Basi di dati II — 20 giugno 2014 — Compito B

Domanda 5 (15%) Considerare il seguente scenario in cui tre client diversi inviano richieste ad un gestore del controllo di concorrenza. Ciascun client può inviare una richiesta solo dopo che è stata eseguita o rifiutata la precedente (se invece una richiesta viene bloccata da un lock, allora il client rimane inattivo fino alla concessione o allo scadere del timeout). Si supponga che, in caso di stallo, abortisca la transazione che ha avanzato la richiesta per prima. In caso di abort, si supponga che il client rilanci la stessa transazione (subito dopo l'esecuzione delle altre azioni in attesa sullo stesso dato).

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

Considerare uno scheduler con controllo di concorrenza basato su **2PL** e livello di isolamento **READ COMMITTED** su tutte le transazioni.
Mostrare il comportamento dello scheduler, supponendo che il valore iniziale dell'oggetto x sia 200. Indicare, nell'ordine, le operazioni che vengono eseguite da ciascun client, specificando, per ciascuna, il valore che viene letto o scritto. In conclusione, dire se si verificano o meno anomalie.

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

Considerare uno scheduler con controllo di concorrenza basato su **2PL** e livello di isolamento **READ COMMITTED** su tutte le transazioni.
Mostrare il comportamento dello scheduler, supponendo che il valore iniziale dell'oggetto x sia 200. Indicare, nell'ordine, le operazioni che vengono eseguite da ciascun client, specificando, per ciascuna, il valore che viene letto o scritto. In conclusione, dire se si verificano o meno anomalie.

client 1	client 2	client 3
		rlock(x) read(x) 200 unlock(x)
rlock(x) read(x) 200 unlock(x)		
	rlock(x) read(x) 200 unlock(x)	
wlock(x) write(x) 210 commit unlock(x)		
		rlock(x) read(x) 210 unlock(x)
	wlock(x) write(x) 220 commit unlock(x)	

Basi di dati II

Prova parziale — 15 maggio 2017 — Compito A

Tempo a disposizione: un'ora.

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

Domanda 1 (30%)
Considerare il seguente scenario in cui tre client diversi inviano richieste ad un gestore del controllo di concorrenza. Ciascun client può inviare una richiesta solo dopo che è stata eseguita o rifiutata la precedente (se invece una richiesta viene bloccata da un lock, allora il client rimane inattivo fino alla concessione o allo scadere del timeout). Si supponga che, in caso di stallo, abortisca la transazione che ha avanzato la richiesta per prima. In caso di abort, si supponga che il client rilanci la stessa transazione (subito dopo l'esecuzione delle altre azioni in attesa sullo stesso dato).

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

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

Considerare uno scheduler con controllo di concorrenza basato su **2PL** con livello di isolamento **SERIALIZABLE** sulle prime due transazioni e **READ COMMITTED** sulla terza. Mostrare il comportamento dello scheduler, supponendo che il valore iniziale dell'oggetto x sia ancora **5000**.

client 1	client 2	client 3

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

Considerare uno scheduler con controllo di concorrenza basato su **2PL** con livello di isolamento **SERIALIZABLE** sulle prime due transazioni e **READ COMMITTED** sulla terza. Mostrare il comportamento dello scheduler, supponendo che il valore iniziale dell'oggetto x sia ancora **5000**.

client 1	client 2	client 3
rlock(x) read(x) 5000		rlock(x) read(x) 5000 unlock(x)
	rlock(x) read(x) 5000 wlock(x) --- attesa	
wlock(x) --- attesa		
write(x) 5010 unlock commit		
	abort rlock(x) --- attesa read(x) 5010 wlock(x) write(x) 5030 unlock commit	
		(supponiamo che passi molto tempo) rlock(x) read(x) 5030 unlock(x)