Transakcie - sériovateľnosť

Transakcia = databázový program, ktorý sa musí buď celý vykonať, alebo sa z neho nesmie vykonať nič.

{set transaction
...
commit (rollback);}

Program každého užívateľa je postupnosť transakcií. Pre databázy je charakteristické, že sa viac programov vykonáva súčasne.

Príklad: Rezervácia letenky (Bratislava - Singapore).
(Bratislava - Viedeň - Rím - Kalkata - Singapúr)

Vlastnostitransakci í - ACID

- Atomičnosť (všetko alebo nič).
- Consistency transformuje databázu z konzistentného stavu do konzistentného stavu.
- Nezávislosť (Isolation) činnosť transakcie je neovplyvniteľná činnosťou iných transakcií.
- Trvanlivosť (Durability) výsledky transakcie pretrvajú po jej skončení v databáze.

Príklad:

transaction T_1:{a:= a+2; b:=3×b;}
transaction T_2:{a:= 3×a; b:=b+2;}

T_1T_2 a=3a+6, b=3b+2
T_2T_1 a=3a+2, b=3b+6

Výsledok *S:* a=3a+6, b=3b+6

Rozvrh: *S*

krok	T_1	T_2
1	A	-
2	-	A
3	-	B
4	B	-

Jemnejšie členenie akcií: **read,** compute, **write.**
Kritické akcie read a write.

Rozbitie kritických akcií:
operačný systém: **lock** a; read a; **unlock** a;
lock a; write a; **unlock** a;

Deadlocka livelock - opakovanie z operačných systémov

Deadlock - niekoľko transakcií sa dostane do stavu, že žiadna z nich nemôže pokračovať.

Livelock - (starvation) stav, keď sa transakcia nikdy nedostane k uskutočneniu ďalšej (aj prvej) akcie.

Zisťovanie - recovery - prevencia

Graf čakania: Uzly grafu sú „aktívne" transakcie, hrana z T_1 do T_2 práve vtedy, keď T_1 čaká na prostriedok pridelený T_2.

Veta 1: *Cyklus grafu čakania = transakcie v deadlocku.*

Livelock sa nedá zistiť - prakticky: keď transakcia čaká vo fronte, alebo „trčí v spracovaní" dlhšie než stanovený limit.

Sériovateľnosť

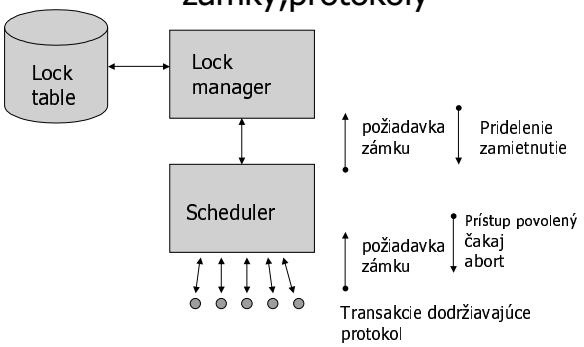
Def: Rozvrh (schedule) je sériovateľný ak je „ekvivalentný" nejakému sériovému vykonaniu transakcií.

T_1	T_2
read a	
a:=a-1	
write a	
read b	
b:=b+1	
write b	
	read b
	b:=b-2
	write b
	read c
	c:=c+2
	write c

T_1	T_2
read a	
	read b
a:=a-1	
	b:=b-2
write a	
	write b
read b	
	read c
b:=b+1	
	c:=c+2
write b	
	write c

T_1	T_2
read a	
a:=a-1	
	read b
	b:=b-2
write a	
read b	
	write b
	read c
b:=b+1	
write b	
	c:=c+2
c:=c+2	
write c	

Nástroje - plánovače (schedulers), zámky,protokoly



„Sémantika"transakcií

Vo všeobecnosti nie sme schopní analyzovať čo transakcie počítajú (robia). Predpokladáme, že pri každom odomknutí premennej *a* sa realizuje priradenie:

a := f(a, všetky premenné dostupné niekedy medzi lock a unlock a);

kde *f* je pri každom odomykaní nová funkcia. Znalosť, že niektoré časti transakcií sa opakujú alebo sú rovnaké nevyužívame.

Príklad:

T_1	T_2	T_3
lock a	lock b	lock a
lock b	lpck c	lock c
unlock a $f_1(a, b)$	unlock b $f_3(b, c)$	unlock c $f_6(a, c)$
unlock b $f_2(a, b)$	lock a	unlock a $f_7(a, c)$
	unlock c $f_4(a, b, c)$	
	unlock a $f_5(a, b, c)$	

Krok	akcia	a	b	c
1	T_1:lock a			
2	T_2:lock b			
3	T_2:lock c			
4	T_2:unlock b	φ	$f_3(b, c)$	
5	T_1:lock b			
6	T_1:unlock a	$f_1(a, f_3(b, c))$		ψ
7	T_2:lock a			
8	T_2:unlock c			$f_4(f_1(a, f_3(b, c)), b, c)$
9	T_2:unlock a	$f_5(\varphi, b, c)$		
10	T_3:lock a			
11	T_3:lock c			
12	T_1:unlock b		$f_2(a, f_3(b, c))$	
13	T_3:unlock c			$f_6(f_5(\varphi, b, c), \psi)$
14	T_3:unlock a	$f_7(f_5(\varphi, b, c), \psi)$		

Test sériovateľnosti

Graf sériovateľnosti:

- uzly sú transakcie
- hrana $T_i \rightarrow T_j$ práve vtedy keď pre nejaké x v rozvrhu T_i: unlock x predchádza T_j: lock x.

Veta: Ak graf sériovateľnosti neobsahuje cyklus. Rozvrh je sériovateľný a topologické utriedenie grafu sériovateľnosti reprezentuje ekvivalentný sériový rozvrh.

Protokol sa nazýva ***dvojfázový*** ak v každej jeho transakcii všetky operácie zamykania (lock) predchádzajú prvú operáciu odomykania (unlock) v danej transakcii.

Veta: Dvojfázový protokol je sériovateľný.

Realistickýmodel - rlock / wlock

Read (shared) lock: transakcia bude len čítať zamknutú premennú (prípadne ju použije k výpočtu inej premennej). **Rlock** bráni iným transakciám zmeniť zamknutú premennú, ale nebráni jej čítaniu.

Write (exclusive) lock: Zámky v predošlom zmysle. Len jedna transakcia môže mať **wlock** na danú premennú v danom okamihu.

Kompatibilita zámkov:

požadovaný zámok \ existujúci zámok	rlock	wlock
rlock	Yes	No
wlock	No	No

Ďalšie zámky – ilock

Zámok pre zvýšenie alebo zníženie hodnoty
a += i; resp. a -=d; (Použitie napríklad v bankomatoch.)
Takéto zámky môžeme zaviesť pre komutatívne a asociatívne operácie.

Kompatibilita zámkov:

požadovaný zámok \ existujúci zámok	rlock	wlock	ilock
rlock	Yes	No	No
wlock	No	No	No
ilock	No	No	Yes

Sériovateľnosť

Zmena definície hrán grafu sérivateľnosti:

- Nech T_i má rlock alebo wlock na premennú *a*. Nech T_j je nasledujúca transakcia požadujúca wlock na *a*. Potom hrana $T_i \rightarrow T_j$.
- Nech T_i má wlok na premennú *a*. Nech T_m je nasledujúca transakcia požadujúca rlock na *a* potom, čo ho T_i odomkne. Potom hrana $T_i \rightarrow T_m$.

Veta: Acyklický graf je sériovateľný. Topologické triedenie dáva evivalentný sériový rozvrh.

Veta: Dvojfázový protokol zaručuje sériovateľnosť.

Neúspešnétransakcie

Dôvody neúspechu (failure) transakcií:

- Prerušenie užívateľom, zlyhanie aritmetickej operácie. Nedostatok práv k prístupu alebo nedostatok prostriedkov.
- Plánovač odhalí deadlock a rozhodne sa transakciu zrušiť.
- Plánovač odvolá transakciu potom, čo detekoval nesério-vateľnosť.
- Zlyhanie software alebo hardware.

commit - posledný príkaz úspešnej transakcie

Neúspešná (aborting) transakcia má za následok **rollback.** Nečisté (dirty) dáta - dáta zapisané transakciou, ktorá nebola ešte potvrdená (committed).

Journal (log)

Kaskádovitý rollback

	T_1	T_2
1	lock a	
2	read a	
3	a:=a-1	
4	write a	
5	unlock a	
6		lock a
7		read a
8		a:=a+2
9		write a
10		unlock a
11		commit
12	lock b	
13	read b	
14	b:=b/a	

Hoci T_2 je committed. Fail T_1 vyvolal neplatnosť premennej a, tým aj nutnosť zrušenia transakcie T_2 .

Striktná dvojfázovosť:

- Transakcia nesmie písať do databázy pokiaľ nedosiahla commit point.
- Transakcia nesmie uvoľniť žiaden zámok pokiaľ nezapísala do databázy.

Agresívnaadefenzívnastratégia

- Agresívna stratégia snaží sa, aby spracovanie bolo čo najrýchlejšie. Začína transakcie aj keď je to spojené s rizikom, že budú odvolané.
- Defenzívna (konzervatívna) stratégia snaží sa minimalizovať riziko abortu transakcií. Nezačína transakcie pokiaľ nie je isté, že skončia.

Checkpoints - kontrolnébody

Checkpoint = konzistentný stav bázy dát (stav, čas)

Backup - podobné ale na náhradnom médiu

- Dočasne začínanie nových transakcií pokiaľ všetky aktívne transakcie nie sú committed alebo aborted.
- Nájde všetky bloky modifikované v dočasných súboroch a stránky v hlavnej pamäti, ktoré neboli zapísané do databázy.
- Zapamätá v predošlom odstavci nájdené bloky do databázy
- Do journálu (logu) poznamená výskyt checkpointu (dátum, stav, druh checkpointu)

Časovérazítka

Základná myšlienka: každá transakcia dostane časové razítko - okamih začatia.

Sériový rozvrh = rozvrh v poradí časových razítok.

Pravidlá sériovateľnosti:

- Transakcia nemôže čítať hodnoty, ktoré boli napísané neskôr začatou transakciou
- Transakcia nemôže písať hodnoty, ktoré boli prečítané neskôr začatou transakciou

Implementácia namiesto zámku dvojica časov $<t_r, t_w>$; transakcia s časovým razítkom t:

- *read* and $t > t_w$: {*read*; **if** $t_r < t$ **then** $t_r = t$; }
- *write* and $t \geq t_r$ and $t \geq t_w$: {*write*; $t_w := t$; }
- *write* and $t_r \leq t < t_w$: { do nothing }
- otherwise abort;

Neporovnateľnosť sériovateľnosti časovýmirazítkamiazámkami

S

	T_1	T_2
1		read b
2	read a	
3	write c	
4		write c

T

	T_1	T_2	T_3
1	read a		
2		read a	
3			read d
4			write d
5			write a
6		read c	
7	write b		
8		write b	

Rozvrh *S* je sériovateľný zámkami, ale nie je sériovateľný časovými razítkami. Rozvrh *T* naopak.