Ceng352 - Database Management Systems
Written Assignment 2

Spring 2019

Q1 Transactions $T1$, $T2$, $T3$ are to be run concurrently. The following table gives details of the proposed schedule of read/write operations and the time when each such operation is scheduled.

Time	T1	T2	T3
1		read(C)	
2	read(A)		
3	write(A)		
4		read(A)	
5			read(B)
6			write(B)
7		write(A)	
8		write(C)	
9	write(B)		
10			commit
11		commit	
12	commit		

When answering the following questions, indicate shared locks by s_i and exclusive locks by x_i where i is the transaction number. Also indicate the operations of transactions as $R_i(X)$ and $W_i(X)$ for read and write operations respectively where i is the transaction number and X is a data item.

(a) Describe how the **strict two-phase locking with deadlock detection** would handle the schedule by filling in the following table.

Operation	Given LOCKS on data items			Wait for graph
	A	B	C	
$R_2(C)$			S_2	
$R_1(A)$	S_1		S_2	
$W_1(A)$	X_1		S_2	
$R_2(A)$	X_1		S_2	$T_2 \xrightarrow{A} T_1$, T_2 delays

$R_3(B)$	X_1	S_3	S_2	$T_2 \xrightarrow{A} T_1$
$W_3(B)$	X_1	X_3	S_2	$T_2 \xrightarrow{A} T_1$
$W_1(B)$	X_1	~~X_3~~	S_2	$T_2 \xrightarrow{A} T_1 \xrightarrow{B} T_3$, T_1 delays
C_3	X_1	X_1	S_2	$T_2 \xrightarrow{A} T_1$, T_1 continues
C_1	S_2	-	S_2	
$W_2(A)$	X_2	-	S_2	
$W_2(C)$	X_2	-	X_2	
C_2	-	-	-	

(b) Describe how the **strict two-phase locking with wound wait deadlock prevention** would handle the schedule. Assume that $TS(T1) = 1$, $TS(T2) = 2$, $TS(T3) = 3$.

wound-wait → old has priority, old kills new one

Operation	Given LOCKS on data items A	 B	 C	Wait for graph
$R_2(C)$	-	-	S_2	
$R_1(A)$	S_1	-	S_2	
$W_1(A)$	X_1	-	S_2	
$R_2(A)$	X_1	-	S_2	$T_2 \xrightarrow{A} T_1$, T_2 is newer, it will wait
$R_3(B)$	X_1	S_3	S_2	$T_2 \xrightarrow{A} T_1$
$W_3(B)$	X_1	X_3	S_2	$T_2 \xrightarrow{A} T_1$
$W_1(B)$	X_1	X_1	S_2	$T_2 \xrightarrow{A} T_1$, T_3 has lock on B. T_3 is newer it will be killed and will's started later
C_1	S_2	-	S_2	T_2 is awaken, it will continue.
$R_3(B)$	S_2	S_3	S_2	I assumed T_3 is restarted.
$W_2(A)$	X_2	S_3	S_2	
$W_2(C)$	X_2	S_3	X_2	
$W_3(B)$	X_2	X_3	X_2	
C_2	-	X_3	-	
C_3	-	-	-	

$W_u(X)$ --- $R_T(X)$ → data önceki txn yazmış, bitmiş OK.

→ check $TS(u) < TS(T)$

$\quad\quad 1 \quad\quad 3$

Q2 Consider the schedule H below. The symbol $r_i(x)$ stands for a read by transaction Ti to item x and $w_i(x)$ stands for a write by Ti to item x. Suppose **timestamp-based scheduler** is used as the concurrency control protocol.

$$H : r_1(A)r_2(B)w_1(C)r_3(B)r_3(C)w_2(B)w_3(A)$$

Describe what happens as each operation below executes if

(a) $TS(T1) = 1, TS(T2) = 2, TS(T3) = 3$

(b) $TS(T1) = 1, TS(T2) = 3, TS(T3) = 2$

Justify whether each operation is accepted or rejected, and show how the RTS and WTS timestamps of the data items are updated in each step.

Note: If an access is rejected. its parent transaction is aborted; so you can ignore (remove from the schedule) all the subsequent accesses by that transaction)

(a) $TS(T1) = 1, TS(T2) = 2, TS(T3) = 3$

→ no problem in read-read

Operation	A		B		C	
	RTS 0	WTS 0	RTS 0	WTS 0	RTS 0	WTS 0
$r_1(A)$	1	-	-	-	-	-
$r_2(B)$	1	-	2	-	-	-
$w_1(C)$	1	-	2	-	-	1
$r_3(B)$	1	-	3	-	-	1
$r_3(C)$	1	-	3	-	3	1
$w_2(B)$ Rollback	1	-	3	-	3	1
$w_3(A)$	1	3	3	-	3	1

$\underbrace{RT(X)}_{3} > \underbrace{TS(T)}_{2}$ → benden sonra gelen txn(3), çoktan okumuş. Roll back T_2

(b) $TS(T1) = 1. TS(T2) = 3, TS(T3) = 2$

Operation	A		B		C	
	RTS	WTS	RTS	WTS	RTS	WTS
$r_1(A)$	1	-	-	-	-	-
$r_2(B)$	1	-	3	-	-	-
$w_1(C)$	1	-	3	-	-	1

$RT(X) > \underbrace{TS(T)}$

$1 > 3$

no ✓

$\underbrace{WT}(X) > TS(T)$

$0 > 3$

Benden önce gelen yazmış.
üstine yazabilirim. ✓

read-read sıkıntısı yok.

$r_3(B)$	1	–	3	–	–	1
$r_3(C)$	1	–	3	–	2	1
$w_2(B)$	1	–	3	3	2	1
$w_3(A)$	1	2	3	3	2	1

$WT(X) > \underbrace{TS(T)}$

$\underline{1} > 2$ → Benden önce gelen adam yazmış. Okuyabilirim

$RT(X) > TS(T)$

$3 > 3$ ok

$\underline{WT(X)} > \underline{TS(T)}$

$- > 3$ ok

→ yazabilirim

$\underline{RT(X)} > T\underline{S}(T)$

$1 > 2$ no ✓

$\underline{WT(X)} > \underline{TS(T)}$ no, ok ✓

$-$ 2