CS 186 Fall 2022 Introduction to Database Systems

MIDTERM

INSTRUCTIONS

This is your exam. Complete it either at exam.cs61a.org or, if that doesn't work, by emailing course staff with your solutions before the exam deadline.

This exam is intended for the student with email address `<EMAILADDRESS>`. If this is not your email address, notify course staff immediately, as each exam is different. Do not distribute this exam PDF even after the exam ends, as some students may be taking the exam in a different time zone.

For questions with **circular bubbles**, you should select exactly *one* choice.

- ◯ You must choose either this option
- ◯ Or this one, but not both!

For questions with **square checkboxes**, you may select *multiple* choices.

- ☐ You could select this choice.
- ☐ You could select this one too!

You may start your exam now. Your exam is due at <DEADLINE> Pacific Time. Go to the next page to begin.

Preliminaries

1 CS 186 Fall 2022 Midterm 2

Unless you have special accommodations, you will have 110 minutes to complete the midterm. Do not share this exam until solutions are released.

1.0.1 Contents:

- The midterm has 7 questions, each with multiple parts, and worth a total of 100 points.

1.0.2 Aids:

- You may use 2 pages (double-sided) of handwritten notes as well as a calculator.
- You must work individually on this exam.

1.0.3 Grading Notes:

- All I/Os must be written as integers. There is no such thing as 1.02 I/Os – that is actually 2 I/Os.
- 1 KB = 1024 bytes. We will be using powers of 2, not powers of 10.
- Unsimplified answers, like those left in log format, will receive a point penalty.

(a) What is your full name?

(b) **(0.5 pt)** What is your student ID number?

(c) SID of the person to your left (Put None if you are taking the exam remotely):

(d) SID of the person to your right (Put None if you are taking the exam remotely):

1. (22 points) Join 186 Course Staff!

For all parts, assume table X has 50 pages and table Y has 10 pages. We have 5 buffer pages.

(a) (3 points) BNLJ

i. (1 pt) 186 TAs wish to join table X and table Y together. What is the minimum number of I/Os it would take to perform a Block Nested Loop Join on X and Y?

210

X BNLJ Y: 50 + ceil(50/3) * 10 = 220 Y BNLJ X: 10 + ceil(10/3) * 50 = 210

We take the smaller cost, with Y as our outer relation and X as our inner relation.

ii. (2 pt) The TAs decide to execute BNLJ again but with a magical RAM. Each time a block from the outer relation is evicted and a new block is read in, the available memory expands by 5 pages. For example, we have B = 5 for reading in the first block of outer relation. In the next iteration, we will have B = 10 to read in the second block, and so on. What is the total cost of X BNLJ Y, where X is the outer relation?

100

read 3 page block of X + 10 pages of Y, read 8 page block of X + 10 pages of Y,
read 13 page block of X + 10 pages of Y, read 18 page block of X + 10 pages of Y, read 8 page block of X + 10 pages of Y

(b) (11 points) SMJ

Raymond and Joy suggest that joining the tables using Sort Merge Join will be more efficient. Assume again that [X] = 50, [Y] = 10, and B = 5. This question walks through the SMJ algorithm step by step.

i. (0.5 pt) How many sorted run(s) do we have at the end of pass 0 for table X?

10

ii. (0.5 pt) How many sorted run(s) do we have at the end of pass 1 for table X?

3

iii. (1 pt) How many pages are in the smallest run(s) from X at the end of pass 1?

10

iv. (0.5 pt) How many sorted run(s) do we have at the end of pass 0 for table Y?

2

v. (0.5 pt) How many sorted run(s) do we have at the end of pass 1 for table Y?

1

vi. (1 pt) How many I/Os in total does it take to sort both tables X and Y?

340

vii. (2 pt) How many I/Os in total will joining tables X and Y with SMJ (without optimization) take, assuming there are no duplicates in either tables?

400

Sort X Pass 0: 10 runs of X, each of 5 pages Pass 1: 3 runs of X, 2 of 20 pages and 1 of 10 pages Pass 2: 1 run of X, of 50 pages

Sort Y Pass 0: 2 runs of Y, each of 5 pages Pass 1: 1 run of Y, of 10 pages

num passes to sort X = 3 Cost(sort X) = 2 * 3 * [X] = 300 num passes to sort Y = 2 cost(sort Y) = 2 * 2 * [Y] = 40

cost(SMJ) = cost(sort X) + cost(sort Y) + [X] + [Y] = 340 + 60 = 400

viii. **(3 pt)** What is the minimum number of buffer pages required for the I/O cost of the previous question to remain the same?

5

Want to sort relation X in 3 passes and relation Y in 2 passes. Smallest B such that B(B-1)^2 >= 50 is 5. We can only sort at most 36 pages of data with B = 4. Likewise, the smallest B such that B(B-1) >= 10 is 4. We need max(5, 4) = 5 buffer pages.

ix. (2 pt) What is the minimum I/O cost of executing SMJ, applying any optimization taught in class? Assume again that there are no duplicates in either table X or table Y.

300

In the penultimate passes of sorting X and Y, we have 3 runs of X and 2 runs of Y. Since 3 + 2 > 5 - 1, we cannot perform the final sorting passes for both tables at once in memory. However, we have enough memory to avoid an extra pair of read/write from disk for one of the tables. We choose table X, which is the larger table. Since runs(X) + 1 <= B - 1, we are able to save 2*[X] I/Os. 400 - 100 = 300. See optimization options detailed in the note on Joins for more info.

(c) (8 points) GHJ

Justin and Rithvik suggest using Grace Hash Join to join the two tables. Recall that [X] = 50, [Y] = 10, and B = 5.

Assume the first hash function we use was imperfect. We end up with partitions after pass 1 that looks like this:

- Partition 1: [X] = 18, [Y] = 3
- Partition 2: [X] = 20, [Y] = 2
- Partition 3: [X] = 10, [Y] = 4
- Partition 4: [X] = 2, [Y] = 1

Assume all remaining passes use hash functions that uniformly partition data.

i. (2 pt) What partitions do we need to recursively partition, if any?

- ☐ A. Partition 1
- ☐ B. Partition 2
- ■ C. Partition 3
- ☐ D. Partition 4
- ☐ E. None of the partitions need to be recursively partitioned

ii. (4 pt) What is the total number of I/Os incurred after finishing all partitioning passes of GHJ (including pass 1)?

150

Pass 1: 2 * (50 + 10) I/Os to read in all pages of X and Y and write out the given partitions Only partition 3 needs to be recursively partitioned Pass 2: 10 + 4 = 14 I/Os to read in partition 3. Then hash into 4 buckets uniformly. Ceil(10/4) = 3 pages of X into each bucket and 1 page of Y in each bucket. In total, 4 pages are sent to each bucket, so we write out 4 * 4 = 16 pages to disk.

Add them together to get 150.

iii. (2 pt) What is the total number of I/Os for the build and probe phase? Recall that we assume we could stream the final output into the next operator.

62

21 (partition 1) + 22 (partition 2) + 3 (partition 4) + 16 (partition 3) = 62 I/Os to read in last passes' partitions and build in memory hash tables.

2. (22 points) Transactions and Concurrency

(a) (5 points) True/False

i. (0.5 pt) It is possible to determine whether a transaction schedule is conflict serializable by using a waits-for graph.

○ True

● False

False. A waits-for graph is used for deadlock detection, not for checking serializability.

ii. (0.5 pt) The order of non-conflicting operations always has no effect on the final state of the database.

● True

○ False

True. If their order is changed, the final state will remain the same—see lecture 14 slide 33.

iii. (0.5 pt) If two schedules are conflict equivalent, then they are also view equivalent.

● True

○ False

True. The set of conflict serializable schedules is a subset of the set of view serializable schedules.

iv. (0.5 pt) If a transaction holds an IX lock on a node, then it is valid under the multiple granularity locking protocol for the transaction to request an IX lock on a child of that node.

● True

○ False

True. To get X or IX on a node, must hold IX or SIX on parent node.

v. (1 pt) Implementing concurrency in a DBMS can decrease the latency of transactions.

● True

○ False

True. Multiple transactions can run at the same time, so one transaction's latency need not be dependent on another unrelated transaction.

vi. (1 pt) If no aborts occur, standard two-phase locking always achieves the same throughput as strict two-phase locking.

○ True

● False

False. Strict 2PL holds locks longer, which can degrade throughput.

vii. (1 pt) Locking at a finer granularity incurs less overhead from lock management than locking at a coarse granularity.

○ True

● False

False. Fine granularity locks incur high overhead in managing locks.

(b) (4 points) Aborts

Consider the following schedule.

Transaction	1	2	3	4	5	6	7	8	9		
T1	r(a)		w(c)				r(b)				
T2		w(b)								r(c)	
T3				w(a)	r(c)			w(c)			w(a)

No transaction will commit in the first 9 timesteps. If a transaction aborts, all changes it has made are undone before the next timestep.

i. (2 pt) Suppose T_1 aborts immediately after timestep 3. Select the following transactions which would cause violations of ACID properties, if committed at timestep 10.

☐ A. T_2

☐ B. T_3

■ C. None of the above

No other transaction has observed T_1's write to c immediately after timestep 3.

ii. (2 pt) Suppose T_2 aborts immediately after timestep 8. Select the following transactions which would cause violations of ACID properties, if committed at timestep 10.

■ A. T_1

■ B. T_3

☐ C. None of the above

T_1 read T_2's write to b (this is a dirty read), so it must abort to preserve isolation. This abort cascades to T_3, which read T_1's write to c.

(c) (10 points) Locking and Serializability

Jason is considering executing two types of transactions that access the pages a, b, c, and x.

- T_1: $r(a)$ $r(x)$ $w(x)$
- T_2: $w(x)$ $r(b)$ $r(c)$

Jason's database of choice is RookieDB, which uses two-phase locking (2PL). To implement 2PL, RookieDB will lock an element *just before* a transaction reads or writes the element. Assume that:

- Transactions never abort.
- Operations within a transaction are executed sequentially.
- Individual read and write operations are atomic, even if a lock is not held on the page being read/written.
- We are lucky so that deadlock does not occur.

i. (2 pt) Jason wants to pick *one* type of transaction to repeatedly execute. This means that they will run as many transactions as possible, either all of type T_1 or all of type T_2.

If all operations take the same amount of time to execute, which type will allow Jason to achieve greater throughput in the best case scenario?

● A. T_1

○ B. T_2

○ C. Either, the throughput would be the same.

Notice that T_1 holds a lock on x starting from its second operation, while T_2 holds a lock on x starting from its first operation. Therefore, T_1 will lock x for a total of *two* operations, while T_2 will lock x for a total of *three* operations.

Running only T_1 will result in a better interleaving. Observe the concurrency of this schedule of T_1's:

r(a)	r(x)	w(x)					
		r(a)	r(x)	w(x)			
				r(a)	r(x)	w(x)	

On the other hand, we cannot overlap the execution of T_2's because of its lock on x:

w(x)	r(b)	r(c)					
			w(x)	r(b)	r(c)		
						w(x)	r(b)	r(c)

ii. (2 pt) Jason now wants to run both T_1 and T_2. However, they aren't happy with the performance of RookieDB, so they decide to optimize RookieDB to **release locks early**.

What is the earliest operation after which transactions of type T_2 can release the lock on x, while still preserving conflict serializability? Transactions of type T_1 will still obey 2PL.

○ A. T_2 never has to lock x

● B. $w(x)$

○ C. $r(b)$

○ D. $r(c)$

If T_2 never locks x, it is possible to get the sequence $r_1(x)$, $w_2(x)$, $w_1(x)$, which is not conflict serializable (this is a lost update).

However, it is safe for T_2 to immediately release the lock on x, because none of its remaining operations conflict with any operations in T_1 or T_2.

iii. (2 pt) While tinkering with RookieDB, Jason accidentally introduced a bug that allows new transactions to be run **without ever acquiring locks**. Vibha decides to exploit this bug.

Jason is still running T_1 and T_2, **which must obey 2PL** (ignore the change from the previous question).

Which of the following transactions, if run by Vibha, could cause violations of conflict serializability? If you select both, assume Vibha is not running T_3 and T_4 concurrently.

☐ A. T_3: $r(b)$ $w(a)$

■ B. T_4: $r(x)$ $w(a)$

☐ C. None of the above

T_3 is safe because its only conflicting operation is $w(a)$ (with T_1's $r(a)$). Given an interleaving of (1) T_3's operations and (2) a serializable schedule involving only T_1 and T_2, we can always order T_3 with respect to T_1 by looking at their operations that access a. Similarly, we can order T_3's with respect to each other via their writes to a.

As an example, in the following schedule:

		1	2	3	4	5	6	7	8		
T1	r(a)						r(x)			w(x)	
T2				w(x)				r(b)			r(c)
T3		r(b)				w(a)					

we would serialize T_1 before T_3 since T_1 accesses a first.

T_4 is not safe. Consider the following partial schedule: $r_1(a)$, $r_4(x)$, $w_1(x)$, $w_4(a)$. The dependency graph will have a cycle between T_1 and T_4 (this anomaly is called *write skew*), and therefore this schedule is not conflict serializable.

iv. (2 pt) After running some transactions, the following schedule is produced.

			1	2	3	4	5	6	7	8	9		
T1	r(a)			r(x)					w(x)				
T2		w(x)			r(b)							r(c)	
T5							w(c)			w(b)			w(a)

How many edges are in the conflict dependency graph? If there's an edge from T1 to T2 and T2 to T1, count them as 2 edges.

4

$T_2 \to T_1$, $T_1 \to T_5$, $T_2 \to T_5$, $T_5 \to T_2$

v. (1 pt) Which of the following serial schedules are conflict equivalent to the schedule in the previous question?

☐ A. $T_1 \to T_2 \to T_5$

☐ B. $T_2 \to T_1 \to T_5$

■ C. None of the above

There's a cycle in the dependency graph involving T_2 and T_5, so the schedule is not conflict serializable.

(d) (4 points) Deadlock

Consider the following schedule.

This schedule consists of multi granularity locks as presented in lecture. The resources are: Tables A and B with pages $A1$, $A2$ and $B1$, $B2$ respectively. Assume every transaction has an IX lock on the database.

There is no queue skipping for this problem. Assume no locks are released in the first 11 timesteps, unless a transaction aborts.

	1	2	3	4	5	6	7	8	9	10	11			
T1	IX(A)						X(A1)						SIX(B)	S(A2)
T2		S(B)										S(A)		
T3			IS(A)						S(A1)					
T4				IX(A)		IX(B)				X(B1)				

The transactions ordered from highest to lowest priority are T_1, T_2, T_3, T_4.

i. (2 pt) Using wait-die deadlock avoidance, which transactions will abort?

- ☐ A. T_1
- ■ B. T_2
- ■ C. T_3
- ■ D. T_4
- ☐ E. No transaction aborts

T_4 will die when trying to acquire IX(B), since T_2 has an S lock on B. T_3 will die when trying to acquire S(A1), since T_1 has an X lock on A1. T_2 will die when trying to acquire S(A), since T_1 holds an IX lock on A.

ii. (2 pt) A CS189 student has trained a machine learning model that detects deadlocks and chooses transactions to abort, such that the total number of aborts to break deadlocks is *minimized*. For the schedule above, which transactions would this deadlock detector abort?

- ☐ A. T_1
- ■ B. T_2
- ☐ C. T_3
- ☐ D. T_4
- ☐ E. None of the above

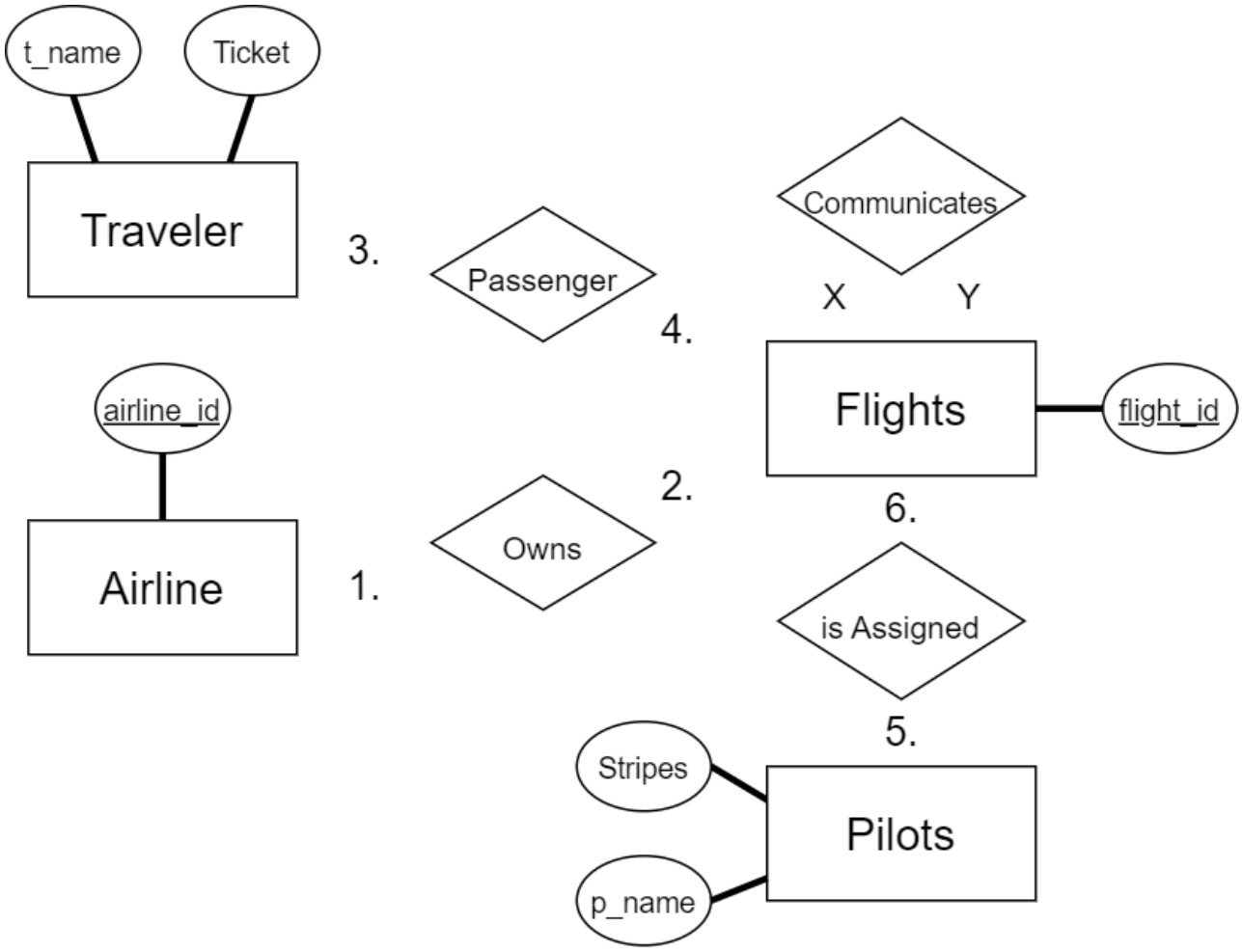
There are two cycles in the waits-for graph. The edges in the waits-for graph are:

- $T_1 \to T_2$ from S(B) and SIX(B)
- $T_2 \to T_1$ from IX(A) and S(A)
- $T_2 \to T_4$ from IX(A) and S(A)
- $T_4 \to T_2$ from S(B) and IX(B)
- $T_3 \to T_1$ from S(A1) and X(A1)

T_2 is involved in both cycles, so we should make sure it aborts. Otherwise, we would end up aborting both T_1 and T_4 to break the cycles.

3. (6.5 points) Ground School

Answer the following questions to fill out this Entity-Relationship Model according to the given constraints.



Constraints

- Each airline owns at least one flight
- Each flight is owned by exactly one airline
- Each flight is assigned exactly one pilot
- Each pilot can be assigned to any number of flights
- Each traveler can be a passenger for at most one flight
- Each flight will have at least one passenger

(a) (0.5 pt) What type of edge should be drawn at 1 between **Airline** and **Owns**?

- ○ A. Thin arrow (key constraint)
- ○ B. Thin line (no constraint)
- ○ C. Bold arrow (key constraint with total participation)
- ● D. Bold line (participation constraint)

(b) **(0.5 pt)** What type of edge should be drawn at 2 between **Flights** and **Owns**?

- ○ A. Thin arrow (key constraint)
- ○ B. Thin line (no constraint)
- ● C. Bold arrow (key constraint with total participation)
- ○ D. Bold line (participation constraint)

(c) **(0.5 pt)** What type of edge should be drawn at 3 between **Traveler** and **Passenger**?

- ● A. Thin arrow (key constraint)
- ○ B. Thin line (no constraint)
- ○ C. Bold arrow (key constraint with total participation)
- ○ D. Bold line (participation constraint)

(d) **(0.5 pt)** What type of edge should be drawn at 4 between **Flights** and **Passenger**?

- ○ A. Thin arrow (key constraint)
- ○ B. Thin line (no constraint)
- ○ C. Bold arrow (key constraint with total participation)
- ● D. Bold line (participation constraint)

(e) **(0.5 pt)** What type of edge should be drawn at 5 between **Pilots** and **Is Assigned**?

- ○ A. Thin arrow (key constraint)
- ● B. Thin line (no constraint)
- ○ C. Bold arrow (key constraint with total participation)
- ○ D. Bold line (participation constraint)

(f) **(0.5 pt)** What type of edge should be drawn at 6 between **Flights** and **Is Assigned**?

- ○ A. Thin arrow (key constraint)
- ○ B. Thin line (no constraint)
- ● C. Bold arrow (key constraint with total participation)
- ○ D. Bold line (participation constraint)

(g) **(0.5 pt)** We now decide to convert **Flights** into a weak entity identified by its corresponding **Airline**. Choose all of the following that must occur.

- ☐ A. Set the edge between Airline and Owns to a Thick Arrow
- ■ B. Bold the Diamond for Owns relationship
- ■ C. Bold the Rectangle for Flights
- ☐ D. Set the edge between Flights and Owns to a Thick Line

(h) **(1 pt)** What attribute(s) uniquely identify the weak entity in this ER diagram from part g? If there are multiple attributes, separate them with commas (order does not matter). Write your answer in the form `table_name.attribute_name`.

Flights.flight_id, Airlines.airline_id

The flight's id becomes a partial key, and it must be appended to the corresponding airline's id because Flights are now a weak entity uniquely identified by a specific Airline. As such, the primary key for Flights needs the attributes "Flights.flight_id, Airlines.airline_id".

(i) **(2 pt)** Due to some CS161 students' exploits, the control center at this airport is no longer functioning. As such, all flights now need to be able to communicate with all other flights to land or take off safely. You may assume that there will always be **at least 2 flights** in the communication network at any time. To satisfy all possible communication networks with the added relationship **Communicates**, what edge type(s) are used to properly fill in both **edge X** and **edge Y**? Select all that apply.

- ☐ A. Thin arrow (key constraint)
- ☐ B. Thin line (no constraint)
- ☐ C. Bold arrow (key constraint with total participation)
- ■ D. Bold line (participation constraint)
- ☐ E. Bold arrow with weak entity

4. (9.5 points) Flight School

Consider the relation `R = PLANES` with the following functional dependencies:

- L → P
- P → L
- AN → E
- PL → NE
- S → PAE

(a) (1 pt) Select all of the following that are candidate keys for the relation `R`.

- ☐ A. `P`
- ■ B. `S`
- ☐ C. `AN`
- ☐ D. `PL`
- ☐ E. `PA`
- ☐ F. `PS`
- ☐ G. None of the above

In the list of options, only `S` is a candidate key. This is because the closure of `S` consists of `PLANES`. By applying `S → PAE`, `P → L`, and `PL → NE`, we get that `S` determines all other columns of `R = PLANES`, making `S` first a superkey. Not only that, if we were to remove `S`, we no longer have any columns so we cannot fully determine `R`. Hence, `S` is a candidate key.

`PS` also fully determines `R`, so it is a superkey. However, if we were to remove `P`, `S` alone still can determine all of `R`. As such, `PS` is a superkey, but not a candidate key. All other remaining options do not fully determine `R`.

(b) (1 pt) Select all of the following that are superkeys for the relation `R`.

- ☐ A. `L`
- ☐ B. `AN`
- ■ C. `PES`
- ☐ D. `PAE`
- ■ E. `PLAS`
- ☐ F. `PLANE`
- ☐ G. None of the above

From the previous part, we already know that `S` is both a superkey and a candidate key. Thus, any option that contains `S` can determine all columns of `R` and be a superkey as well. Hence, `PES` and `PLAS` are both superkeys.

(c) (1 pt) Which of the functional dependencies violate BCNF?

- [x] A. `L → P`
- [x] B. `P → L`
- [x] C. `AN → E`
- [x] D. `PL → NE`
- [ ] E. `S → PAE`
- [ ] F. None of the above

A functional dependency `X → Y` violates BCNF if Y is not a subset of X **AND** X is not a superkey of the relation `R`. In our case:

- `P` is not a subset of `L` and `L` is not a superkey
- `L` is not a subset of `P` and `P` is not a superkey
- `E` is not a subset of `AN` and `AN` is not a superkey
- `NE` is not a subset of `PL` and `PL` is not a superkey
- `S` is a superkey of `R` so `S → PAE` does not violate BCNF

(d) (2 pt) Which of the following relations are in the BCNF decomposition after applying `L → P`? The following functional dependencies are copied from above for your convenience:

- L → P
- P → L
- AN → E
- PL → NE
- S → PAE

- [ ] A. `PL`
- [x] B. `LAS`
- [x] C. `PLNE`
- [ ] D. `NES`
- [ ] E. `LANES`
- [ ] F. `PAN`
- [ ] G. None of the above

`L+`, the closure of `L`, is `PLNE` because `L→P` and `PL→NE`. So, the first relation we have is `PLNE`. Then the second relation consists of `L` union with the original relation `R - L+` = `L` union `R - PLNE` = `L` union `AS` = `LAS`. Thus, the two relations after applying `L → P` are `LAS` and `PLNE`.

(e) **(2 pt)** Which of the following relations are in the BCNF decomposition after applying all dependencies in the order in which they are listed in the question's description?

- ☐ A. `P`
- ☐ B. `PL`
- ☐ C. `PNE`
- ☐ D. `ANE`
- ■ E. `LAS`
- ☐ F. `LANS`
- ☐ G. `PLANE`
- ☐ H. None of the above

From applying `L → P` in the previous question:

- Relation 1: `PLNE`
- Relation 2: `LAS`

`P → L`: The FD `P → L` doesn't break BCNF because P is a superkey for the relation 1 from the previous part since `P → L` and `PL → NE` results in `PLNE`. As such, this FD doesn't need to be applied to decompose the previous relations.

`AN → E`: Nothing is done here because `ANE` aren't within the same relation in any of the relations.

`PL → NE`: Nothing is done here because `PL` is a superkey for `PLNE` (the second relation), so this FD doesn't violate BCNF.

`S → PAE`: Nothing is done here because `SPAE` aren't within the same relation in any of the relations.

Final Solution:

- Relation 1: `PLNE`
- Relation 2: `LAS`

Thus, the only relation in the decomposition from the following options is E. LAS.

(f) **(0.5 pt)** Is the BCNF decomposition from the previous question lossless?

- ● Yes
- ○ No

BCNF decompositions are always lossless.

(g) **(2 pt)** Your flight school instructor uses their own algorithm and comes up with the decomposition of `PL`, `ANS`, `E` for the relation `R`. They claim that their decomposition is dependency preserving for the relation `R`. **Explain if their claim is right or wrong in one sentence.**

No, it violates AN → E, PL → NE, S → PAE.

5. (15 points) Recovery

(a) (4 points) Trivia

i. (2 pt) Which of the following statements are **true**?

- [x] A. No-Steal and No-Force policy requires Redo Logging to ensure ACID properties
- [ ] B. Steal and Force policy requires both Undo and Redo Logging to ensure ACID properties
- [x] C. No-Steal and Force policy requires neither Undo or Redo Logging to ensure ACID properties
- [ ] D. No-Steal and Force policy is the optimal policy in terms of buffer cache performance
- [ ] E. None of the Above

B is false because Force policy does not require Redo logging. D is false because Steal and No-Force policy has better performance. Steal allows buffer pages to be evicted when needed even if a transaction isn't finished. No-Force means pages are only written to disk when absolutely needed.

ii. (2 pt) Which of the following statements are **true**? Assuming they are in the context of ARIES Recovery Model.

- [ ] A. During the Redo Phase, we start from the smallest recLSN and redo all UPDATE and CLR records
- [x] B. During the Undo Phase, we undo all UPDATE records for transactions that are still running or aborting when the system crashes, unless the UPDATE record is already undone indicated by a CLR record. We do not undo CLR records.
- [ ] C. During the Undo Phase, every time we undo an UPDATE record, we need to append the log with a CLR record, which keeps track of the next record to undo using the prevLSN field.
- [ ] D. The number of log records we need to iterate through during the Redo Phase is greater than or equal to that during the Analysis Phase
- [ ] E. None of the above

A is false because there are certain circumstances when we don't want to redo the update or the CLR records. C is false because CLR uses undoNextLSN to keep track of this piece of info, not prevLSN.

(b) (4 points) ARIES - Analysis

Gavin's friend Leafy was working on her research project but her computer accidentally shut down. Please help her go through each phase of the ARIES Recovery Algorithm to recover her hard work.

Below is the log recovered from disk when recovery starts.

LSN	Description	prevLSN
0	MASTER: checkpoint at LSN 40	-
10	T3 UPDATES P4	-
20	T2 UPDATES P1	-
30	T2 UPDATES P3	20
40	Begin Checkpoint	-
50	T2 UPDATES P2	30
60	End Checkpoint	-
70	T1 UPDATES P1	-
80	T3 UPDATES P5	10
90	T3 UPDATES P4	80
100	T3 COMMITS	90
110	T1 UPDATES P5	70
120	T1 ABORTS	110
130	CLR: Undo Write to P5 (T1, LSN 110) UndoNextLSN: 70	120

The End Checkpoint record contains the following tables (transaction table and DPT).

TID	Status	lastLSN
T2	Running	30
T3	Running	10

PID	recLSN
P4	10

The first step to performing recovery is to run the Analysis Phase. Answer the following questions about Analysis.

i. **(2 pt)** What are the recLSNs of each page in the DPT at the end of the Analysis phase?

- ☐ A. P1 - 20
- ■ B. P1 - 70
- ■ C. P2 - 50
- ☐ D. P3 - 30
- ■ E. P4 - 10
- ☐ F. P4 - 90
- ■ G. P5 - 80
- ☐ H. P5 - 130
- ☐ I. None of the above

P4 is already in the DPT with recLSN 10. The Analysis Phase starts at the latest Begin Checkpoint and adds all pages updated not in the DPT with the recLSN as the LSN of the first update encountered. P3 is not in the DPT.

ii. **(1 pt)** How many more records do we need to append to the log after the Analysis phase and before the next phase begins? Write your answer as a single integer.

2

We need to write an END record for T3 and an ABORT record for T2 for a total of 2 more records.

iii. **(1 pt)** At the end of the Analysis Phase, what is the status of transaction T3 in the transaction table?

- ○ A. Commit
- ○ B. End
- ○ C. Abort
- ● D. T3 will not be in the transaction table

A T3 COMMIT record is encountered during Analysis, so it's status will be set to Commit. At the end of Analysis, an END record is written for all committing transactions, and T3 is removed from the transaction table.

(c) (3 points) ARIES - Redo

Assume we are working with the same log as the previous part. The next step is the Redo Phase. For the following parts, assume **pageLSN(disk) < LSN.**

i. (2 pt) Among the following LSNs, select the ones that correspond to an entry in the log that needs to be redone.

- [■] A. 10
- [ ] B. 20
- [ ] C. 30
- [■] D. 50
- [■] E. 70
- [■] F. 130
- [ ] G. None of the above

We redo all UPDATEs and CLRs unless recLSN > LSN or the page is not in the DPT.

ii. (1 pt) Is it possible that the pageLSN of page 5 on disk is 110?

- (●) Yes
- ( ) No

(d) (4 points) ARIES - Undo

Assume we are working with the same log as the previous part. Finally, we perform the Undo phase.

i. (2 pt) How many log records are undone in the Undo Phase?

4

T1 and T2 still have not committed (T1's status is aborting, T2's status is running), so their updates need to be undone. T1 has 1 UPDATE record without a CLR, and T2 has 3.

ii. (2 pt) How many additional log entries do we need to add during the Undo phase?

6

4 CLRs are written for the 4 undone log records, and 2 END records are written for the 2 transactions.

6. (16 points) Trip Optimization

Three friends, Devang, Jeffrey, and Vibha are planning a Thanksgiving trip from Berkeley to get to their dream destination: Boise, Idaho. To get there, they want to take a flight to Las Vegas, a train from there to Los Angeles, and finally a bus to Boise. They would like to see all the possible routes available for them to get there. We have the following tables representing these transportational services and their details for these destinations:

```
CREATE TABLE Flights (
    flight_id INT PRIMARY KEY
    price FLOAT
    departure_time DATETIME
    arrival_time DATETIME
    destination VARCHAR(30)
);
```

```
CREATE TABLE Trains (
    train_id INT PRIMARY KEY
    price FLOAT
    departure_time DATETIME
    arrival_time DATETIME
    destination VARCHAR(10)
);
```

```
CREATE TABLE Buses (
    bus_id INT PRIMARY KEY
    price FLOAT
    departure_time DATETIME
    arrival_time DATETIME
    destination VARCHAR(30)
);
```

Number of pages and records per page:

- Flights: 100 pages, 10 records per page
- Trains: 50 pages, 20 records per page
- Buses: 30 pages, 10 records per page

Indices:

- Flights: Alt 2 clustered index on Flights.destination with h = 2 and 25 leaf pages
- Trains: Alt 1 index on Trains.price with h = 4 and 50 leaf pages
- Buses: Alt 2 unclustered index on Buses.departure_time with h = 3 and 30 leaf pages

Table statistics:

- For all tables' id columns: min = 0, max = 99, 100 distinct values
- Flights.price: min = 0, max = 100, 50 distinct values
- Trains.price: min = 1, max = 15
- Buses.price: min = 0, max = 50, 50 distinct values

(a) (3 points) Selectivity Estimation

Estimate the number of pages of output for the following queries.

Assume that:

- A record from Buses is the same size as a record from Flights.
- Each page in both tables can fit exactly 10 records (10 Flights records or 10 Buses records).
- Pages in all tables are the same size.

i. (1 pt) `SELECT * FROM Flights WHERE price <= 50;`

50

We use the float selectivity formula for <= inequality: (50 - 0) / (100-0) = 1/2. Number of tuples = floor(1/2 * 100 * 10) = 500 tuples. Number of pages = ceil(500 / 10) = 50 pages.

ii. (2 pt)

```
SELECT * FROM Flights, Buses
WHERE (Flights.flight_id <= 24 AND Buses.price <= 20)
    OR (Buses.bus_id > 19 AND Flights.price <= 12.5);
```

5700

Selectivity of Flights.flight_id <= 24: (24 - 0) / (99 - 0 + 1) + (1 / 100) = 25 / 100 = 1/4

Selectivity of Buses.price <= 20: (20 - 0) / (50 - 0) = 4 / 10

Selectivity of (Flights.flight_id <= 24 AND Buses.price <= 20): 1/ 4 * 4/ 10 = 1/10

Selectivity of Buses.bus_id > 19: (99 - 19) / (99 - 0 + 1) = 80 / 100 = 8 / 10

Selectivity of Flights.price <= 12.5: (12.5 - 0) / (100 - 0) = 1/ 8

Selectivity of (Buses.bus_id > 19 AND Flights.price <= 12.5) = 8 / 10 * 1/ 8 = 1/10

Selectivity of (Flights.flight_id <= 24 AND Buses.price <= 20) OR (Buses.bus_id > 19 AND Flights.price <= 12.5) = 2/10 - 1/100

Number of tuples: floor((2/10 - 1/100) * 100 * 10 * 30 * 10) = 60000 - 3000 = 57000

Because we are told to assume a record from Buses is the same size as a record from Flights and that each page in both tables can fit exactly 10 records (10 Flights records or 10 Buses records), we know that each resulting tuple after the join is exactly twice the size. As a result, the number of data pages is: ceil(57000 / 5) = 11400 pages.

(b) (11 points) Query Optimization

To get their routes, Jeffrey decides to execute this query, which we can assume returns the correct result:

```
SELECT
    Flights.flight_id, Trains.train_id, Buses.bus_id
FROM
    Flights INNER JOIN Trains
        ON Flights.arrival_time = Trains.departure_time
    INNER JOIN Buses
        ON Trains.arrival_time = Buses.departure_time
WHERE
    Flights.destination = ‘‘Vegas’’
    AND Trains.destination = ‘‘Los_Angeles’’
    AND Buses.destination = ‘‘Boise’’
ORDER BY
    Flights.price, Trains.price, Buses.price;
```

i. (1 pt) How many I/Os does it take to do a full table scan on `Trains`?

50

ii. (2 pt) What is the estimated I/O cost to do an Index scan on `Flights.destination`? Assume that there are 5 unique flight destinations and the flight destinations are uniformly distributed. Assume we are in Pass 1 of System R.

A. Selectivity = 1/5. Cost of index scan = 2 (h) + 5 (leaf pages) + 20 (data pages) = 27 I/Os

iii. (2 pt) Assuming the given I/O costs for the following scans are correct, which of the following scans will advance to the next pass?

- [x] A. Index scan on Flights.destination - 42 I/Os
- [x] B. Index scan on Buses.price - 63 I/Os
- [x] C. Index scan on Trains.departure_time - 104 I/Os
- [ ] D. Full scan on Flights - 100 I/Os
- [x] E. Full scan on Buses - 30 I/Os
- [x] F. Full scan on Trains - 40 I/Os

Index scan on Flights.destination - 42 I/Os: Moves forward because it's both: cheapest and interesting

Index scan on Buses.departure_time - 63 I/Os: Moves forward because it's interesting (column used in future inner join)

Index scan on Trains.price - 104 I/Os: Moves forward because it's interesting

Full scan on Flights - 100 I/Os: Neither the cheapest, nor interesting

Full scan on Buses - 30 I/Os: Moves forward because it's the cheapest

Full scan on Trains - 40 I/Os: Moves forward because it's the cheapest

iv. (3 pt) What is the minimum estimated I/O cost of joining table `Flights` and table `Trains` with BNLJ, using full scans as single access plans for `Flights` and `Trains`? Assume we are in Pass 2 of System R.

Assume the following information:

- Table `Flights`: 100 pages
- Table `Trains`: 50 pages
- Selectivity of `Flights.destination = ‘‘Vegas’’`: 1/5
- Selectivity of `Trains.destination = ‘‘Los_Angeles’’`: 1/10
- We have B = 7 buffer pages

Feel free to apply materialization and push down selection/projection.

150 I/Os

Push down selectivity for both and do not materialize.

Trains BNLJ Flights: 50 + ceil(5/5)*100 = 150

Partial credit given for the second and third smallest I/O cost join:

Push down selectivity for both and materialize outer.

Trains BNLJ Flights: (50 + 5) + 5 + ceil(5/5)*100 = 160

Push down selectivity for both and materialize inner.

Flights BNLJ Trains: (50 + 5) + (100 + ceil(20/5) * 5) = 175

v. (2 pt) In pass 2 of System R, assuming that the given I/O costs for the joins are correct, which of the following joins will be passed on?

- [x] A. Buses PNLJ Trains - 40 I/Os
- [x] B. Trains SNLJ Flights (Sorted on Trains.departure_time) - 150 I/Os
- [ ] C. Trains BNLJ Buses - 200 I/Os
- [ ] D. Flights SMJ Buses - 150 I/Os
- [x] E. Buses INLJ Trains (Sorted on Buses.price) - 100 I/Os
- [ ] F. Flights BNLJ Trains (Sorted on Flights.destination) - 200 I/Os

Buses PNLJ Trains - 40 I/Os - Moves on because Cheapest

Trains SNLJ Flights (Sorted on Trains.departure_time) - 150 I/Os - Interesting because sorted scan of Trains used in future join

Trains BNLJ Buses - 200 I/Os - Neither interesting, nor cheap

Flights SMJ Buses - 150 I/Os - Cross join :(

Buses INLJ Trains (Sorted on Buses.price) - 100 I/Os - Interesting because sorted scan of Buses used in a future ORDER BY

Flights BNLJ Trains (Sorted on Flights.destination) - 200 I/Os - Neither interesting, nor cheap

vi. (1 pt) How many total passes will the Query Optimizer have to run?

3

(c) (2 points) Operators

Specify the kind of operators present in the following queries:

i. (1 pt) `SELECT * FROM Flights WHERE Flights.price < 100;`

- ● Non-Blocking/Streaming
- ○ Blocking

ii. (1 pt) `SELECT * FROM Flights ORDER BY Flights.price;`

- ○ Non-Blocking/Streaming
- ● Blocking

27. (9 points) Relational Algebra

In this question we have relations $R(A, B)$ and $S(C, D)$ where A, B, C, D are all non `NULL` integers. In the following we write A instead of $R.A$ etc when the context is clear.

(a) (1 pt) The distinct operator δ converts a bag (multiset) relation into a set relation, and is equivalent to no-op otherwise. Assume $R(A, B)$ described above is a bag, express $\delta(R)$ using the relational algebra operators covered in class, or write "N/A" below if it is not possible to do so.

$\delta(R(A, B)) = ?$

$\gamma_{A,B}(R)$

(b) (8 points)

In the following questions assume R and S are sets, and we use set semantics for relational algebra and bag (multiset) semantics for SQL.

Is each of the following relational algebra expression or SQL query equivalent, i.e., they always return the same results regardless of the contents of the input relations?

i. (1 pt) $S \bowtie_{D=A} R =$ `SELECT * FROM R, S WHERE R.A = S.D`

Equivalent?

- (●) Yes
- ( ) No

ii. (1 pt) $\pi_A(R \times S) = \pi_A(R \bowtie_{A=B} S) \cup \pi_A(R \bowtie_{A \neq B} S)$

Equivalent?

- (●) Yes
- ( ) No

iii. (1 pt) $R = \pi_{R.A,R.B}(R \bowtie_{R.A=T.A} \rho_{T(A,B)}(R))$

Equivalent?

- (●) Yes
- ( ) No

iv. (1 pt) `SELECT * FROM R INTERSECT SELECT R.A, R.B FROM R, S WHERE R.B = S.C`

$= \pi_{A,B}(R \bowtie_{B \neq C} S)$

Equivalent?

- ( ) Yes
- (●) No

As a counterexample, consider R(A,B) consisting of tuples (1,1) and (1,2), and S(C,D) with tuples (1,1), (3,1).

v. (2 pt) $\rho_{F(A,count(*)->X)}(\gamma_{A,count(*)}(R \bowtie_{A=C} S) =$
$\rho_{F(A->A,X->X)}(\pi_{A,X}(R \bowtie_{A=T.K} \rho_{T(C->K,count(*)->X)}(\gamma_{C,count(*)}(S)))$

Equivalent?

- [x] Yes
- [ ] No

vi. (2 pt) $\gamma_{C,max(D)}(\sigma_{C=1}(S)) =$

```
SELECT S.C FROM S WHERE S.C = 1 ORDER BY S.C LIMIT 1
```

Equivalent?

- [ ] Yes
- [x] No

The SQL query does not project max(D).

No more questions.