Basi di dati II

Prova parziale — 15 maggio 2017 — Compito A

Tempo a disposizione: un'ora.

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

Domanda 1 (30%)
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)	
x = x + 10		
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 **5000**. 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 — 15 maggio 2017 — 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)	
x = x + 10		
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 **5000**.

client 1	client 2	client 3

Si verificano anomalie?

Basi di dati II — 15 maggio 2017 — Compito A

Domanda 2 (20%)
Considerare nuovamente lo scenario della domanda 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)	
x = x + 10		
write(x)		
	commit	
commit		
		read(x)
		commit

Considerare uno scheduler con controllo di concorrenza basato su **2PL** 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 **5000**.

client 1	client 2	client 3

Si verificano anomalie?

Basi di dati II — 15 maggio 2017 — Compito A

Domanda 3 (25%)
Considerare il rifornimento di benzina presso un distributore self-service come una transazione gestita secondo un protocollo che somiglia al commit a due fasi (ma non è uguale ad esso, anche perchè richiede uno specifico ordine per le operazioni). In effetti, si può pensare che ci sia un coordinatore (il dispositivo attraverso il quale si fanno le richieste), un accettatore di banconote e una pompa di erogazione.

1. Di solito, in caso di malfunzionamento della pompa (o comunque di mancato o parziale rifornimento) questi distributori rilasciano uno scontrino con il quale si può chiedere la restituzione dei soldi o un altro rifornimento. Spiegare questo comportamento e spiegare perchè, nel caso in cui invece dell'accettatore di banconote ci sia un lettore di carta di credito, questo scontrino può non essere rilasciato.

2. Per garantire un buon funzionamento del distributore, è opportuno (anche se ovviamente non è sempre vero, come discuteremo sotto) che il dispositivo di controllo si guasti molto raramente e, in caso di crash, se possibile, riparta subito. Perché?

3. In caso di guasto serio del dispositivo, fra l'accettazione delle banconote e l'erogazione del carburante, come si può pensare di ovviare alle difficoltà o almeno a fornire ragionevoli garanzie all'utente?
   (Individuare una opportuna soluzione che includa aspetti tecnici e soluzioni pragmatiche, fra queste ultime ad esempio un numero di telefono da contattare).

Basi di dati II — 15 maggio 2017 — Compito A

Domanda 4 (25%)
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 di blocchi 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 (supporre per semplicità che non ci 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):

Come nel caso precedente, ma con riferimento ad un programma che, per realizzare i 25.000 inserimenti, utilizzi complessivamente $k = 5000$ transazioni, ognuna con 5 inserimenti (supporre di nuovo che non ci 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

Prova parziale — 15 maggio 2017 — Compito B

Tempo a disposizione: un'ora.

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

Domanda 1 (30%)
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 + 10		
write(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 **2000**. 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 — 15 maggio 2017 — 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)	
		begin
		read(x)
x = x + 10		
write(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 **2000**.

client 1	client 2	client 3

Si verificano anomalie?

Basi di dati II — 15 maggio 2017 — Compito B

Domanda 2 (20%)
Considerare nuovamente lo scenario della domanda precedente, ripetuto qui sotto per comodità.

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

Considerare uno scheduler con controllo di concorrenza basato su **2PL** 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 **2000**.

client 1	client 2	client 3

Si verificano anomalie?

Basi di dati II — 15 maggio 2017 — Compito B

Domanda 3 (25%)
Considerare il rifornimento di benzina presso un distributore self-service come una transazione gestita secondo un protocollo che somiglia al commit a due fasi (ma non è uguale ad esso, anche perchè richiede uno specifico ordine per le operazioni). In effetti, si può pensare che ci sia un coordinatore (il dispositivo attraverso il quale si fanno le richieste), un lettore di carte di credito e una pompa di erogazione. Di solito, sulla carta di credito viene addebitato l'importo corrispondente alla benzina effettivamente erogata, quindi dopo l'erogazione.

1. Illustrare un possibile comportamento del sistema, specificando l'ordine con cui vengono eseguite le operazioni, nel caso in cui vadano tutte a buon fine.

2. Illustrare che cosa si può supporre che succeda in caso di guasto della pompa e quindi di mancata erogazione.

3. Illustrare come si può ipotizzare che venga gestito un guasto (che potrebbe essere serio e lungo, anche se certamente raro) del dispositivo di coordinamento, che avvenga fra il momento in cui si conclude l'erogazione e la comunicazione della quantità di benzina (che la pompa di erogazione deve comunicare al dispositivo di coordinamento).

Basi di dati II — 15 maggio 2017 — Compito B

Domanda 4 (25%)
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 di blocchi 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 (supporre per semplicità che non ci 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):

Come nel caso precedente, ma con riferimento ad un programma che, per realizzare i 50.000 inserimenti, utilizzi complessivamente $k = 10.000$ transazioni, ognuna con 5 inserimenti (supporre di nuovo che non ci 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

Prova parziale — 15 maggio 2017 — Compito C

Tempo a disposizione: un'ora.

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

Domanda 1 (30%)
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)	
x = x + 10		
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 **7000**. 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 — 15 maggio 2017 — 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)	
x = x + 10 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 **7000**.

client 1	client 2	client 3

Si verificano anomalie?

Basi di dati II — 15 maggio 2017 — Compito C

Domanda 2 (20%)
Considerare nuovamente lo scenario della domanda 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)	
x = x + 10		
write(x)		
	commit	
commit		
		read(x)
		commit

Considerare uno scheduler con controllo di concorrenza basato su **2PL** 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 **7000**.

client 1	client 2	client 3

Si verificano anomalie?

Basi di dati II — 15 maggio 2017 — Compito C

Domanda 3 (25%)
Considerare il rifornimento di benzina presso un distributore self-service come una transazione gestita secondo un protocollo che somiglia al commit a due fasi (ma non è uguale ad esso, anche perchè richiede uno specifico ordine per le operazioni). In effetti, si può pensare che ci sia un coordinatore (il dispositivo attraverso il quale si fanno le richieste), un accettatore di banconote e una pompa di erogazione.

1. Di solito, in caso di malfunzionamento della pompa (o comunque di mancato o parziale rifornimento) questi distributori rilasciano uno scontrino con il quale si può chiedere la restituzione dei soldi o un altro rifornimento. Spiegare questo comportamento e spiegare perchè, nel caso in cui invece dell'accettatore di banconote ci sia un lettore di carta di credito, questo scontrino può non essere rilasciato.

2. Per garantire un buon funzionamento del distributore, è opportuno (anche se ovviamente non è sempre vero, come discuteremo sotto) che il dispositivo di controllo si guasti molto raramente e, in caso di crash, se possibile, riparta subito. Perché?

3. In caso di guasto serio del dispositivo, fra l'accettazione delle banconote e l'erogazione del carburante, come si può pensare di ovviare alle difficoltà o almeno a fornire ragionevoli garanzie all'utente?
(Individuare una opportuna soluzione che includa aspetti tecnici e soluzioni pragmatiche, fra queste ultime ad esempio un numero di telefono da contattare).

Basi di dati II — 15 maggio 2017 — Compito C

Domanda 4 (25%)
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 di blocchi 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 (supporre per semplicità che non ci 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):

Come nel caso precedente, ma con riferimento ad un programma che, per realizzare i 100.000 inserimenti, utilizzi complessivamente $k = 10.000$ transazioni, ognuna con 10 inserimenti (supporre di nuovo che non ci 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

Prova parziale — 15 maggio 2017 — Compito D

Tempo a disposizione: un'ora.

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

Domanda 1 (30%)
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 + 10		
write(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 **4000**. 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 — 15 maggio 2017 — 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)	
		begin
		read(x)
x = x + 10		
write(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 **4000**.

client 1	client 2	client 3

Si verificano anomalie?

Basi di dati II — 15 maggio 2017 — Compito D

Domanda 2 (20%)
Considerare nuovamente lo scenario della domanda precedente, ripetuto qui sotto per comodità.

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

Considerare uno scheduler con controllo di concorrenza basato su **2PL** 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 **4000**.

client 1	client 2	client 3

Si verificano anomalie?

Basi di dati II — 15 maggio 2017 — Compito D

Domanda 3 (25%)
Considerare il rifornimento di benzina presso un distributore self-service come una transazione gestita secondo un protocollo che somiglia al commit a due fasi (ma non è uguale ad esso, anche perchè richiede uno specifico ordine per le operazioni). In effetti, si può pensare che ci sia un coordinatore (il dispositivo attraverso il quale si fanno le richieste), un lettore di carte di credito e una pompa di erogazione. Di solito, sulla carta di credito viene addebitato l'importo corrispondente alla benzina effettivamente erogata, quindi dopo l'erogazione.

1. Illustrare un possibile comportamento del sistema, specificando l'ordine con cui vengono eseguite le operazioni, nel caso in cui vadano tutte a buon fine.

2. Illustrare che cosa si può supporre che succeda in caso di guasto della pompa e quindi di mancata erogazione.

3. Illustrare come si può ipotizzare che venga gestito un guasto (che potrebbe essere serio e lungo, anche se certamente raro) del dispositivo di coordinamento, che avvenga fra il momento in cui si conclude l'erogazione e la comunicazione della quantità di benzina (che la pompa di erogazione deve comunicare al dispositivo di coordinamento).

Basi di dati II — 15 maggio 2017 — Compito D

Domanda 4 (25%)
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 di blocchi 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 (supporre per semplicità che non ci 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):

Come nel caso precedente, ma con riferimento ad un programma che, per realizzare i 50.000 inserimenti, utilizzi complessivamente $k = 5000$ transazioni, ognuna con 10 inserimenti (supporre di nuovo che non ci 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

Prova parziale — 15 maggio 2017 — Compito A

Cenni sulle soluzioni

Tempo a disposizione: un'ora.

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

Domanda 1 (30%)
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)	
x = x + 10 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 **5000**. 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. **Correggere soluzione**

client 1	client 2	client 3
		begin read(x) *legge 5000*
begin read(x) *legge 5000*		
	begin read(x) *legge 5000* x = x + 20 write(x) *scrive 5020*	
x = x + 10 xlock(x) *attesa*		
	commit	
write(x) *scrive 5010* commit		
		read(x) *legge 5000* commit

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

Basi di dati II — 15 maggio 2017 — 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)	
x = x + 10 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 **5000**.

client 1	client 2	client 3
		begin read(x) *legge 5000*
begin read(x) *legge 5000*		
	begin read(x) *legge 5000* x = x + 20 write(x) *scrive 5020*	
x = x + 10 xlock(x) *attesa*		
	commit	
write(x) *prova a scrivere* abort		
begin read(x) *legge 5020* x = x + 10 write(x) *scrive 5030* commit		
		read(x) *legge 5030* commit

Si verificano anomalie?
Lettura inconsistente per il client 3

Basi di dati II — 15 maggio 2017 — Compito A

Domanda 2 (20%)
Considerare nuovamente lo scenario della domanda 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)	
x = x + 10 write(x)		
	commit	
commit		
		read(x) commit

Considerare uno scheduler con controllo di concorrenza basato su **2PL** 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 **5000**.
Ci sono due possibili soluzioni. Qui supponiamo che lo stallo venga rilevato dopo la conclusione della transazione del client 3 e non ci sono anomalie. Se invece venisse rilevato prima, ci potrebbero essere diverse, con anomalia di lettura inconsistente sul client 3.

client 1	client 2	client 3
		begin read(x) *legge 5000*
begin read(x) *legge 5000*		
	begin read(x) *legge 5000* x = x + 20 xlock *attesa*	
x = x + 10 xlock(x) *attesa*		
		read(x) *legge 5000* commit
	abort (per stallo)	
write(x) *scrive 5010* commit		
	begin read(x) *legge 5010* x = x + 20 write(x) *scrive 5030* commit	

Si verificano anomalie?
vedere sopra

Basi di dati II — 15 maggio 2017 — Compito A

Domanda 3 (25%)
Considerare il rifornimento di benzina presso un distributore self-service come una transazione gestita secondo un protocollo che somiglia al commit a due fasi (ma non è uguale ad esso, anche perchè richiede uno specifico ordine per le operazioni). In effetti, si può pensare che ci sia un coordinatore (il dispositivo attraverso il quale si fanno le richieste), un accettatore di banconote e una pompa di erogazione.

1. Di solito, in caso di malfunzionamento della pompa (o comunque di mancato o parziale rifornimento) questi distributori rilasciano uno scontrino con il quale si può chiedere la restituzione dei soldi o un altro rifornimento. Spiegare questo comportamento e spiegare perchè, nel caso in cui invece dell'accettatore di banconote ci sia un lettore di carta di credito, questo scontrino può non essere rilasciato.

In caso di pagamento in contanti, lo scontrino è una forma ragionevole di transazione compensativa (più "sicuro" rispetto alla restituzione delle banconote). In caso di carta di credito è invece molto semplice annullare la transazione o effettuare uno storno

2. Per garantire un buon funzionamento del distributore, è opportuno (anche se ovviamente non è sempre vero, come discuteremo sotto) che il dispositivo di controllo si guasti molto raramente e, in caso di crash, se possibile, riparta subito. Perché?

Il dispositivo di controllo gestisce le operazioni e senza di esso non si può fare quasi nulla. La ripartenza veloce, con l'ausilio di opportuni log, permette di completare (o annullare) la transazione in corso (o le transazioni, se ci sono più pompe).

3. In caso di guasto serio del dispositivo, fra l'accettazione delle banconote e l'erogazione del carburante, come si può pensare di ovviare alle difficoltà o almeno a fornire ragionevoli garanzie all'utente?
(Individuare una opportuna soluzione che includa aspetti tecnici e soluzioni pragmatiche, fra queste ultime ad esempio un numero di telefono da contattare).

Sono possibile ovviamente varie soluzioni. Quella ipotizzata prevede la possibilità di contattare un call center, che prenda nota di orario e altre informazioni sull'utente e sul servizio, che possano poi essere confrontate con i log del sistema (del controllore e della pompa), per attivare il rimborso. È come sempre essenziale la disponibilità di log

Basi di dati II — 15 maggio 2017 — Compito A

Domanda 4 (25%)
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:

Le possibili risposte sono in **grassetto** Indicare il numero di scritture di blocchi 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 (supporre per semplicità che non ci siano altre transazioni attive)

- numero di scritture di pagine di log:
  almeno $N = 25.000$, 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, $N = 25.000$; probabilmente di meno, anche solo $N/f = 125$, una per blocco (dove f indica il fattore di blocco $f = D/L =$ 200 nei compiti A e C e 400 nei compiti B e D); questo perché le scritture vengono ottimizzate ed eseguite "quando fa comodo"
  - strategia undo-only (no-redo):
    in questo caso certamente una per transazione, $N = 25.000$, 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 25.000 inserimenti, utilizzi complessivamente $k = 5000$ transazioni, ognuna con 5 inserimenti (supporre di nuovo che non ci 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 $N/k = 5$ ennuple e quindi le relative scritture su log occupano $N/k \times L \times 3 =$ 300B, che entrano in un blocco. In totale, per $k = 5000$ transazioni, circa $k = 5000$ 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 $N/f = 125$, una per blocco
  - strategia undo-only (no-redo):
    una per ogni transazione, $k = 5000$ scritture
  - strategia redo-only (no-undo):
    anche in questo caso, si può pensare che ogni blocco si scriva una volta sola, quindi ancora $N/f = 125$

Basi di dati II

Prova parziale — 15 maggio 2017 — Compito B

Cenni sulle soluzioni

Tempo a disposizione: un'ora.

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

Domanda 1 (30%)
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 + 10		
write(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 **2000**. 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 2000*		
	begin	
	read(x) *legge 2000*	
		begin
		read(x) *legge 2000*
x = x + 10		
write(x) *scrive 2010*		
	x = x + 20	
	xlock(x) *attesa*	
commit		
	write(x) *scrive 2020*	
	commit	
		read(x) *legge 2000*
		commit

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

Basi di dati II — 15 maggio 2017 — 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)	
		begin read(x)
x = x + 10 write(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 **2000**.

client 1	client 2	client 3
begin read(x) *legge 2000*		
	begin read(x) *legge 2000*	
		begin read(x) *legge 2000*
x = x + 10 write(x) *scrive 2010*		
	x = x + 20 xlock(x) *attesa*	
commit		
	write(x) *prova a scrivere* abort	
	begin read(x) *legge 2010* x = x + 20 write(x) *scrive 2030* commit	
		read(x) *legge 2030* commit

Si verificano anomalie?
Lettura inconsistente per il client 3

Basi di dati II — 15 maggio 2017 — Compito B

Domanda 2 (20%)
Considerare nuovamente lo scenario della domanda precedente, ripetuto qui sotto per comodità.

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

Considerare uno scheduler con controllo di concorrenza basato su **2PL** 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 **2000**.
Ci sono due possibili soluzioni. Qui supponiamo che lo stallo venga rilevato dopo la conclusione della transazione del client 3 e non ci sono anomalie. Se invece venisse rilevato prima, ci potrebbero essere diverse, con anomalia di lettura inconsistente sul client 3.

client 1	client 2	client 3
begin read(x) *legge 2000*		
	begin read(x) *legge 2000*	
		begin read(x) *legge 2000*
x = x + 10 xlock(x) *attesa*		
	x = x + 20 xlock(x) *attesa*	
		read(x) *legge 2000* commit
abort		
	write(x) *scrive 2020* commit	
begin read(x) *legge 2020* x = x + 10 write(x) *scrive 2030* commit		

Si verificano anomalie?
vedere sopra

Basi di dati II — 15 maggio 2017 — Compito B

Domanda 3 (25%)
Considerare il rifornimento di benzina presso un distributore self-service come una transazione gestita secondo un protocollo che somiglia al commit a due fasi (ma non è uguale ad esso, anche perchè richiede uno specifico ordine per le operazioni). In effetti, si può pensare che ci sia un coordinatore (il dispositivo attraverso il quale si fanno le richieste), un lettore di carte di credito e una pompa di erogazione. Di solito, sulla carta di credito viene addebitato l'importo corrispondente alla benzina effettivamente erogata, quindi dopo l'erogazione.

1. Illustrare un possibile comportamento del sistema, specificando l'ordine con cui vengono eseguite le operazioni, nel caso in cui vadano tutte a buon fine.

possibile soluzione

(a) inserimento carta di credito

(b) verifica della validià della carta di credito ("preautorizzazione") e (di solito) restituzione della carta

(c) comunicazione alla pompa di erogazione

(d) erogazione

(e) comunicazione dalla pompa al controllore con l'importo

(f) addebito dell'importo

vspace*1cm

2. Illustrare che cosa si può supporre che succeda in caso di guasto della pompa e quindi di mancata erogazione.

Nessun problema particolare, visto l'addebito viene eseguito dopo l'erogazione

3. Illustrare come si può ipotizzare che venga gestito un guasto (che potrebbe essere serio e lungo, anche se certamente raro) del dispositivo di coordinamento, che avvenga fra il momento in cui si conclude l'erogazione e la comunicazione della quantità di benzina (che la pompa di erogazione deve comunicare al dispositivo di coordinamento).

La pompa deve registrare (in un proprio log) le operazioni svolte, in modo che, al ripristino del dispositivo di coordinamento sia possibile comunicare ad esso l'avvenuta erogazione e possa quindi essere registrato il conseguente addebito

Basi di dati II — 15 maggio 2017 — Compito B

Domanda 4 (25%)
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:

Le possibili risposte sono in **grassetto** Indicare il numero di scritture di blocchi 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 (supporre per semplicità che non ci siano altre transazioni attive)

- 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 $f = D/L =$ 200 nei compiti A e C e 400 nei compiti B e D); questo perché le scritture vengono ottimizzate ed eseguite "quando fa comodo"
  - 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 $k = 10.000$ transazioni, ognuna con 5 inserimenti (supporre di nuovo che non ci 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 $T/k = 5$ ennuple e quindi le relative scritture su log occupano $T/k \times L \times 3 =$ 300B, che entrano in un blocco. In totale, per $k = 10.000$ transazioni, circa $k = 10.000$ 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):
    una per ogni transazione, $k = 10.000$ scritture
  - 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

Prova parziale — 15 maggio 2017 — Compito C

Cenni sulle soluzioni

Tempo a disposizione: un'ora.

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

Domanda 1 (30%)
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)	
x = x + 10 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 **7000**. 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 7000*
begin read(x) *legge 7000*		
	begin read(x) *legge 7000* x = x + 20 write(x) *scrive 7020*	
x = x + 10 xlock(x) *attesa*		
	commit	
write(x) *prova a scrivere* abort		
begin read(x) *legge 7020* x = x + 10 write(x) *scrive 7030* commit		
		read(x) *legge 7030* commit

Si verificano anomalie?
Lettura inconsistente per il client 3

Basi di dati II — 15 maggio 2017 — 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)	
x = x + 10 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 **7000**. **Correggere soluzione**

client 1	client 2	client 3
		begin read(x) *legge 7000*
begin read(x) *legge 7000*		
	begin read(x) *legge 7000* x = x + 20 write(x) *scrive 7020*	
x = x + 10 xlock(x) *attesa*		
	commit	
write(x) *scrive 7010* commit		
		read(x) *legge 7000* commit

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

Basi di dati II — 15 maggio 2017 — Compito C

Domanda 2 (20%)
Considerare nuovamente lo scenario della domanda 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)	
x = x + 10 write(x)		
	commit	
commit		
		read(x) commit

Considerare uno scheduler con controllo di concorrenza basato su **2PL** 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 **7000**.
Ci sono due possibili soluzioni. Qui supponiamo che la transazione sul client 3 si concluda rapidamente e quindi le altre due tollerino l'attesa. Si ha l'anomalia di perdita di aggiornamento fra 1 e 2. Se invece scattasse qualche timeout, il tutto dipenderebbe dall'ordine di esecuzione e potrebbero fortunosamente evitarsi le anomalie

client 1	client 2	client 3
		begin read(x) *legge 7000*
begin read(x) *legge 7000*		
	begin read(x) *legge 7000* x = x + 20 xlock *attesa*	
x = x + 10 xlock(x) *attesa*		
		read(x) *legge 7000* commit
write(x) *scrive 7010* commit		
	write(x) *scrive 7020* commit	

Si verificano anomalie?
vedere sopra

Basi di dati II — 15 maggio 2017 — Compito C

Domanda 3 (25%)
Considerare il rifornimento di benzina presso un distributore self-service come una transazione gestita secondo un protocollo che somiglia al commit a due fasi (ma non è uguale ad esso, anche perchè richiede uno specifico ordine per le operazioni). In effetti, si può pensare che ci sia un coordinatore (il dispositivo attraverso il quale si fanno le richieste), un accettatore di banconote e una pompa di erogazione.

1. Di solito, in caso di malfunzionamento della pompa (o comunque di mancato o parziale rifornimento) questi distributori rilasciano uno scontrino con il quale si può chiedere la restituzione dei soldi o un altro rifornimento. Spiegare questo comportamento e spiegare perchè, nel caso in cui invece dell'accettatore di banconote ci sia un lettore di carta di credito, questo scontrino può non essere rilasciato.

In caso di pagamento in contanti, lo scontrino è una forma ragionevole di transazione compensativa (più "sicuro" rispetto alla restituzione delle banconote). In caso di carta di credito è invece molto semplice annullare la transazione o effettuare uno storno

2. Per garantire un buon funzionamento del distributore, è opportuno (anche se ovviamente non è sempre vero, come discuteremo sotto) che il dispositivo di controllo si guasti molto raramente e, in caso di crash, se possibile, riparta subito. Perché?

Il dispositivo di controllo gestisce le operazioni e senza di esso non si può fare quasi nulla. La ripartenza veloce, con l'ausilio di opportuni log, permette di completare (o annullare) la transazione in corso (o le transazioni, se ci sono più pompe).

3. In caso di guasto serio del dispositivo, fra l'accettazione delle banconote e l'erogazione del carburante, come si può pensare di ovviare alle difficoltà o almeno a fornire ragionevoli garanzie all'utente?
(Individuare una opportuna soluzione che includa aspetti tecnici e soluzioni pragmatiche, fra queste ultime ad esempio un numero di telefono da contattare).

Sono possibile ovviamente varie soluzioni. Quella ipotizzata prevede la possibilità di contattare un call center, che prenda nota di orario e altre informazioni sull'utente e sul servizio, che possano poi essere confrontate con i log del sistema (del controllore e della pompa), per attivare il rimborso. È come sempre essenziale la disponibilità di log

Basi di dati II — 15 maggio 2017 — Compito C

Domanda 4 (25%)
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:

Le possibili risposte sono in **grassetto** Indicare il numero di scritture di blocchi 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 (supporre per semplicità che non ci siano altre transazioni attive)

- numero di scritture di pagine di log:
  almeno $M = 100.000$, 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, $M = 100.000$; probabilmente di meno, anche solo $M/f = 500$, una per blocco (dove f indica il fattore di blocco $f = D/L =$ 200 nei compiti A e C e 400 nei compiti B e D); questo perché le scritture vengono ottimizzate ed eseguite "quando fa comodo"
  - strategia undo-only (no-redo):
    in questo caso certamente una per transazione, $M = 100.000$, 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 100.000 inserimenti, utilizzi complessivamente $k = 10.000$ transazioni, ognuna con 10 inserimenti (supporre di nuovo che non ci 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 $M/k = 10$ ennuple e quindi le relative scritture su log occupano $M/k \times L \times 3 =$ 600B, che entrano in un blocco. In totale, per $k = 10.000$ transazioni, circa $k = 10.000$ 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 $M/f = 500$, una per blocco
  - strategia undo-only (no-redo):
    una per ogni transazione, $k = 10.000$ scritture
  - strategia redo-only (no-undo):
    anche in questo caso, si può pensare che ogni blocco si scriva una volta sola, quindi ancora $M/f = 500$

Basi di dati II

Prova parziale — 15 maggio 2017 — Compito D

Cenni sulle soluzioni

Tempo a disposizione: un'ora.

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

Domanda 1 (30%)
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 + 10		
write(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 **4000**. 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 4000*		
	begin	
	read(x) *legge 4000*	
		begin
		read(x) *legge 4000*
x = x + 10		
write(x) *scrive 4010*		
	x = x + 20	
	xlock(x) *attesa*	
commit		
	write(x) *prova a scrivere*	
	abort	
	begin	
	read(x) *legge 4010*	
	x = x + 20	
	write(x) *scrive 4030*	
	commit	
		read(x) *legge 4030*
		commit

Si verificano anomalie?
Lettura inconsistente per il client 3

Basi di dati II — 15 maggio 2017 — 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)	
		begin read(x)
x = x + 10 write(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 **4000**.

client 1	client 2	client 3
begin read(x) *legge 4000*		
	begin read(x) *legge 4000*	
		begin read(x) *legge 4000*
x = x + 10 write(x) *scrive 4010*		
	x = x + 20 xlock(x) *attesa*	
commit		
	write(x) *scrive 4020* commit	
		read(x) *legge 4000* commit

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

Basi di dati II — 15 maggio 2017 — Compito D

Domanda 2 (20%)
Considerare nuovamente lo scenario della domanda precedente, ripetuto qui sotto per comodità.

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

Considerare uno scheduler con controllo di concorrenza basato su **2PL** 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 **4000**.
Ci sono due possibili soluzioni. Qui supponiamo che la transazione sul client 3 si concluda rapidamente e quindi le altre due tollerino l'attesa. Si ha l'anomalia di perdita di aggiornamento fra 1 e 2. Se invece scattasse qualche timeout, il tutto dipenderebbe dall'ordine di esecuzione e potrebbero fortunosamente evitarsi le anomalie

client 1	client 2	client 3
begin		
read(x) *legge 4000*		
	begin	
	read(x) *legge 4000*	
		begin
		read(x) *legge 4000*
x = x + 10		
xlock(x) *attesa*		
	x = x + 20	
	xlock(x) *attesa*	
		read(x) *legge 4000*
		commit
write(x) *scrive 4010*		
commit		
	write(x) *scrive 4020*	
	commit	

Si verificano anomalie?
vedere sopra

Basi di dati II — 15 maggio 2017 — Compito D

Domanda 3 (25%)
Considerare il rifornimento di benzina presso un distributore self-service come una transazione gestita secondo un protocollo che somiglia al commit a due fasi (ma non è uguale ad esso, anche perchè richiede uno specifico ordine per le operazioni). In effetti, si può pensare che ci sia un coordinatore (il dispositivo attraverso il quale si fanno le richieste), un lettore di carte di credito e una pompa di erogazione. Di solito, sulla carta di credito viene addebitato l'importo corrispondente alla benzina effettivamente erogata, quindi dopo l'erogazione.

1. Illustrare un possibile comportamento del sistema, specificando l'ordine con cui vengono eseguite le operazioni, nel caso in cui vadano tutte a buon fine.

possibile soluzione

(a) inserimento carta di credito

(b) verifica della validià della carta di credito ("preautorizzazione") e (di solito) restituzione della carta

(c) comunicazione alla pompa di erogazione

(d) erogazione

(e) comunicazione dalla pompa al controllore con l'importo

(f) addebito dell'importo

vspace*1cm

2. Illustrare che cosa si può supporre che succeda in caso di guasto della pompa e quindi di mancata erogazione.

Nessun problema particolare, visto l'addebito viene eseguito dopo l'erogazione

3. Illustrare come si può ipotizzare che venga gestito un guasto (che potrebbe essere serio e lungo, anche se certamente raro) del dispositivo di coordinamento, che avvenga fra il momento in cui si conclude l'erogazione e la comunicazione della quantità di benzina (che la pompa di erogazione deve comunicare al dispositivo di coordinamento).

La pompa deve registrare (in un proprio log) le operazioni svolte, in modo che, al ripristino del dispositivo di coordinamento sia possibile comunicare ad esso l'avvenuta erogazione e possa quindi essere registrato il conseguente addebito

Basi di dati II — 15 maggio 2017 — Compito D

Domanda 4 (25%)
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:

Le possibili risposte sono in **grassetto** Indicare il numero di scritture di blocchi 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 (supporre per semplicità che non ci siano altre transazioni attive)

- numero di scritture di pagine di log:
  almeno $R = 50.000$, 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, $R = 50.000$; probabilmente di meno, anche solo $R/f = 125$, una per blocco (dove f indica il fattore di blocco $f = D/L =$ 200 nei compiti A e C e 400 nei compiti B e D); questo perché le scritture vengono ottimizzate ed eseguite "quando fa comodo"
  - strategia undo-only (no-redo):
    in questo caso certamente una per transazione, $R = 50.000$, 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 $k = 5000$ transazioni, ognuna con 10 inserimenti (supporre di nuovo che non ci 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 $R/k = 10$ ennuple e quindi le relative scritture su log occupano $R/k \times L \times 3 =$ 300B, che entrano in un blocco. In totale, per $k = 5000$ transazioni, circa $k = 5000$ 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 $R/f = 125$, una per blocco
  - strategia undo-only (no-redo):
    una per ogni transazione, $k = 5000$ scritture
  - strategia redo-only (no-undo):
    anche in questo caso, si può pensare che ogni blocco si scriva una volta sola, quindi ancora $R/f = 125$