Basi di dati II — 21 settembre 2016

Tempo a disposizione: due ore e trenta minuti.

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

Domanda 1 (15%) *Nota bene: l'argomento di questa domanda, come tale, non è stato trattato nel corso, ma è senz'altro possibile rispondere ai quesiti sulla base delle conoscenze acquisite nel corso stesso.*
Come certamente noto, ci sono molte applicazioni su Internet nelle quali è necessaria una grande scalabilità e che vengono quindi realizzate con architetture parallele con centinaia o migliaia di processori. Queste architetture sono di solito di tipo "shared-nothing," cioè ogni processore ha la propria memoria centrale e i propri dischi. Sulla base delle conoscenze relative alla gestione dei lock e dei buffer, spiegare perché invece una architettura "shared-disk," in cui ogni processore ha la propria memoria ma l'insieme dei dischi è condiviso, risulterebbe poco scalabile, rispetto alle prestazioni della base di dati, nonostante l'ampia disponibilità di memoria centrale.

Per queste applicazioni vengono usati di solito sistemi che premettono di eseguire solo operazioni semplici e localizzate sui singoli nodi dell'architettura. Di solito, i dati venogno partizionati fra i vari nodi sulla base di un campo chiave (o pseudochiave selettiva) e ogni operazione fa riferimento a uno o pochi valori della chiave. Il criterio di distribuzione è noto e il suo utilizzo non richiede l'accesso ai singoli nodi (dato un valore della chiave si individua direttamente il nodo competente). Spiegare perché questa "localizzazione" è fondamentale per le transazioni (cioè perché transazioni su più nodi sarebbero inefficienti).

Basi di dati II — 21 settembre 2016

Domanda 2 (15%)
Con riferimento al progetto "Carriere studenti," si considerino gli aspetti che riguardano:

- gli studenti, identificati dal codice fiscale, con data di nascita, indirizzo di residenza e coorte (che supponiamo sia specificata con l'anno di prima iscrizione all'università in assoluto); si noti che, come nel progetto realizzato, non viene qui associato allo studente il corso di studio cui è iscritto
- l'anno accademico di riferimento (una stringa nella forma x-1/x, ad esempio 2014/2015);
- il corso di studi, con un codice, un titolo, una durata legale, un'utenza sostenibile (il numero massimo di studenti ammissibili), un tipo ("corso di laurea triennale", "corso di laurea magistrale");
- i vari corsi, con un codice, un nome, un numero di CFU e un settore scientifico (ad esempio ING-INF/05).

Si supponga di disporre della seguente sorgente di dati che riporta gli eventi relativi al superamento di esami.

ESAMI

Studente (*stringa*)	Codicecorso (*stringa*)	Corsodistudi (*stringa*)	Voto (*numerico*)	Data (*tipo data*)	AnnoAccademico (*stringa*)
{1234-5678-1203}	503040	20802178	23	2016-01-31	2015/2016
{1234-5678-1203}	507045	20802178	29	2016-02-05	2015/2016
{1234-5678-6273}	503042	20202181	31	2016-01-31	2014/2015
{1234-5678-6273}	507045	20202181	28	2015-09-05	2014/2015
...	...	...	...	...	...

Si mostri lo schema dimensionale di un data mart da utilizzare per effettuare analisi sul numero di CFU conseguiti e sui voti riportati da ciascuno studente per data solare, anno accademico, corso e corso di studi. Mostrare lo schema relazionale (in qualunque modo compatto e comprensibile), indicando gli attributi (e le chiavi) di ciascuna delle relazioni. Si tratta sostanzialmente di quanto descritto sopra, versione semplificata di quello utilizzato nel progetto.

Domanda 3 (10%)
Con riferimento alla domanda 2, si supponga di voler gestire le variazioni nel tempo dell'indirizzo di residenza degli studenti e di voler tenere in considerazione, nelle analisi, il valore valido ad una determinata data.

1. Modificare opportunamente lo schema dimensionale progettato nella domanda 2 per tenere conto di questo aspetto.
2. Spiegare (anche in modo informale, ma chiaro) quali specificità dovrebbe avere il flusso di ETL utilizzato per acquisire i dati.

Domanda 4 (15%)
Spiegare, con una dimostrazione schematica ma ragionevolmente completa, che la conflict-seriallizzabilità implica la view-serializzabilità (ma non necessariamente viceversa) e che il 2PL stretto implica la conflict-serializzabilità (ma non necessariamente viceversa)

Basi di dati II — 21 settembre 2016

Domanda 5 (15%)
Considerare un'operazione di emissione biglietto ferroviario presso uno sportello automatico (che operi come terminale con capacità elaborativa ridotta, ma non nulla). Essa prevede, fra le altre, le tre azioni seguenti:

- effettuazione del pagamento con carta di credito
- emissione del biglietto
- commit della transazione

Rispondere alle seguenti domande:

1. la prima azione viene certamente eseguita per prima; in quale ordine vanno eseguite le altre due?

2. spiegare perché se le azioni venissero eseguite in ordine inverso (rispetto a quello indicato nella prima risposta) si avrebbero conseguenze inaccettabili

3. spiegare come potrebbe essere gestito, in tal caso, un guasto (caduta del sistema centrale, del terminale o del collegamento) verificatosi fra la seconda e la terza azione (notare che il pagamento avviene solo per mezzo di carta di credito; spiegare quali difficoltà aggiuntive si avrebbero per il pagamento in contanti)

Basi di dati II — 21 settembre 2016

Domanda 6 (15%)
Considerare le operazioni di raggruppamento, specificate in SQL con clausole `GROUP BY` e funzioni aggregative sui gruppi. Ad esempio:

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

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

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

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

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

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

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

A	B	C
101	a	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	d	2
111	a	2
112	k	4
113	a	1
114	k	2
115	e	3
116	d	3

Basi di dati II — 21 settembre 2016

Domanda 7 (15%)
Si consideri una base di dati con le relazioni

- R1(A,B,C,D) con
  - vincolo di integrità referenziale fra l'attributo D e la chiave E della relazione R2
  - L_1=2.000.000 ennuple e fattore di blocco f_1=200
  - b=200.000 valori diversi sull'attributo B (tutti gli interi compresi fra 1 e b)
  - c=20 valori diversi sull'attributo C (tutti gli interi compresi fra 1 e c)
  - una struttura disordinata, un indice sulla chiave primaria A e un altro sull'attributo B;
- R2(E,F,G) con
  - L_2=4.000.000 ennuple e fattore di blocco f_2=40
  - una struttura disordinata, un indice sulla chiave primaria E

Supponendo che:

- gli indici abbiano tutti i=4 livelli (contando anche radice e foglie) e fattore di blocco massimo f_i=100
- il sistema esegua sempre i join come nested loop (con accesso diretto tramite indice, se possibile)
- ogni operazione possa contare su un numero di pagine di buffer pari a circa q=120,

valutare il costo (indicandolo in modo sia simbolico sia numerico) di ciascuna delle interrogazioni seguenti:

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

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

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

Basi di dati II — 21 settembre 2016
Cenni sulle soluzioni

Tempo a disposizione: due ore e trenta minuti.

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

Domanda 1 (15%) *Nota bene: l'argomento di questa domanda, come tale, non è stato trattato nel corso, ma è senz'altro possibile rispondere ai quesiti sulla base delle conoscenze acquisite nel corso stesso.*
Come certamente noto, ci sono molte applicazioni su Internet nelle quali è necessaria una grande scalabilità e che vengono quindi realizzate con architetture parallele con centinaia o migliaia di processori. Queste architetture sono di solito di tipo "shared-nothing," cioè ogni processore ha la propria memoria centrale e i propri dischi. Sulla base delle conoscenze relative alla gestione dei lock e dei buffer, spiegare perché invece una architettura "shared-disk," in cui ogni processore ha la propria memoria ma l'insieme dei dischi è condiviso, risulterebbe poco scalabile, rispetto alle prestazioni della base di dati, nonostante l'ampia disponibilità di memoria centrale.

In una architettura shared-disk, la tabella dei lock e i buffer dovrebbero essere presenti su ogni nodo, con evidenti difficoltà di sincronizzazione

Per queste applicazioni vengono usati di solito sistemi che premettono di eseguire solo operazioni semplici e localizzate sui singoli nodi dell'architettura. Di solito, i dati venogno partizionati fra i vari nodi sulla base di un campo chiave (o pseudochiave selettiva) e ogni operazione fa riferimento a uno o pochi valori della chiave. Il criterio di distribuzione è noto e il suo utilizzo non richiede l'accesso ai singoli nodi (dato un valore della chiave si individua direttamente il nodo competente). Spiegare perché questa "localizzazione" è fondamentale per le transazioni (cioè perché transazioni su più nodi sarebbero inefficienti).

Se le transazioni non sono locali, serve il commit a due fasi, che porta a significativi overhead di scambi di messaggi e attese di sincronizzazione.

Basi di dati II — 21 settembre 2016

Domanda 2 (15%)
Con riferimento al progetto "Carriere studenti," si considerino gli aspetti che riguardano:

- gli studenti, identificati dal codice fiscale, con data di nascita, indirizzo di residenza e coorte (che supponiamo sia specificata con l'anno di prima iscrizione all'università in assoluto); si noti che, come nel progetto realizzato, non viene qui associato allo studente il corso di studio cui è iscritto
- l'anno accademico di riferimento (una stringa nella forma x-1/x, ad esempio 2014/2015);
- il corso di studi, con un codice, un titolo, una durata legale, un'utenza sostenibile (il numero massimo di studenti ammissibili), un tipo ("corso di laurea triennale", "corso di laurea magistrale");
- i vari corsi, con un codice, un nome, un numero di CFU e un settore scientifico (ad esempio ING-INF/05).

Si supponga di disporre della seguente sorgente di dati che riporta gli eventi relativi al superamento di esami.

ESAMI

Studente (*stringa*)	Codicecorso (*stringa*)	Corsodistudi (*stringa*)	Voto (*numerico*)	Data (*tipo data*)	AnnoAccademico (*stringa*)
{1234-5678-1203}	503040	20802178	23	2016-01-31	2015/2016
{1234-5678-1203}	507045	20802178	29	2016-02-05	2015/2016
{1234-5678-6273}	503042	20202181	31	2016-01-31	2014/2015
{1234-5678-6273}	507045	20202181	28	2015-09-05	2014/2015
...	...	...	...	...	...

Si mostri lo schema dimensionale di un data mart da utilizzare per effettuare analisi sul numero di CFU conseguiti e sui voti riportati da ciascuno studente per data solare, anno accademico, corso e corso di studi. Mostrare lo schema relazionale (in qualunque modo compatto e comprensibile), indicando gli attributi (e le chiavi) di ciascuna delle relazioni. Si tratta sostanzialmente di quanto descritto sopra, versione semplificata di quello utilizzato nel progetto.

Dimensioni

- STUDENTE(KStud, CF, Coorte, DataNascita, AnnoNascita, IndirizzoResidenza, ComuneResidenza, ...)
- ANNOACCADEMICO(KAA, AnnoAccademico, AnnoAccademicoX, ...)
- CORSODISTUDI(KCdS, CodiceCdS, TitoloCdS, ...)
- CORSO(KCorso, CodiceCorso, NomeCorso, CFUCorso, SSDCorso, ...)
- TEMPO(KTempo, Anno, Mese, Giorno, ...)

Tabella dei fatti (la grana è il singolo esame)

- FATTIESAMI(KStud, KAA, KCdS, KCorso, KTempo, Voto, NumeroCFU)

Basi di dati II — 21 settembre 2016

Domanda 3 (10%)
Con riferimento alla domanda 2, si supponga di voler gestire le variazioni nel tempo dell'indirizzo di residenza degli studenti e di voler tenere in considerazione, nelle analisi, il valore valido ad una determinata data.

1. Modificare opportunamente lo schema dimensionale progettato nella domanda 2 per tenere conto di questo aspetto.
2. Spiegare (anche in modo informale, ma chiaro) quali specificità dovrebbe avere il flusso di ETL utilizzato per acquisire i dati.

- La dimensione studente evolve lentamente ed è opportuno gestirla con la cosiddetta tecnica di tipo 2, cioè con una ennupla per ogni "versione" dello studente con diverso indirizzo. Possono essere utili date di inizio e fine validità della versione.
- Il flusso di ETL deve verificare la data in cui è stato sostenuto l'esame, per scegliere la versione corretta dello studente;

Basi di dati II — 21 settembre 2016

Domanda 4 (15%)
Spiegare, con una dimostrazione schematica ma ragionevolmente completa, che la conflict-seriallizzabilità implica la view-serializzabilità (ma non necessariamente viceversa) e che il 2PL stretto implica la conflict-serializzabilità (ma non necessariamente viceversa)

Vedi libro o lucidi

Basi di dati II — 21 settembre 2016

Domanda 5 (15%)
Considerare un'operazione di emissione biglietto ferroviario presso uno sportello automatico (che operi come terminale con capacità elaborativa ridotta, ma non nulla). Essa prevede, fra le altre, le tre azioni seguenti:

- effettuazione del pagamento con carta di credito
- emissione del biglietto
- commit della transazione

Rispondere alle seguenti domande:

1. la prima azione viene certamente eseguita per prima; in quale ordine vanno eseguite le altre due?

prima il commit e poi l'emissione del biglietto

2. spiegare perché se le azioni venissero eseguite in ordine inverso (rispetto a quello indicato nella prima risposta) si avrebbero conseguenze inaccettabili

Perché un crash dopo l'emissione del biglietto non permetterebbe di tenere traccia dell'emissione del biglietto né, tantomeno, di annullarla.

3. spiegare come potrebbe essere gestito, in tal caso, un guasto (caduta del sistema centrale, del terminale o del collegamento) verificatosi fra la seconda e la terza azione (notare che il pagamento avviene solo per mezzo di carta di credito; spiegare quali difficoltà aggiuntive si avrebbero per il pagamento in contanti)

La transazione verrebbe registrata (con il commit). L'emissione del biglietto da parte del componente meccanico invece no e quindi potrebbe venire richiesta, al riavvio del sistema, una "transazione compensativa" che porti al riaccredito delle somme pagate. Non si potrebbe rimettere il biglietto, perché non si avrebbe la garanzia della presenza dell'utente davanti al terminale. In caso di pagamento in contanti non si potrebbe restituire l'importo (sempre in assenza della garanzia della presenza)

Basi di dati II — 21 settembre 2016

Domanda 6 (15%)
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	a	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	d	2
111	a	2
112	k	4
113	a	1
114	k	2
115	e	3
116	d	3

Run su disco

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

109	a	5
111	a	2
110	d	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	f	3	

108	f	4	←
112	f	4	

115	e	3	←
114	f	2	

Record prodotti dalle prime 3 `next()`

a	36
d	12
e	6

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

Basi di dati II — 21 settembre 2016

Domanda 7 (15%)
Si consideri una base di dati con le relazioni

- R1(A,B,C,D) con
  - vincolo di integrità referenziale fra l'attributo D e la chiave E della relazione R2
  - L_1=2.000.000 ennuple e fattore di blocco f_1=200
  - b=200.000 valori diversi sull'attributo B (tutti gli interi compresi fra 1 e b)
  - c=20 valori diversi sull'attributo C (tutti gli interi compresi fra 1 e c)
  - una struttura disordinata, un indice sulla chiave primaria A e un altro sull'attributo B;
- R2(E,F,G) con
  - L_2=4.000.000 ennuple e fattore di blocco f_2=40
  - una struttura disordinata, un indice sulla chiave primaria E

Supponendo che:

- gli indici abbiano tutti i=4 livelli (contando anche radice e foglie) e fattore di blocco massimo f_i=100
- il sistema esegua sempre i join come nested loop (con accesso diretto tramite indice, se possibile)
- ogni operazione possa contare su un numero di pagine di buffer pari a circa q=120,

valutare il costo (indicandolo in modo sia simbolico sia numerico) di ciascuna delle interrogazioni seguenti:

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

$$\frac{L_1}{f_1} + \frac{L_1}{c}(i - 2 + 1) = \text{ ca. } 310.000$$

- scansione di R1 per la selezione: $\frac{L_1}{f_1}$
- un accesso diretto a R2 per ogni ennupla nel risultato della selezione, costo unitario come sopra

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

- usando l'indice: $\frac{L_1}{f_1} + L_1(i - 2 + 1) = $ ca. .6.000.000: scansione di R1, $\frac{L_1}{f_1}$, più un accesso diretto a R2 per ogni ennupla di R1: costo unitario profondità p dell'indice, meno 2 per i livelli nel buffer più uno per l'accesso al record
- senza usare l'indice (ma i buffer per ridurre il numero dei cicli) $\frac{L_1}{f_1} + \frac{L_1}{f_1} \times \frac{L_2}{f_2} \times \frac{1}{p} = $ ca. 1.000.000

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

$$i + \frac{L_1}{b} + \frac{L_1}{b}(i - 1 + 1) = \text{ ca. } 55$$

- accesso diretto a R1 per la selezione: i per l'indice e $\frac{L_1}{b}$ per le ennuple
- un accesso diretto a R2 per ogni ennupla nel risultato della selezione, costo unitario come sopra, con la differenza che si può ipotizzare la radice nel buffer, ma non gli altri livelli