CS 186 Fall 2021 Introduction to Database Systems

Midterm 2

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 W186 Fall 2021 Midterm 2

`If you do not have special accommodations, you will have 110 minutes to complete the midterm.`

1.0.1 Contents:

- The midterm has 6 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.
- Do not share this exam until solutions are released.

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.

1.0.4 Taking the exam remotely:

- Do NOT post on piazza if you want to ask a question, use the clarification form. We reserve the right to deduct points from the midterm if you use piazza during the exam.
- You may print this exam to work on it.
- For each question, submit only your final answer on examtool.
- For numerical answers, do not input any suffixes (i.e. if your answer is 5 I/Os, only input 5 and not 5 I/Os or 5 IOs).
- Make sure to save your answers in examtool at the end of the exam, although the website should autosave your answers as well.

(a) What is your full name?

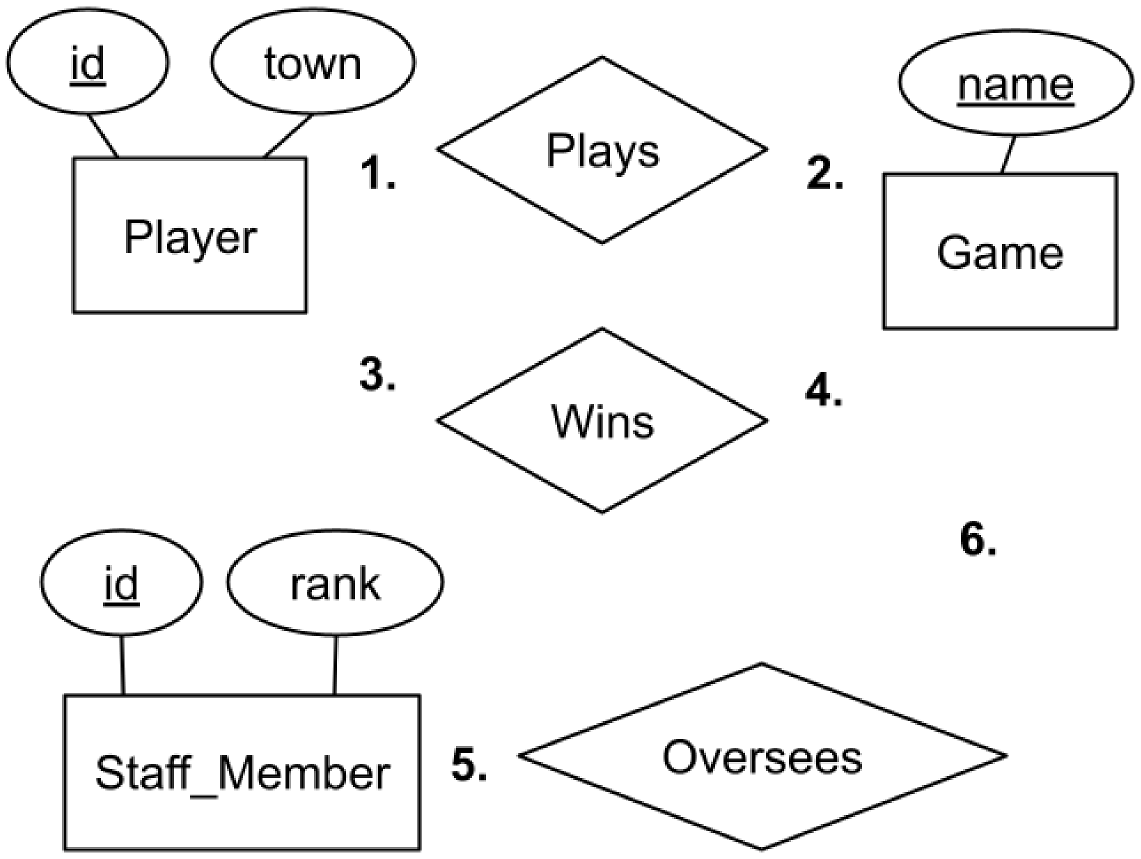
(b) What is your student ID number?

(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. **(8 points) ER Game**

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



Constraints

- Each player must play in at least one game, and each game may have any number of players
- Each player may win any number of games, and each game may have any number of winners
- Each game must have exactly one staff member that oversees it, and each staff member may oversee at most one game

(a) **(1 pt)** What type of edge should be drawn at 1 between **Player** and **Plays**?

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

(b) (1 pt) What type of edge should be drawn at 2 between **Player** and **Game**?

○ A. Thin arrow (key constraint)
[✓ B. Thin line (no constraint)]
○ C. Bold arrow (key constraint with total participation)
○ D. Bold line (participation constraint)

(c) (1 pt) What type of edge should be drawn at 3 between **Player** and **Wins**?

○ A. Thin arrow (key constraint)
[✓ B. Thin line (no constraint)]
○ C. Bold arrow (key constraint with total participation)
○ D. Bold line (participation constraint)

(d) (1 pt) What type of edge should be drawn at 4 between **Wins** and **Game**?

○ A. Thin arrow (key constraint)
[✓ B. Thin line (no constraint)]
○ C. Bold arrow (key constraint with total participation)
○ D. Bold line (participation constraint)

(e) (1 pt) What type of edge should be drawn at 5 between **Staff_Member** and **Oversees**?

[✓ A. Thin arrow (key constraint)]
○ B. Thin line (no constraint)
○ C. Bold arrow (key constraint with total participation)
○ D. Bold line (participation constraint)

(f) (1 pt) What type of edge should be drawn at 6 between **Oversees** and **Game**?

○ A. Thin arrow (key constraint)
○ B. Thin line (no constraint)
[✓ C. Bold arrow (key constraint with total participation)]
○ D. Bold line (participation constraint)

(g) (1 pt) We now decide to convert Game to a weak entity identified by its corresponding Staff_Member. Choose all of the following that must be bolded.

[✓ A. Rectangle surrounding “Game”]
☐ B. Oval surrounding “name”
☐ C. Rectangle surrounding “Staff_Member”
☐ D. CORRECT Diamond surrounding “Oversees”

(h) **(1 pt)** Continuing from part g, write down the attributes in the primary key for a game. Use dot notation to name the attributes and separate them by commas. Order does not matter.

Game.name, Staff_Member.id

The game's name becomes a partial key, and it must be appended to the corresponding staff member's id. Therefore, the primary key needs the attributes "Game.name, Staff_Member.id".

2. (9 points) Functional Dependency Arcade

(a) **(1 pt)** Consider the relation `R = {SPELUNKY}`. Decompose `R` into BCNF in the order of the following functional dependencies: `S -> PELUNKY; Y -> UN; KE -> PL`.

Which of the following relations are in the decomposition after applying `S -> PELUNKY`?

- ☐ A. `S, PELUNKY`
- ☐ B. `S`
- ☐ C. `PELUNKY`
- ☐ D. `S, SPELUNKY`
- ■ E. None of the above

`S -> PELUNKY` already satisfies BCNF since `S` is a superkey.

(b) **(2 pt)** Which of the following relations are in the BCNF decomposition after applying all dependencies in the order from the previous subquestion?

- ☐ A. `S`
- ☐ B. `EUNY`
- ☐ C. `SYK`
- ■ D. `SEKY`
- ☐ E. `PLK`
- ■ F. `PELK`
- ☐ G. `SPELUNKY`

The other relation that is in the final decomposition is `UNY`. The results after applying the decompositions are:

`S -> PELUNKY`: `SPELUNKY`

`Y -> UN`: `SPELKY, UNY`

`KE -> PL`: `SEKY, PELK, UNY`

(c) **(1 pt)** Which of the following are superkeys of the relation?

- ■ A. `SPL`
- ☐ B. `PELUNKY`
- ☐ C. `Y`
- ■ D. `S`
- ☐ E. None of the above

The `S -> PELUNKY` dependency means that the `S` column can determine all other columns, so any set of columns with `S` can also determine all other columns. Therefore both `S` and `SPL` are superkeys.

(d) (1 pt) Which of the following are candidate keys?

☐ A. `SPL`

☐ B. `PELUNKY`

☐ C. `Y`

■ D. None of the above

Although `SPL` is a superkey, the `P` and `L` columns can be removed, and the `S` column alone can still determine the rest of the relation's columns. Therefore, `SPL` is not a candidate key. Additionally, since the other options were not even superkeys, they cannot be candidate keys.

(e) (1 pt) Is the decomposition lossless? Write "Yes" if it is, or "No" it is not.

Yes

BCNF decompositions are always lossless.

(f) (1 pt) Is the decomposition dependency preserving? Write "Yes" if it is, or "No" it is not.

No

The `S -> PELUNKY` dependency has been lost.

(g) (2 pt) Suppose that the `METROID` relation has 80 tuples. A lossy decomposition of the original relation results in the relations `MTOID, ETRID`. How many tuples might be in the relation after joining the `MTID` and `EROID` relations back together? (Choose all possible amounts)

☐ A. 20

☐ B. 40

■ C. 100

■ D. 120

A lossy decomposition results in extra "junk" data when the relations are joined in an attempt to form the original relation. We would have over 80 tuples in our recombined relation, so 100 and 120 are both correct possibilities.

3. (23 points) Query Optimization

Consider the following tables:

```
CREATE TABLE R {
    x FLOAT,
    y INTEGER,
    z INTEGER
}

CREATE TABLE S {
    x FLOAT,
    y INTEGER,
    z INTEGER
}

CREATE TABLE T {
    x FLOAT,
    y INTEGER,
    z INTEGER
}
```

Number of pages and records per page:

```
R: 50 pages, 100 records/page
S: 150 pages, 100 records/page
T: 250 pages, 100 records/page
```

Index:

```
R: Alt 2 clustered index on R.y with h = 3 and 90 leaf pages
S: Alt 2 unclustered index on S.x with h = 2 and 120 leaf pages
```

Table stats (assume uniform distribution):

```
R.y: min = 0, max = 50, 51 unique values
S.x: min = 100, max = 300, 500 unique values
S.z: min = 25, max = 50, 26 unique values
T.x: min = 150, max = 200, 25 unique values
```

(a) (2 points) Selectivity Practice

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

i. (1 pt)

```
SELECT * FROM R WHERE y < 34;
```

34

Selectivity: (34 - 0) / (50 - 0 + 1) = 34/51

Number of tuples: floor(34/51 * 50 * 100) = 3333 tuples

Number of pages: ceil(3333 / 100) = 34

ii. (1 pt)

```
SELECT * FROM S WHERE x >= 150 OR NOT (z < 38);
```

$$\boxed{\textbf{132}}$$

Selectivity of (x >= 150): (300 - 150) / (300 - 100) = 150/200 = 3/4

Selectivity of (NOT (z < 38)): 1 - ((38 - 25) / (50 - 25 + 1)) = 1 - (13 / 26) = 1/2

Combined selectivity: 3/4 + 1/2 - 3/4 * 1/2 = 7/8

Number of tuples: floor(7/8 * 150 * 100) = 13125 tuples

Number of pages: ceil(13125/100) = 132

(b) (21 points) Query Optimizer

The following tables and information are repeated from above for your convenience:

```
CREATE TABLE R {
    x FLOAT,
    y INTEGER,
    z INTEGER
}

CREATE TABLE S {
    x FLOAT,
    y INTEGER,
    z INTEGER
}

CREATE TABLE T {
    x FLOAT,
    y INTEGER,
    z INTEGER
}
```

Number of pages and records per page:

```
R: 50 pages, 100 records/page
S: 150 pages, 100 records/page
T: 250 pages, 100 records/page
```

Index:

```
R: Alt 2 clustered index on R.y with h = 3 and 90 leaf pages
S: Alt 2 unclustered index on S.x with h = 2 and 120 leaf pages
```

Table stats (assume uniform distribution):

```
R.y: min = 0, max = 50, 51 unique values
S.x: min = 100, max = 300, 500 unique values
S.z: min = 25, max = 50, 26 unique values
T.x: min = 150, max = 200, 25 unique values
```

We want to execute the following query:

```
SELECT *
    FROM R INNER JOIN S ON R.x = S.x OR R.y >= S.z
           INNER JOIN T ON S.z = T.z
    WHERE T.x >= 160 AND R.y < 17
    ORDER BY S.x
```

i. **(1 pt)** How many I/Os would a full scan of R take? Assume all header pages are cached.

50

Full scan reads all 50 pages.

ii. **(2 pt)** How many I/Os would an index scan of S using the index on S.x take (you must use the index in the index scan)?

Assume we are in pass 1 of the System R (Selinger) query optimizer.

15122

There are no single column conditions relating to S.x so no selections are pushed down.

2 inner nodes

120 leaf nodes

150 pages * 100 records/page = 15000 records

2 + 120 + 15000 (1 I/O per matching record) = 15122 I/Os

iii. **(2 pt)** How many I/Os would an index scan of R using the index on R.y take (you must use the index in the index scan)?

Assume we are in pass 1 of the System R (Selinger) query optimizer.

50

We push down the the single column condition R.y $<$ 17

Selectivity: (17 - 0) / (50 - 0 + 1) = 17/51 = 1/3

3 inner nodes

1/3 * 90 = 30 leaf nodes

floor(1/3 * 50 * 100) = 1666 records

ceil(1666 / 100) = 17 pages

3 + 30 + 17 (1 I/O per page of matching records) = 50 I/Os

iv. **(2 pt)** Assume in the first pass of the Selinger Optimizer S and T are both accessed with a full scan and streamed directly into the second pass. Assume there is no materialization and we are still executing the query above. What is the minimum I/O cost when doing a BNLJ between these two relations, including the cost of the 1st pass?

Assume that the number of buffer pages `B = 12` and the selectivity for `T.x >= 160` equals `4/5`.

3250

We push down the single column condition T.x $>=$ 160

Selectivity: (200 - 160) / (200 - 150) = 40/50 = 4/5 (or use the given selectivity)

We use T and S as the outer relation and inner relation respectively

250 + (200 / 10) * 150 = 3250

v. **(3 pt)** Here is the query we want to execute again:

```
SELECT *
    FROM R INNER JOIN S ON R.x = S.x OR R.y >= S.z
           INNER JOIN T ON S.z = T.z
    WHERE T.x >= 160 AND R.y < 17
    ORDER BY S.x
```

Which of the following query plans will be retained by the first pass of the Selinger Optimizer given total estimated I/O cost and output order?

Assume the following query plans are independent of the information given about R, S and T earlier.

```
A: Full scan on R   I/Os: 3000  Order: None
B: Index scan on R  I/Os: 2800  Order: R.x
C: Index Scan on R  I/Os: 2000  Order: R.y
D: Full scan on S   I/Os: 2000  Order: None
E: Index scan on S  I/Os: 2500  Order: S.x
F: Index scan on S  I/Os: 2300  Order: S.y
G: Index scan on S  I/Os: 4000  Order: S.z
H: Full scan on T   I/Os: 3000  Order: None
I: Index scan on T  I/Os: 2000  Order: T.x
J: Index scan on T  I/Os: 1500  Order: T.z
```

☐ A

☐ B

[✓ C]

[✓ D]

[✓ E]

☐ F

[✓ G]

☐ H

☐ I

[✓ J]

B is not an interesting order because R.x is used in a nonequijoin (R.x = S.x OR R.y >= S.z) which means having R.x in order doesn't help joining R and S. C is the minimum cost access on table R and is also an interesting order and is used in a downstream join. D is the minimum cost access on table S. E is an interesting order and is used in a downstream join. G is is an interesting order and is used in a downstream join. J is the minimum cost access on table T and is also an interesting order and is used in a downstream join.

vi. **(3 pt)** Here is the query we want to execute again:

```
SELECT *
    FROM R INNER JOIN S ON R.x = S.x OR R.y >= S.z
           INNER JOIN T ON S.z = T.z
    WHERE T.x >= 160 AND R.y < 17
    ORDER BY S.x
```

Which of the following query plans will be returned by the second pass of the Selinger Optimizer given total estimated I/O cost and output order?

Assume the following query plans are independent of the information given about R, S and T earlier.

```
A: R BNLJ T I/Os: 4000  Order: None
B: T BNLJ R I/Os: 3800  Order: None
C: R SMJ T  I/Os: 5000  Order: R.y
D: R BNLJ S I/Os: 3500  Order: None
E: R SMJ S  I/Os: 3550  Order: R.y
F: S SMJ R  I/Os: 3800  Order: S.x
G: T BNLJ S I/Os: 2200  Order: None
H: S SMJ T  I/Os: 4500  Order: T.z
I: S SMJ T  I/Os: 2300  Order: T.x
J: T SMJ S  I/Os: 3800  Order: S.y
```

☐ A

☐ B

☐ C

[✓ D]

☐ E

[✓ F]

[✓ G]

☐ H

☐ I

☐ J

Selinger Optimizer does not consider cross joins so any query plan joining R and T is not considered. D is the minimum cost join between R,S. F is an interesting order and is used later in the ORDER BY. G is the minimum cost join between S,T. Clarification: in reality, we won't be able to do R SMJ S or S SMJ R because of the nonequijoin condition on R and S.

vii. (2 pt) Select all statements regarding Selinger Optimizer which are true. **There could be zero, one, or more than one correct choices. In the case that there are zero correct choices, mark E. None of the above.**

☐ A. We always retain the minimum cost join between every pair of relations in pass 2

☐ B. Left deep query plans still require us to write the result of a join to disk every time

☐ C. We always prefer to keep index scans over full scans on a table in pass 1 because index scans yield an interesting order

☐ D. We can always reduce I/Os by pushing down selects on both the left and right tables in nested loop joins.

■ E. None of the above

A. is false, we don't want to consider cross joins

B. is false, left deep query plans are fully pipelined

C. is false, if a full scan has a lower I/O cost we also want to keep it.

D. is false, because we have to scan through the right table every time in a nested loop join, there is no improvement in I/O cost.

viii. (6 pt) Dylan is very excited to have been given the opportunity to join David's research group at Berkeley RISE Lab. For his first assignment, David wants Dylan to join two tables together to count the number of shoulders in Berkeley, while also having the smallest I/O cost possible.

We are given the following information:

```
Person: 100 pages
Info: 5000 pages
50 pages of records where Person.height >= 170
800 pages of records where Info.major = 1
```

Buffer pages:

```
B = 12
```

What is the minimum I/O cost needed to execute the following query? Feel free to materialize if it helps reduce the I/O cost. Assume we use a full scan to access the Person and Info tables and BNLJ is used for the join.

```
SELECT Person.shoulder
    FROM Person INNER JOIN Info on Person.pid = Info.pid
WHERE Person.height >= 170 AND Info.major = 1
```

9150

To minimize the amount of I/Os in this query, we would want Info as our outer table and Person as our inner table. We want to push down both selects so it is right above the corresponding single table accesses and materialize Person after the select and before the BNLJ.

The total I/O cost will be: (100 + 50) + (5000 + ceil(800 / 10) * 50) = 9150 I/Os

4. (20 points) Transactions and Concurrency

Clarification: Assume no queue-skipping.

Conor starts a transaction to read and write to the DARE database. Below is a table describing Conor's transaction, with the time of reads and writes as columns.

	1	2	3	4	5	6	7	8
Conor			R(a)			W(b)		

(a) (4 pt) Shadaj wants to read the DARE database at the same time. However, he is very evil, and he knows that the database **does not guarantee isolation**. Which of these schedules will expose isolation violations from inconsistent reads?

Assume that transactions do not use locks and do not abort. Select *all* possible options below.

Schedule A

	1	2	3	4	5	6	7	8
Conor			R(a)			W(b)		
Shadaj	R(b)			R(b)				

Schedule B

	1	2	3	4	5	6	7	8
Conor			R(a)			W(b)		
Shadaj				R(b)			R(b)	

Schedule C

	1	2	3	4	5	6	7	8
Conor			R(a)			W(b)		
Shadaj	R(b)						R(b)	

Schedule D

	1	2	3	4	5	6	7	8
Conor			R(a)			W(b)		
Shadaj							R(b)	R(b)

☐ A. Schedule A

[✓ B. Schedule B]

[✓ C. Schedule C]

☐ D. Schedule D

(b) (1 pt) Are the schedules `A`,`B`,`C`,`D`,`E` above **conflict serializable**? Clarification: Schedule `E` does not exist.

- A. They are all conflict serializable.
- [✓ B. Some are conflict serializable, and some are not.]
- C. None of them are conflict serializable.

(c) (9 points) Deadlock detection

Lisa, another PhD student in RISE, comes to the rescue and fixes the database such that it now **guarantees isolation**. In fact, it guarantees *all* ACID properties.

To resolve potential deadlocks, she's testing out different deadlock prevention schemes and priorities. For each scheme and priority, she wants to know whether Conor or Shadaj will wait for a lock or abort their transactions.

Assume we're using Schedule B:

	1	2	3	4	5	6	7	8
Conor			R(a)			W(b)		
Shadaj				R(b)			R(b)	

i. (1 pt) If we are using wound-wait, and Conor has higher priority. What happens?

- A. Conor aborts
- B. Conor waits
- [✓ C. Shadaj aborts]
- D. Shadaj waits

ii. (1 pt) If we are using wait-die, and Shadaj has higher priority. What happens?

- [✓ A. Conor aborts]
- B. Conor waits
- C. Shadaj aborts
- D. Shadaj waits

(d) (4 pt) Audrey and Samyu want to introduce a subtle complication. They want to write their transactions such that the schedule is **view serializable**, but not **conflict serializable**. Help them fill in the right operation.

	1	2	3	4	5	6	7	8
Conor			R(a)			R(b)		
Samyu		R(a)					W(a)	
Audrey				W(b)	?			

What can **?** be to make the schedule **view serializable**, but not **conflict serializable**? Select all valid options.

☐ A. R(b)

☐ B. W(b)

☐ C. R(a)

[✓ D. W(a)]

(e) (3 pt) Shadaj decides to step up his game and introduce deadlock. Which one of the following schedules will deadlock?

Assume that transactions wait for locks and do not abort. Select *all* possible options below.

Schedule X

	1	2	3	4	5	6	7	8
Conor			R(a)			W(b)		
Shadaj				W(a)				
Lily					R(b)		R(a)	

Schedule Y

	1	2	3	4	5	6	7	8
Conor			R(a)			W(b)		
Shadaj		R(a)						R(b)
Lily					R(b)		W(a)	

Schedule Z

	1	2	3	4	5	6	7	8
Conor			R(a)			R(b)		
Shadaj		R(a)		W(b)				
Lily					R(b)		W(a)	

[✓ A. Schedule X]

[✓ B. Schedule Y]

☐ C. Schedule Z

(f) (6 points) Lock Hierarchy

Professor Hellerstein notices Shadaj's malicious behavior and terminates his PhD. Shadaj is now working an unfulfilling but handsomely-paid job at Facebook. Conor now wants to write a SQL statement and see what kind of locks he'll need to obtain.

He is presented with this existing schema:

```
CREATE TABLE Students(
  enough BOOLEAN
);
```

i. **(3 pt)** Write a single SQL query using the table above, such that **SIX locks** will be used. Credit will only be given if your SQL query is syntactically correct. **Do not** use nested queries.

Here are some examples of SQL syntax to jog your memories:

```
SELECT column FROM table WHERE condition
UPDATE table SET column = value WHERE condition
INSERT INTO table (column) VALUES (value)
DELETE FROM table WHERE condition
```

A query in the following format:
UPDATE Students SET enough = true WHERE enough = false or DELETE . . .

ii. **(3 points)**

Write the locks acquired for the SQL query above. If no lock needs to be acquired, write `NL`. **If different locks must be acquired depending on the circumstances, write all of them.**

A. **(1 pt)** What is the lock(s) acquired for the table `Students`?

SIX

B. **(1 pt)** What is the lock(s) acquired for a **page** in the table of `Students`?

IX

C. **(1 pt)** What is the lock(s) acquired for **individual items** in the table `Students`?

{X, NL} or {X}, based on the interpretation of the question's wording

5. (20 points) Recovering from Lakshya's Jokes

(a) (2 pt) Select all true statements.

■ A. ARIES recovery uses a No Force, Steal policy.

■ B. In Redo logging, we create a new Redo log entry when writing values in memory.

☐ C. In the ARIES algorithm, the Undo phase ensures durability and the Redo phase ensures atomicity.

☐ D. The LastLSN is associated with the dirty page table, and is the LSN that is associated with the last operation that dirtied the page.

(b) (5 points) ARIES Analysis

After hearing Lakshya's jokes, the database has collapsed and we're given the log below after rebooting. Answer the following questions based on the log. Assume for this question that the dirty page table read in is empty.

Log

LSN	Record	prevLSN
50	T1 updates P2	null
60	Begin Checkpoint	-
70	T2 updates P2	null
80	T1 updates P3	50
90	T3 updates P3	null
100	T2 updates P1	70
110	T3 aborts	90
120	End Checkpoint	-
130	T1 updates P1	80
140	T1 updates P2	130
150	Begin Checkpoint	-
160	T1 updates P3	140
170	T1 Commits	160
180	T2 updates P3	100
190	T3 End	110
200	End Checkpoint	-

Transaction Table

Transaction	Status	lastLSN
T1	Running	160
T2	Running	100
T3	Aborting	110

i. (1 pt) Write the value of the recLSN for **P1** if it is in the dirty page table after performing the analysis phase. Assume the dirty page table starts off empty. If the page is not modified, write in the value -1.

-1

When dealing with multiple checkpoints, we always start from the last successful checkpoint.

ii. (1 pt) Write the value of the recLSN for **P2** if it is in the dirty page table after performing the analysis phase. If the page is not modified, write in the value -1.

-1

iii. (1 pt) Write the value of the recLSN for **P3** if it is in the dirty page table after performing the analysis phase. If the page is not modified, write in the value -1.

160

iv. (1 pt) What is the status of T2 at the end of analysis?

- ○ A. Running
- [✓ B. Aborting]
- ○ C. Committing
- ○ D. Not in the transaction table

v. (1 pt) What is the status of T3 at the end of analysis?

- ○ A. Running
- ○ B. Aborting
- ○ C. Committing
- [✓ D. Not in the transaction table]

(c) (6 points) Redo

After the analysis phase of ARIES, we now enter the redo phase. **Note that the given logs and tables are independent of the previous part - do NOT refer back to the previous tables**. Assume that pageLSN(disk) $<$ LSN. You may assume that an end operation counts as a record.

Log

LSN	Record	prevLSN
10	T1 updates P1	null
20	Begin Checkpoint	-
30	T2 updates P2	null
40	T3 updates P1	null
50	T1 updates P3	10
60	T2 commits	30
70	End Checkpoint	-
80	T1 updates P2	50
90	T3 updates P2	40
100	T2 End	30
110	T1 aborts	80
120	CLR undo T1 LSN 80	110
130	T3 updates P3	90
	CRASH	

Transaction Table

Transaction	Status	lastLSN
T1	Aborting	110
T3	Aborting	130

Dirty Page Table

Page ID	recLSN
P2	30
P3	50

i. (1 pt) At which LSN does the first REDO operation occur?

30

ii. (5 pt) At which LSNs do we perform REDO operations?

☐ A. 10

☐ B. 20

[✓ C. 30]

☐ D. 40

[✓ E. 50]

☐ F. 60

☐ G. 70

[✓ H. 80]

[✓ I. 90]

☐ J. 100

☐ K. 110

[✓ L. 120]

[✓ M. 130]

(d) (7 points) Undo

The next stage of the ARIES algorithm is the UNDO phase - let's run through the same given log, transaction table, and dirty page table from the redo phase. The tables are copied below for your convenience. **You may assume that an end operation counts as a record.**

Log

LSN	Record	prevLSN
10	T1 updates P1	null
20	Begin Checkpoint	-
30	T2 updates P2	null
40	T3 updates P1	null
50	T1 updates P3	10
60	T2 commits	30
70	End Checkpoint	-
80	T1 updates P2	50
90	T3 updates P2	40
100	T2 End	30
110	T1 aborts	80
120	CLR undo T1 LSN 80	110
130	T3 updates P3	90
	CRASH	

Transaction Table

Transaction	Status	lastLSN
T1	Aborting	120
T3	Aborting	130

Dirty Page Table

Page ID	recLSN
P2	30
P3	50

i. (1 pt) Which transactions do we need to write additional CLRs for?

[✓ A. T1]

☐ B. T2

[✓ C. T3]

Note that on the exam, we had some inconsistencies with the in person and remote version answer choices. To fix this, answer choice C was omitted from grading.

ii. (1 pt) How many additional records are needed to undo **T1**?

3

Don't forget to write the End record! It was bolded on the exam that it counts as a record.

iii. **(1 pt)** What is the first LSN for transaction **T1** that we write an additional CLR for during the Undo phase? If no additional CLRs are written for transaction T1, answer N/A.

50

iv. **(1 pt)** How many additional records are needed to undo **T2**?

0

v. **(1 pt)** What is the first LSN for transaction **T2** that we write an additional CLR for during the Undo phase? If no additional CLRs are written for transaction T2, answer N/A.

N/A

vi. **(1 pt)** How many additional records are needed to undo **T3**?

4

vii. **(1 pt)** What is the first LSN for transaction **T3** that we write an additional CLR for during the Undo phase? If no additional CLRs are written for transaction T3, answer N/A.

130

6. (20 points) Students Join Chancellor

Chancellor Christ wants to comb through campus data to determine whether library hours should be extended. She has asked you to help her perform optimal query joins for her queries. She gives you the following information along with a set of questions:

- *Students* table: 40 pages with 100 records per page
- *Seats* table: 25 pages with 40 records per page
- *Hours* table: 30 pages with 10 records per page
- Our buffer has 7 pages (B = 7)
- Alternative 1 B+ tree of height 4 on *Seats.library_id*
  - Assume each tuple in *Students* matches 1 tuple in *Seats*
- Unclustered Alternative 3 B+ tree of height 4 on *Hours.libary_id*
  - Assume each tuple in *Seats* matches 3 tuples in *Hours*

(a) (2 points) Library Capacity

First, the Chancellor wants to find out whether the libraries have enough capacity for students. How many I/Os will the following joins on *Students* and *Seats* take (for every question, choose the join order that minimizes the I/O cost)?

i. (1 pt) Block Nested Loop Join

225

25 + (5 * 40)

ii. (1 pt) Index Nested Loop Join

20040

40 + (4000 * 5)

(b) (8 points) Students in Seats

After mapping students to seats, the Chancellor created a new SeatPlan table that has 100 pages with 50 records per page. Believing Block Nested Loop Join and Index Nested Loop Join to be less optimal, you decide to try to join SeatPlan and Hours using **Sort Merge Join**. The university has received a donation of 6 more buffer pages, so B = 13.

- *SeatPlan* table: 100 pages with 50 records per page
- *Hours* table: 30 pages with 10 records per page
- Our buffer has 13 pages (B = 13)

i. (2 pt) What is the cost of sorting *SeatPlan*?

400

2 * 2 * 100 = 400 I/Os

ii. (2 pt) What is the cost of sorting *Hours*?

120

2 * 2 * 30 = 120 I/Os

iii. (4 pt) What is the overall cost of using Sort Merge Join to join *SeatPlan* and *Hours*? Perform optimization if possible and omit the cost of the final write.

390

We can use the optimization because 8 (# runs of SeatPlan) + 3 (# runs of Hours) <= B - 1 = 12

Without optimization: 400 + 120 + 100 + 30 = 650 I/Os

With optimization: 650 - 2 (100 + 30) = 390 I/Os

(c) (10 points) Timing is Key

Believing other joins to be less optimal, you decide to try to join *SeatPlan* and *Hours* using Grace Hash Join.

For your convenience, the assumptions are repeated from previous question:

- *SeatPlan* table: 100 pages with 50 records per page
- *Hours* table: 30 pages with 10 records per page
- Our buffer has 13 pages (B = 13)
- Two hash functions:
  - A: $\frac{1}{2}$ of the keys in the first bucket, $\frac{1}{4}$ of the keys in the second bucket, rest of the keys evenly distributed across the remaining buckets
  - B: $\frac{1}{4}$ of the keys in the first bucket, $\frac{1}{4}$ of the keys in the second bucket, rest of the keys evenly distributed across the remaining buckets

i. (2 pt) You decide to use hash function A for pass 1. What is the I/O cost for reading and writing the *SeatPlan* partitions?

205

First bucket: 50 pages, Second bucket: 25 pages, 3rd-12th buckets: 3 pages

100 (read) + 50 + 25 + 3*10 (write) = 205 I/Os

ii. (2 pt) You decide to use hash function A for pass 1. What is the I/O cost for reading and writing the *Hours* partitions?

63

First bucket: 15 pages, Second bucket: 8 pages, 3rd-12th buckets: 1 page

30 (read) + 15 + 8 + 1*10 (write) = 63 I/Os

iii. (2 pt) If we use hash function B for the second partitioning pass, will recursive partitioning be needed after the second pass?

○ Yes

[✓ No]

iv. (4 pt) What is the total I/O cost of this Grace Hash Join?

554

For SeatPlan:

We have to recursively partition the first bucket from the first pass.

First bucket: 50 (read) + 13 (write first bucket) + 13 (write second bucket) + 3 * 10 (write for 3rd - 12th buckets) = 106 I/Os

For Hours:

We have to recursively partition the first bucket from the first pass.

First bucket: 15 (read) + 4 (write for first bucket) + 4 (write for second bucket) + 1 * 10 (write for 3rd - 12th buckets) = 33 I/Os

106 + 33 = 139 I/Os

Partitioning cost: 205 + 63 + 139 = 407 I/Os

Read SeatPlan: 13 + 13 + 3 * 10 + 25 + 3 * 10 = 111 I/Os

Read Hours: 4 + 4 + 1 * 10 + 8 + 1 * 10 = 36 I/Os

407 + 111 + 36 = 554 I/Os

No more questions.