CS 143 Final Exam

JASON LESS

TOTAL POINTS

89 / 100

QUESTION 1

Problem A 28 pts

1.1 Problem A Q1 **4 / 4**

✓ **- 0 pts** Correct

- 4 pts Wrong

1.2 Problem A Q2 **4 / 4**

✓ **- 0 pts** Correct

- 1 pts Minor mistake in wait-for graph. It should have and only have 3 edges. T2 -> T1, T1 -> T3, T3 -> T1, any missing edge/extra edge/wrong arrow direction is a mistake

- 2 pts Answer is correct. But the question is to prove deadlock by the appropriate graph. This graph means the wait-for graph. It should have and only have 3 edges. T2 -> T1, T1 -> T3, T3 -> T1.

- 4 pts Wait-for graph is incorrect. It should have and only have 3 edges. T2 -> T1, T1 -> T3, T3 -> T1.

1.3 Problem A Q3 **4 / 4**

✓ **- 0 pts** Correct

- 2 pts The answer is partially correct. But miss the key points to point out that T1 wounds (forces rollback of) T3 when read(C). After that, T1 complete and release its X-lock on A, whereby T2 also complete. T3 can restart and also complete.

- 4 pts The correct answer is T1 wounds (forces rollback of) T3 when read(C). After that, T1 complete and release its X-lock on A, whereby T2 also complete. T3 can restart and also complete.

1.4 Problem A Q4 **4 / 4**

✓ **- 0 pts** Correct

- 4 pts Correct answer should be yes, strict 2PL can avoid cascading rollbacks because it prevents dirty read (because all x-locks are released only after transaction commit/abort)

- 2 pts Wrong explanation. Correct explanation is strict 2PL can avoid cascading rollbacks because it prevents dirty read (because all x-locks are released only after transaction commit/abort). Any explanation related to timestamps, transaction orders, deadlocks, wound-wait, using ad-hoc examples is not correct.

- 1 pts Explanation may only give a definition of the cascadeless rollbacks. Correct explanation is strict 2PL can avoid cascading rollbacks because it prevents dirty read (because all x-locks are released only after transaction commit/abort).

1.5 Problem A Q5 **4 / 4**

✓ **- 0 pts** Correct

- 3 pts No deadlock. But the correct result schedule is: T1 write(A) Wts(A)=T1; T2 read(A) Rts(A)= T2; T3 write(C) Wts(C)=T3; T1 read(C) and abort; T2 read(C) and abort; T3 read(A) and complete; Then T1, T2 finish.

- 2 pts Correct. But the schedule is not shown.

- 4 pts No deadlock. The correct result schedule is: T1 write(A) Wts(A)=T1; T2 read(A) Rts(A)= T2; T3 write(C) Wts(C)=T3; T1 read(C) and abort; T2 read(C) and abort; T3 read(A) and complete; Then T1, T2 finish.

1.6 Problem A Q6 **4 / 4**

✓ **- 0 pts** Correct

- 4 pts Wrong

1.7 Problem A Q7 **4 / 4**

✓ **- 0 pts** Correct

- 4 pts Wrong, not because deadlock, keeping orders, making it simple or consistent. It is because starvation

- **1 pts** Catch the similar idea, but not see the precise word -- starvation

QUESTION 2

Problem B 24 pts

2.1 Problem B Q1 2 / 2

✓ - **0 pts** **Correct**

- **2 pts** wrong

2.2 Problem B Q2 2 / 2

✓ - **0 pts** **Correct**

- **2 pts** wrong

2.3 Problem B Q3 2 / 2

✓ - **0 pts** **Correct**

- **2 pts** wrong

2.4 Problem B Q4 2 / 2

✓ - **0 pts** **Correct**

2.5 Problem B Q5 0 / 2

- **0 pts** Correct

✓ - **2 pts** **wrong**

2.6 Problem B Q6 2 / 2

✓ - **0 pts** **Correct**

- **2 pts** wrong

2.7 Problem B Q7 0 / 3

- **0 pts** Correct

- **1 pts** FDs should be cname, sname -> mayor and sname -> popl.
sname->popl violates 3NF. sname is not a key and popl is not contained in a key.

- **2 pts** sname->popl violates 3NF. sname is not a key and popl is not contained in a key.

✓ - **3 pts** **The answer is No.**

2.8 Problem B Q8 3 / 3

✓ - **0 pts** **Correct**

- **1 pts** 629 tuples should be updated

- **2 pts** 629 tuples should be updated rather than leave inconsistent data in database. Large amount of updates is the anomaly.

- **3 pts** For a state where the population changed, we have to update popl in N=629 tuples, where N denotes the number of cities in the state.

2.9 Problem B Q9 3 / 3

✓ - **0 pts** **Correct**

- **1 pts** citiesInSts(cname, sname, Mayor), States(sname, popl)

- **2 pts** citiesInSts(cname, sname, Mayor), States(sname, popl)

- **3 pts** citiesInSts(cname, sname, Mayor), States(sname, popl)

2.10 Problem B Q10 0 / 3

- **0 pts** Correct

- **1 pts** citiesInSts(cname, sname, Mayor): cname and sname are the key
States(sname, popl): sname is the key

- **1 pts** sname in citiesStates is a foreign key reference to State sname. It's incorrect with opposite reference or double direction reference.

- **2 pts** citiesInSts(cname, sname, Mayor): cname and sname are the key and sname is FK reference to States relation.
States(sname, popl): sname is the key

✓ - **3 pts** **citiesInSts(cname, sname, Mayor): cname and sname are the key and sname is FK reference to States relation.**
States(sname, popl): sname is the key

QUESTION 3

Problem C 20 pts

3.1 Problem C Q1 0 / 4

+ **2 pts** Correctly marked false

+ **2 pts** Explanation provides justification for answer. This may include: checkpoint role in (expediting) recovery, cascading rollbacks handled by transaction manager and/or concurrency control, checkpoints not having knowledge of dirty reads, or policies that affect cascading rollbacks (eg strict 2PL). This does not include an analysis that makes assumptions about the schedules, as no such assumptions were given in the question. Answers that only reword the problem statement (eg "Checkpoints do not manage cascading rollbacks") are not rewarded points.

✓ + **0 pts** **Incorrectly marked true**

3.2 Problem C Q2 **3 / 4**

✓ + **2 pts** **Correct option (FALSE).**

✓ + **2 pts** **Correct explanation that justifies answer. Explanation should mention at least one of: role in expediting recovery, automatic use by recovery manager, recovery manager responsibilities, lack of decision input on transaction commit/abort.**

+ **0 pts** Incorrect option selected (TRUE)

- 1 **Point adjustment**

Point taken off for unclear handwriting. Answer sheet instructions state that answers must be written clearly.

This one point deduction applies for all of problem C (ie you will not be deducted repeatedly on other problems).

3.3 Problem C Q3 **4 / 4**

✓ + **2 pts** **Correctly marked True**

✓ + **2 pts** **Correct explanation that mentions how checkpoints specifically speed up recovery (ie by reducing number of log records required to store/read, reducing redo operations, or reducing number of transactions to consider)**

+ **0 pts** Incorrectly marked False

3.4 Problem C Q4 **4 / 4**

✓ + **2 pts** **Correctly marked False**

✓ + **2 pts** **Correct justification, eg pointing out that checkpoints do not force transactions to commit.**

+ **0 pts** Incorrectly marked True

3.5 Problem C Q5 **4 / 4**

✓ + **2 pts** **True marked (correct)**

✓ + **2 pts** **Justification mentioning ability to skip checkpoints (eg using an older undamaged one) and/or how they are not necessary for recovery (but losing them may slow down the process)**

+ **0 pts** False marked (incorrect)

QUESTION 4

Problem D 28 pts

4.1 Problem D Q1 **7 / 7**

✓ - **0 pts** **Correct**

- **7 pts** Empty

4.2 Problem D Q2 **7 / 7**

✓ - **0 pts** **Correct**

- **7 pts** Empty

- **4 pts** Canonical form doesn't need to eliminate FDs using transitive rule

4.3 Problem D Q3 **7 / 7**

✓ - **0 pts** **Correct**

- **2 pts** Using AC->B as violation

- **4 pts** Incomplete decomposition

- **1 pts** Wrong consideration order of rules

- **2 pts** Using C->DE rather than canonical form

- **2 pts** Wrong result

- **7 pts** Empty

4.4 Problem D Q4 **7 / 7**

✓ - **0 pts** **Correct**

- **3 pts** Missing the lost FD

- **7 pts** Wrong Answer

QUESTION 5

Extra Credit 0 pts

5.1 EC Q1 **0 / 0**

+ **2 pts** Correct

✓ + **0 pts** **Incorrect Answer, Correct one should be A**

5.2 EC Q2 **2 / 0**

✓ **+ 2 pts** Correct

+ 0 pts Incorrect. Correct answer is D.

5.3 EC Q3 **0 / 0**

+ 2 pts Correct

✓ **+ 0 pts** Incorrect. Correct answer is B.

+ 0 pts Only B is correct. Multiple selection is not correct.

CS143, Fall 2017: Final Exam Answer Sheet

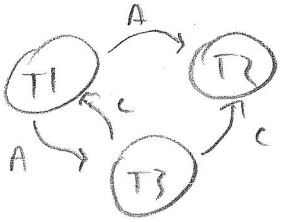
Please write all your answers in the designated area of each problem clearly. Circle your choice and explain if required. Do not use the answer sheet as scratch paper. Do not write in the back.

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

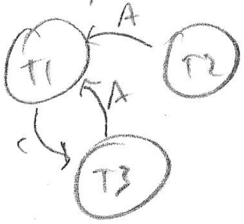
2. • Under Strict 2PL ⇒ transactions won't release lock-X's until the end of the transaction

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 de-dlock 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 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 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)

10. cities. sname → states.sname

UID: 404-640-158 NAME: Jeran 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 → B is fine (doesn't violate BCNF)
B → A violates BCNF as B → A is not trivial nor is B a candidate key

2. No, not canonical as C → DE has 2 attributes on the right side

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

3. • lossless if ① the attributes of the union of the decomposition 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 ∪ R2 = R ② R1 ∩ R2 → R1 or R1 ∩ R2 → 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 = R1(AB), R3(DE), R5(CD), R6(BC)
⇒ ① R1 ∪ R3 ∪ R4 = ABCDE = R ✓
② R1 ∩ R3 ∩ R4 = ∅, and

only lossy decomposition exists

4. Yes [✓ No]

• AC → B and D → 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: Jaron Less

(Extra sheet if you need to use)