ordinati, ciascuno a partire da tre blocchi del file originario. L'ordinamento di ciascun run può 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 — 21 giugno 2018 — Compito C

Domanda 4 (10%, solo per la prova completa)
Si consideri un B-tree con nodi intermedi che contengono due chiavi e tre puntatori e foglie con due chiavi, in cui vengano inserite chiavi (a partire dall'albero vuoto) nel seguente ordine: 42, 55, 14, 32, 15, 23, 28, 31, 34, 35, 36. Mostrare l'albero dopo l'inserimento di cinque chiavi, di otto chiavi e alla fine.

Vengono mostrate le soluzioni per il compito A. Le altre sono analoghe (isomorfe).





Basi di dati II — 21 giugno 2018 — Compito C

Domanda 5 (10%, solo per la prova completa)
Per ciascuno degli schedule sotto riportati, indicare, scrivendo **sì** o **no** nelle varie caselle, a quali classi appartiene: S (seriale, rispetto a letture e scritture, ignorare commit e abort), CSR (conflict-serializzabile), S2PL (cioè generabile da uno scheduler basato su 2PL stretto), MV (cioè generabile da uno scheduler multiversion con controllo di serializzabilità: "a serializable transaction cannot modify or lock rows changed by other transactions after the serializable transaction began"). Negli schedule, s_i indica l'inizio della transazione i e c_i il suo commit.

Soluzioni per il compito A

	S	CSR	S2PL	MV
$s_2, s_1, r_2(x), w_2(x), r_1(x), w_1(x), c_2, c_1$	sì	sì	no	no
$s_2, r_2(x), w_2(x), s_1, c_2, r_1(x), w_1(x), c_1$	sì	sì	sì	no
$s_1, s_2, r_1(x), r_2(x), w_2(x), r_2(y), w_2(y), c_2, r_1(y), c_1$	no	no	no	no
$s_1, s_2, r_1(x), r_2(x), w_1(x), c_1, w_2(x), c_2$	no	no	no	sì

Domanda 6 (10%, solo per prova completa)
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)? no
Quale decisione può prendere il coordinatore?
Poiché il caso non si può verificare, non ha senso parlare della decisione che viene presa

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

Si può verificare (sì o no)? sì
Quale decisione può prendere il coordinatore?
Nessuna, perché il vecchio coordinatore è anche un partecipante e non sappiamo come ha deciso (potrebbe avere per qualche motivo abortito)

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)? sì
Quale decisione può prendere il coordinatore?
Abort (oppure attendere, ma non è opportuno)

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

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

Basi di dati II — 21 giugno 2018 — Compito C

Domanda 7 (15% solo per prova completa)
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				
		`ready(`T_x`)`		`ready(`T_x`)`	
			`ready(`T_x`)→`N_1		`ready(`T_x`)→`N_1
				crash	
`commit(`T_x`)`					
	`commit(`T_x`)→`N_2`,`N_3				
		`commit(`T_x`)`			
			`ack(`T_x`)→`N_1		
		`prep(`T_y`,`N_1`,`N_3`)`			
			`prep(`T_y`)→`N_1`,`N_3		
`ready(`T_y`)`					
	`ready(`T_y`)→`N_2				
crash					
		`abort(`T_y`)`			
			`abort(`T_y`)→`N_1`,`N_3		
restart					
	`commit(`T_x`)→`N_2`,`N_3				
			`ack(`T_x`)→`N_1		
			`abort(`T_y`)→`N_1`,`N_3		
`abort(`T_y`)`					
	`ack(`T_y`)→`N_2				
				restart	
			`abort(`T_y`)→`N_3		
				`abort(`T_y`)`	
					`ack(`T_y`)→`N_2
		`complete(`T_y`)`			
	`commit(`T_x`)→`N_3				
				`commit(`T_x`)`	
					`ack(`T_x`)→`N_1
`complete(`T_x`)`					

Basi di dati II — Esame — 21 giugno 2018 — Compito D
Cenni sulle soluzioni

Tempo a disposizione: 1h15 per la prova breve e 2h45 per la prova completa.

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

Scrivere in modo ordinato e leggibile, negli spazi a disposizione.

Basi di dati II — 21 giugno 2018 — Compito D

Domanda 1 (15% per la prova completa, 40% per la prova breve)
Si consideri uno schema dimensionale con la seguente tabella dei fatti relativa alle carriere degli studenti presso una università:

- FattiEsami(KInsegnamento, KSessione, KCorsoDiStudio, NumeroEsami, Media)

Con riferimento a tale schema, si supponga che

- complessivamente la tabella contenga 50.000 ennuple
- vi siano 1000 insegnamenti e 10 sessioni e per ogni sessione e per ogni insegnamento ci sia almeno un esame verbalizzato
- vi siano 50 corsi di studio
- ogni insegnamento abbia studenti di mediamente 5 corsi di studio (e che in ogni sessione e per ogni insegnamento ci sia almeno uno studente per ciascuno di tali corsi di studio)
- le operazioni più frequenti siano quelle che producono:
  1. numero esami per un insegnamento e ciascuna sessione (ad esempio, Basi di dati II, in ciascuna delle 10 sessioni; quindi il risultato contiene 10 ennuple) con frequenza $f_1 = 10$
  2. numero esami complessivo per ciascun insegnamento (quindi il risultato contiene 1000 ennuple) con frequenza $f_2 = 1000$
  3. numero esami in una sessione (ad esempio, la sessione estiva del 2017) per ciascun corso di studio (quindi vanno prodotte 50 ennuple, una per corso di studio) con frequenza $f_3 = 10$
- sia possibile realizzare una sola vista materializzata
- nelle selezioni, il costo sia pari al numero di ennuple estratte dalla relazione (prima dell'eventuale aggregazione)

Scegliere, fra le tre viste sostanzialmente corrispondenti alle tre interrogazioni, quella che si ritiene supporti meglio il carico applicativo. Indicare lo schema di ciascuna vista.

Indichiamo con $c = 50$ il numero di corsi di studio, $s = 10$ il numero di sessioni, $i = 1000$ il numero di insegnamenti e $N = 50.000$ il numero di ennuple nella tabella dei fatti.
Le tre viste:

1. EsamiInsegnSessione(KInsegnamento, KSessione, NumeroEsami, Media), cardinalità $i \times s = 1000 \times 10 = 10.000$, supporta l'interrogazione 1 e 2
2. EsamiInsegn(KInsegnamento, NumeroEsami,Media), cardinalità $i = 1000$, supporta l'interrogazione 2
3. EsamiSessioneCdS(KSessione, KCorsoDiStudio, NumeroEsami, Media), cardinalità $s \times c = 10 \times 50 = 500$, supporta l'interrogazione 3

Notare che le operazioni 1 e 3 chiedono un sottoinsieme delle ennuple della vista (rispettivamente, quelle relative ad un insegnamento e quelle relative ad una sessione)

Costo unitario per ogni operazione con ciascuna vista e costo complessivo:

	con vista 1	con vista 2	con vista 3
c_1	Si deve accedere alle ennuple della vista relative a un insegnamento, per tutte le sessioni; costo unitario: $s = 10$	Si deve usare la relazione base, accedendo a tutte le ennuple di un insegnamento: $N/i = 50.000/1000 = 50$	Si deve usare la relazione base, accedendo a tutte le ennuple di un insegnamento: $N/i = 50.000/1000 = 50$
c_2	Si deve accedere a tutte le ennuple della vista 1: $i \times s = 1000 \times 10 = 10.000$	Si deve accedere a tutte le ennuple della vista 2: $i = 1000$	Si deve accedere a tutte le ennuple della relazione base: $N = 50.000$
c_3	Si deve usare la relazione base, accedendo a tutte le ennuple di una sessione: $N/s = 50.000/10 = 5.000$	Si deve usare la relazione base, accedendo a tutte le ennuple di una sessione: $N/s = 50.000/10 = 5.000$	Si deve accedere alle ennuple della vista relative alla sessione, una per corso di studio: $c = 50$
$\Sigma_i\ c_i \times f_i$	$10 \times 10 + 10.000 \times 1000 + 5.000 \times 10$	$50 \times 10 + 1000 \times 1000 + 5.000 \times 10$	$50 \times 10 + 50.000 \times 1000 + 50 \times 10$

Conviene quindi scegliere la vista 2

Basi di dati II — 21 giugno 2018 — Compito D

Domanda 2 (20% per la prova completa, 60% per la prova breve)
Si consideri la seguente porzione dello schema dell'archivio delle carriere degli studenti di una università:

- Studenti(Matricola, Cognome, Nome, DataNascita, CorsoDiStudio, AnnoDiCorso)
- Insegnamenti(Codice, Titolo, CFU, Settore)
- Esami(Studente, Insegnamento, Data, Voto)
- CorsiDiStudio(CodiceCdS, Titolo, Livello, Classe, Dipartimento)
- Dipartimento(CodiceDip, NomeDip)
- Classe(CodiceClasse, NomeClasse)

Progettare uno schema dimensionale che permetta di rispondere, fra le altre, alle seguenti interrogazioni:

- calcolare il numero di studenti (con la relativa media dei voti) che hanno superato l'esame di un certo insegnamento in un certo anno accademico (si supponga che, per la data, l'unico dettaglio rilevante sia l'anno accademico e che esista un modo univoco per associare un anno accademico ad una data di esame)
- come nel caso precedente, ma anche distinto per corso di studio (oppure per dipartimento, oppure per livello, oppure per classe)
- come nel caso precedente, ma anche distinto per anno di iscrizione (ad esempio, primo, secondo, terzo, quarto; si supponga che l'anno sia un numero che può crescere indefinitamente)
- calcolare il numero di CFU complessivamente conseguiti in un anno accademico, in un corso di studio, divisi per settore (come si vede dallo schema, Settore è una proprietà di ciascun insegnamento; ad esempio, per Basi di dati II il settore è ING-INF/05)

Assumere che, per ragioni di privatezza e di compattezza, sia opportuno limitare la cardinalità della tabella dei fatti, utilizzando la minima cardinalità che permetta la risposta alle precedenti interrogazioni.
Mostrare lo schema dimensionale, specificando la grana scelta.

Grana: esami di un certo insegnamento, superati da studenti iscritti ad un certo anno di corso di un certo corso di studio, in un certo anno accademico.

Tabella dei fatti:
FattiEsami(KInsegnamento, KCdS, KAnnoAccademico, KAnnoDiCorso,NumStudenti, VotoMedio, CFU)

Dimensioni:
Insegnamento(KInsegnamento, TitoloIns, Settore ...)
CorsoDiStudio(KCdS, CodiceCdS, TitoloCdS,Livello, CodiceClasse, NomeClasse, CodiceDip, NomeDip...)
AnnoAccademico(KAnnoAccademico, ...)
AnnoDiCorso(KAnnoDiCorso, ...)

AnnoDiCorso potrebbe essere degenere

Basi di dati II — 21 giugno 2018 — Compito D

Descrivere, informalmente, ma in modo il più possibile strutturato e comprensibile, il processo di ETL che porta alla tabella dei fatti mostrata in risposta alla domanda precedente

Ci si aspettava una risposta con un po' di dettagli, relativi ai seguenti aspetti

- join delle relazioni Esami, Insegnamenti, Studenti
- calcolo dell'anno accademico a partire dalla data
- raggruppamento per insegnamento, corso di studio, anno accademico e anno di corso, con conteggio delle ennuple, media dei voti e somma dei CFU
- sostituzione degli identificatori con le chiavi surrogate delle dimensioni

Basi di dati II — 21 giugno 2018 — Compito D

Domanda 3 (20%, solo per la prova completa)
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	d	14
102	d	2
103	d	4
104	a	14
105	k	3
106	d	1
107	e	3
108	k	4
109	a	5
110	a	2
111	a	2
112	k	4
113	a	1
114	k	2
115	e	3
116	d	3

Run su disco

104	a	14
101	d	14
102	d	2
103	d	4
106	d	1
105	k	3

109	a	5
110	a	2
111	a	2
107	e	3
108	k	4
112	k	4

113	a	1
116	d	3
115	e	3
114	k	2

Buffer

106	d	1	←
105	k	3	

108	k	4	←
112	k	4	

115	e	3	←
114	k	2	

Record prodotti dalle prime 3 `next()`

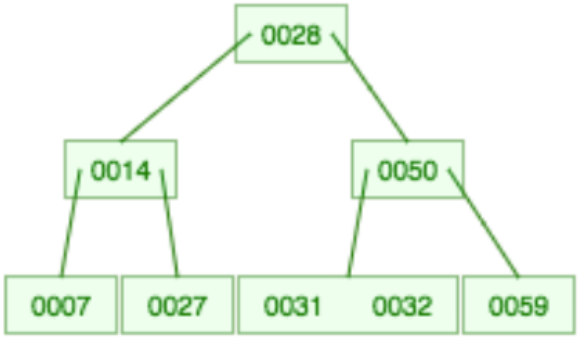
a	24
d	24
e	6

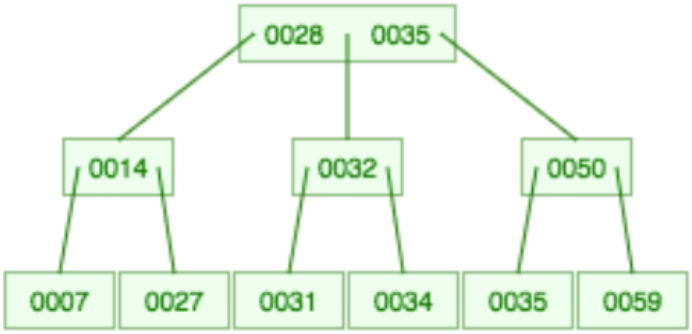
Il numero ideale di buffer (da utilizzare sia nella prima sia nella seconda passata) è pari alla radice quadrata del numero dei blocchi e quindi a tre.
Nella prima passata, si costruiscono quindi tre run ordinati, ciascuno a partire da tre blocchi del file originario. L'ordinamento di ciascun run può 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 — 21 giugno 2018 — Compito D

Domanda 4 (10%, solo per la prova completa)
Si consideri un B-tree con nodi intermedi che contengono due chiavi e tre puntatori e foglie con due chiavi, in cui vengano inserite chiavi (a partire dall'albero vuoto) nel seguente ordine: 39, 50, 12, 32, 20, 22, 28, 31, 34, 35, 36. Mostrare l'albero dopo l'inserimento di cinque chiavi, di otto chiavi e alla fine.

Vengono mostrate le soluzioni per il compito A. Le altre sono analoghe (isomorfe).





Basi di dati II — 21 giugno 2018 — Compito D

Domanda 5 (10%, solo per la prova completa)
Per ciascuno degli schedule sotto riportati, indicare, scrivendo **sì** o **no** nelle varie caselle, a quali classi appartiene: S (seriale, rispetto a letture e scritture, ignorare commit e abort), CSR (conflict-serializzabile), S2PL (cioè generabile da uno scheduler basato su 2PL stretto), MV (cioè generabile da uno scheduler multiversion con controllo di serializzabilità: "a serializable transaction cannot modify or lock rows changed by other transactions after the serializable transaction began"). Negli schedule, s_i indica l'inizio della transazione i e c_i il suo commit.

Soluzioni per il compito A

	S	CSR	S2PL	MV
$s_1, s_2, r_1(x), r_2(x), w_2(x), r_2(y), w_2(y), c_2, r_1(y), c_1$	sì	sì	no	no
$s_1, s_2, r_1(x), r_2(x), w_1(x), c_1, w_2(x), c_2$	sì	sì	sì	no
$s_2, s_1, r_2(x), w_2(x), r_1(x), w_1(x), c_2, c_1$	no	no	no	no
$s_2, r_2(x), w_2(x), s_1, c_2, r_1(x), w_1(x), c_1$	no	no	no	sì

Domanda 6 (10%, solo per prova completa)
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)? no
Quale decisione può prendere il coordinatore?
Poiché il caso non si può verificare, non ha senso parlare della decisione che viene presa

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)? sì
Quale decisione può prendere il coordinatore?
Abort (oppure attendere, ma non è opportuno)

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

Si può verificare (sì o no)? sì
Quale decisione può prendere il coordinatore?
Nessuna, perché il vecchio coordinatore è anche un partecipante e non sappiamo come ha deciso (potrebbe avere per qualche motivo abortito)

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)? sì
Quale decisione può prendere il coordinatore?
Commit

Basi di dati II — 21 giugno 2018 — Compito D

Domanda 7 (15% solo per prova completa)
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(T1)→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(T1,Y,Z)`					
	`prep(T1)→Y,Z`				
		`ready(T1)`		`ready(T1)`	
			`ready(T1)→X`		`ready(T1)→X`
				crash	
`commit(T1)`					
	`commit(T1)→Y,Z`				
		`commit(T1)`			
			`ack(T1)→X`		
		`prep(T2,X,Z)`			
			`prep(T2)→X,Z`		
`ready(T2)`					
	`ready(T2)→Y`				
crash					
		`abort(T2)`			
			`abort(T2)→X,Z`		
restart					
	`commit(T1)→Y,Z`				
			`ack(T1)→X`		
			`abort(T2)→X,Z`		
`abort(T2)`					
	`ack(T2)→Y`				
				restart	
			`abort(T2)→Z`		
				`abort(T2)`	
					`ack(T2)→Y`
		`complete(T2)`			
	`commit(T1)→Z`				
				`commit(T1)`	
					`ack(T1)→X`
`complete(T1)`					