Tecnologie e architetture per la gestione dei dati

Prova parziale — 10 maggio 2024

Tempo a disposizione: un'ora.

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

Tecnologie e architetture per la gestione dei dati — 10 maggio 2024

Domanda 1 (25%) *Come ausilio a questa domanda tenere presente gli "Spunti sui vincoli di chiave e loro verifica" recentemente pubblicati sul sito del corso e riportati nell'ultima pagina di questo fascicolo (che può essere staccata dal fascicolo stesso, per comodità).*

Considerare una base di dati con la tabella creata e popolata con le seguenti operazioni

```
create table conti (num integer primary key, saldo integer);
insert into conti values (1,0)
```

e i seguenti scenari in cui due client diversi inviano richieste ad un gestore del controllo di concorrenza. In ciascun caso, mostrare il comportamento di uno scheduler con controllo di concorrenza basato su **Multiversioni** (come in Postgres). **Indicare, nell'ordine, le operazioni che vengono eseguite da ciascun client, specificando quando vengono poste in attesa**. In caso di stallo, si supponga che venga abortita la transazione che ha effettuato per prima la richiesta. Le transazioni abortite non vanno rilanciate.

Scenario 1

`start transaction isolation level serializable;`
`insert into conti values (2,0);`
 	
`commit;`
	
	`start transaction isolation level serializable;`
	`insert into conti values (2,1000);`
	
	`commit;`

Risposta

client 1	client 2
Quali ennuple sono contenute nella relazione `conti` alla fine?	

Scenario 2

`start transaction isolation level serializable;`
`insert into conti values (2,0);`
 	
`rollback;`
	
	`start transaction isolation level serializable;`
	`insert into conti values (2,1000);`
	
	`commit;`

Risposta

client 1	client 2
Quali ennuple sono contenute nella relazione `conti` alla fine?	

Tecnologie e architetture per la gestione dei dati — 10 maggio 2024

Scenario 3

Nota bene: la relazione `conti` contiene già una ennupla con valore 1 per la chiave.

`start transaction isolation level serializable;`
`insert into conti values (1,0);`
 	
`commit;`
	
	`start transaction isolation level serializable;`
	`insert into conti values (2,1000);`
	
	`commit;`

Risposta

client 1	client 2
Quali ennuple sono contenute nella relazione `conti` alla fine?	

Scenario 4

`start transaction isolation level serializable;`
`delete from conti where num=1;`
 	
`commit;`
	
	`start transaction isolation level serializable;`
	`insert into conti values (1,1000);`
	
	`commit;`

Risposta

client 1	client 2
Quali ennuple sono contenute nella relazione `conti` alla fine?	

Scenario 5

Uguale allo Scenario 3, ma avendo definito la verifica dei vincoli come "differita"

Risposta

client 1	client 2
Quali ennuple sono contenute nella relazione `conti` alla fine?	

Tecnologie e architetture per la gestione dei dati — 10 maggio 2024

Domanda 2 (25%) Si supponga di dover eseguire un'interrogazione che calcola statistiche su una base di dati (ad esempio: trovare per ogni corso la media dei voti assegnati). Indicare (con un brevissimo commento, nello spazio disponibile) quale livello di isolamento (`READ UNCOMMITTED`, `READ COMMITTED`, `REPEATABLE READ` o `SERIALIZABLE`) si potrebbe scegliere in ciascuno dei seguenti casi

1. per tutti i corsi sono già presenti molti esami e l'interrogazione è eseguita mentre vengono inseriti alcuni esami, comunque pochissimi per corso (rispetto a quelli già presenti); sono accettabili risultati "approssimati"
2. per tutti i corsi sono già presenti molti esami e l'interrogazione è eseguita mentre vengono inseriti alcuni esami, comunque pochissimi per corso (rispetto a quelli già presenti); **non** sono accettabili risultati "approssimati"
3. per tutti i corsi sono già presenti molti esami e l'interrogazione è eseguita mentre vengono corretti (cioè modificati) i voti di alcuni esami, comunque pochissimi per corso (rispetto a quelli già presenti) e senza inserirne di nuovi; **non** sono accettabili risultati "approssimati"
4. per tutti i corsi sono già presenti molti esami e l'interrogazione è eseguita mentre vengono corretti (cioè modificati) i voti di alcuni esami, comunque pochissimi per corso (rispetto a quelli già presenti) e senza inserirne di nuovi; sono accettabili risultati "approssimati"
5. per tutti i corsi sono già presenti pochi esami e l'interrogazione è eseguita mentre vengono corretti (cioè modificati) i voti di molti esami, e senza inserirne di nuovi; sono accettabili risultati "approssimati"

1.	2.	3.	4.	5.

Domanda 3 (25%)
Si consideri una base di dati su cui una applicazione effettua moltissimi inserimenti e aggiornamenti, con operazioni tutte molto semplici ma estremamente numerose. Considerare le due situazioni seguenti:

A. concorrentemente alle operazioni sopra citate non viene eseguita nessun'altra operazione
B. concorrentemente alle operazioni sopra citate vengono eseguite molte piccole interrogazioni (e nessun altro aggiornamento)

Commentare brevemente per ciascuna delle due situazioni quali potrebbero essere i vantaggi e gli svantaggi (con riferimento alle prestazioni in termini di tempo di risposta sia per la normale operatività sia in caso di guasto e conseguente necessità di recovery) delle seguenti due scelte per l'applicazione che esegue inserimenti e aggiornamenti:

i. eseguire ciascuna operazione in una transazione separata
ii. riunire tutte le operazioni in un'unica transazione

	A	B
i.		
ii.		In caso di MV, come sopra, perché le interrogazioni leggono le proprie versioni. In caso di 2PL, l'unica transazione manterrebbe i lock fino al commit, bloccando le interrogazioni

Domanda 4 (25%)
Considerare un sistema distribuito con tre nodi X, Y e Z, che eseguono due transazioni T_1 e T_2 che coinvolgono i tre nodi, in modo diverso. Per la prima transazione X è il coordinatore, mentre per la seconda il coordinatore è Y. I due coordinatori inviano, come riportato nello schema sottostante, le richieste di `prepare`. Il nodo Z va in crash subito dopo aver risposto alla prima richiesta (senza avere il tempo di ricevere il messaggio di `commit`) e prima di ricevere la seconda. Indicare, nello schema sottostante, una possibile sequenza di scritture sui log e invio di messaggi (che includa anche i passi sopra illustrati), supponendo che entrambi i nodi siano ripristinati abbastanza presto (ma che vengano persi alcuni messaggi di risposta, ad esempio inviati a seguito di una decisione). Per i messaggi si usi la notazione *tipo(transaz)→destinatari* (come nell'esempio: `prepare(T1)→Y,Z`). Supporre che nel log del coordinatore si scrivano solo i record di `prepare`, `commit` e `complete`, con i messaggi gestiti invece in memoria. Indicare ragionevoli istanti per i timeout, che permettano di concludere il protocollo per entrambe le transazioni.

Nodo X		Nodo Y		Nodo Z	
Log	Messaggi	Log	Messaggi	Log	Messaggi
`prep(T1,Y,Z)`					
	`prep(T1)→Y,Z`				
				crash	
		`prep(T2,X,Z)`			
			`prep(T2)→X,Z`		
crash					
restart					
				restart	

Tecnologie e architetture per la gestione dei dati

Prova parziale — 10 maggio 2024

Cenni sulle soluzioni

Tempo a disposizione: un'ora.

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

Tecnologie e architetture per la gestione dei dati — 10 maggio 2024

Domanda 1 (25%) *Come ausilio a questa domanda tenere presente gli "Spunti sui vincoli di chiave e loro verifica" recentemente pubblicati sul sito del corso e riportati nell'ultima pagina di questo fascicolo (che può essere staccata dal fascicolo stesso, per comodità).*

Considerare una base di dati con la tabella creata e popolata con le seguenti operazioni

```
create table conti (num integer primary key, saldo integer);
insert into conti values (1,0)
```

e i seguenti scenari in cui due client diversi inviano richieste ad un gestore del controllo di concorrenza. In ciascun caso, mostrare il comportamento di uno scheduler con controllo di concorrenza basato su **Multiversioni** (come in Postgres). **Indicare, nell'ordine, le operazioni che vengono eseguite da ciascun client, specificando quando vengono poste in attesa**. In caso di stallo, si supponga che venga abortita la transazione che ha effettuato per prima la richiesta. Le transazioni abortite non vanno rilanciate.

Scenario 1

`start transaction isolation level serializable;`
`insert into conti values (2,0);`
 	
`commit;`
	
	`start transaction isolation level serializable;`
	`insert into conti values (2,1000);`
	
	`commit;`

Risposta

`start transaction ...` **OK** `insert ... (2,0);` **OK** `commit ...` **OK** ;	 `start transaction ...` **OK** `insert ... (2,1000);` **attesa** **riprende, va in errore (violazione del vincolo di chiave) e non esegue il commit**;

Quali ennuple sono contenute nella relazione `conti` alla fine?
(1,0) e (2,0)

Scenario 2

`start transaction isolation level serializable;`
`insert into conti values (2,0);`
 	
`rollback;`
	
	`start transaction isolation level serializable;`
	`insert into conti values (2,1000);`
	
	`commit;`

Risposta

`start transaction ...` **OK** `insert ... (2,0);` **OK** `rollback ...` **OK** ;	 `start transaction ...` **OK** `insert ... (2,1000);` **attesa** **riprende** `insert ... (2,1000);` **OK** `commit;` **OK**

Quali ennuple sono contenute nella relazione `conti` alla fine?
(1,0) e (2,1000)

Tecnologie e architetture per la gestione dei dati — 10 maggio 2024

Scenario 3

Nota bene: la relazione `conti` contiene già una ennupla con valore 1 per la chiave.

`start transaction isolation level serializable;`
`insert into conti values (1,0);`
 	
`commit;`
	
	`start transaction isolation level serializable;`
	`insert into conti values (2,1000);`
	
	`commit;`

Risposta

`start transaction ...` **OK** `insert ... (1,0);` **va in errore per violazione vincolo di chiave (verifica immediata) e viene abortita**	 `start transaction ...` **OK** `insert ... (2,1000);` **OK** `commit;` **OK**

Quali ennuple sono contenute nella relazione `conti` alla fine?
(1,0) e (2,1000)

Scenario 4

`start transaction isolation level serializable;`
`delete from conti where num=1;`
 	
`commit;`
	
	`start transaction isolation level serializable;`
	`insert into conti values (1,1000);`
	
	`commit;`

Risposta

`start transaction ...` **OK** `delete ...;` **OK** `commit ...` **OK ;**	 `start transaction ...` **OK** `insert ... (1,1000);` **attesa** **riprende** `insert ... (1,1000);` **OK** `commit;` **OK**

Quali ennuple sono contenute nella relazione `conti` alla fine?
(1,1000)

Scenario 5

Uguale allo Scenario 3, ma avendo definito la verifica dei vincoli come "differita"

Risposta

`start transaction ...` **OK** `insert ... (1,0);` **OK** `commit;` **NO — alla richiesta di commit viene verificato il vincolo di chiave (verifica differita) e la transazione viene abortita**	 `start transaction ...` **OK** `insert ... (2,1000);` **OK** `commit;` **OK**

Quali ennuple sono contenute nella relazione `conti` alla fine?
(1,0) e (2,1000)

Tecnologie e architetture per la gestione dei dati — 10 maggio 2024

Domanda 2 (25%) Si supponga di dover eseguire un'interrogazione che calcola statistiche su una base di dati (ad esempio: trovare per ogni corso la media dei voti assegnati). Indicare (con un brevissimo commento, nello spazio disponibile) quale livello di isolamento (`READ UNCOMMITTED`, `READ COMMITTED`, `REPEATABLE READ` o `SERIALIZABLE`) si potrebbe scegliere in ciascuno dei seguenti casi

1. per tutti i corsi sono già presenti molti esami e l'interrogazione è eseguita mentre vengono inseriti alcuni esami, comunque pochissimi per corso (rispetto a quelli già presenti); sono accettabili risultati "approssimati"
2. per tutti i corsi sono già presenti molti esami e l'interrogazione è eseguita mentre vengono inseriti alcuni esami, comunque pochissimi per corso (rispetto a quelli già presenti); **non** sono accettabili risultati "approssimati"
3. per tutti i corsi sono già presenti molti esami e l'interrogazione è eseguita mentre vengono corretti (cioè modificati) i voti di alcuni esami, comunque pochissimi per corso (rispetto a quelli già presenti) e senza inserirne di nuovi; **non** sono accettabili risultati "approssimati"
4. per tutti i corsi sono già presenti molti esami e l'interrogazione è eseguita mentre vengono corretti (cioè modificati) i voti di alcuni esami, comunque pochissimi per corso (rispetto a quelli già presenti) e senza inserirne di nuovi; sono accettabili risultati "approssimati"
5. per tutti i corsi sono già presenti pochi esami e l'interrogazione è eseguita mentre vengono corretti (cioè modificati) i voti di molti esami, e senza inserirne di nuovi; sono accettabili risultati "approssimati"

1.	2.	3.	4.	5.
RC	**S**	**RR**	**RC**	**RR**

Tecnologie e architetture per la gestione dei dati — 10 maggio 2024

Domanda 3 (25%)
Si consideri una base di dati su cui una applicazione effettua moltissimi inserimenti e aggiornamenti, con operazioni tutte molto semplici ma estremamente numerose. Considerare le due situazioni seguenti:

A. concorrentemente alle operazioni sopra citate non viene eseguita nessun'altra operazione
B. concorrentemente alle operazioni sopra citate vengono eseguite molte piccole interrogazioni (e nessun altro aggiornamento)

Commentare brevemente per ciascuna delle due situazioni quali potrebbero essere i vantaggi e gli svantaggi (con riferimento alle prestazioni in termini di tempo di risposta sia per la normale operatività sia in caso di guasto e conseguente necessità di recovery) delle seguenti due scelte per l'applicazione che esegue inserimenti e aggiornamenti:

i. eseguire ciascuna operazione in una transazione separata
ii. riunire tutte le operazioni in un'unica transazione

	A	B
i.	Oneroso nel caso ordinario, perché ci sono molti commit e molte scritture sincrone nel log. Poco oneroso in caso di guasto perché le transazioni andate in commit non vanno ripetute	Come nel caso A per il recovery. Per la concorrenza, in caso di MV, non cambia nulla, i molti inserimenti sono semplici e, ammesso che interferiscano fra loro, lo fanno per poco tempo. Stesso discorso in caso di 2PL.
ii.	Poco oneroso nel caso ordinario, perché non ci sono esigenze di scritture sincrone nel log. Molto oneroso in caso di guasto perché tutto il lavoro svolto va ripetuto	In caso di MV, come sopra, perché le interrogazioni leggono le proprie versioni. In caso di 2PL, l'unica transazione manterrebbe i lock fino al commit, bloccando le interrogazioni

Tecnologie e architetture per la gestione dei dati — 10 maggio 2024

Domanda 4 (25%)
Considerare un sistema distribuito con tre nodi X, Y e Z, che eseguono due transazioni T_1 e T_2 che coinvolgono i tre nodi, in modo diverso. Per la prima transazione X è il coordinatore, mentre per la seconda il coordinatore è Y. I due coordinatori inviano, come riportato nello schema sottostante, le richieste di `prepare`. Il nodo Z va in crash subito dopo aver risposto alla prima richiesta (senza avere il tempo di ricevere il messaggio di `commit`) e prima di ricevere la seconda. Indicare, nello schema sottostante, una possibile sequenza di scritture sui log e invio di messaggi (che includa anche i passi sopra illustrati), supponendo che entrambi i nodi siano ripristinati abbastanza presto (ma che vengano persi alcuni messaggi di risposta, ad esempio inviati a seguito di una decisione). Per i messaggi si usi la notazione *tipo(transaz)→destinatari* (come nell'esempio: `prepare(T1)→Y,Z`). Supporre che nel log del coordinatore si scrivano solo i record di `prepare`, `commit` e `complete`, con i messaggi gestiti invece in memoria. Indicare ragionevoli istanti per i timeout, che permettano di concludere il protocollo per entrambe le transazioni.

Nodo X		Nodo Y		Nodo Z	
Log	Messaggi	Log	Messaggi	Log	Messaggi
prep(T_1,Y,Z)					
	prep(T_1)→Y,Z				
		ready(T_1)		ready(T_1)	
			ready(T_1)→X		ready(T_1)→X
				crash	
commit(T_1)					
	commit(T_1)→Y,Z				
		commit(T_1)			
			ack(T_1)→X		
		prep(T_2,X,Z)			
			prep(T_2)→X,Z		
ready(T_2)					
	ready(T_2)→Y				
crash					
		abort(T_2)			
			abort(T_2)→X,Z		
restart					
	commit(T_1)→Y,Z				
			ack(T_1)→X		
			abort(T_2)→X,Z		
abort(T_2)					
	ack(T_2)→Y				
				restart	
			abort(T_2)→Z		
				abort(T_2)	
					ack(T_2)→Y
		complete(T_2)			
	commit(T_1)→Z				
				commit(T_1)	
					ack(T_1)→X
complete(T_1)					