Basi di dati II — Prova parziale — 27 maggio 2019 — Compito A

Tempo a disposizione: un'ora.

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

Basi di dati II — 27 maggio 2019 — Compito A

Domanda 1 (25%)
Considerare il seguente scenario in cui due client 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).

`start transaction`
`  isolation level read committed;`
`read(x);`
`x = x + 200;`
`write(x);`
 	
`read(y);`
`y = y + 100;`
`write(y);`
`commit`
	`start transaction`
	`  isolation level read committed;`
	`read(y);`
	`y = y + 100;`
	`write(y)`
	`read(x);`
	`x = x + 100;`
	`write(x);`
	`commit`

Considerare uno scheduler con controllo di concorrenza basato su **Multversioni** (come in Postgres). Mostrare il comportamento dello scheduler, supponendo che il valore iniziale degli oggetti x e y sia per entrambi 300. Indicare, nell'ordine, le operazioni che vengono eseguite da ciascun client, specificando, per ciascuna, il valore che viene letto o scritto.

Basi di dati II — 27 maggio 2019 — Compito A

Domanda 2 (25%)
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)?
Quale decisione può prendere il coordinatore?

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

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

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

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

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

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

Domanda 3 (25%)
Considerare i seguenti scenari in cui due client inviano richieste ad un gestore del controllo di concorrenza. Per il secondo scenario si utilizza una notazione intuitiva anche se non ammissibile in Postgres (con le variabili `XX` e `YY`).

```
start transaction
  isolation level serializable;
select * from impiegati;
```

```
start transaction
  isolation level serializable;
select * from impiegati;
```

```
insert into impiegati
  (matricola, cognome, conteggio)
  values (101,'Rossi',1);
commit
```

```
insert into impiegati
  (matricola, cognome, conteggio)
  values (102,'Bruni',2);
commit
```

```
start transaction
  isolation level serializable;
XX = select count(*) from impiegati;
```

```
start transaction
  isolation level serializable;
YY = select count(*) from impiegati;
```

```
insert into impiegati
  (matricola, cognome, conteggio)
  values (101,'Rossi', XX+1);
commit
```

```
insert into impiegati
  (matricola, cognome, conteggio)
  values (102,'Bruni', YY+1));
commit
```

In entrambi i casi, la seconda transazione viene abortita al momento del commit.

- Spiegare brevemente perché

(continua a pagina seguente)

Se invece il livello di isolamento fosse stato `repeatable read`, allora entrambi in entrambi gli scenari le transazioni si sarebbero concluse con accettazione del commit.

- Spiegare brevemente perché

- Commentare i risultati ottenuti nei due scenari e spiegare perché uno dei due va considerato indesiderabile. Allo scopo, mostrare il contenuto della relazione dopo l'inserimento, supponendola vuota all'inizio. Per comprendere meglio il comportamento, tenere presente che l'attributo `conteggio`, in ciascuna ennupla, serve in sostanza ad indicare la cardinalità della relazione subito dopo l'inserimento della ennupla stessa.

In sostanza, si può osservare che, per ciascun livello di isolamento, i due scenari vengono trattati nello stesso modo: con `serializable` vengono rifiutati entrambi, anche se uno dei due è accettabile, mentre con `repeatable read` vengono accettati entrambi, anche se uno dei due è indesiderabile.

- Spiegare brevemente perché

Basi di dati II — 27 maggio 2019 — 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 in memoria secondaria necessarie per realizzare i 25.000 inserimenti, supponendo che i record di log abbiano una lunghezza pari a circa il triplo di quella dei record del file, con riferimento ad un programma che utilizzi una transazione separata per ciascun inserimento

- numero di scritture di pagine di log:

- numero di scritture di pagine della relazione, assumendo che il sistema utilizzi una strategia undo-redo senza vincoli particolari

Come nel caso precedente, ma con riferimento ad un programma che, per realizzare i 25.000 inserimenti, utilizzi complessivamente $k = 5$ transazioni, ognuna con 5000 inserimenti (e supponendo che non vi siano altre transazioni attive)

- numero di scritture di pagine di log:

- numero di scritture di pagine della relazione, sempre assumendo una strategia undo-redo senza vincoli particolari

Basi di dati II — Prova parziale — 27 maggio 2019 — Compito B

Tempo a disposizione: un'ora.

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

Basi di dati II — 27 maggio 2019 — Compito B

Domanda 1 (25%)
Considerare il seguente scenario in cui due client 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).

`start transaction`
`  isolation level serializable;`
`read(x);`
`x = x + 300;`
`write(x);`
 	
`read(y);`
`y = y + 100;`
`write(y);`
`commit`
	
	`start transaction`
	`  isolation level serializable;`
	`read(y);`
	`y = y + 100;`
	`write(y)`
	`read(x);`
	`x = x + 100;`
	`write(x);`
	`commit`

Considerare uno scheduler con controllo di concorrenza basato su **Multversioni** (come in Postgres). Mostrare il comportamento dello scheduler, supponendo che il valore iniziale degli oggetti x e y sia per entrambi 100. Indicare, nell'ordine, le operazioni che vengono eseguite da ciascun client, specificando, per ciascuna, il valore che viene letto o scritto.

Basi di dati II — 27 maggio 2019 — Compito B

Domanda 2 (25%)
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)? Quale decisione può prendere il coordinatore?

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

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

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

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

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

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

Domanda 3 (25%)
Considerare i seguenti scenari in cui due client inviano richieste ad un gestore del controllo di concorrenza. Per il secondo scenario si utilizza una notazione intuitiva anche se non ammissibile in Postgres (con le variabili `XX` e `YY`).

```
start transaction
  isolation level serializable;
select * from impiegati;
```

```
start transaction
  isolation level serializable;
select * from impiegati;
```

```
insert into impiegati
  (matricola, cognome, conteggio)
  values (101,'Rossi',1);
commit
```

```
insert into impiegati
  (matricola, cognome, conteggio)
  values (102,'Bruni',2);
commit
```

```
start transaction
  isolation level serializable;
XX = select count(*) from impiegati;
```

```
start transaction
  isolation level serializable;
YY = select count(*) from impiegati;
```

```
insert into impiegati
  (matricola, cognome, conteggio)
  values (101,'Rossi', XX+1);
commit
```

```
insert into impiegati
  (matricola, cognome, conteggio)
  values (102,'Bruni', YY+1));
commit
```

In entrambi i casi, la seconda transazione viene abortita al momento del commit.

- Spiegare brevemente perché

(continua a pagina seguente)

Se invece il livello di isolamento fosse stato `repeatable read`, allora entrambi in entrambi gli scenari le transazioni si sarebbero concluse con accettazione del commit.

- Spiegare brevemente perché

- Commentare i risultati ottenuti nei due scenari e spiegare perché uno dei due va considerato indesiderabile. Allo scopo, mostrare il contenuto della relazione dopo l'inserimento, supponendola vuota all'inizio. Per comprendere meglio il comportamento, tenere presente che l'attributo `conteggio`, in ciascuna ennupla, serve in sostanza ad indicare la cardinalità della relazione subito dopo l'inserimento della ennupla stessa.

In sostanza, si può osservare che, per ciascun livello di isolamento, i due scenari vengono trattati nello stesso modo: con `serializable` vengono rifiutati entrambi, anche se uno dei due è accettabile, mentre con `repeatable read` vengono accettati entrambi, anche se uno dei due è indesiderabile.

- Spiegare brevemente perché

Basi di dati II — 27 maggio 2019 — 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 $R = 50.000$ ennuple, con valori della chiave tutti diversi fra loro e da quelli già nella relazione (quindi il sistema verifica il soddisfacimento del vincolo di chiave e ammette tutte le operazioni).
Rispondere alle domande seguenti, indicando formule e valori numerici:

Indicare il numero di scritture in memoria secondaria necessarie per realizzare i 50.000 inserimenti, supponendo che i record di log abbiano una lunghezza pari a circa il triplo di quella dei record del file, con riferimento ad un programma che utilizzi una transazione separata per ciascun inserimento

- numero di scritture di pagine di log:

- numero di scritture di pagine della relazione, assumendo che il sistema utilizzi una strategia undo-redo senza vincoli particolari

Come nel caso precedente, ma con riferimento ad un programma che, per realizzare i 50.000 inserimenti, utilizzi complessivamente $k = 5$ transazioni, ognuna con 10.000 inserimenti (e supponendo che non vi siano altre transazioni attive)

- numero di scritture di pagine di log:

- numero di scritture di pagine della relazione, sempre assumendo una strategia undo-redo senza vincoli particolari

Basi di dati II — Prova parziale — 27 maggio 2019 — Compito C

Tempo a disposizione: un'ora.

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

Basi di dati II — 27 maggio 2019 — Compito C

Domanda 1 (25%)
Considerare il seguente scenario in cui due client 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).

`start transaction`
`  isolation level read committed;`
`read(x);`
`x = x + 400;`
`write(x);`
 	
`read(y);`
`y = y + 100;`
`write(y);`
`commit`
	`start transaction`
	`  isolation level read committed;`
	`read(y);`
	`y = y + 100;`
	`write(y)`
	`read(x);`
	`x = x + 100;`
	`write(x);`
	`commit`

Considerare uno scheduler con controllo di concorrenza basato su **Multversioni** (come in Postgres). Mostrare il comportamento dello scheduler, supponendo che il valore iniziale degli oggetti x e y sia per entrambi 200. Indicare, nell'ordine, le operazioni che vengono eseguite da ciascun client, specificando, per ciascuna, il valore che viene letto o scritto.

Basi di dati II — 27 maggio 2019 — Compito C

Domanda 2 (25%)
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)?
Quale decisione può prendere il coordinatore?

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

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

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

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

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

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

Domanda 3 (25%)
Considerare i seguenti scenari in cui due client inviano richieste ad un gestore del controllo di concorrenza. Per il secondo scenario si utilizza una notazione intuitiva anche se non ammissibile in Postgres (con le variabili `XX` e `YY`).

`start transaction`
`  isolation level serializable;`
`select * from impiegati;`	

	`start transaction`
	`  isolation level serializable;`
	`select * from impiegati;`

`insert into impiegati`
`  (matricola, cognome, conteggio)`
`  values (101,'Rossi',1);`
`commit`	

	`insert into impiegati`
	`  (matricola, cognome, conteggio)`
	`  values (102,'Bruni',2);`
	`commit`

`start transaction`
`  isolation level serializable;`
`XX = select count(*) from impiegati;`	

	`start transaction`
	`  isolation level serializable;`
	`YY = select count(*) from impiegati;`

`insert into impiegati`
`  (matricola, cognome, conteggio)`
`  values (101,'Rossi', XX+1);`
`commit`	

	`insert into impiegati`
	`  (matricola, cognome, conteggio)`
	`  values (102,'Bruni', YY+1));`
	`commit`

In entrambi i casi, la seconda transazione viene abortita al momento del commit.

- Spiegare brevemente perché

(continua a pagina seguente)

Se invece il livello di isolamento fosse stato `repeatable read`, allora entrambi in entrambi gli scenari le transazioni si sarebbero concluse con accettazione del commit.

- Spiegare brevemente perché

- Commentare i risultati ottenuti nei due scenari e spiegare perché uno dei due va considerato indesiderabile. Allo scopo, mostrare il contenuto della relazione dopo l'inserimento, supponendola vuota all'inizio. Per comprendere meglio il comportamento, tenere presente che l'attributo `conteggio`, in ciascuna ennupla, serve in sostanza ad indicare la cardinalità della relazione subito dopo l'inserimento della ennupla stessa.

In sostanza, si può osservare che, per ciascun livello di isolamento, i due scenari vengono trattati nello stesso modo: con `serializable` vengono rifiutati entrambi, anche se uno dei due è accettabile, mentre con `repeatable read` vengono accettati entrambi, anche se uno dei due è indesiderabile.

- Spiegare brevemente perché

Basi di dati II — 27 maggio 2019 — 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 in memoria secondaria necessarie per realizzare i 100.000 inserimenti, supponendo che i record di log abbiano una lunghezza pari a circa il triplo di quella dei record del file, con riferimento ad un programma che utilizzi una transazione separata per ciascun inserimento

- numero di scritture di pagine di log:

- numero di scritture di pagine della relazione, assumendo che il sistema utilizzi una strategia undo-redo senza vincoli particolari

Come nel caso precedente, ma con riferimento ad un programma che, per realizzare i 100.000 inserimenti, utilizzi complessivamente $k = 10$ transazioni, ognuna con 10.000 inserimenti (e supponendo che non vi siano altre transazioni attive)

- numero di scritture di pagine di log:

- numero di scritture di pagine della relazione, sempre assumendo una strategia undo-redo senza vincoli particolari

Basi di dati II — Prova parziale — 27 maggio 2019 — Compito D

Tempo a disposizione: un'ora.

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

Basi di dati II — 27 maggio 2019 — Compito D

Domanda 1 (25%)
Considerare il seguente scenario in cui due client 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).

`start transaction`
`  isolation level serializable;`
`read(x);`
`x = x + 600;`
`write(x);`
 	
`read(y);`
`y = y + 100;`
`write(y);`
`commit`
	
	`start transaction`
	`  isolation level serializable;`
	`read(y);`
	`y = y + 100;`
	`write(y)`
	`read(x);`
	`x = x + 100;`
	`write(x);`
	`commit`

Considerare uno scheduler con controllo di concorrenza basato su **Multversioni** (come in Postgres). Mostrare il comportamento dello scheduler, supponendo che il valore iniziale degli oggetti x e y sia per entrambi 500. Indicare, nell'ordine, le operazioni che vengono eseguite da ciascun client, specificando, per ciascuna, il valore che viene letto o scritto.

Basi di dati II — 27 maggio 2019 — Compito D

Domanda 2 (25%)
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)?
Quale decisione può prendere il coordinatore?

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

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

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

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

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

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

Domanda 3 (25%)
Considerare i seguenti scenari in cui due client inviano richieste ad un gestore del controllo di concorrenza. Per il secondo scenario si utilizza una notazione intuitiva anche se non ammissibile in Postgres (con le variabili `XX` e `YY`).

`start transaction`
`  isolation level serializable;`
`select * from impiegati;`	

	`start transaction`
	`  isolation level serializable;`
	`select * from impiegati;`

`insert into impiegati`
`  (matricola, cognome, conteggio)`
`  values (101,'Rossi',1);`
`commit`	

	`insert into impiegati`
	`  (matricola, cognome, conteggio)`
	`  values (102,'Bruni',2);`
	`commit`

`start transaction`
`  isolation level serializable;`
`XX = select count(*) from impiegati;`	

	`start transaction`
	`  isolation level serializable;`
	`YY = select count(*) from impiegati;`

`insert into impiegati`
`  (matricola, cognome, conteggio)`
`  values (101,'Rossi', XX+1);`
`commit`	

	`insert into impiegati`
	`  (matricola, cognome, conteggio)`
	`  values (102,'Bruni', YY+1));`
	`commit`

In entrambi i casi, la seconda transazione viene abortita al momento del commit.

- Spiegare brevemente perché

(continua a pagina seguente)

Se invece il livello di isolamento fosse stato `repeatable read`, allora entrambi in entrambi gli scenari le transazioni si sarebbero concluse con accettazione del commit.

- Spiegare brevemente perché

- Commentare i risultati ottenuti nei due scenari e spiegare perché uno dei due va considerato indesiderabile. Allo scopo, mostrare il contenuto della relazione dopo l'inserimento, supponendola vuota all'inizio. Per comprendere meglio il comportamento, tenere presente che l'attributo `conteggio`, in ciascuna ennupla, serve in sostanza ad indicare la cardinalità della relazione subito dopo l'inserimento della ennupla stessa.

In sostanza, si può osservare che, per ciascun livello di isolamento, i due scenari vengono trattati nello stesso modo: con `serializable` vengono rifiutati entrambi, anche se uno dei due è accettabile, mentre con `repeatable read` vengono accettati entrambi, anche se uno dei due è indesiderabile.

- Spiegare brevemente perché

Basi di dati II — 27 maggio 2019 — 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 $T = 50.000$ ennuple, con valori della chiave tutti diversi fra loro e da quelli già nella relazione (quindi il sistema verifica il soddisfacimento del vincolo di chiave e ammette tutte le operazioni).
Rispondere alle domande seguenti, indicando formule e valori numerici:

Indicare il numero di scritture in memoria secondaria necessarie per realizzare i 50.000 inserimenti, supponendo che i record di log abbiano una lunghezza pari a circa il triplo di quella dei record del file, con riferimento ad un programma che utilizzi una transazione separata per ciascun inserimento

- numero di scritture di pagine di log:

- numero di scritture di pagine della relazione, assumendo che il sistema utilizzi una strategia undo-redo senza vincoli particolari

Come nel caso precedente, ma con riferimento ad un programma che, per realizzare i 50.000 inserimenti, utilizzi complessivamente $k = 10$ transazioni, ognuna con 5000 inserimenti (e supponendo che non vi siano altre transazioni attive)

- numero di scritture di pagine di log:

- numero di scritture di pagine della relazione, sempre assumendo una strategia undo-redo senza vincoli particolari

Basi di dati II — Prova parziale — 27 maggio 2019 — Compito A

Cenni sulle soluzioni

Tempo a disposizione: un'ora.

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

Basi di dati II — 27 maggio 2019 — Compito A

Domanda 1 (25%)
Considerare il seguente scenario in cui due client 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).

`start transaction`
`  isolation level read committed;`
`read(x);`
`x = x + 200;`
`write(x);`
 	
`read(y);`
`y = y + 100;`
`write(y);`
`commit`
	`start transaction`
	`  isolation level read committed;`
	`read(y);`
	`y = y + 100;`
	`write(y)`
	`read(x);`
	`x = x + 100;`
	`write(x);`
	`commit`

Considerare uno scheduler con controllo di concorrenza basato su **Multversioni** (come in Postgres). Mostrare il comportamento dello scheduler, supponendo che il valore iniziale degli oggetti x e y sia per entrambi 300. Indicare, nell'ordine, le operazioni che vengono eseguite da ciascun client, specificando, per ciascuna, il valore che viene letto o scritto.

Con entrambi i livello di isolamento, si genera uno stallo (che in Postgres porta all'abort della prima transazione, mentre qui si assume che abortisca la seconda; comunque, dal punto di vista applicativo, non si hanno anomalie perché una transazione viene completata e l'altra abortita; se, come ipotizzato, quella abortita viene rilanciata, tutto funziona)

Basi di dati II — 27 maggio 2019 — Compito A

Domanda 2 (25%)
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

Domanda 3 (25%)
Considerare i seguenti scenari in cui due client inviano richieste ad un gestore del controllo di concorrenza. Per il secondo scenario si utilizza una notazione intuitiva anche se non ammissibile in Postgres (con le variabili `XX` e `YY`).

`start transaction`
`  isolation level serializable;`
`select * from impiegati;`	

	`start transaction`
	`  isolation level serializable;`
	`select * from impiegati;`

`insert into impiegati`
`  (matricola, cognome, conteggio)`
`  values (101,'Rossi',1);`
`commit`	

	`insert into impiegati`
	`  (matricola, cognome, conteggio)`
	`  values (102,'Bruni',2);`
	`commit`

`start transaction`
`  isolation level serializable;`
`XX = select count(*) from impiegati;`	

	`start transaction`
	`  isolation level serializable;`
	`YY = select count(*) from impiegati;`

`insert into impiegati`
`  (matricola, cognome, conteggio)`
`  values (101,'Rossi', XX+1);`
`commit`	

	`insert into impiegati`
	`  (matricola, cognome, conteggio)`
	`  values (102,'Bruni', YY+1));`
	`commit`

In entrambi i casi, la seconda transazione viene abortita al momento del commit.

- Spiegare brevemente perché

  Possibile risposta: con `serializable`, viene rilevato un ciclo fra le letture e scritture delle due transazioni, sugli stessi dati. Nota bene: l'inserimento fantasma non è rilevante (e anche con il tipico esempio: lettura-inserimento-lettura, tutto funzionerebbe, perché verrebbe letta la versione a inizio transazione)

(continua a pagina seguente)

Se invece il livello di isolamento fosse stato `repeatable read`, allora entrambi in entrambi gli scenari le transazioni si sarebbero concluse con accettazione del commit.

- Spiegare brevemente perché
  con `repeatable read`, non vengono cercati i cicli, ma solo gli aggiornamenti sui dati scritti da ciascuna transazione, che sono in questo caso diversi; non si fa alcun riferimento alle letture
- Commentare i risultati ottenuti nei due scenari e spiegare perché uno dei due va considerato indesiderabile. Allo scopo, mostrare il contenuto della relazione dopo l'inserimento, supponendola vuota all'inizio. Per comprendere meglio il comportamento, tenere presente che l'attributo `conteggio`, in ciascuna ennupla, serve in sostanza ad indicare la cardinalità della relazione subito dopo l'inserimento della ennupla stessa.

  Nel primo scenario, conteggio viene inserito "a mano" e quindi i valori sono rispettivamente 1 e 2, cosa corretta (nell'esempio banale): in effetti, letture e scritture non interferiscono, perchè il valore della lettura non viene utilizzato. Nel secondo scenario, la concorrenza mal gestita fa inserire in entrambi i casi 1: le letture e le scritture interferiscono realmente, in quanto ciascuna scrittura scrive un dato calcolato a partire dalla corrispondente lettura

In sostanza, si può osservare che, per ciascun livello di isolamento, i due scenari vengono trattati nello stesso modo: con `serializable` vengono rifiutati entrambi, anche se uno dei due è accettabile, mentre con `repeatable read` vengono accettati entrambi, anche se uno dei due è indesiderabile.

- Spiegare brevemente perché

  Il controllo di concorrenza ignora la "semantica delle operazioni": sa solo che ci sono letture e scritture, ma non sa se le letture influenzano le scritture, cosa che avviene in uno scenario (che quindi andrebbe evitato) ma non nell'altro (che non crea problemi).

Basi di dati II — 27 maggio 2019 — 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 in memoria secondaria necessarie per realizzare i 25.000 inserimenti, supponendo che i record di log abbiano una lunghezza pari a circa il triplo di quella dei record del file, con riferimento ad un programma che utilizzi una transazione separata per ciascun inserimento

- numero di scritture di pagine di log:
  almeno $N = 25.000$, una per transazione
- numero di scritture di pagine della relazione, assumendo che il sistema utilizzi una 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 $f = D/L = 400$ nei compiti A e C e 200 nei compiti B e D)

Come nel caso precedente, ma con riferimento ad un programma che, per realizzare i 25.000 inserimenti, utilizzi complessivamente $k = 5$ transazioni, ognuna con 5000 inserimenti (e supponendo che non vi siano altre transazioni attive)

- numero di scritture di pagine di log:
  per ogni transazione si debbono scrivere le pagine di log corrispondenti. Ogni transazione scrive N/k ennuple e quindi le relative scritture su log occupano $N/k \times L \times 3 = 300$KB e cioè $N/k \times L \times 3 * 1/D =$ ca.75 blocchi. In totale, per $k = 5$ transazioni, circa $N \times L \times 3 * 1/D =$ ca.375 scritture
- numero di scritture di pagine della relazione, sempre assumendo una strategia undo-redo senza vincoli particolari non si può dire con precisione, anche solo $N/f = 125$, una per blocco

Basi di dati II — Prova parziale — 27 maggio 2019 — Compito B

Cenni sulle soluzioni

Tempo a disposizione: un'ora.

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

Basi di dati II — 27 maggio 2019 — Compito B

Domanda 1 (25%)
Considerare il seguente scenario in cui due client 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).

`start transaction`
`  isolation level serializable;`
`read(x);`
`x = x + 300;`
`write(x);`
 	
`read(y);`
`y = y + 100;`
`write(y);`
`commit`
	
	`start transaction`
	`  isolation level serializable;`
	`read(y);`
	`y = y + 100;`
	`write(y)`
	`read(x);`
	`x = x + 100;`
	`write(x);`
	`commit`

Considerare uno scheduler con controllo di concorrenza basato su **Multversioni** (come in Postgres). Mostrare il comportamento dello scheduler, supponendo che il valore iniziale degli oggetti x e y sia per entrambi 100. Indicare, nell'ordine, le operazioni che vengono eseguite da ciascun client, specificando, per ciascuna, il valore che viene letto o scritto.

Con entrambi i livello di isolamento, si genera uno stallo (che in Postgres porta all'abort della prima transazione, mentre qui si assume che abortisca la seconda; comunque, dal punto di vista applicativo, non si hanno anomalie perché una transazione viene completata e l'altra abortita; se, come ipotizzato, quella abortita viene rilanciata, tutto funziona)

Basi di dati II — 27 maggio 2019 — Compito B

Domanda 2 (25%)
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

Domanda 3 (25%)
Considerare i seguenti scenari in cui due client inviano richieste ad un gestore del controllo di concorrenza. Per il secondo scenario si utilizza una notazione intuitiva anche se non ammissibile in Postgres (con le variabili `XX` e `YY`).

`start transaction`
`  isolation level serializable;`
`select * from impiegati;`	

	`start transaction`
	`  isolation level serializable;`
	`select * from impiegati;`

`insert into impiegati`
`  (matricola, cognome, conteggio)`
`  values (101,'Rossi',1);`
`commit`	

	`insert into impiegati`
	`  (matricola, cognome, conteggio)`
	`  values (102,'Bruni',2);`
	`commit`

`start transaction`
`  isolation level serializable;`
`XX = select count(*) from impiegati;`	

	`start transaction`
	`  isolation level serializable;`
	`YY = select count(*) from impiegati;`

`insert into impiegati`
`  (matricola, cognome, conteggio)`
`  values (101,'Rossi', XX+1);`
`commit`	

	`insert into impiegati`
	`  (matricola, cognome, conteggio)`
	`  values (102,'Bruni', YY+1));`
	`commit`

In entrambi i casi, la seconda transazione viene abortita al momento del commit.

- Spiegare brevemente perché

  Possibile risposta: con `serializable`, viene rilevato un ciclo fra le letture e scritture delle due transazioni, sugli stessi dati. Nota bene: l'inserimento fantasma non è rilevante (e anche con il tipico esempio: lettura-inserimento-lettura, tutto funzionerebbe, perché verrebbe letta la versione a inizio transazione)

(continua a pagina seguente)

Se invece il livello di isolamento fosse stato `repeatable read`, allora entrambi in entrambi gli scenari le transazioni si sarebbero concluse con accettazione del commit.

- Spiegare brevemente perché
  con `repeatable read`, non vengono cercati i cicli, ma solo gli aggiornamenti sui dati scritti da ciascuna transazione, che sono in questo caso diversi; non si fa alcun riferimento alle letture
- Commentare i risultati ottenuti nei due scenari e spiegare perché uno dei due va considerato indesiderabile. Allo scopo, mostrare il contenuto della relazione dopo l'inserimento, supponendola vuota all'inizio. Per comprendere meglio il comportamento, tenere presente che l'attributo `conteggio`, in ciascuna ennupla, serve in sostanza ad indicare la cardinalità della relazione subito dopo l'inserimento della ennupla stessa.

  Nel primo scenario, conteggio viene inserito "a mano" e quindi i valori sono rispettivamente 1 e 2, cosa corretta (nell'esempio banale): in effetti, letture e scritture non interferiscono, perchè il valore della lettura non viene utilizzato. Nel secondo scenario, la concorrenza mal gestita fa inserire in entrambi i casi 1: le letture e le scritture interferiscono realmente, in quanto ciascuna scrittura scrive un dato calcolato a partire dalla corrispondente lettura

In sostanza, si può osservare che, per ciascun livello di isolamento, i due scenari vengono trattati nello stesso modo: con `serializable` vengono rifiutati entrambi, anche se uno dei due è accettabile, mentre con `repeatable read` vengono accettati entrambi, anche se uno dei due è indesiderabile.

- Spiegare brevemente perché

  Il controllo di concorrenza ignora la "semantica delle operazioni": sa solo che ci sono letture e scritture, ma non sa se le letture influenzano le scritture, cosa che avviene in uno scenario (che quindi andrebbe evitato) ma non nell'altro (che non crea problemi).

Basi di dati II — 27 maggio 2019 — 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 $R = 50.000$ ennuple, con valori della chiave tutti diversi fra loro e da quelli già nella relazione (quindi il sistema verifica il soddisfacimento del vincolo di chiave e ammette tutte le operazioni).
Rispondere alle domande seguenti, indicando formule e valori numerici:

Indicare il numero di scritture in memoria secondaria necessarie per realizzare i 50.000 inserimenti, supponendo che i record di log abbiano una lunghezza pari a circa il triplo di quella dei record del file, con riferimento ad un programma che utilizzi una transazione separata per ciascun inserimento

- numero di scritture di pagine di log:
  almeno $R = 50.000$, una per transazione
- numero di scritture di pagine della relazione, assumendo che il sistema utilizzi una 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 $f = D/L = 400$ nei compiti A e C e 200 nei compiti B e D)

Come nel caso precedente, ma con riferimento ad un programma che, per realizzare i 50.000 inserimenti, utilizzi complessivamente $k = 5$ transazioni, ognuna con 10.000 inserimenti (e supponendo che non vi siano altre transazioni attive)

- numero di scritture di pagine di log:
  per ogni transazione si debbono scrivere le pagine di log corrispondenti. Ogni transazione scrive R/k ennuple e quindi le relative scritture su log occupano $R/k \times L \times 3 = 300$KB e cioè $R/k \times L \times 3 * 1/D =$ ca.75 blocchi. In totale, per $k = 5$ transazioni, circa $R \times L \times 3 * 1/D =$ ca.375 scritture
- numero di scritture di pagine della relazione, sempre assumendo una strategia undo-redo senza vincoli particolari non si può dire con precisione, anche solo $R/f = 125$, una per blocco

Basi di dati II — Prova parziale — 27 maggio 2019 — Compito C

Cenni sulle soluzioni

Tempo a disposizione: un'ora.

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

Basi di dati II — 27 maggio 2019 — Compito C

Domanda 1 (25%)
Considerare il seguente scenario in cui due client 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).

`start transaction`
`  isolation level read committed;`
`read(x);`
`x = x + 400;`
`write(x);`
 	
`read(y);`
`y = y + 100;`
`write(y);`
`commit`
	`start transaction`
	`  isolation level read committed;`
	`read(y);`
	`y = y + 100;`
	`write(y)`
	`read(x);`
	`x = x + 100;`
	`write(x);`
	`commit`

Considerare uno scheduler con controllo di concorrenza basato su **Multversioni** (come in Postgres). Mostrare il comportamento dello scheduler, supponendo che il valore iniziale degli oggetti x e y sia per entrambi 200. Indicare, nell'ordine, le operazioni che vengono eseguite da ciascun client, specificando, per ciascuna, il valore che viene letto o scritto.

Con entrambi i livello di isolamento, si genera uno stallo (che in Postgres porta all'abort della prima transazione, mentre qui si assume che abortisca la seconda; comunque, dal punto di vista applicativo, non si hanno anomalie perché una transazione viene completata e l'altra abortita; se, come ipotizzato, quella abortita viene rilanciata, tutto funziona)

Basi di dati II — 27 maggio 2019 — Compito C

Domanda 2 (25%)
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

Domanda 3 (25%)
Considerare i seguenti scenari in cui due client inviano richieste ad un gestore del controllo di concorrenza. Per il secondo scenario si utilizza una notazione intuitiva anche se non ammissibile in Postgres (con le variabili `XX` e `YY`).

```
start transaction
  isolation level serializable;
select * from impiegati;
```

```
start transaction
  isolation level serializable;
select * from impiegati;
```

```
insert into impiegati
  (matricola, cognome, conteggio)
  values (101,'Rossi',1);
commit
```

```
insert into impiegati
  (matricola, cognome, conteggio)
  values (102,'Bruni',2);
commit
```

```
start transaction
  isolation level serializable;
XX = select count(*) from impiegati;
```

```
start transaction
  isolation level serializable;
YY = select count(*) from impiegati;
```

```
insert into impiegati
  (matricola, cognome, conteggio)
  values (101,'Rossi', XX+1);
commit
```

```
insert into impiegati
  (matricola, cognome, conteggio)
  values (102,'Bruni', YY+1));
commit
```

In entrambi i casi, la seconda transazione viene abortita al momento del commit.

- Spiegare brevemente perché

  Possibile risposta: con `serializable`, viene rilevato un ciclo fra le letture e scritture delle due transazioni, sugli stessi dati. Nota bene: l'inserimento fantasma non è rilevante (e anche con il tipico esempio: lettura-inserimento-lettura, tutto funzionerebbe, perché verrebbe letta la versione a inizio transazione)

(continua a pagina seguente)

Se invece il livello di isolamento fosse stato `repeatable read`, allora entrambi in entrambi gli scenari le transazioni si sarebbero concluse con accettazione del commit.

- Spiegare brevemente perché
  con `repeatable read`, non vengono cercati i cicli, ma solo gli aggiornamenti sui dati scritti da ciascuna transazione, che sono in questo caso diversi; non si fa alcun riferimento alle letture
- Commentare i risultati ottenuti nei due scenari e spiegare perché uno dei due va considerato indesiderabile. Allo scopo, mostrare il contenuto della relazione dopo l'inserimento, supponendola vuota all'inizio. Per comprendere meglio il comportamento, tenere presente che l'attributo `conteggio`, in ciascuna ennupla, serve in sostanza ad indicare la cardinalità della relazione subito dopo l'inserimento della ennupla stessa.

  Nel primo scenario, conteggio viene inserito "a mano" e quindi i valori sono rispettivamente 1 e 2, cosa corretta (nell'esempio banale): in effetti, letture e scritture non interferiscono, perchè il valore della lettura non viene utilizzato. Nel secondo scenario, la concorrenza mal gestita fa inserire in entrambi i casi 1: le letture e le scritture interferiscono realmente, in quanto ciascuna scrittura scrive un dato calcolato a partire dalla corrispondente lettura

In sostanza, si può osservare che, per ciascun livello di isolamento, i due scenari vengono trattati nello stesso modo: con `serializable` vengono rifiutati entrambi, anche se uno dei due è accettabile, mentre con `repeatable read` vengono accettati entrambi, anche se uno dei due è indesiderabile.

- Spiegare brevemente perché

  Il controllo di concorrenza ignora la "semantica delle operazioni": sa solo che ci sono letture e scritture, ma non sa se le letture influenzano le scritture, cosa che avviene in uno scenario (che quindi andrebbe evitato) ma non nell'altro (che non crea problemi).

Basi di dati II — 27 maggio 2019 — 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 in memoria secondaria necessarie per realizzare i 100.000 inserimenti, supponendo che i record di log abbiano una lunghezza pari a circa il triplo di quella dei record del file, con riferimento ad un programma che utilizzi una transazione separata per ciascun inserimento

- numero di scritture di pagine di log:
  almeno $M = 100.000$, una per transazione
- numero di scritture di pagine della relazione, assumendo che il sistema utilizzi una 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 $f = D/L = 400$ nei compiti A e C e 200 nei compiti B e D)

Come nel caso precedente, ma con riferimento ad un programma che, per realizzare i 100.000 inserimenti, utilizzi complessivamente $k = 10$ transazioni, ognuna con 10.000 inserimenti (e supponendo che non vi siano altre transazioni attive)

- numero di scritture di pagine di log:
  per ogni transazione si debbono scrivere le pagine di log corrispondenti. Ogni transazione scrive M/k ennuple e quindi le relative scritture su log occupano $M/k \times L \times 3 = 600$KB e cioè $M/k \times L \times 3 * 1/D =$ ca.150 blocchi. In totale, per $k = 10$ transazioni, circa $M \times L \times 3 * 1/D =$ ca.1500 scritture
- numero di scritture di pagine della relazione, sempre assumendo una strategia undo-redo senza vincoli particolari non si può dire con precisione, anche solo $M/f = 500$, una per blocco

Basi di dati II — Prova parziale — 27 maggio 2019 — Compito D

Cenni sulle soluzioni

Tempo a disposizione: un'ora.

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

Basi di dati II — 27 maggio 2019 — Compito D

Domanda 1 (25%)
Considerare il seguente scenario in cui due client 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).

`start transaction`
`  isolation level serializable;`
`read(x);`
`x = x + 600;`
`write(x);`
 	
`read(y);`
`y = y + 100;`
`write(y);`
`commit`
	
	`start transaction`
	`  isolation level serializable;`
	`read(y);`
	`y = y + 100;`
	`write(y)`
	`read(x);`
	`x = x + 100;`
	`write(x);`
	`commit`

Considerare uno scheduler con controllo di concorrenza basato su **Multversioni** (come in Postgres). Mostrare il comportamento dello scheduler, supponendo che il valore iniziale degli oggetti x e y sia per entrambi 500. Indicare, nell'ordine, le operazioni che vengono eseguite da ciascun client, specificando, per ciascuna, il valore che viene letto o scritto.

Con entrambi i livello di isolamento, si genera uno stallo (che in Postgres porta all'abort della prima transazione, mentre qui si assume che abortisca la seconda; comunque, dal punto di vista applicativo, non si hanno anomalie perché una transazione viene completata e l'altra abortita; se, come ipotizzato, quella abortita viene rilanciata, tutto funziona)

Basi di dati II — 27 maggio 2019 — Compito D

Domanda 2 (25%)
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

Domanda 3 (25%)
Considerare i seguenti scenari in cui due client inviano richieste ad un gestore del controllo di concorrenza. Per il secondo scenario si utilizza una notazione intuitiva anche se non ammissibile in Postgres (con le variabili `XX` e `YY`).

`start transaction`
`  isolation level serializable;`
`select * from impiegati;`	

	`start transaction`
	`  isolation level serializable;`
	`select * from impiegati;`

`insert into impiegati`
`  (matricola, cognome, conteggio)`
`  values (101,'Rossi',1);`
`commit`	

	`insert into impiegati`
	`  (matricola, cognome, conteggio)`
	`  values (102,'Bruni',2);`
	`commit`

`start transaction`
`  isolation level serializable;`
`XX = select count(*) from impiegati;`	

	`start transaction`
	`  isolation level serializable;`
	`YY = select count(*) from impiegati;`

`insert into impiegati`
`  (matricola, cognome, conteggio)`
`  values (101,'Rossi', XX+1);`
`commit`	

	`insert into impiegati`
	`  (matricola, cognome, conteggio)`
	`  values (102,'Bruni', YY+1));`
	`commit`

In entrambi i casi, la seconda transazione viene abortita al momento del commit.

- Spiegare brevemente perché

  Possibile risposta: con `serializable`, viene rilevato un ciclo fra le letture e scritture delle due transazioni, sugli stessi dati. Nota bene: l'inserimento fantasma non è rilevante (e anche con il tipico esempio: lettura-inserimento-lettura, tutto funzionerebbe, perché verrebbe letta la versione a inizio transazione)

(continua a pagina seguente)

Se invece il livello di isolamento fosse stato `repeatable read`, allora entrambi in entrambi gli scenari le transazioni si sarebbero concluse con accettazione del commit.

- Spiegare brevemente perché
  con `repeatable read`, non vengono cercati i cicli, ma solo gli aggiornamenti sui dati scritti da ciascuna transazione, che sono in questo caso diversi; non si fa alcun riferimento alle letture
- Commentare i risultati ottenuti nei due scenari e spiegare perché uno dei due va considerato indesiderabile. Allo scopo, mostrare il contenuto della relazione dopo l'inserimento, supponendola vuota all'inizio. Per comprendere meglio il comportamento, tenere presente che l'attributo `conteggio`, in ciascuna ennupla, serve in sostanza ad indicare la cardinalità della relazione subito dopo l'inserimento della ennupla stessa.

  Nel primo scenario, conteggio viene inserito "a mano" e quindi i valori sono rispettivamente 1 e 2, cosa corretta (nell'esempio banale): in effetti, letture e scritture non interferiscono, perchè il valore della lettura non viene utilizzato. Nel secondo scenario, la concorrenza mal gestita fa inserire in entrambi i casi 1: le letture e le scritture interferiscono realmente, in quanto ciascuna scrittura scrive un dato calcolato a partire dalla corrispondente lettura

In sostanza, si può osservare che, per ciascun livello di isolamento, i due scenari vengono trattati nello stesso modo: con `serializable` vengono rifiutati entrambi, anche se uno dei due è accettabile, mentre con `repeatable read` vengono accettati entrambi, anche se uno dei due è indesiderabile.

- Spiegare brevemente perché

  Il controllo di concorrenza ignora la "semantica delle operazioni": sa solo che ci sono letture e scritture, ma non sa se le letture influenzano le scritture, cosa che avviene in uno scenario (che quindi andrebbe evitato) ma non nell'altro (che non crea problemi).

Basi di dati II — 27 maggio 2019 — 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 $T = 50.000$ ennuple, con valori della chiave tutti diversi fra loro e da quelli già nella relazione (quindi il sistema verifica il soddisfacimento del vincolo di chiave e ammette tutte le operazioni).
Rispondere alle domande seguenti, indicando formule e valori numerici:

Indicare il numero di scritture in memoria secondaria necessarie per realizzare i 50.000 inserimenti, supponendo che i record di log abbiano una lunghezza pari a circa il triplo di quella dei record del file, con riferimento ad un programma che utilizzi una transazione separata per ciascun inserimento

- numero di scritture di pagine di log:
  almeno $T = 50.000$, una per transazione
- numero di scritture di pagine della relazione, assumendo che il sistema utilizzi una strategia undo-redo senza vincoli particolari : al massimo una per transazione, $T = 50.000$; probabilmente di meno, anche solo $T/f = 125$, una per blocco (dove f indica il fattore di blocco $f = D/L = 400$ nei compiti A e C e 200 nei compiti B e D)

Come nel caso precedente, ma con riferimento ad un programma che, per realizzare i 50.000 inserimenti, utilizzi complessivamente $k = 10$ transazioni, ognuna con 5000 inserimenti (e supponendo che non vi siano altre transazioni attive)

- numero di scritture di pagine di log:
  per ogni transazione si debbono scrivere le pagine di log corrispondenti. Ogni transazione scrive T/k ennuple e quindi le relative scritture su log occupano $T/k \times L \times 3 = 300$KB e cioè $T/k \times L \times 3 * 1/D =$ ca.38 blocchi. In totale, per $k = 10$ transazioni, circa $T \times L \times 3 * 1/D =$ ca.380 scritture
- numero di scritture di pagine della relazione, sempre assumendo una strategia undo-redo senza vincoli particolari non si può dire con precisione, anche solo $T/f = 125$, una per blocco