CAS CS 660: Grad. Intro to Database Systems
Data-intensive Systems and Computing Lab
Department of Computer Science
College of Arts and Sciences, Boston University
http://bu-disc.github.io/CS660/

CS660 Fall 2024 – Written Assignment 6

Title: *Transactions, Locking, Concurrency Control, Recovery*

Problem 1: Transactions. [25pts]

For each of the following schedules, draw the precedence graph. Explain whether or not the schedule is conflict serializable. If the schedule is conflict serializable, give one possible equivalent serial schedule.
Note that, R_i denotes transaction *i* reads an item, while W_i denotes a write.

- $R_1(A)$ - $W_1(A)$ - $R_2(B)$- $R_2(C)$ - $W_3(A)$ - $W_2(B)$ - $R_4(B)$ - $R_4(A)$ - $R_4(E)$ - $R_2(A)$ - $R_3(B)$ - $W_1(B)$
- $R_1(A)$ - $R_4(B)$ - $W_1(B)$ - $W_3(A)$ - $R_2(B)$ - $R_2(A)$ - $R_4(C)$ - $R_4(B)$ - $R_2(A)$ - $R_3(C)$

Problem 2: Concurrency Control. [25pts]

Assume we have a strict 2PL protocol. For each of the following schedules, insert the appropriate locks (both shared and exclusive). Additionally, explain what happens to the scheduler as each schedule is being executed.
Note that, R_i denotes transaction *i* reads an item, while W_i denotes a write. If a transaction blocks because of an operation, the transaction with the next operation in the schedule will continue. When a transaction unblocks, it resumes its operations.

If you have a deadlock, you need to choose a transaction to abort and follow all the proper protocols to allow the rest of the schedule to continue. Additionally, aborted transactions will be required to restart again at some point along the schedule.

Lastly write the *actual* executed schedule at the end.

- $R_1(A) - R_2(A) - W_3(C) - R_1(B) - W_2(A) - W_3(B)$
- $R_1(A) - R_2(B) - R_3(C) - R_1(B) - R_2(C) - R_3(A) - W_1(A) - W_2(B) - W_3(D)$

Problem 3: Locking. [25pts]

Consider a database organized in terms of the following hierarchy of objects: The database itself is an object (D), and it contains two files (F1 and F2), each of which contains 1000 pages (P1 … P1000 and P1001 … P2000, respectively). Each page contains 100 records, and records are identified as p:i, where p is the page identifier and i is the slot of the record on that page. The system uses multiple-granularity locking, with S, X, IS, IX and SIX locks, and database-level, file-level, page-level and record-level locking. For each of the following operations, indicate the sequence of lock requests that must be generated by a transaction that wants to carry out (just) these operations:

1. Read record P1200:5.
2. Read records P1200:98 through P1205:2.
3. Read all (records on all) pages in file F1.
4. Read pages P500 through P520.
5. Read pages P10 through P980.
6. Read all pages in F1 and (based on the values read) modify 10 pages.
7. Delete record P1200:98. (This is a blind write.)
8. Delete the first record from each page. (Again, these are blind writes.)
9. Delete all records.

Problem 4: Recovery. [25pts]

Consider the log below

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

In this log, we store information about 3 transactions. After the log record with LSN 100, the system crashes and then we restart. We use the ARIES recovery algorithm discussed in Chapter 18 in the book. Based on that, answer the following questions:

a. What is done during the Analysis phase?
b. What is done during the Redo phase?
c. What is done during the Undo phase?"
d. Show the log when recovery is complete, including all non-null prevLSN and undonextLSN values in the log records.