Innlevering 2 IN3020

Pilasilda A. George

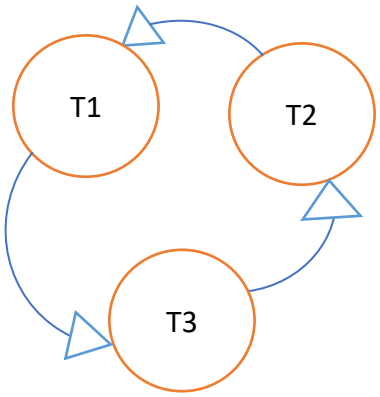
pilasila@uio.no

05.februar.2019

Oppgave 1

a) $S_1 = r_1(a); r_3(d); r_1(b); r_2(b); w_1(b); r_3(a); w_3(a); r_1(c); w_1(c); r_2(d); w_2(d);$

Presedensgrafen for S_1:



På grunn av sykel i presedensgrafen er den ikke-konfliktserialiserbar.

b) $S_1 = r_1(a); r_3(d); r_1(b); r_2(b); w_1(b); r_3(a); w_3(a); r_1(c); w_1(c); r_2(d); w_2(d);$

$T1 = Sl_1(a); r_1(a); xl_1(b); r_1(b); w_1(b); xl_1(c); r_1(c); w_1(c); ul_1(a); ul_1(b); ul_1(c);$

$T2 = Sl_2(b); r_2(b); xl_2(d); r_2(d); w_2(d); ul_2(b); ul_2(d);$

$T3 = Sl_3(d); r_3(d); xl_3(a); r_3(a); w_2(d); ul_3(d); ul_2(d);$

c)

T1	T2	T3
$Sl_1(a)$;		
$R_1(a)$;		
		$Sl_3(d)$;
		$R_3(d)$;
$Sl_1(b)$;		
$R_1(b)$;		
	$Sl_2(b)$;	
	$R_2(b)$;	
$X_1(b)$; - vent		
		$Sl_3(a)$;
		$R_3(a)$;
$X_l(c)$;		$X_l(a)$; - vent
$R_1(c)$;		
$W_1(c)$;		
	$Sl_2(d)$;	
	$R_2(d)$;	
	$Xl_2(d)$; - vent	

T_1 setter leselås på a, a blir lest. T_3 fortsetter setter leselås på d og leser d. T1 fortsetter setter leselås på b og leser b.

T_2 fortsetter setter leselås på b og leser b. T_1 fortsetter og prøver å skrive b, men må vente til T_2 har unlocket b.

T_3 fortsetter setter leselås på a, leser a og prøver å skrive a, men må vente fordi T_1 allerede har låst a og T_3 må vente til T_1 har unlocket a.

T_1 fortsetter setter leselås på c, leser så c og skriver c.

T_2 fortsetter setter leselås på d, leser d og prøver å sette skrivelås på d, men kan ikke det fordi T_3 allerede har satt lås på d dermed blir det deadlock siden alle transaksjoner må vente på hverandre.

Oppgave 2

a) T_1 ber om en skrivelås på b, men siden T_2 har leselås på b og T_1 er eldre enn T_2, så må T_1 vente.

T_2 ber om en skrivelås på d, men siden T_3 har leselås på d og T_2 er eldre enn T_3, så må T_2 vente.

T_3 ber om en skrivelås på a, men siden T_1 har leselås på a og T_3 er yngre enn T_1, så må T_3 abortert/rullet tilbake.

b) T_1 ber om en skrivelås på a, men siden T_2 har leselås på b og T_1 er eldre enn T_2, så tvinges T_2 til å abortere/rulle tilbake.

T_2 ber om en skrivelås på d, men siden T_3 har leselås på d og T_2 er eldre enn T_3, så tvinges T_3 til å abortere/rulle tilbake.

T_3 ber om en skrivelås på a, men siden T_1 har leselås på a og T_3 er yngre enn T_1, så blir T_3 tunget til å vente til låsen er frigjort.

Oppgave 3

a) Serializable: før man committer, ble transaksjonen abortert fordi det er ikke mulig å gjøre update-operasjoner på serialiaserbare isolasjonsnivåer. Jeg gjør update operasjoner på begge transkasjonene i to terminaler kan ikke den ene transaksjonen se hvilke updates eller inserts som er gjort i den andre transkasjonen. Kjører så commit for begge transaksjonene får jeg en melding som ser følgende ut: ERROR: could not serialize access due to read/write dependencies among transactions DETAIL: Reason code: Canceled on identification as a pivot, during commit attempt. HINT: The transaction might succeed if retried. Dette fordi begge transaksjonene har endret det den andre transaksjonen ville ha lest. Selecter jeg tabellen i begge terminalene, får jeg samme resultat for begge. Dette fordi at begge transaksjonene kjører serielt/parallelt og kan ikke gjøre update på begge transaksjonene samtidig. Dermed blir det kun T_2 som gjør en update på tabellen og begge transaksjonene får samme resultat.

b) Repeatable read: tilstanden til databasen blir opprettholdt fra starten av transaksjonen. Hvis du oppdaterer verdien i transaksjon 1 og det samme i transaksjon 2, da vil verdien oppdateres i begge transaksjonene. Kjører du commit deretter i begge transaksjonene, vil du få den opprinnelige tilstanden til tabellen, altså da det ble gjort insert.

Oppgave 4

x, y og z er lokale variable for T_1. Dette gjør at:

T1:

X: a; **Y:** b; **Y:** x + y; **Z:** c; **Z:** z+1; **C:** z;

A = 13, B = 17 og C = 19, som er de intielle verdiene.

a) Post i undo-logg, for T1
START T1
OLDV(T1, b, 17)
OLDV(T1, c, 19)
COMMIT T1

b) Loggpostene skrives først til log for også skrive til disk i tabeller før den commiter.

c) Postene for undo/redo-loggen for T1:
<T1, x, x-verdi(gammel), x-verdi(ny)>
START T1
UPD(T1, b, 17,30)
UPD(T1, C, 19,20)
COMMIT T1

d) Loggpostene kan kan som sagt skrive til disk før o getter transaksjonene har commitet. Siden det ikke er spesielle krav for hvordan undo/redo loggen commiter, så kan loggene overføres til disk når det passer.

Oppgave 5

Tabellen R(x,y) har 100000 tupler T(R), dette innebærer at hver blokk har 500 tupler. Dette bidrar til at:

100000/500 = 200 blokker B(R)

Vx(R) = antall distinkte verdier for x i R

Som konklusjon vil jeg si at Vx(R) er lavere jo estimert kostnad en får. Hvis det er under 500 distinkte verdier vil det ikke lønne seg å bruke indeksen.