CARNEGIE MELLON UNIVERSITY
COMPUTER SCIENCE DEPARTMENT
15-445/645 – DATABASE SYSTEMS (SPRING 2023)
PROF. CHARLIE GARROD

Homework #4 (by Abigale Kim)
Due: **Friday April 7, 2023 @ 11:59pm**

IMPORTANT:

- Enter all of your answers into **Gradescope by 11:59pm on Friday April 7, 2023**.
- **Plagiarism**: Homework may be discussed with other students, but all homework is to be completed **individually**.

For your information:

- Graded out of **100** points; **4** questions total
- Rough time estimate: $\approx$ 2 - 4 hours (0.5 - 1 hours for each question)

Revision : 2023/03/27 16:09

Question	Points	Score
Query Execution, Planning, and Optimization	25	
Serializability and 2PL	29	
Hierarchical Locking	20	
Multi-Version Concurrency Control	26	
Total:	100	

Question 1: Query Execution, Planning, and Optimization . . . . . [25 points]

(a) **[5 points]** The zone map optimization is more useful in speeding up OLAP queries compared to OLTP queries.
☐ True ☐ False

(b) **[5 points]** The thread per worker process model allows for the opportunity of using intra-query parallelism, whereas the process per worker process model does not.
☐ True ☐ False

(c) **[5 points]** The vectorized query processing model (which uses SIMD instructions to parallelize operations) is an example of intra-query parallelism.
☐ True ☐ False

(d) **[5 points]** An index scan is always better (fewer I/O operations, faster run-time) than a heap scan if the query contains an `ORDER BY` clause matching the index key.
☐ True ☐ False

(e) **[5 points]** Database management systems estimate the cost of every plan for a query, to pick the optimal plan for execution.
☐ True ☐ False

Question 2: Serializability and 2PL.....[29 points]

(a) True/False Questions:

i. **[2 points]** One downside of regular (i.e., not strong strict) 2PL is that it can lead to a domino effect, where multiple transactions are aborted.
☐ True ☐ False

ii. **[2 points]** Using strong strict 2PL guarantees a conflict serializable schedule.
☐ True ☐ False

iii. **[2 points]** Using regular (i.e., not strong strict) 2PL guarantees a conflict serializable schedule.
☐ True ☐ False

iv. **[2 points]** View serializable schedules never produce unrepeatable reads.
☐ True ☐ False

v. **[2 points]** Strong strict 2PL prevents deadlocks during transaction execution.
☐ True ☐ False

vi. **[2 points]** Regular (i.e., not strong strict) 2PL prevents the lost update problem.
☐ True ☐ False

(b) Serializability:
Consider the schedule of 4 transactions in Table 1. R(·) and W(·) stand for 'Read' and 'Write', respectively, and time increases from left to right. (This is in contrast to the diagrams in class, where time proceeded downward.)

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
T_1		W(B)		W(A)					R(C)	
T_2	W(E)							R(A)		
T_3			R(B)							R(D)
T_4					R(D)	R(E)	W(C)			

Table 1: A schedule with 4 transactions

i. **[2 points]** Is this schedule serial?
☐ Yes ☐ No

ii. **[2 points]** Is this schedule serializable?
☐ Yes ☐ No

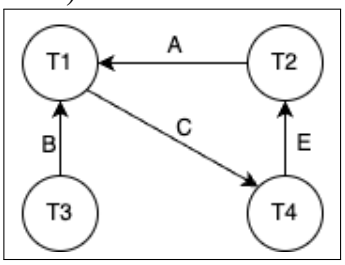
iii. **[2 points]** Is this schedule conflict serializable?
☐ Yes ☐ No

iv. **[2 points]** Is this schedule view serializable?
☐ Yes ☐ No

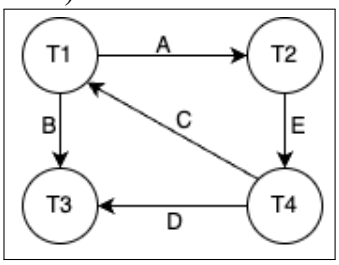
Question 2 continues...

v. **[4 points]** Choose the correct dependency graph of the schedule given above. Each edge in the dependency graph looks like this: '$T_x \rightarrow T_y$ with Z on the arrow indicating that there is a conflict on Z where T_x read/wrote on Z before T_y'.

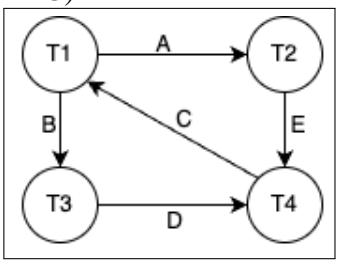
☐ A)



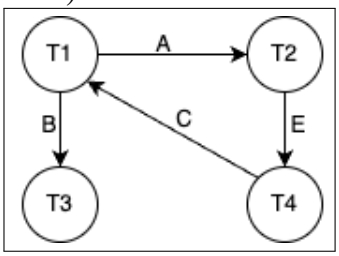
☐ B)



☐ C)



☐ D)



vi. **[3 points]** Select all possible changes that would produce a conflict serializable schedule.

☐ Remove only `T1`
☐ Remove only `T2`
☐ Remove only `T3`
☐ Remove only `T4`
☐ Original schedule is already conflict serializable, remove no transactions

vii. **[2 points]** Is this schedule possible under regular 2PL?

☐ Yes
☐ No

Question 3: Hierarchical Locking . . . . . [20 points]

Consider a database D consisting of two tables A (which stores information about musical artists) and R (which stores information about the artists' releases). Specifically:

- R(rid, name, artist_credit, language, status, genre, year, number_sold)
- A(id, name, type, area, gender, begin_date_year)

Table R spans 1000 pages, which we denote R1 to R1000. Table A spans 50 pages, which we denote A1 to A50. Each page contains 100 records. We use the notation R3.20 to denote the twentieth record on the third page of table R. There are no indexes on these tables.

Suppose the database supports shared and exclusive hierarchical intention locks (S, X, IS, IX and SIX) at four levels of granularity: database-level (D), table-level (R and A), page-level (e.g., R10), and record-level (e.g., R10.42). We use the notation IS(D) to mean a shared database-level intention lock, and X(A2.20-A3.80) to mean a set of exclusive locks on the records from the 20th record on the second page to the 80th record on the third page of table A.

For each of the following operations below, what sequence of lock requests should be generated to maximize the potential for concurrency while guaranteeing correctness?

(a) **[4 points]** Fetch the record where artist `name = 'Taylor Swift'`.
- ☐ IX(D), X(A)
- ☐ S(D)
- ☐ IS(D), IS(A)
- ☐ IS(D), S(A)

(b) **[4 points]** Modify the 15^{th} record on R645.
- ☐ IX(D), IX(R), IX(R645), IX(R645.15)
- ☐ IX(D), IX(R), IX(R645), X(R645.15)
- ☐ SIX(D), SIX(R), SIX(R645), IX(R645.15)
- ☐ IS(D), IS(R), IS(R645), X(R645.15)

(c) **[4 points]** Scan all the records on pages A41 through A49, and modify A32.75.
- ☐ IS(D), IS(A), S(A41-A49), IX(A32), IX(A32.75)
- ☐ IS(D), IS(A), S(A41-A49), IX(A32), X(A32.75)
- ☐ IX(D), SIX(A), IX(A32), X(A32.75)
- ☐ SIX(D), SIX(A), IX(A32), X(A32.75)

(d) **[4 points]** Scan all records in R and modify the 23^{rd} record on R7.
- ☐ IX(D), SIX(R), IX(R7), X(R7.23)
- ☐ IS(D), IS(R), IS(R7), X(R7.23)
- ☐ S(D), IX(R), X(R7.23)
- ☐ IS(D), SIX(R), X(R7.23)

(e) **[4 points]** Delete records in R if `number_sold < 3`.
- ☐ SIX(D), S(R)
- ☐ SIX(D), X(R)
- ☐ IX(D), X(R)
- ☐ IX(D), SIX(R)

Question 4: Multi-Version Concurrency Control . . . . . [26 points]

Scout is a database management system designer who has decided to implement multi-versioning for her system. She is now contemplating design decisions given the current workloads she wants to run, given the following constraints:

1. Most of the data needed in this database has already been loaded.
2. The DBMS is optimized for the speed of highly-concurrent reads.
3. I/O is expensive on this DBMS platform. The DBMS should minimize number of I/O requests per read.
4. Reads often access snapshots of the database taken at different times.
5. The database contains secondary indexes to help speed up read performance.

For each of the design decisions below, provide a brief response ($\sim$3 sentences) that explains which approach Scout should use, along with the pros and cons of that approach.

(a) **[6 points]** Concurrency Control (Two-Phase Locking, Timestamp Ordering, or Optimistic Concurrency Control)

(b) **[6 points]** Version Storage (Append-Only Storage, Time-Travel Storage, or Delta Storage)

(c) **[6 points]** Garbage Collection (Background Vacuuming, Cooperative Cleaning, or Transaction-level GC)

(d) **[6 points]** Index Management (Logical Pointers or Physical Pointers)

(e) **[2 points]** In a DBMS with MVCC implemented, read-only transactions can block write transactions.

☐ True ☐ False