Tecnologie e architetture per la gestione dei dati

Prova parziale — 19 maggio 2022

Tempo a disposizione: un'ora e venti minuti.

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

Tecnologie e architetture per la gestione dei dati — 19 maggio 2022

Domanda 1 (40%)
Considerare il seguente scenario in cui due 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 e che, in caso di abort, il client rilanci la stessa transazione (subito dopo l'esecuzione delle altre azioni in attesa sullo stesso dato).

`start transaction isolation level ....  ;`
`read(x) ;`
 	
`read(y);`
`y = y + x + 1`
`write(y);`
`commit;`
	
	`start transaction isolation level ....;`
	`read(y);`
	
	`read(x);`
	`x = x + y + 100;`
	`write(x)`
	`commit;`

A. Considerare uno scheduler con controllo di concorrenza basato su **Multiversioni** (come in Postgres) e livello di isolamento **SERIALIZABLE** su entrambe le transazioni.
Mostrare il comportamento dello scheduler, supponendo che i valori iniziali degli oggetti x e y siano entrambi **1000**. Indicare, nell'ordine, le operazioni che vengono eseguite da ciascun client, specificando, per ciascuna, il valore che viene letto o scritto. Per le transazioni rilanciate dopo abort, non è necessario scrivere tutti i dettagli, ma è sufficiente indicare sinteticamente le azioni e il risultato delle scritture.
In conclusione, dire se si verificano o meno anomalie.

client 1	client 2
Si verificano anomalie?(sì o no e in caso affermativo quale)	

Considerare nuovamente lo scenario della domanda precedente, con riferimento ai contesti diversi mostrati nella pagina seguente e rispondere a quanto richiesto sopra.

Tecnologie e architetture per la gestione dei dati — 19 maggio 2022

B. Controllo di concorrenza **Multiversioni** (come in Postgres) e livello **REPEATABLE READ**

client 1	client 2
Si verificano anomalie?(sì o no e in caso affermativo quale)	

C. Controllo di concorrenza basato su **2PL** e livello di isolamento **SERIALIZABLE**

client 1	client 2
Si verificano anomalie?(sì o no e in caso affermativo quale)	

D.Controllo di concorrenza basato su **2PL** e livello di isolamento **REPEATABLE READ**

client 1	client 2
Si verificano anomalie?(sì o no e in caso affermativo quale)	

Tecnologie e architetture per la gestione dei dati — 19 maggio 2022

Domanda 2 (20%)
Considerare un sistema distribuito con tre nodi A, B e C, che eseguono due transazioni T_1 e T_2 che coinvolgono i tre nodi, in modo diverso. Per la prima transazione A è il coordinatore, mentre per la seconda il coordinatore è B. I due coordinatori inviano, come riportato nello schema sottostante, le richieste di `prepare`. Il nodo C va in crash subito dopo aver risposto alla prima richiesta (senza avere il tempo di ricevere il messaggio di `commit`) e prima di ricevere la seconda. Poi va in crash anche il nodo A. Indicare, nello schema sottostante, una possibile sequenza di scritture sui log e invio di messaggi (che includa anche i passi sopra illustrati), supponendo che entrambi i nodi siano ripristinati abbastanza presto (ma che vengano persi alcuni messaggi di risposta, ad esempio inviati a seguito di una decisione). Per i messaggi si usi la notazione *tipo(transaz)→destinatari* (come nell'esempio: `prepare(`T_1`)→B,C`). Supporre che nel log del coordinatore si scrivano solo i record di `prepare`, `commit` e `complete`, con i messaggi gestiti invece in memoria. Indicare ragionevoli istanti per i timeout, che permettano di concludere il protocollo per entrambe le transazioni.

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

Tecnologie e architetture per la gestione dei dati — 19 maggio 2022

Domanda 3 (20%) 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)?
Perché?

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)?
Perché?

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)?
Perché?

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)?
Perché?

Quale decisione può prendere il coordinatore?

Tecnologie e architetture per la gestione dei dati — 19 maggio 2022

Domanda 4 (20%)
Considerare un sistema che utilizzi blocchi di lunghezza $D = 4$ KB (approssimabili a 4000 byte) e una tabella R con una struttura fisica heap con record a lunghezza fissa che occupano $L = 20$ byte ciascuno, in cui vengono inserite $N = 25.000$ ennuple, con valori della chiave tutti diversi fra loro e da quelli già nella relazione (quindi il sistema verifica il soddisfacimento del vincolo di chiave e ammette tutte le operazioni).
Rispondere alle domande seguenti, indicando formule e valori numerici:

Indicare il numero di scritture in memoria secondaria necessarie per realizzare i 25.000 inserimenti, supponendo che i record di log abbiano una lunghezza pari a circa il triplo di quella dei record del file, con riferimento ad un programma che utilizzi una transazione separata per ciascun inserimento

- numero di scritture di pagine di log:

- numero di scritture di pagine della relazione, nei tre casi seguenti:
  - strategia undo-redo senza vincoli particolari
  - strategia redo-only (no-undo)
  - strategia undo-only (no-redo)

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

- numero di scritture di pagine di log:

- numero di scritture di pagine della relazione, nei tre casi seguenti:
  - strategia undo-redo senza vincoli particolari
  - strategia redo-only (no-undo)
  - strategia undo-only (no-redo)

Tecnologie e architetture per la gestione dei dati

Prova parziale — 19 maggio 2022

Cenni sulle soluzioni

Tempo a disposizione: un'ora e venti minuti.

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

Tecnologie e architetture per la gestione dei dati — 19 maggio 2022

Domanda 1 (40%)
Considerare il seguente scenario in cui due 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 e che, in caso di abort, il client rilanci la stessa transazione (subito dopo l'esecuzione delle altre azioni in attesa sullo stesso dato).

`start transaction isolation level ....  ;`
`read(x) ;`
 	
`read(y);`
`y = y + x + 1`
`write(y);`
`commit;`
	
	`start transaction isolation level ....;`
	`read(y);`
	
	`read(x);`
	`x = x + y + 100;`
	`write(x)`
	`commit;`

A. Considerare uno scheduler con controllo di concorrenza basato su **Multiversioni** (come in Postgres) e livello di isolamento **SERIALIZABLE** su entrambe le transazioni.
Mostrare il comportamento dello scheduler, supponendo che i valori iniziali degli oggetti x e y siano entrambi **1000**. Indicare, nell'ordine, le operazioni che vengono eseguite da ciascun client, specificando, per ciascuna, il valore che viene letto o scritto. Per le transazioni rilanciate dopo abort, non è necessario scrivere tutti i dettagli, ma è sufficiente indicare sinteticamente le azioni e il risultato delle scritture.
In conclusione, dire se si verificano o meno anomalie.

Risposta/commento:
Le operazioni vengono tutte eseguite nell'ordine in cui vengono richieste, fino alla scrittura da parte del secondo client, che viene rifiutata, per la presenza del ciclo lettura scrittura (la transazione 2 scrive x dopo che la 1 lo ha letto e la transazione 1 scrive y dopo che la 2 lo ha letto). Il sistema, alla richiesta di scrittura, mostra il messaggio:

```
ERROR: could not serialize access due to read/write dependencies among transactions
DETAIL: Reason code:  Canceled on identification as a pivot, during write.
```

Se poi la transazione viene rilanciata, procede regolarmente

Considerare nuovamente lo scenario della domanda precedente, con riferimento ai contesti diversi mostrati nella pagina seguente e rispondere a quanto richiesto sopra.

B. Controllo di concorrenza **Multiversioni** (come in Postgres) e livello **REPEATABLE READ**

Risposta/commento:
Le operazioni vengono tutte eseguite nell'ordine in cui vengono richieste. Si verifica una anomalia, perché in effetti il risultato che si ottiene non corrisponde ad alcune esecuzione seriale. Il valore di y che si ottiene è 2001 il che è corretto, mentre per x si ottiene 2100. In una esecuzione seriale, il secondo valore aggiornato dovrebbe essere 3101.

C. Controllo di concorrenza basato su **2PL** e livello di isolamento **SERIALIZABLE**

Risposta/commento:
Si genera uno stallo per le richieste di lock in scrittura, viene abortita la seconda transazione, che poi riparte e tutto si conclude senza anomalie

D.Controllo di concorrenza basato su **2PL** e livello di isolamento **REPEATABLE READ**

Risposta/commento:
Si genera uno stallo per le richieste di lock in scrittura, viene abortita la seconda transazione, che poi riparte e tutto si conclude senza anomalie

Tecnologie e architetture per la gestione dei dati — 19 maggio 2022

Domanda 2 (20%)
Considerare un sistema distribuito con tre nodi A, B e C, che eseguono due transazioni T_1 e T_2 che coinvolgono i tre nodi, in modo diverso. Per la prima transazione A è il coordinatore, mentre per la seconda il coordinatore è B. I due coordinatori inviano, come riportato nello schema sottostante, le richieste di `prepare`. Il nodo C va in crash subito dopo aver risposto alla prima richiesta (senza avere il tempo di ricevere il messaggio di `commit`) e prima di ricevere la seconda. Poi va in crash anche il nodo A. Indicare, nello schema sottostante, una possibile sequenza di scritture sui log e invio di messaggi (che includa anche i passi sopra illustrati), supponendo che entrambi i nodi siano ripristinati abbastanza presto (ma che vengano persi alcuni messaggi di risposta, ad esempio inviati a seguito di una decisione). Per i messaggi si usi la notazione *tipo(transaz)→destinatari* (come nell'esempio: `prepare(T1)→B,C`). Supporre che nel log del coordinatore si scrivano solo i record di `prepare`, `commit` e `complete`, con i messaggi gestiti invece in memoria. Indicare ragionevoli istanti per i timeout, che permettano di concludere il protocollo per entrambe le transazioni.

Nodo A		Nodo B		Nodo C	
Log	Messaggi	Log	Messaggi	Log	Messaggi
prep(T_1,B,C)					
	prep(T_1)→B,C				
		ready(T_1)		ready(T_1)	
			ready(T_1)→A		ready(T_1)→A
				crash	
commit(T_1)					
	commit(T_1)→B,C				
		commit(T_1)			
			ack(T_1)→A		
		prep(T_2,A,C)			
			prep(T_2)→A,C		
ready(T_2)					
	ready(T_2)→B				
crash					
		abort(T_2)			
			abort(T_2)→A,C		
restart					
	commit(T_1)→B,C				
			ack(T_1)→A		
			abort(T_2)→A,C		
abort(T_2)					
	ack(T_2)→B				
				restart	
			abort(T_2)→C		
				abort(T_2)	
					ack(T_2)→B
		complete(T_2)			
	commit(T_1)→C				
				commit(T_1)	
					ack(T_1)→A
complete(T_1)					

Tecnologie e architetture per la gestione dei dati — 19 maggio 2022

Domanda 3 (20%) 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
Perché?
I partecipanti contattati hanno ricevuto tutti e accettato
Quale decisione può prendere il coordinatore?
Nessuna, perché il vecchio coordinatore è anche un partecipante e non sappiamo se ha deciso

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
Perché?
Non può essere stata presa la decisione positiva se qualcuno non ha accettato
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)? sì
Perché?
Qualche partecipante non ha ricevuto o ha ricevuto e non accettato e la decisione non è stata presa
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
Perché?
Alcuni hanno ricevuto il messaggio della seconda fase e gli altri no
Quale decisione può prendere il coordinatore?
Commit

Tecnologie e architetture per la gestione dei dati — 19 maggio 2022

Domanda 4 (20%)
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, 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 $f = D/L = 400$ nei compiti A e C e 200 nei compiti B e D)
  - strategia redo-only (no-undo) : come per la strategia undo-redo
  - 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

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, 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 redo-only (no-undo) per ogni transazione, si debbono scrivere i blocchi coinvolti, se le scritture sono sequenziali, ogni blocco si scrive una volta sola (a parte un po' di sfrido) quindi $N/f = 125$
  - strategia undo-only (no-redo) anche in questo caso, si può pensare che ogni blocco si scriva una volta sola, quindi ancora $N/f = 125$