

Carnegie Mellon University in Qatar

Database Applications

15-415 - Spring 2020

Problem Set 5

Out: April 18, 2020

Due: April 23, 2020

1 Serializability and Locking Protocols [20 Points]

Consider Schedule *A* given below in **Table 1** below. R(.) and W(.) denote 'Read' and 'Write', respectively. Ignore the lock T1:S(Y), for the moment.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
T1	S(Y)	R(Y)																			R(X)	
T2				W(X)																		
T3															R(X)		W(Z)					
T4									R(Z)		W(Y)											

Table 1: Schedule *A* with 4 transactions

- (a) Is schedule *A* serializable? Explain.
- (b) Is schedule *A* allowed by 2PL? If no, briefly explain why. If yes, fill in **Table 1** with the lock/unlock requests that could have happened.

Notes:

- Make sure that the 2PL protocol is obeyed.
- Use the notations S(.), X(.), and U(.) to denote Shared lock, eXclusive lock, and Unlock, respectively.

- (c) Is schedule *A* allowed by *strict* 2PL? Explain.

2 Deadlock Detection [25 Points]

Consider the following two schedules, 1 and 2, shown in Table 1 and Table 2, respectively.

	1	2	3	4
T1	S(A)			S(B)
T2		X(A)		
T3			X(B)	

Table 2: Schedule 1

	1	2	3	4	5
T4	S(D)				S(F)
T5	X(D)				
T6			X(B)	X(D)	

Table 3: Schedule 2

- (a) For Schedule 1, assuming no other transactions exist, list which lock requests will be granted or blocked by the lock manager.
- (b) Give the **wait-for** graph for Schedule 1.
- (c) For Schedule 1, indicate whether or not there will be a deadlock at the end of the schedule. Explain briefly.
- (d) For Schedule 2, assuming no other transactions exist, list which lock requests will be granted or blocked by the lock manager.
- (e) Give the **wait-for** graph for schedule 2.
- (f) For Schedule 2, indicate whether or not there will be a deadlock at the end of the schedule. Explain briefly.

3 B^+ Tree Locking [25 Points]

Consider the B^+ tree in **Figure 2** below. Use the non-conservative **lock-coupling** algorithm, *Bayer-Schkolnick*, to lock the B^+ tree. The algorithm is described in lecture 24, as well as in page 561, Section 17.5.2 in the textbook.

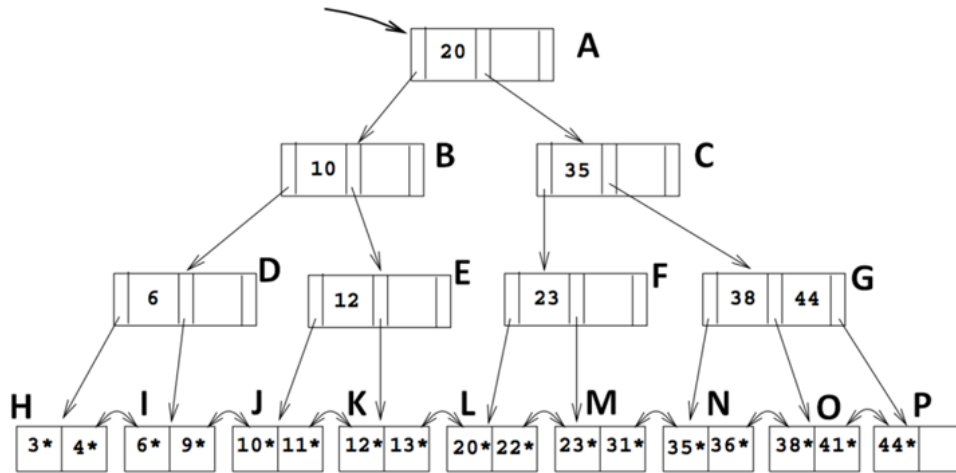


Figure 1: A sample B^+ tree

For each of the following transactions give the sequence of lock/unlock requests. As in question 1, use the notations S(.), X(.), and U(.) to denote **S**hared lock, **eX**clusive lock, and **U**nlock, respectively.

- (a) **T1**: Search for the data entry 25^*
- (b) **T2**: Insert the data entry 39^*
- (c) **T3**: Insert the data entry 59^*
- (d) **T4**: Delete the data entry 13^*

Handout continues on the next page(s)

4 Recovery using ARIES [30 Points]

Consider the execution history shown in **Figure 2**. *In addition*, the system crashes during recovery after writing two log records to stable storage and **again** after writing another log record. Assume that we run the **ARIES** algorithm to recover from crashes. Answer the following questions:

LSN	LOG
00	begin_checkpoint
10	end_checkpoint
20	update: T1 writes P1
30	update: T2 writes P2
40	update: T3 writes P3
50	update: T4 writes P4
60	T4 commit
70	T2 commit
80	update: T3 writes P2
90	T2 end
100	update: T1 writes P5
110	T3 abort
	CRASH, RESTART

Figure 2: Execution with Multiple Crashes

- What is done during the **Analysis** phase? In particular, show how the records in the **Dirty Page** and the **Transaction tables** are populated/altered/deleted during **Analysis** phase.
- What is done during the **Redo** phase? In particular, show how the **ARIES** algorithm proceeds with and finishes the **Redo** phase. Also, describe an execution that illustrates the use of the first condition in the **Redo** phase.
- What is done during the **Undo** phase? In particular, show how the **ARIES** algorithm proceeds with and finishes the **Undo** phase.
- Show the log when recovery is complete, including all non-null **prevLSN** and **undoNextLSN** values in log records.