Advanced Database Systems - CSCI-GA.2434-001 - Fall 2022

Professor: Dennis Shasha

Homework 1 - Due: Friday, 4:30 PM October 21, 2022

Each question is worth 10 points. You may work with one partner and sign both of your names to your paper. Please submit as follows:

```
- Sign in to https://brightspace.nyu.edu with your NYU NetID and password.
- Under the 'My Courses' section, select the course 'Advanced Database Systems CSCI-GA.2434-00
- Navigate to the 'Assignment' tab.
- Please submit your solution as a .pdf file/.zip file.
```

1. For each of the following example executions, determine if it is serializable, assuming each active transaction ultimately commits. Say why or why not.
   (a) $\text{Read}_1(x)$; $\text{Write}_2(x)$; $\text{Write}_2(y)$; $\text{Read}_3(y)$; $\text{Read}_3(z)$
   (b) $\text{Read}_1(x)$; $\text{Write}_2(x)$; $\text{Write}_2(y)$; $\text{Read}_3(y)$; $\text{Read}_3(z)$; $\text{Write}_1(z)$
   (c) $\text{Read}_1(x)$; $\text{Write}_2(x)$; $\text{Write}_2(y)$; $\text{Read}_3(y)$; $\text{Write}_1(z)$; $\text{Read}_3(z)$
   (d) $\text{Read}_1(x)$; $\text{Write}_2(y)$; $\text{Read}_1(y)$; $\text{Write}_2(z)$; $\text{Read}_3(z)$; $\text{Write}_3(z)$; $\text{Read}_1(z)$

2. Try to complete the proof of the serialization graph theorem by showing the final write portion.

3. Consider the two phase locking protocol with reads and writes. Is two phase locking deadlock-free? If so, prove it. If not, illustrate a deadlock situation. That is, you must illustrate a situation in which there is a cycle in the waits-for graph.

4. In class, we discussed strict two phase locking which is two phase locking with the added constraint that all locks must be released at the end of the transaction and a transaction acquires locks immediately before it first accesses a database item. Suppose you remove these added constraints (so locks can be released earlier than the end of the transaction provided no new locks are acquired afterwards and locks may be obtained way before they are needed), giving "pure" two phase locking.

   An execution is order-preserving serializable if it is serializable and the following holds: whenever all operations of some transaction $T1$ precede all operations of some other transaction $T2$ (i.e. whenever $T1$ completes before $T2$ begins), then there exists an equivalent serial execution in which $T1$ precedes $T2$. Prove that pure two phase locking produces order-preserving serializable executions.

5. Imagine an intention locking protocol in which normal locks are acquired in a two phase manner, but intention locks can be released at any time. Would such a protocol be serializable? If so, prove it. If not, show a counterexample.

6. Recall the Kung and Robinson optimistic concurrency control algorithm as discussed in class. How would you modify that algorithm to ensure that every transaction eventually completes (i.e., no transaction is restarted forever)?

7. Consider a B+ tree allowing splits and free-on-empty. In structure, this is a standard B+ tree (no "horizontal" links, only parent to child pointers). Free-on-empty means that the B+ tree does not use merges, but rather waits until a node is empty before freeing it. Different processes execute on the B+ tree in a concurrent fashion and a process may delay arbitrarily long. Assume that each split works as follows: lock a node n, transfer the highest half of the keys from n to some newly allocated node n', adjust the fences (representing the insets) appropiately, release lock on n, lock parent(n), change pointers and key separators in parent(n) appropriately, and release lock on parent(n). Assume that free-on-empty works like this: lock(n), confirm that n is empty, set its fence to the empty set (this is a shorthand for the following operation: the fence field in node n contains what the inset should be; normally that value is an interval between *a* and *b* where *a* and *b* are the separators of parent(n); in this case however we will set the fence to the empty set as a signal to searchers that this node will soon have an empty keyset), release lock on n, lock parent(n), adjust pointers in parent(n), release lock on parent(n). Here are your tasks: (a) Please show an example of these operations on a data structure containing 15 data items (all data itmes in a B+ tree are at the leaves), a fanout of three, and at most three data items per node. The two structure-changing operations are split and free-at-empty. (b) Specify the search algorithm (hint: use a give-up scheme, but you don't necessarily need to give up by restarting at the root; state where to give up to). (c) Show that the structure and search invariant conditions hold as specified in the lecture notes on concurrent search structure algorithms. (d) Say when to free a node that has had its inset(n) set to empty.

8. Show that the multiversion read consistency algorithm ensures serializability. That is, read-only transactions use the multiversion technique whereas read-write transactions use strict two phase locking.

9. In the Oracle database system, transactions (including read-write ones) use multiversion read consistency for any SELECT statement, but acquire exclusive row locks for SELECT FOR UPDATE statements and hold those locks until the end of the transaction (including an end-of-file lock to prevent phantoms). Suppose some transactions in an execution issue SELECT statements followed by INSERT statements and issue no SELECT FOR UPDATE statements. (INSERT statements also acquire exclusive locks on the data items they insert and hold those locks until the end of the transaction.) Can this lead to a non-serializable execution? If so, show one. Otherwise, say why not.

10. (AQuery) (i) Given a schema ticks(ID, date, endofdayprice) find the maximum and minimum return of each stock where return is the ratio of the price of day i compared with the price of day i-1 (for running ratios, you can use the "ratios" function). Use AQuery built-ins wherever possible. If you would like examples of ratios, please see the links from the course website.

    Example of this query:

```
ticks(ID, date, endofdayprice)
s1, 1, 10
s2, 1, 100
s1, 2, 13
s2, 2, 200
s1, 3, 10
```

就是和sql不一样的是他可以排序
ASSUMING ASC date 这个就是按照date排序
max和min就是最大最小值

```
s2, 3, 150
s1, 4, 15
s2, 4, 140

Max return for s1 is 15/10.
Min return for s1 is 10/13.
Max return for s2 is 200/100
Min return for s2 is 150/200.
```

(ii) Assuming ticks were already sorted by ID (but not necessarily by time), how would you process the above query if you had complete control of the optimization process?