CARNEGIE MELLON UNIVERSITY
COMPUTER SCIENCE DEPARTMENT
15-445/645 – DATABASE SYSTEMS (SPRING 2025)
PROF. JIGNESH PATEL

Homework #5 (by Aditya Bhatnagar)

Due: **Sunday April 06, 2025 @ 11:59pm**

IMPORTANT:

- Enter all of your answers into **Gradescope by 11:59pm on Sunday April 06, 2025**.
- **Plagiarism**: Homework may be discussed with other students, but all homework is to be completed **individually**.

For your information:

- Graded out of **100** points; **3** questions total
- Rough time estimate: ≈ 2 - 4 hours (0.5 - 1 hours for each question)

Revision : 2025/03/25 10:50

Question	Points	Score
Serializability and 2PL	34	
Hierarchical Locking	30	
Optimistic Concurrency Control	36	
Total:	100	

Question 1: Serializability and 2PL ....[34 points]

(a) True/False Questions:

i. **[3 points]** Strong strict Two-Phase Locking (2PL) prevents the occurrence of cascading aborts and inherently avoids deadlocks without the need for additional prevention or detection techniques.
☐ True ☐ False

ii. **[3 points]** Using regular (i.e., not strong strict) 2PL does not guarantee a conflict serializable schedule.
☐ True ☐ False

iii. **[3 points]** Conflict-serializable schedules prevent unrepeatable reads and dirty reads.
☐ True ☐ False

iv. **[3 points]** 2PL is a pessimistic concurrency control protocol.
☐ True ☐ False

v. **[3 points]** Every conflict-serializable schedule is always conflict-equivalent to at least one view-serializable schedule.
☐ 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(C)				R(D)
T_2			W(A)	W(B)						
T_3		R(C)			R(A)					
T_4							W(B)	R(D)	W(A)	

Table 1: A schedule with 4 transactions

i. **[3 points]** Is this schedule serial?
☐ Yes ☐ No

ii. **[3 points]** Is this schedule conflict serializable?
☐ Yes ☐ No

iii. **[3 points]** Is this schedule view serializable?
☐ Yes ☐ No

iv. **[7 points]** Compute the conflict dependency graph for the schedule in Table 1, selecting all edges (and the object that caused the dependency) that appear in the graph.

☐ $T_1 \rightarrow T_2$ ☐ $T_2 \rightarrow T_3$ ☐ $T_2 \rightarrow T_4$
☐ $T_2 \rightarrow T_1$ ☐ $T_3 \rightarrow T_2$ ☐ $T_4 \rightarrow T_2$
☐ $T_1 \rightarrow T_3$ ☐ $T_1 \rightarrow T_4$ ☐ $T_3 \rightarrow T_4$
☐ $T_3 \rightarrow T_1$ ☐ $T_4 \rightarrow T_1$ ☐ $T_4 \rightarrow T_3$

v. **[3 points]** Is this schedule possible under regular 2PL?
☐ Yes
☐ No

Question 2: Hierarchical Locking .... [30 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 2000 pages, which we denote R1 to R2000. Table A spans 100 pages, which we denote A1 to A100. Each page contains 200 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]** Update the type of all musical artists in table A with the name = 'Eminem' to 'Slim Shady'
- ☐ X(D)
- ☐ IX(D), X(A)
- ☐ SIX(D), X(A)
- ☐ IX(D), IX(A)

(b) **[5 points]** Find the release record whose year equals the oldest year in all releases.
- ☐ S(D)
- ☐ IS(D), IS(R)
- ☐ IX(D), X(R)
- ☐ IS(D), S(R)

(c) **[5 points]** Increment the number_sold for the 10^{th} record on R1999.
- ☐ IX(D), IX(R), IX(R1999), IX(R1999.10)
- ☐ IX(D), IX(R), IX(R1999), X(R1999.10)
- ☐ SIX(D), SIX(R), SIX(R1999), X(R1999.10)
- ☐ IS(D), IS(R), IS(R1999), X(R1999.10)

(d) **[5 points]** Scan all records between R10 and R100 and modify the 2^{nd} record on R50
- ☐ IX(D), SIX(R), IX(R50), X(R50.2)
- ☐ IS(D), S(R), IX(R50.2)
- ☐ IX(D), IX(R), IX(R10-R100), IX(R50), X(R50.2)
- ☐ IS(D), SIX(R), X(R50)

(e) **[5 points]** Delete records in A if type = 'Band'.
- ☐ SIX(D), SIX(A)
- ☐ IX(D), X(A)

- ☐ `IX(D), IX(A)`
- ☐ `SIX(D), X(A)`

(f) **[6 points]** Two users are trying to access data. User A is scanning all the records in `R` to read, while User B is trying to modify the 15^{th} record in `A10`. Which of the following sets of locks are most suitable for this scenario?

- ☐ `User A: SIX(D), S(R), User B: SIX(D), IX(A), X(A10.15)`
- ☐ `User A: S(D), User B: X(D)`
- ☐ `User A: IS(D), S(R), User B: IX(D), IX(A), X(A10.15)`
- ☐ `User A: IS(D), S(R), User B: SIX(D), IX(A), X(A10.15)`

Question 3: Optimistic Concurrency Control . . . . [36 points]

Consider the following set of transactions accessing a database with object *A, B, C, D*. The questions below assume that the transaction manager is using **optimistic concurrency control** (OCC). Assume that a transaction begins its read phase with its first operation and switches from the READ phase immediately into the VALIDATION phase after its last operation executes.

Note: VALIDATION may or may not succeed for each transaction. If validation fails, the transaction will get immediately **aborted**.

You can assume that the DBMS is using the serial validation protocol discussed in class where only one transaction can be in the validation phase at a time, and each transaction is doing **backward validation** (i.e. Each transaction, when validating, checks whether it intersects its read/write sets with any transactions that have already committed. You may assume there are no other transactions in addition to the ones shown below.)

time	T_1	T_2	T_3
1		READ(A)	
2	READ(A)		
3			
4	WRITE(A)	WRITE(A)	
5		WRITE(B)	
6		WRITE(C)	
7		VALIDATE?	
8	READ(B)	WRITE?	READ(C)
9	VALIDATE?		
10	WRITE?		
11			
12			
13			WRITE(C)
14			
15			READ(D)
16			WRITE(D)
17			VALIDATE?
18			WRITE?

Figure 1: An execution schedule

(a) **[5 points]** When is each transaction's timestamp assigned in the transaction process?
- ☐ The start of the write phase.
- ☐ Timestamps are not necessary for OCC.
- ☐ The start of the validation phase.
- ☐ The start of the read phase.

(b) **[5 points]** When time = 8, will T_3 read C written by T_2?
☐ Yes ☐ No

(c) **[5 points]** Will T1 abort?
☐ Yes
☐ No

(d) **[5 points]** Will T2 abort?
☐ Yes
☐ No

(e) **[5 points]** Will T3 abort?
☐ Yes
☐ No

(f) **[3 points]** OCC works best when concurrent transactions access the same subset of data in a database.
☐ True ☐ False

(g) **[3 points]** Transactions can suffer from *phantom reads* in OCC.
☐ True ☐ False

(h) **[3 points]** Aborts due to OCC are wasteful because they happen after a transaction has already finished executing.
☐ True ☐ False

(i) **[2 points]** All Objects in the read-set of a transaction are read as a whole when the transaction starts.
☐ True ☐ False