# CS W186 Spring 2020 Midterm 2 (Online)

**Do not share this exam until solutions are released.**

## Contents:

- The midterm has *6 questions*, each with multiple parts, and worth a total of *85 points*.

## Taking the exam:

- You have *150 minutes* to complete the midterm.
- You may print this exam or download it onto an electronic device to work on it.
- For each question, submit only your *final answer* on Gradescope.
- 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) and do not use LaTeX.
- Questions tagged with [**EXPLANATION**] require you to show work and not doing so will result in **no credit**. You can do this by inputting an explanation in the text field or submitting a file (photo, PDF) of your work. The text field supports LaTeX by using $$insert expression here$$.
- Make sure to submit on Gradescope at the end of the exam.

## Aids:

- This exam is open-book, open-calculator, and open-Internet.
- **You must work individually on this exam.**

## 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 Pre-Exam (1 point)

1. (1 point) Please read and sign the Honor Code Statement on Gradescope.

## 2 Join Zoom Meeting (12 points)

We want to join the `Meetings(zoom_id, class_id, host)` table with the `Classes(cid, dept, course_no, professor)` on the join condition: `class_id = cid`.

- The `Meetings` table has 80 pages with 100 records per page.
- The `Classes` table has 40 pages with 50 records per page.
- We have B = 12 buffer pages.
- We have a clustered Alternative 2, height 3 B+ tree on `Meetings.class_id`. Assume that each class has 4 matches in the `Meetings` table.
- We have no indexes on the `Classes` table.
- For every question, choose the join order that minimizes the I/O cost.

Questions 3-5 walk through doing an unoptimized Sort Merge Join and questions 6-10 walk through doing a Grace Hash Join. All other questions are independent of each other.

1. (2 points) How many I/Os will a Block Nested Loop Join cost?

**Solution:**
Meetings as the outer table: 80 + ceil(80/10) * 40 = 400
Classes as the outer table: 40 + ceil(40/10) * 80 = **360 I/Os**

2. (2 points) [**EXPLANATION**] How many I/Os will an Index Nested Loop Join cost?

**Solution:** We must choose Classes as the outer table because we only have an index on the meetings table. The inner table must have an index built on it so we can use it to find matching records.
$[C] + |C|$ * (cost to reach leaf + cost to read data pages)
The cost to read data pages is only 1 because it is a clustered index and each page can hold more than 4 records.
40 + 2000 * (4 + 1) = **10,040 I/Os**

3. (1 point) How many I/Os does the merging phase (2nd half) of Sort Merge Join take on average?

**Solution:**
There are only 4 matches per value, so we will never have to backtrack an entire page. Therefore the merging pass is only a linear pass over each table:
$[C] + [M]$ =**120 I/Os**

4. (1 point) [**EXPLANATION**] How many I/Os does the Sort Merge Join cost in total?

**Solution:** Cost of SMJ is: Sort(Meetings) + Sort(Classes) + Merge(Meetings, Classes)
Sort meetings:
$[M] > B$, so can't be sorted in 1 pass
$[M] <= B(B-1)$, so can be sorted in 2 passes
2 * (# passes) * $[M] = 2 * 2 * 80 = 320$

Sort Classes:
$[C] > B$, so can't be sorted in 1 pass
$[C] <= B(B-1)$, so can be sorted in 2 passes
2 * (# passes) * [C] = 2 * 2 * 40 = 160

Total cost: 320 + 160 + 120 = **600 I/Os**

5. (1 point) [**EXPLANATION**] What is the minimum number of buffer pages we could have and not increase the number of I/Os in the previous question?

**Solution:**
As long as we have 3 buffer pages we can do merging as usual.
To avoid increasing the I/Os we need to be able to sort each table in 2 passes.
Therefore $B(B-1) >= N$
$B(B-1) >= 80$
We could solve for B, but I'll do guess and check...
9(8) = 72, so that doesn't work.
10(9) = 90, so that does work.
**10 buffer pages is the min to avoid increasing I/Os.**

For the rest of this problem we will go through a Grace Hash Join step-by-step. Assume that we have hash functions that divide the data uniformly into partitions.

6. (1 point) How large are the partitions for the `Meetings` table?

   **Solution:**
   80 pages get divided into 11 partitions (we need one input buffer). $80/11 = 7.3 \rightarrow$ **8 pages per partition**.

7. (1 point) How large are the partitions for the `Classes` table?

   **Solution:**
   40 pages get divided into 11 partitions. $40/11 = 3.6 \rightarrow$ **4 pages per partition.**

8. (1 point) How many partitioning passes will we need to do in total (consider a pass to be a full pass over each relation, so a and b in total is 1 pass).

   **Solution:**
   **1 partitioning pass (both the Classes and meetings relations can fit in memory).**

9. (1 point) How many I/Os are spent just on the partitioning passes?

   **Solution:**
   We only need to do one pass, so we need to read in each relation and write out all the partitions.
   80 + 8 * 11 + 40 + 4 * 11 = **252 I/Os**

10. (1 point) How many I/Os are spent just on the build and probe phase? Recall that (like all joins) the final output is not materialized.

    **Solution:**
    We need to read in all the partitions we wrote out during the last hashing pass. We don't need to write the result back to disk because it is streamed to the operators above it.
    8 * 11 + 4 * 11 = **132**

## 3 Query (O_O) (23 points)

This question will be optimizing the following query:

```
SELECT * FROM R
  INNER JOIN S on S.a = R.a OR S.b > R.b
  INNER JOIN T on T.c = S.c
WHERE R.b > 100 AND S.c != 9
GROUP BY S.c;
```

There are 3 relations R(a,b,c), S(a,b,c), T(a,b,c):

- R - 500 pages with 3 tuples/page
- S - 300 pages with 4 tuples/page
- T - 200 pages with 2 tuples/page

There are 2 indexes:

- Alternative 2 height 3 **unclustered index on S.c** with 35 leaf pages
- Alternative 2 height 4 **clustered index on R.a** with 60 leaf pages

We have the following data (all integers) on table column values:

- R.a ranges from [35-45] inclusive.
- S.a ranges from [40-50] inclusive.
- S.c ranges from [1-50] inclusive.
- T.c ranges from [1-100] inclusive.

To optimize this query, we must first find the best way to access our relations. The table below can be used to compare different access methods.

| Relation | Access Method | I/O Cost | Interesting Order | Retained | Output Size |
|---|---|---|---|---|---|
| R | File Scan | 500 | None | Yes | |
| | Index scan (R.a) | | | | |
| S | File Scan | 300 | None | Yes | |
| | Index scan (S.c) | | S.c | | |
| T | File Scan | 200 | None | Yes | 200 |

1. (2 points) **[EXPLANATION]** What is the I/O cost of performing an index scan on R using the index on R.a?

**Solution:** It takes 4 + 60 + 500 = **564 I/Os** to perform an index scan as the index has height = 4, 60 leaf pages, and is clustered. Since there is no selection predicate on R.a in the query, a full index scan must be performed.

2. (1 point) Does the index scan on R yield an interesting order?
   A. Yes - R.a
   [✓ B. No]

**Solution:** No, the index scan on R.a does not yield an interesting order because R.a is not used in any ORDER BY or GROUP BY clause and Sort Merge Join cannot be used in a non-equijoin.

3. (1 point) Is the index scan on R retained?
   A. Yes
   [✓ B. No]

**Solution:** No, the index scan is not retained because it has higher cost than file scan and maintains no interesting order

4. (1 point) What is the output size of R in pages?

**Solution:** 1/10 * 500 = **50 pages**. We push down selections and since no data on the column values of R.b is known, the selectivity of the predicate, R.b > 100, is 1/10.

5. (2 points) **[EXPLANATION]** What is the I/O cost of performing an index scan on S using the index on S.c?

**Solution:** 3 + 35 + (49/50)*1200 = **1214 I/Os**. We need to read of all the index's leaf pages but only read 49/50 of the records since we want to filter by S.c != 9.

6. (1 point) What is the output size of S in pages?

**Solution:** 49/50 * 300 = **294**. We push down selections and the selectivity of S.c ! = 9 is $(1-1/50) =$ 49/50.

After finishing the first pass, we use the 2nd pass to find the best way to join sets of 2 tables. The table below can be used to visualize the results of the 2nd pass after answering the questions below.

There are **12 buffer pages** and for the sake of simplicity, assume that there are no duplicates in the join columns. **Disregard the output sizes you computed for the first pass and use R - 60 pages, S - 270 pages for any calculations in the 2nd pass**.

| Relations | Best Join | I/O Cost | Interesting Order | Output Size |
|---|---|---|---|---|
| {S,R} | | | | |
| {T,S} | | 1670 | | |

7. (1.5 points) **[EXPLANATION]** What is the best join for the set of tables {S,R}?
   **A. BNLJ**
   B. INLJ
   C. SMJ
   D. GHJ

**Solution:** Since the join between R and S is not an equijoin, the best option is BNLJ since INLJ, SMJ, and GHJ can't be used.

8. (1.5 points) What is the I/O cost of the best join for {S,R}?

**Solution:** 60 + 60/10 * 270 = **1680 I/Os**. Join order matters when performing BNLJ and R join S yields the smaller number of I/Os as S join R results in 270 + 270/10 * 60 = 1890 I/Os .

9. (1 point) Is there an interesting order as a result of joining {S,R}?

   A. Yes - S.a/R.a

   B. Yes - S.b

   C. Yes - R.b

   **D. No**

**Solution:** No interesting orders are produced from BNLJ.

10. (1.5 points) What is the output size of the resulting relation after joining {S,R} in pages?

**Solution:** The selectivity of S.a = R.a OR S.b > R.b is (1/11 + 1/10 - 1/110). Therefore, the estimated output of the join is selectivity * 60 * 270 = **2946 pages.**

11. (3 points) **[EXPLANATION]** What is the best join for the set of tables {T,S}?

   A. BNLJ

   B. INLJ

   **C. SMJ**

   D. GHJ

**Solution:**
BNLJ:
T join S = 200 + 200/10 * 270 = **5600 I/Os**
S join T = 270 + 270/10 * 200 = 5670 I/Os

INLJ:
T join S = 200 + 400*5 = **2200 I/Os**

**SMJ:**
sort(T) + merge = 3 * 2(200) + 270 + 200 = **1670 I/Os**

GHJ:
partition(S) + partition(T) + conquer = 270 + 200 + 2(25(11) + 19(11)) + 2(3(121) + 2(121)) = 470 + 968 + 1210 = **2648 I/Os**

12. (1 point) Is there an interesting order as a result of joining {T,S}?

   **A. Yes - S.c/T.c**

   B. No

**Solution:** Yes, we eventually want to GROUP BY on S.c and SMJ produces output sorted on the join column, S.c. In this problem, the best join just happened to be the join with the relevant interesting order so we don't need to keep any other joins from this pass.

13. (1 point) What is the output size of the resulting relation after joining {T,S} in pages?

**Solution:** 1/100 * 270 * 200 = **540 pages**. The selectivity of T.c = S.c is 1/max(100, 50) = 1/100.

Upon finishing the 2nd pass, we've found the best plan for joining any 2 tables together. In the final pass, we will find the best plan for joining any 3 tables together using the estimated costs from the last pass.

**Disregard the output sizes you computed for the second pass and use {S, R} - 2900 pages for any calculations in the 3rd pass.**

| Relations | Best Join | I/O Cost | Interesting Order | Output Size |
|---|---|---|---|---|
| {{S,R}, T} |  | 15618 |  | 5800 |
| {{T,S}, R} | BNLJ | 3500 | None | 5700 |

14. (3 points) **[EXPLANATION]** What is best join for the set of tables {{S,R},T}?

    A. BNLJ

    B. INLJ

    C. SMJ

    **D. GHJ**

**Solution:**
BNLJ:
{S,R} join T = 2900 + 2900/10 * 200 = **60900 I/Os**
don't consider T join {S, R} since not left deep

INLJ:
no index on T.c

SMJ:
sort({S,R} + sort(T) + merge = 2(2900) * 4 + 2(200) * 3 + 2900 + 200 = **27500 I/Os**

**GHJ**:
partition({S,R}) + partition(T) + conquer = 2900 + 200 + 2(264(11) + 19(11)) + 2(24(121) + 2(121)) = 3100 + 6226 + 6292 = **15618 I/Os**.

15. (1.5 points) **[EXPLANATION]** What is the final plan that QO takes?

    **A. {{S,R},T}**

    B. {{T,S},R}

**Solution:** {{S,R},T} will be taken since it yields the lowest overall cost for executing the query. The result of executing the join creates an interesting order on S.c for grouping so nothing more needs to be done.

Even though the join for {{T,S},R} yields a smaller number of I/Os, adding on an operator to fulfill the GROUP BY clause makes the plan extremely expensive.

sorting {{T,S},R} = 2(5700) * 4 = 45600 I/Os
hashing {{T,S},R} = 5700 + 2(519 * 11) + 2(48 * 121) + 3(5 * 1331) = 45669 I/Os

## 4 Parallel Patients (13 points)

The hospital want to improve the efficiency of their database, and they hired you to help them query the database in parallel. You have access to 5 machines and the following schemas and sizes of three tables:

`Patients(name, resident_id, level_of_sickness, state, hospital_id)`: 1000 pages
`Doctors(name, resident_id, hospital_id)`: 100 pages
`Hospitals(hospital_id, hname, address)`: 10 pages

`Patients` has a column called `level_of_sickness`, which is an integer between 1 and 10 (inclusive) that indicates how severe the sickness is, 1 being the least severe and 10 being the most severe, and every level of sickness exists in `Patients`. You want to find the more severe patients, so you run the following query:

```
SELECT * FROM Patients WHERE level_of_sickness > 7;
```

1. (1 point) What is the I/O cost of the above query if `Patients` is range partitioned on `level_of_sickness` to the 5 machines, with the following values and sizes for each machine?

| **Machine** | **Values** | **Size (number of pages)** |
|---|---|---|
| M1 | 1, 2 | 270 |
| M2 | 3 | 170 |
| M3 | 4, 5, 6 | 300 |
| M4 | 7, 8 | 60 |
| M5 | 9, 10 | 200 |

**Solution:** 260. We need to scan everything in M4 and M5.

2. (1 point) True or False: It is **possible** to achieve a better I/O cost with hash partition.

**Solution:** True. If the hash function sends values of 8, 9, 10 to one machine and other values to other machines, then we only need to scan that one machine, which results in fewer I/Os because we don't need to scan the 7s as M4 does above.

You want to find all the patients in California, so you run the following query:

```
SELECT * FROM Patients WHERE state = 'CA';
```

3. (1 point) What is the I/O cost of the above query if `Patients` is round-robin partitioned to the 5 machines?

**Solution:** 1000. We need to scan everything.

4. (1 point) True or False: It is **guaranteed** to achieve a better I/O cost with hash partition on `state` using any hash function such that none of the partitions are empty.

**Solution:** False. We only need to scan the machine with 'CA', but if all other machines get one record each, and all of them combined make up less than a full page, than the machine with 'CA' still has all the pages, so the I/O cost is not better.

You want to perform the following join using **Parallel Grace Hash Join**:

```
SELECT * FROM Doctors D, Hospitals H WHERE D.hospital_id = H.hospital_id;
```

Both `Doctors` and `Hospitals` are hash partitioned on `hospital_id` to the 5 machines, with the following numbers of pages:

| **Machine** | **Number of pages from `Doctors`** | **Number of pages from `Hospitals`** |
|---|---|---|
| M1 | 40 | 5 |
| M2 | 30 | 2 |
| M3 | 10 | 1 |
| M4 | 10 | 1 |
| M5 | 10 | 1 |

5. (2 points) **[EXPLANATION]** Each machine has 8 buffer pages, and each I/O takes 1ms. How long does it take the perform the join, in ms?

   Assume the hash partitioning is already done (so don't include that in your answer).

   **Solution:** 45ms. M1 has the most pages for both tables, so that will be the bottle neck. Since 5 pages from `Hospitals` fit in B-2=6, we can directly build and probe, so the latency is $(40 + 5) \cdot 1\text{ms} = 45\text{ms}$.

6. (1 point) True or False: It is possible to reduce the time in the previous question by increasing the number of buffer pages.

   **Solution:** False. The smaller table on each machine already fits in buffer, so there is already no recursive partitioning and increasing the number of buffer pages won't help.

7. (1 point) True or False: It is possible to reduce the time in the previous question by using a different hash function for the initial hash partitioning.

   **Solution:** True. A more uniform hash function can relax the bottle neck of M1 and reduce the time.

Unfortunately, some doctors are also sick, and you want to find out the sick doctors. You perform the following join using **parallel un-optimized Sort Merge Join** on the 5 machines (M1, M2, M3, M4, M5):

```
SELECT * FROM Patients P, Doctors D WHERE P.resident_id = D.resident_id;
```

8. (2 points) **[EXPLANATION]** Initially, everything is on M1. Assume we can range partition both `Patients` and `Doctors` on `resident_id` perfectly so that each machine gets the same number of pages. What is the network cost of this initial partitioning in pages?

   **Solution:** 880. 1/5 of the pages will stay on M1, and 4/5 need to be sent to the other machines. So the network cost is $4/5 \cdot (1000 + 100) = 880$ pages.

9. (3 points) **[EXPLANATION]** Each machine has 8 buffer pages, and each I/O takes 1ms. How long does it take to perform this join (in ms) after partitioning evenly data across machines? Assume that the "conquer" phase of sorting is streamed from the network, so that each machine does not incur a read I/O after data is received from the network.

   **Solution:** 1720.

   Each machine gets 200 pages of `Patients` and 20 pages of `Doctors`, and since they run in parallel, we can just look at one machine.

   As the assumption says, we don't count the read from the initial partitioning phase, so we start from the initial write after the "conquer" phase, which has $200 + 20 = 220$ I/Os.

   Sorting `Patients` takes 3 passes, so $2 \cdot 2 \cdot 200 = 800$ I/Os.

   Sorting `Doctors` takes 2 passes, so $1 \cdot 2 \cdot 20 = 40$ I/Os.

   The final pass for merging them takes $200 + 20 = 220$ I/Os.

   So each machine takes 1280 I/Os, or 1280ms.

## 5 New Whip, Who Dis? (17 points)

Ever since you helped Alon Tusk optimize Flux Motors' database of car orders, you've become the go-to Database Engineer in the company! Flux has recently started taking orders for the new car they released, and it's a major hit! Once again, Alon Tusk needs your help making sure the transactions run smoothly.

Alon hands you a snippet of a schedule and asks you to analyze it. Consider the following schedule for all questions:

| **Transaction** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|---|---|---|---|---|---|---|---|---|
| T1 | R(A) | | | | W($C_1$) | R(Y) | | |
| T2 | | | W(B) | | | | W(C) | |
| T3 | | R($A_1$) | | | | | | W($B_2$) |
| T4 | | | | R($B_1$) | | | | |

For the entirety of this problem, consider a database X with a table Y. Y has pages A, B, and C. Page A has tuples $A_1$, $A_2$, page B has tuples $B_1$, $B_2$, and page C has tuples $C_1$, $C_2$. Additionally, use the multi-granularity locking (with minimum privilege).

1. (3 points) What is the set of edges of the waits-for graph for the schedule? We write $(T_i, T_j)$ if there exists a directed edge from $T_i$ to $T_j$.

   A. $\{(T_2, T_1), (T_1, T_2), (T_2, T_3), (T_2, T_4)\}$

   **B. $\{(T_1, T_2), (T_2, T_1), (T_3, T_2), (T_4, T_2)\}$**

   C. $\{(T_1, T_2), (T_2, T_3), (T_2, T_4), (T_4, T_3)\}$

   D. $\{(T_2, T_1), (T_3, T_2), (T_4, T_2), (T_3, T_4)\}$

   E. None of the above

   **Solution:** B. $\{(T_1, T_2), (T_2, T_1), (T_3, T_2), (T_4, T_2)\}$

2. (1 point) True or False: This schedule has deadlock.

   **Solution:** True. There is a cycle between $T_1$ and $T_2$

3. (5 points) [**EXPLANATION**] List all the locks that T1 has after timestep 8 in the order of acquisition with the first lock acquired being on the top left and most recent lock being on the bottom right. If a lock gets promoted, it should stay in the same position (as done in the project). You may or may not need all the boxes. Leave unused boxes blank.

| | | | | |
|---|---|---|---|---|
| Lock 1 | Lock 2 | Lock 3 | Lock 4 | Lock 5 |
| Lock 6 | Lock 7 | Lock 8 | Lock 9 | Lock 10 |

**Solution:** IX(X), IX(Y), S(A), IX(C), X($C_1$)

After timestep 1, T1 will have acquired IS(X), IS(Y), and IS(A). After timestep 5, T1 will promote its IS locks on X and Y to IX locks (keeping time of acquisition following promote rules) then acquire IX(C) and X($C_1$). At timestep 6, T1 will try to release IX(Y) and acquire SIX(Y), which will be blocked due to T2 having an IX lock already. Remember, you cannot promote to SIX!

Worried about transactions becoming deadlocked, Alon gives you the task of preventing deadlocks from happening. Having taken CS W186, you remember learning about some deadlock prevention strategies!

For the following four questions, assume that $T_1$ started first, $T_2$ started second, $T_3$ started third, and $T_4$ started last. If a transaction gets aborted, it will immediately release all locks it has acquired and the wait queue will be processed. Additionally, if a transaction gets placed on a waiting queue (blocked), locks that the transaction acquired up to that point are not released and the actions for that transaction **while it is blocked** do not happen (as if the action was never on the schedule). Additionally, assume promotes are implemented as described on Project 4.

4. (2 points) [**EXPLANATION**] If you use the **wound-wait** approach, which transaction(s) will end up getting aborted? **Select all correct choices. If no transaction gets aborted, select None**

   A. $T_1$

   **B.** $T_2$

   C. $T_3$

   D. $T_4$

   E. None

   **Solution:** B. At timestep 6, T1 aborts T2 because it has higher priority and wants a SIX lock on Y and T2 is holding a conflicting IX lock on Y.

5. (2 points) Which transaction(s) will end up blocked **by the end of timestep 8** if using wound-wait? **Select all correct choices. If no transaction gets blocked, select None**

   A. $T_1$

   B. $T_2$

   **C.** $T_3$

   D. $T_4$

   E. None

   **Solution:** C. Transaction T4 gets blocked due to wanting IS(B) by T2 already holding X(B), but T2 gets aborted at timestep 6 by T1. Then at timestep 8, T3 tries getting IX(Y) but is blocked due to T1 holding SIX(Y).

6. (2 points) If you use the **wait-die** approach, which transaction(s) will end up getting aborted? **Select all correct choices. If no transaction gets aborted, select None**

   A. $T_1$

   **B.** $T_2$

   **C.** $T_3$

   **D.** $T_4$

   E. None

**Solution:** B, C, D. At timestep 4, T4 wants IS(B), but since it has lower priority than T2, it will abort. At timestep 7, T2 wants X(C) but T1 has IX(C) and is higher priority, therefore T2 aborts. Then at timestep 8, T3 tries to get an IX(Y), but is aborted due to T1 having an SIX(Y).

7. (2 points) [**EXPLANATION**] Which transaction(s) will end up blocked **by the end of timestep 8** if using wait-die? **Select all correct choices. If no transaction gets blocked, select None**

   A. $T_1$

   B. $T_2$

   C. $T_3$

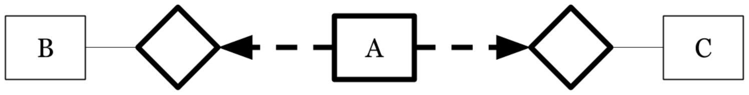
   D. $T_4$

   **E. None**

**Solution:** E. At timestep 6, T1 wants SIX(Y), but is blocked due to T2 holding IX(Y). However, at timestep 7 T2 gets aborted and T2's locks are dropped and the queue gets processed allowing T1 to get the SIX lock on Y. All other transactions other than T1 are aborted.

## 6 A Zoomed Relationship (19 points)

In this problem, you may encounter the concept of a "multi-weak entity" which is a weak entity that can be uniquely identified by an owner entity belonging to one of multiple entity sets. This means that every entity in a multi-weak entity set must participate only once across all its identifying relationship sets. Visually, we will denote the participation of a multi-week entity in an identifying relationship with a **bold dashed arrow**. For example, in the following ER diagram, every entity in A can be uniquely identified by an owner entity that belongs to either B or C.



Also, assume weak and multi-weak entities can participate in non-identifying relationships.

Now onto the problem. With all classes moving online for the rest of the semester, Berkeley is partnering with Zoom Video Communications to develop a new model for tracking online learning. Fill in the ER diagram (attached on page 19) with the series of following constraints. It may be helpful to fill out the entire diagram with all the constraints given on page 19 before answering questions 1-14.

For questions 1-4, what edges should be drawn with the following constraints? Students must take at least 1 course, courses must have at least 1 student taking it and at least 1 professor instructing it, and professors can teach at most 1 course.

1. (1 point) What type of edge should be drawn between **Student** and **Takes**?
   - A. Thin Line
   - B. Thin Arrow
   - **C. Bold Line**
   - D. Bold Arrow
   - E. Bold Dashed Arrow

**Solution:** A Student must take at least 1 Course.

2. (1 point) What type of edge should be drawn between **Course** and **Takes**?
   - A. Thin Line
   - B. Thin Arrow
   - **C. Bold Line**
   - D. Bold Arrow
   - E. Bold Dashed Arrow

**Solution:** A Course must have at least 1 Student taking it.

3. (1 point) What type of edge should be drawn between **Professor** and **Instructs**?
   - A. Thin Line
   - **B. Thin Arrow**
   - C. Bold Line
   - D. Bold Arrow
   - E. Bold Dashed Arrow

**Solution:** A Professor can instruct up to 1 Course.

4. (1 point) What type of edge should be drawn between **Course** and **Instructs**?
   - A. Thin Line
   - B. Thin Arrow
   - **C. Bold Line**
   - D. Bold Arrow
   - E. Bold Dashed Arrow

**Solution:** A Course must have at least 1 Professor instructing it.

For questions 5-8, what edges should be drawn with the following constraints? Students can now optionally sign up exactly once as a Zoom User while professors are required to sign up and only once. Each Zoom User account belongs to exactly 1 person (student or professor). Interacting in any form with a Zoom Lecture (attending, lecturing, etc.) can only be done through a Zoom User.

5. (1 point) What type of edge should be drawn between **Student** and **Sign Up (Student)**?
   - A. Thin Line
   - **B. Thin Arrow**
   - C. Bold Line
   - D. Bold Arrow
   - E. Bold Dashed Arrow

**Solution:** A Student can optionally sign up exactly once as a Zoom User.

6. (1 point) What type of edge should be drawn between **Zoom User** and **Sign Up (Student)**?
   - A. Thin Line
   - B. Thin Arrow
   - C. Bold Line
   - D. Bold Arrow
   - **E. Bold Dashed Arrow**

**Solution:** A Zoom User is a multi-weak set since it can be uniquely identified by either a Student or a Professor i.e. it can only belong to either a Student of a Professor.

7. (1 point) What type of edge should be drawn between **Professor** and **Sign Up (Professor)**?
   - A. Thin Line
   - B. Thin Arrow
   - C. Bold Line
   - **D. Bold Arrow**
   - E. Bold Dashed Arrow

**Solution:** A Professor must sign up exactly once as a Zoom User.

8. (1 point) What type of edge should be drawn between **Zoom User** and **Sign Up (Professor)**?
   - A. Thin Line
   - B. Thin Arrow
   - C. Bold Line
   - D. Bold Arrow
   - **E. Bold Dashed Arrow**

**Solution:** Same reasoning as 6.

For questions 9-12, what edges should be drawn with the following constraints? To test the online platform, all Zoom Users are required to attend at least 1 Zoom Lecture. A Zoom Lecture must have at least 1 Zoom User lecturing but there is no requirement on the number of Zoom Users attending.

9. (1 point) What type of edge should be drawn between **Zoom User** and **Attends**?
   - A. Thin Line
   - B. Thin Arrow
   - **C. Bold Line**
   - D. Bold Arrow
   - E. Bold Dashed Arrow

   **Solution:** All Zoom Users must attend at least 1 Zoom Lecture.

10. (1 point) What type of edge should be drawn between **Zoom Lecture** and **Attends**?
    - **A. Thin Line**
    - B. Thin Arrow
    - C. Bold Line
    - D. Bold Arrow
    - E. Bold Dashed Arrow

    **Solution:** There is no requirement on the number of Zoom Users attending a Zoom lecture.

11. (1 point) What type of edge should be drawn between **Zoom User** and **Lectures**?
    - **A. Thin Line**
    - B. Thin Arrow
    - C. Bold Line
    - D. Bold Arrow
    - E. Bold Dashed Arrow

    **Solution:** A Zoom User is not required to lecture for any Zoom lectures, and there is also no constraint on the number of Zoom Lectures a Zoom User can lecture.

12. (1 point) What type of edge should be drawn between **Zoom Lecture** and **Lectures**?
    - A. Thin Line
    - B. Thin Arrow
    - **C. Bold Line**
    - D. Bold Arrow
    - E. Bold Dashed Arrow

**Solution:** A Zoom Lecture must have at least 1 Zoom User lecturing it.

For questions 13-14, what edges should be drawn with the following constraints? A Zoom Lecture can optionally be part of at most 1 Course, but a Course must have at least 1 Zoom Lecture be a part of it.

13. (1 point) What type of edge should be drawn between **Zoom Lecture** and **Part of**?

    A. Thin Line

    **B. Thin Arrow**

    C. Bold Line

    D. Bold Arrow

    E. Bold Dashed Arrow

**Solution:** A Zoom Lecture can be part of at most 1 Course.

14. (1 point) What type of edge should be drawn between **Course** and **Part of**?

    A. Thin Line

    B. Thin Arrow

    **C. Bold Line**

    D. Bold Arrow

    E. Bold Dashed Arrow

**Solution:** A Course must have at least one Zoom Lecture be part of it.

Consider the following attribute set R = {SOCIALDTNG}

15. (3 points) **[EXPLANATION]** Decompose R into BCNF in the order of the following FDs: S $\rightarrow$ CLD, D $\rightarrow$ NG, AO $\rightarrow$ S, G $\rightarrow$ IT. Which of the following tables are included in the final decomposition?

    **A. SCLD**

    **B. AOS**

    **C. DNG**

    **D. GIT**

    E. SOANG

    F. SOIAT

**Solution:** S $\rightarrow$ CLD decomposes R into SCILDTNG and SOA. D $\rightarrow$ NG decomposes SCILDTNG into IDTNG and DSCL. AO $\rightarrow$ S is ignored since AO is a superkey. G $\rightarrow$ IT decomposes IDTNG into GIT and GDN. The final decomposition is: GIT, GDN, DSCL, SOA.

16. (1 point) **[EXPLANATION]** Suppose we decompose R into SOCLDNG and OIATG. Is this decomposition dependency preserving with respect to the FDs in the previous question?

    A. Yes

    **B. No**

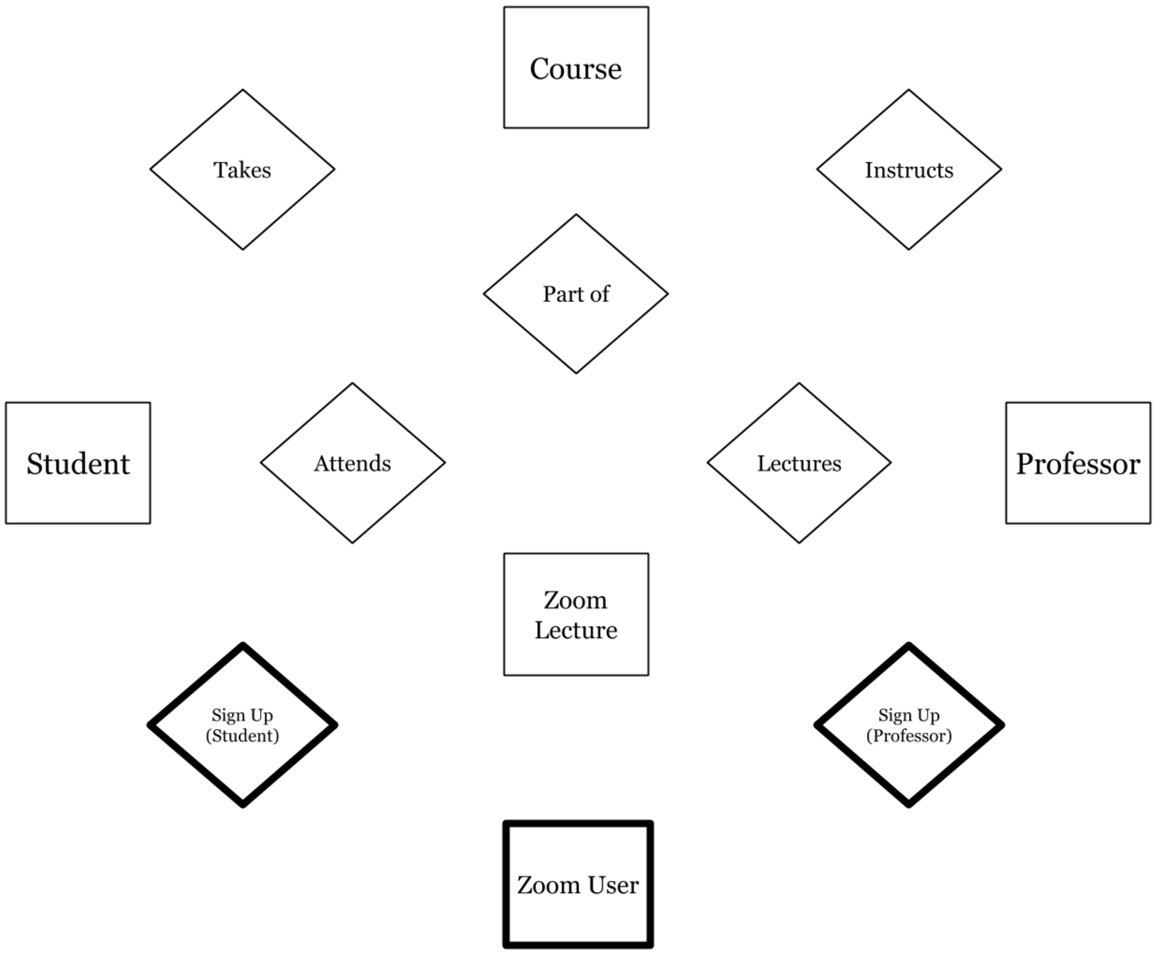
**Solution:** AO $\rightarrow$ S is not preserved.

17. (1 point) **[EXPLANATION]** Is the same decomposition lossless with respect to the FDs in the previous question?

    A. Yes

    **B. No**

**Solution:** SOCLDNG $\cap$ OIATG = OG which is not a key of either relation.



**All Constraints**

1. Students must take at least 1 course, courses must have at least 1 student taking it and at least 1 professor instructing it, and professors can teach at most 1 course.
2. Students can now optionally sign up exactly once as a Zoom User while professors are required to sign up and only once. Each Zoom User account belongs to exactly 1 person (student or professor). Interacting in any form with a Zoom Lecture (attending, lecturing, etc.) can only be done through a Zoom User.
3. To test the online platform, all Zoom Users are required to attend at least 1 Zoom Lecture. A Zoom Lecture must have at least 1 Zoom User lecturing but there is no requirement on the number of Zoom Users attending.
4. A Zoom Lecture can optionally be part of at most 1 Course, but a Course must have at least 1 Zoom Lecture be a part of it.