TDT4145 Datamodellering og databasesystemer

Transaksjoner

a) Hvilke av de følgende historiene er konfliktsserialiserbare? Bestem evt. den ekvivalente serielle historien.

H1: r1 (X); r3 (X); w1(X); r2(X); w3(X)

H2: r1 (X); r3 (X); w3(X); w1(X); r2(X)

H3: r3 (X); r2 (X); w3(X); r1(X); w1(X)

H4 r3 (X); r2 (X); r1(X); w3(X); w1(X)

(H1) This schedule is not serializable because T3 must run before T1 (r3(X) happens before w1(X) and T1 must run before T3 (r1(X) happens before w3(X)).

(H2) This schedule is not serializable.

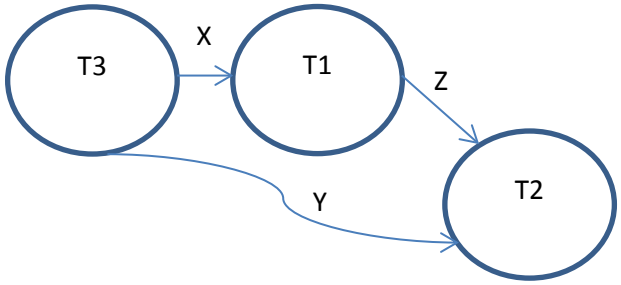
(H3) This schedule is serializable because all conflicting operations of T3 happens before all conflicting operation of T1. T2 has only one operation, which is a read on X (r2(X)), which does not conflict with any other operation. Thus this serializable schedule is equivalent to r2(X); r3(X); w3(X); r1(X); w1(X) (e.g., T2 ->T3->T1) serial schedule.

(H4) This is not a serializable schedule.

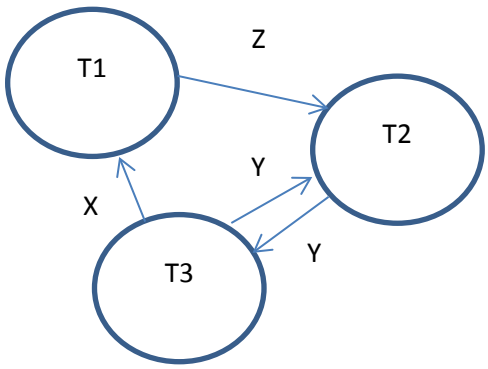
b) Se på de tre transaksjonene T1, T2, and T3 og historiene H5 og H6 gitt under. Tegn presedensgrafen for H5 og H6 og avgjør om historiene er serialiserbare eller ikke. Hvis historien er serialserbar, skriv ned den ekvivalente serielle historien.

T1: r1(x); r1(z); w1(x)
T2: r2(z); r2(y); w2(z); w2(y)
T3: r3(x); r3(y); w3(y)
H5: r1(x); r2(z); r1(z); r3(x); r3(y); w1(x); w3(y); r2(y); w2(z); w2(y)
H6: r1(x); r2(z); r3(x); r1(z); r2(y); r3(y); w1(x); w2(z); w3(y); w2(y)

Schedule H5: It is a serializable schedule because there is no cycle in the graph:



Schedule H6 is not a serializable schedule because there a cycle in the graph:



c) Se på historiene H7, H8 og H9 under. Bestem om hver historie er *strict*, *cascadeless*, *recoverable* eller *nonrecoverable*.
H7: r1(x); r2(z); r1(z); r3(x); r3(y); w1(x); c1; w3(y); c3; r2(y); w2(z); w2(y);c2
H8: r1(x); r2(z); r1(z); r3(x); r3(y); w1(x); w3(y); r2(y); w2(z); w2(y); c1; c2; c3;
H9: r1(x); r2(z); r3(x); r1(z); r2(y); r3(y); w1(x); w2(z); w3(y); w2(y); c3; c2;

Strict schedule: A schedule is strict if it satisfies the following conditions:

1. Tj reads a data item X ***after*** Ti has written to X and Ti is terminated (aborted or committed)

2. Tj writes a data item X ***after*** Ti has written to X and Ti is terminated (aborted or committed)

Historie H7 er **strikt** fordi T2 leser og skriver Y etter at T3 som skrev Y har committet.

Historie H8 er **ikke strikt** fordi T2 leser Y etter at T3 har skrevet Y og før T3 har committet.

Historie H9 er **ikke strikt** fordi T2 skriver Y etter at T3 har skrevet Y og før T3 har committet.

Cascadeless schedule: A schedule is cascadeless if the following condition is satisfied:

Tj reads X only ***after*** Ti has written to X and terminated (aborted or committed).

Historie H7 er ***cascadeless*** da den er strikt.

Historie H8 er ikke ***cascadeless*** fordi T2 leser Y etter at T3 har skrevet Y men før T3 har committet.

Historie H9 er ***cascadeless*** da det er ingen lesinger av ucommittede verdier.

Recoverable schedule: A schedule is recoverable if the following condition is satisfied:

Tj commits after Ti if Tj has read any data item written by Ti.

Historie H7 er **recoverable** da den er strikt.

Historie H8 er **nonrecoverable** da T2 leser Y som T3 har skrevet og T2 committer før T3.

Historie H9 er **recoverable** da den er cascadeless.

d) La A,B,C og D være dataelementer med angitte startverdier og gitt loggen under med loggposter på formatet:
[LSN,Operation,Transaction,DataItem,BeforeImage,AfterImage]

	A	B	C	D
	30	15	40	20
[101,start_trans,T3]				
[102,read,T3,C]				
[103,write,T3,B,15,12]		12		
[104,start_trans,T2]				
[105,read,T2,B]				

[106,write,T2,B,12,18]		18		
[107,start,T1]				
[108,read,T1,A]				
[109,read,T1,D]				
[110,write,T1,D,20,25]				25
[111,read,T2,D]				
[112,write,T2,D,25,26]				26
[113,read,T3,A]				

NB: Det er ikke vanlig å logge read-operasjoner, med mindre man har behov for å replikere operasjoner og låser via loggposter. De er logget her for å kunne lage spørsmålet under ☺

Hvilken recoveryegenskap (strict, cascadeless, recoverable, ingen av delene) har denne historien?

Historien er ikke strict fordi T2 både leser og skriver B før T3 committer/aborterer.
Historien er ikke cascadeless fordi T2 leser en ucommitted operasjon av T3.
Historien er recoverable hvis T2 committer etter T3 og T1.

e) Hvis T2 rulles tilbake som en konsekvens av konflikten med T3, hvilke verdier vil dataelementene A, B, C, D ha etter at T2 har blitt rullet tilbake?
Da ville verdiene være A=30, B=12, C=40, D=25.

f) Anta A, B, C, D er datasider det skal gjøres recovery på. Hvilke loggposter blir det gjort REDO på i recovery av denne loggen når DPT har følgende tilstand:

(B,recLSN=106),(D,recLSN=112)

etter analysen og blokkene har følgende tilstand på disken:

(A,pageLSN=100,value=30),
(B, pageLSN=106,value=18),
(C,pageLSN=50,value=40),
(D,pageLSN=110,value=25)

?

Ser bare på write-operasjonene:

[101,start_trans,T3]	
[102,read,T3,C]	
[103,write,T3,B,15,12]	Ser i DPT at den ikke trenger REDO
[104,start_trans,T2]	
[105,read,T2,B]	
[106,write,T2,B,12,18]	Ser i DPT at den kanskje trenger REDO. Ser i blokka at den ikke trenger REDO.
[107,start_trans,T1]	
[108,read,T1,A]	
[109,read,T1,D]	
[110,write,T1,D,20,25]	Ser i DPT at den ikke trenger REDO
[111,read,T2,D]	
[112,write,T2,D,25,26]	Ser i DPT at den kanskje trenger REDO og i blokka at den trenger REDO.

[113,read,T3,A]	

g) Skriv om sekvensen med operasjoner slik at den blir strict. Anta alle tre transaksjonene commiter. Du må renummerere loggpostene og legge til noen commit-loggposter.

	A	B	C	D
	30	15	40	20
[101,start_trans,T3]				
[102,read,T3,C]				
[103,write,T3,B,15,12]		12		
[104,read,T3,A]				
[105,commit,T3]				
[106,start_trans,T2]				
[107,read,T2,B]				
[108,write,T2,B,12,18]		18		
[109,start_trans,T1]				
[110,read,T1,A]				
[111,read,T1,D]				
[112,write,T1,D,20,25]				25
[113,commit,T1]				
[114,read,T2,D]				
[115,write,T2,D,25,26]				26
[116,commit,T2]				

Svein Erik Bratsberg, 9/4-2014