CS3223 Tutorial 8: MVCC

Wk 11, Sem 2, 2022/23

Questions to be discussed: 1, 2 & 4.

1. (a) Write down a serializable Snapshot Isolation (SSI) schedule that is not a conflict serializable schedule (CSS).

   (b) Write down a conflict serializable schedule (CSS) that is not a serializable Snapshot Isolation (SSI) schedule.

   (c) Write down a serializable Snapshot Isolation (SSI) schedule S (involving transactions T_1, T_2, and T_3) such that DSG(S) consists of two concurrent rw-dependency edges $T_1 \overset{rw}{\dashrightarrow} T_2$ and $T_2 \overset{rw}{\dashrightarrow} T_3$.

Solution:

(a) The following schedule is a SSI schedule that is multiversion view equivalent to (T_2, T_1). However, it is a not a valid monoversion schedule and therefore not a CSS.

T_1:	$W_1(x_1)$,			$W_1(y_1)$, C_1,	
T_2:		$R_2(x_0)$, $R_2(y_0)$,			C_2

(b) The following schedule is CSS schedule that is conflict equivalent to (T_1, T_2). However, under SI, it is impossible for both of the tranactions to have committed due to their concurrent updates to y. Specifically, T_2 would have been aborted (either when $W_2(y)$ is attempted under FUW or when C_2 is attempted under FCW).

T_1:	$R_1(x)$,		$W_1(y)$,	C_1,	
T_2:		$R_2(x)$,			$W_2(y)$, C_2

(c) The following schedule is a SSI schedule that is multiversion view equivalent to (T_1, T_2, T_3). Observe that $T_1 \overset{rw}{\dashrightarrow} T_2$ (due to $R_1(x_0)$ and $W_2(x_2)$), and $T_2 \overset{rw}{\dashrightarrow} T_3$ (due to $R_2(y_0)$ and $W_3(y_3)$),

T_1:	$R_1(x)$,			C_1,			
T_2:		$W_2(x)$,	$R_2(y)$,		C_2,		
T_3:					$W_3(y)$,		C_3

2. For each of the following multiversion schedules, state whether it is multiversion view serializable (MVSS). If the schedule is MVSS, write down a serial monoversion schedule that is multiversion view equivalent to it.

(a)

T_1:			$R_1(x_4)$		$W_1(x_1)$				$R_1(y_0)$	$W_1(z_1)$		
T_2:				$R_2(y_0)$				$R_2(x_0)$				$R_2(z_3)$
T_3:						$W_3(x_3)$	$W_3(z_3)$					
T_4:	$W_4(y_4)$	$W_4(x_4)$										

(b)

T_1:			$R_1(x_4)$		$W_1(x_1)$				$R_1(y_4)$	$W_1(z_1)$		
T_2:				$R_2(y_0)$				$R_2(x_1)$				$R_2(z_3)$
T_3:						$W_3(x_3)$	$W_3(z_3)$					
T_4:	$W_4(y_4)$	$W_4(x_4)$										

(c)

T_1:			$R_1(x_4)$		$W_1(x_1)$				$R_1(y_4)$	$W_1(z_1)$		
T_2:				$R_2(y_0)$				$R_2(x_3)$				$R_2(z_3)$
T_3:						$W_3(x_3)$	$W_3(z_3)$					
T_4:	$W_4(y_4)$	$W_4(x_4)$										

(d)

T_1:			$R_1(x_3)$		$W_1(x_1)$				$R_1(y_0)$	$W_1(z_1)$		
T_2:				$R_2(y_4)$				$R_2(x_4)$				$R_2(z_1)$
T_3:						$W_3(x_3)$	$W_3(z_3)$					
T_4:	$W_4(y_4)$	$W_4(x_4)$										

Solution:

To determine whether a schedule S is MVSS, we can construct a directed graph (denoted by MVSG(S)) to capture the read-from relations among the transactions. Each node in MVSG(S) represents a transaction with the following edges: (1) (T_j, T_i) if T_i reads from T_j, and (2) (T_j, T_i) if T_j read some object O from the initial database & T_i update object O. If MVSG(S) is cyclic, then S is not MVSS. If MVSG(S) is acyclic, then S is MVSS iff there exists a serial monoversion schedule produced from a topological ordering of MVSG(S) that is multiversion view equivalent to S.

(a) Not MVSS. Since $R_1(x_4)$, we have $T_4 \rightarrow T_1$ in MVSG(S). Since $R_1(y_0)$ & $W_4(y_4)$, we have $T_1 \rightarrow T_4$ in MVSG(S). Since MVSG(S) is cyclic, S is not MVSS.

(b) Not MVSS. Since $R_1(x_4)$, we have $T_4 \rightarrow T_1$ in MVSG(S). Since $R_2(y_0)$ & $W_4(y_4)$, we have $T_2 \rightarrow T_4$ in MVSG(S). Since $R_2(x_1)$, we have $T_1 \rightarrow T_2$ in MVSG(S). Since MVSG(S) is cyclic, S is not MVSS.

(c) MVSS. Since $R_1(x_4)$, we have $T_4 \rightarrow T_1$ in MVSG(S). Since $R_2(y_0)$ & $W_4(y_4)$, we have $T_2 \rightarrow T_4$ in MVSG(S). Since $R_2(x_3)$, we have $T_3 \rightarrow T_2$ in MVSG(S). The schedule is multiversion view equivalent to serial monoversion schedule (T_3, T_2, T_4, T_1).

(d) MVSS. Since $R_1(x_3)$, we have $T_3 \rightarrow T_1$ in MVSG(S). Since $R_1(y_0)$ & $W_4(y_4)$, we have $T_1 \rightarrow T_4$ in MVSG(S). Since $R_2(y_4)$, we have $T_4 \rightarrow T_2$ in MVSG(S). Since $R_2(z_1)$, we have $T_1 \rightarrow T_2$ in MVSG(S). The schedule is multiversion view equivalent to serial monoversion schedule (T_3, T_1, T_4, T_2).

3. (Exercise 17.3, R&G) For each of the following schedules, state whether it is a Snapshot Isolation schedule under the First-Committer-Wins rule.
   (a) R_1(X), R_2(X), W_1(X), W_2(X), $Commit_1$, $Commit_2$
   (b) W_1(X), R_2(Y), R_1(Y), R_2(X), $Commit_1$, $Commit_2$
   (c) R_1(X), R_2(Y), W_3(X), R_2(X), R_1(Y), $Commit_1$, $Commit_2$, $Commit_3$
   (d) R_1(X), R_1(Y), W_1(X), R_2(Y), W_3(Y), W_1(X), R_2(Y), $Commit_1$, $Commit_2$, $Commit_3$
   (e) R_1(X), W_2(X), W_1(X), $Commit_2$, $Commit_1$
   (f) W_1(X), R_2(X), W_1(X), $Commit_2$, $Commit_1$
   (g) R_2(X), W_3(X), $Commit_3$, W_1(Y), $Commit_1$, R_2(Y), W_2(Z), $Commit_2$
   (h) R_1(X), W_2(X), $Commit_2$, W_1(X), $Commit_1$, R_3(X), $Commit_3$
   (i) R_1(X), W_2(X), W_1(X), R_3(X), $Commit_1$, $Commit_2$, $Commit_3$

Solution:

(a) Not SI-FCW: T_2 must abort due to $W_2(X)$.
(b) SI-FCW.
(c) SI-FCW.
(d) SI-FCW.
(e) Not SI-FCW: T_1 must abort due to $W_1(X)$.
(f) SI-FCW.
(g) SI-FCW.
(h) Not SI-FCW: T_1 must abort due to $W_1(X)$.
(i) Not SI-FCW: T_2 must abort due to $W_2(X)$.

4. Consider the following schedule involving transactions T_1, T_2, T_3, and T_4 which are executed in a database system using the Snapshot Isolation protocol with the First Committer Wins Rule.

   Assume the following values for the state of the database before the start of the schedule:

$$a_0 = 10, \quad b_0 = 20, \quad c_0 = 30.$$

Timestamp	T_3	T_1	T_2	T_4	**Comments**
1	$R_3(c)$				
2			$R_2(b)$		
3			$W_2(b)$		T_2 updates b to 45.
4			$Commit_2$		
5	$R_3(b)$				
6	$W_3(b)$				T_3 updates b to 60.
7		$R_1(a)$			
8		$R_1(b)$			
9		$W_1(a)$			T_1 updates a to 100.
10				$R_4(b)$	
11				$R_4(a)$	
12				$W_4(b)$	T_4 updates b to 16.
13	$R_3(a)$				
14	$W_3(c)$				T_3 updates c to 200.
15		$W_1(c)$			T_1 updates c to 40.
16		$Commit_1$			
17				$R_4(b)$	
18				$R_4(c)$	

   (a) Write down the value read by $R_3(b)$ at timestamp 5.
   (b) Write down the value read by $R_1(b)$ at timestamp 8.
   (c) Write down the value read by $R_4(b)$ at timestamp 10.
   (d) Write down the value read by $R_4(a)$ at timestamp 11.
   (e) Write down the value read by $R_3(a)$ at timestamp 13.
   (f) Write down the value read by $R_4(b)$ at timestamp 17.
   (g) Write down the value read by $R_4(c)$ at timestamp 18.

Solution: The database snapshot for a transaction T consists of (1) the versions written by transactions that have committed before the start of T, and (2) any versions written by T itself. Thus, T can't read any version written by a concurrent transaction. For each $R_i(O)$, the most recent version of O in T_i's database snapshot is returned.

(a) Write down the value read by $R_3(b)$ at timestamp 5. (20)

(b) Write down the value read by $R_1(b)$ at timestamp 8. (45)

(c) Write down the value read by $R_4(b)$ at timestamp 10. (45)

(d) Write down the value read by $R_4(a)$ at timestamp 11. (10)

(e) Write down the value read by $R_3(a)$ at timestamp 13. (10)

(f) Write down the value read by $R_4(b)$ at timestamp 17. (16)

(g) Write down the value read by $R_4(c)$ at timestamp 18. (30)