CS143, Fall 2017: Final Exam Answer Sheet

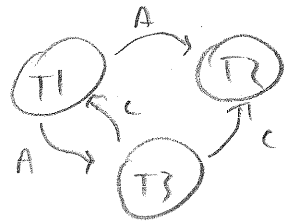
Please write all your answers in the designated area of each problem clearly. Circle your choice and explain if required. Do not use the answer sheet as scratch paper. Do not write in the back.

UID: 404-640-158 NAME: Jason Less

## Problem A

1. Yes **No** Explanation:

- Can display this with a precedence graph ⇒
- If acyclic ⇒ then conflict serializable

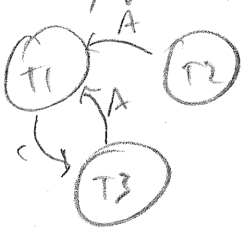



As a cycle exists, it is not conflict serializable

2. · Under Strict 2PL ⇒ transactions won't release lock-X's until the end of the transaction

| T1 | T2 | T3 |
|---|---|---|
| lock-X(A) | | |
| write A | | |
| | wait for lock-S(A) | |
| | | lock-X(C) |
| | | write(C) |
| wait for lock-S(C) | | |
| | | wait for lock-S(A) |

· Wait-for graph ⇒ If cycle, then the schedule is in deadlock




· Cycle exists, so $\boxed{\text{deadlock exists}}$

· As T1 and T3 are waiting for locks held by each other

3. · With wound-wait ⇒ · Older transactions will wound (rollback) younger transactions
· Younger transactions can wait for data items held by older transactions

· Now, the $\boxed{\text{schedule will complete.}}$ Instead of waiting for a lock on C, T1 will rollback T3, and grab the necessary locks and finish

· T2 will wait for T1 to release locks and finish, and upon restarting (depending on when), T3 will either wait for locks held by T2 and then finish. So, the order they will complete is T1, T2, T3

4. **Yes** No Explanation:

· For a schedule to be cascadeless ⇒ If $T_j$ reads a data item written previously by $T_i$, then $T_i$ must commit before $T_j$ reads it. As the schedule in Q3 is under strict 2PL, transactions won't release write locks (lock-X's) until the end of their transaction. In addition, as it is wound-wait, younger transactions will wait for the locks held by older transactions. Therefore, it will guarantee cascadeless schedules, as the older transactions will release their locks at the end of their transactions, and then commit before the younger ones will read that data item.

~~5.~~

5) · Timestamp-based protocol ⇒

| T1 | T2 | T3 |
|---|---|---|
| start | | |
| write(A) | | |
| | start | |
| | | start |
| | read(A) | |
| | | write(C) |
| read(C) | | |
| | read(C) | |
| write(B) | | |
| | | read(A) |

① W-TS(A) = TS(T1)

② TS(T2) > W-TS(A), so read executes, R-TS(A) = TS(T2)

③ W-TS(C) = TS(T3)

④ TS(T1) < W-TS(C), so value of C that $T_1$ wanted was already overwritten. So read rejected, T1 rolled back

⑤ TS(T2) < W-TS(C), so read rejected, T2 rolled back

⑥ R-TS(A) = TS(T3)

· T3 will then commit

~~6. Yes No~~

~~7.~~

6) **Yes** No ⇒ Timestamp-based protocols prevent deadlock

7) · This is to $\boxed{\text{prevent starvation}}$ ⇒ Thus, with competition to acquire locks in the future, older transactions can wait for locks from younger ones, instead of continuously being rolled back

UID: 404-640-158 NAME: Jason Less

Problem B

1. True False

2. True False

3. True False

4. True False

5. True False

6. True False

7. Yes No

Explanation: - A relation is 3NF if for all FDs either (1) Trivial or (2) left side is a candidate key (3) Right side is part of a candidate key

- As the FDs for the given schema either have the left-side as candidate keys, or the right sides are part of the candidate key

8. - Will require the updating of all records associated with the 629 cities

9. Cities(sname, cname, mayor)

States(sname, popl)

cities. sname → states.sname

10.

UID: 404-640-158 NAME: Jeremy Less

## Problem C

**1.** True False

Explanation:
- The checkpoint's main purpose is to streamline the recovery process
  - The goal of the checkpoint is to avoid having to read back the whole log during recovery, and redo/undo all transactions
    - Instead, a checkpoint makes it so that committed transactions before the checkpoint are ignored
- Therefore, cascading of rollbacks can be prevented if a checkpoint is put in place (thus committing transactions before)
  - Thus, even though it isn't its main goal, checkpoints help manage cascading rollbacks

**2.** True False

Explanation:
- A checkpoint consists of a checklist of active transactions that have not yet committed
  - Therefore, recovery can ignore already committed transactions ⇒ streamlining recovery
- However, the decision to commit/abort is not up to the DB administrator, but the procedure of the redo/undo phases of the recovery process
  - If a transaction (at a given system failure) is active at the latest checkpoint, it will be undone if no commit/abort instruction was found
  - Thus, DB administrator doesn't decide on commit/aborts

**3.** True False

Explanation:
- For reasons discussed in (1) and (2), the main function is to expedite recovery
  - Ignoring transactions committed before the last checkpoint
  - Only considering active transactions

**4.** True False

Explanation:
- Execution of a checkpoint forces the buffer for all running transactions to be written out to disk
  - But it doesn't actually force them to commit
  - If a crash occurred, then those transactions could potentially be undone

**5.** True False

Explanation:
- Yes, it can still complete recovery
  - The loss of checkpoints will however, slow the recovery process down
  - As, the recovery may have to look further for the "latest" checkpoint (not damaged), and may repeat recovery on some transactions that had already committed (i.e. redo them)

UID: 404-690-158 NAME: Jason Less

## Problem D

1. Yes **No**

Explanation: • For a relation to be BCNF, each FD must either be (1) trivial or (2) the left-hand side is a candidate key

$R = ABCDE$
$Key(R) = AC, BC$
$AC \to B$ is fine (doesn't violate BCNF)
$B \to A$ violates BCNF as $B \to A$ is not trivial nor is B a candidate key

2. No, $\boxed{\text{not canonical}}$ as $C \to DE$ has 2 attributes on the right side

• $C \to DE$ can be replaced with $C \to D$, $C \to E$ ⇒ Thus, FDs ⇒
(1) $AC \to B$
(2) $B \to A$
(3) $D \to E$
(4a) $C \to D$
(4b) $C \to E$

3. • lossless if (1) The attributes of the union of the decompositions has all attributes in R, and
(2) The intersection of the attributes of the decomposition relations is a FD to any of the decomposition relations ⇒ (i.e. if R decomposes to R1 and R2, then (1) $R1 \cup R2 = R$ (2) $R1 \cap R2 \to R1$ or $R1 \cap R2 \to R2$
(3) The intersection is not the empty set

$R = ABCDE$
$Key(R) = AC, BC$

$B+ = AB$
$R1(AB)$ ✓
$Key(R1) = B$

$R2(BCDE)$
$Key(R2) = BCDE$
$D+ = DE$
$R3(DE)$ ✓
$Key(R3) = D$

$R4(BCD)$
$Key(R4) = BCD$
$R5(CD)$ $R6(BC)$

⇒ Decomposition = $\boxed{R1(AB), R3(DE), R5(CD), R6(BC)}$
(1) $R1 \cup R3 \cup R4 = ABCDE = R$ ✓
(2) $R1 \cap R3 \cap R4 = \phi$, and

only lossy decomposition exists

4. Yes **No**

• $AC \to B$ and $D \to E$ are lost

## Extra Credit Problems

1. A **B** C

2. A B C **D**

3. A B **C** D

UID: 424-690-158

NAME: Faron Less

(Extra sheet if you need to use)