Tranzakcijų valdymas

- Blogų duomenų problema
- Rollback kaskadavimas
- Rollback valdymas
- Rollback blokams
- Rollback mažiems duomenų elementams
- Commit grupės
- Deadlocks

Blogų duomenų problema

T_1	T_2	A	B
		25	25
$l_1(A); r_1(A);$			
$A := A + 100;$			
$w_1(A); l_1(B); u_1(A);$		125	
	$l_2(A); r_2(A);$		
	$A := A*2;$		
	$w_2(A);$	250	
	$l_2(B)$ **Denied**		
$r_1(B);$			
Abort; $u_1(B);$			
	$l_2(B); u_2(A); r_2(B);$		
	$B := B*2$		
	$w_2(B); u_2(B);$		50

Blogų duomenų problema

T_1	T_2	T_3	A	B	C
200	150	175	RT=0	RT=0	RT=0
			WT=0	WT=0	WT=0
	$w_2(B)$;			WT=150	
$r_1(B)$;					
	$r_2(A)$;		RT=150		
		$r_3(C)$;			RT=175
	$w_2(C)$;				
	Abort;			WT=0	
		$w_3(A)$;	WT=175		

Rollback

- Rollback blokams
  - Kietas bolkavimas – kai tranzakcija negali irasyti duomeu, kol ji ne bus baigta
- Rollback maziems bazes elementams
  1. Skaitomi duomenys is disko ir attitinkamai keisti buferio turini
  2. Jei yra logai, tai pradine reiksme gaunama tiesiogiai is logo
  3. Pagrindineje atmintyje isaugoti logai kiekvienia tranzakcijai

Commit grupe

T_1 raso X, raso i loga, bet logas lieka buferyje. T_2 skaito X.

- Ir T_1 ir T_2 neturi teises rasyti i diska. Tada nutraukiamas abieju trancakciju darbas.
- T_1 irase, o T_2 ne. Galimi 2 variantai: T_2 neskaite X po neutralios tranzakcijos, nutraukiama, arba neturi itakos duomenu bazei.
- Abu irasyti. Tai X yra teisingi, pries skaitant T_2.

Kol nebaigta tranzakcija, neleisti rasyti i diska, ir kol nebus uzregistruota loguose.

Loginiai logai

- Lengviau pasalinti bloga trancakcija, kai duomenu bazes elementai yra blokai arba puslapiai. Problemos
- Loginimo metu reikalauja naujos arba senos duomenu bazes reiksmes, arba abieju is karto, kad butu irasyta i loga.. Jei pakeitimai mazi, tai loguose yra daug bereikalingos informacijos.
- Reikalavimas, kad irasas butu atsatytas pabaigus blokavima tiktai po commito.

Polirafinis ir testai View-Serializability

T_1	T_2	T_3	T_4
	$r_2(A)$;		
$r_1(A)$; $w_1(C)$;			
		$r_3(C)$;	
$w_1(B)$;			
			$r_4(B)$;
		$w_3(A)$;	
			$r_4(C)$;
	$w_2(D)$; $r_2(B)$;		
			$w_4(A)$; $w_4(B)$;

Deadlocks – laukimo grafas

T_1: $l_1(A)$; $r_1(A)$; $l_1(B)$; $w_1(B)$; $u_1(A)$; $u_1(B)$;
T_2: $l_2(C)$; $r_2(C)$; $l_2(A)$; $w_2(A)$; $u_2(C)$; $u_2(A)$;
T_3: $l_3(B)$; $r_3(B)$; $l_3(C)$; $w_3(C)$; $u_3(B)$; $u_3(C)$;
T_4: $l_4(D)$; $r_4(D)$; $l_4(A)$; $w_4(A)$; $u_4(D)$; $u_4(A)$;

T_1	T_2	T_3	T_4
$l_1(A)$; $r_1(A)$;			
	$l_2(C)$; $r_2(C)$;		
		$l_3(B)$; $r_3(B)$;	
			$l_4(D)$; $r_4(D)$;
	$l_2(A)$; Denied		
		$l_3(C)$; Denied	
			$l_4(A)$; Denied
$l_1(B)$; Denied			

Deadlock panaikinimas

T_1: $l_1(A)$; $r_1(A)$; $l_1(B)$; $w_1(B)$; $u_1(A)$; $u_1(B)$;
T_2: $l_2(A)$; $l_2(C)$; $r_2(C)$; $w_2(A)$; $u_2(C)$; $u_2(A)$;
T_3: $l_3(B)$; $r_3(B)$; $l_3(C)$; $w_3(C)$; $u_3(B)$; $u_3(C)$;
T_4: $l_4(A)$; $l_4(D)$; $r_4(D)$; $w_4(A)$; $u_4(D)$; $u_4(A)$;

Deadlock panaikinimas

T_1	T_2	T_3	T_4
$l_1(A);\ r_1(A);$			
	$l_2(A);$ Denied		
		$l_3(B);\ r_3(B);$	
			$l_4(A);$ Denied
		$l_3(C);\ w_3(C);$	
		$u_3(B);\ u_3(C);$	
$l_1(B);\ w_1(B);$			
$u_1(A);\ u_1(B);$			
	$l_2(A);\ l_2(C);$		
	$r_2(C);\ w_2(A);$		
	$u_2(A);\ u_2(C);$		
			$l_4(A);\ r_4(D);$
			$r_4(D);\ w_4(A);$
			$u_4(A);\ u_4(D);$

Deadlocks aptikimas Timestamps

T_1	T_2	T_3	T_4
$l_1(A);\ r_1(A);$			
	$l_2(A);$ **Dies**		
		$l_3(B);\ r_3(B);$	
			$l_4(A);$ **Dies**
		$l_3(C);\ w_3(C);$	
		$u_3(B);\ u_3(C);$	
$l_1(B);\ w_1(B);$			
$u_1(A);\ u_1(B);$			
			$l_4(A);\ l_4(D);$
	$l_2(A);$ **Waits**		
			$r_4(D);\ w_4(A);$
			$u_4(D);\ u_4(A);$
	$l_2(A);\ l_2(C);$		
	$r_2(C);\ w_2(A);$		
	$u_2(A);\ u_2(C);$		

Deadlocks aptikimas Timestamps

T_1	T_2	T_3	T_4
$l_1(A);\ r_1(A);$			
	$l_2(A);$ **Waits**		
		$l_3(B);\ r_3(B);$	
			$l_4(A);$ **Dies**
$l_1(B);\ w_1(B);$		**Wounded**	
$u_1(A);\ u_1(B);$			
	$l_2(A);\ l_2(C);$		
	$r_2(C);\ w_2(A);$		
	$u_2(A);\ u_2(C);$		
			$l_4(A);\ l_4(D);$
			$r_4(D);\ w_4(A);$
			$u_4(D);\ u_4(A);$
		$l_3(B);\ r_3(B);$	
		$l_3(C);\ w_3(C);$	
		$u_3(B);\ u_3(C);$	