CS W186 Fall 2019 Midterm 2

Do not turn this page until instructed to start the exam.

Contents:

- You should receive one *single-sided answer sheet* and a 14-page *exam packet*.
- The midterm has *6 questions*, each with multiple parts, and worth a total of *74 points*.

Taking the exam:

- You have *110 minutes* to complete the midterm.
- All answers should be written on the answer sheet. The exam packet will be collected but not graded.
- For each question, place only your *final answer* on the answer sheet; *do not show work*.
- For multiple choice questions, please *fill in the bubble or box completely* as shown on the left below. ***Answers marking the bubble or box with an X or check mark may receive a point penalty***.

- A blank page is provided at the end of the exam packet for use as scratch paper.

Aids:

- You are allowed **one handwritten** 8.5” × 11” double-sided pages of notes.
- *No electronic devices are allowed on this exam.* No calculators, tablets, phones, smartwatches, etc.

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 Questions (0 points)

1. (0.0001 points) Pick an integer in $[0, 750)$ that no one else taking this exam picked.
2. (0.0001 points) How many pearls are in a typical 24oz boba drink?

2 Grace Hopper Joins (12 points)

You are the organizer for the 2020 Grace Hopper Conference. You are given two tables: Table R, which contains information about all students taking CS W186, and Table S, which contains information about all UC Berkeley students attending the conference. You wanted to know how many students in CS W186 are attending the conference, and whether they have enough slip minutes to put off their project and have a great time in Orlando. You decide to get this information by using Block Nested Loop Join (BNLJ) on both tables. **For the next 2 parts, you may assume that the size of Table R is 100 pages, and the size of Table S is 50 pages. You can also assume that you have 52 pages of buffer space.**

1. (1 point) Calculate the I/O cost of BNLJ, assuming that Table R is the outer relation, and Table S is the inner relation.

2. (1 point) Calculate the I/O cost of BNLJ, assuming that Table S is the outer relation, and Table R is the inner relation.

3. (1 point) True/False: Using the smaller relation as the outer one in the BNLJ algorithm will always be cheaper than using the larger relation as the outer one in BNLJ.

Now, we will perform Grace Hash Join using tables R and S. Operate under the following assumptions:

- $[R] = 60$, $[S] = 20$
- B (number of buffer pages) = 6

4. (2 points) Assuming that we use perfect hash functions and uniformly partition our data at every step of the join process, what is the I/O cost of executing Grace Hash Join?

5. (4 points) Now, assume that the first hash function we use was imperfect. As a result, we end up with partitions after pass 0 that look like this:

   - Partition 1: [R] = 20, [S] = 4
   - Partition 2: [R] = 20, [S] = 7
   - Partition 3: [R] = 10, [S] = 2
   - Partition 4: [R] = 4, [S] = 5
   - Partition 5: [R] = 6, [S] = 2

   Assuming that we use a perfect hash function on each partition following this initial mishap of a partitioning phase, what is the total number of I/Os, from start to finish, needed to execute a Grace Hash Join between R and S?

For the next 3 parts, refer to the following pseudo-code for sort merge join:

```
do {
    if (!mark) {
        while (r < s) { advance r }
        while (r > s) { advance s }
        mark = s                          (i)
    }
    if (r == s) {
        result = <r, s>
        advance s
        return result
    } else {
        reset s to mark                   (ii)
        advance r
        mark = NULL                       (iii)
    }
}
```

Everybody makes mistakes, everybody has those days. Whilst implementing SMJ, you may have forgotten some lines of code. For the following questions, indicate the problem you would run into if you were to forget the line of code as indicated by (i), (ii), and (iii). Assume that each question is independent of one another (i.e. you're only minorly flawed -but otherwise you're perfect- and forgot one line of code in each of the questions), also assume that the do loop terminates once one or both iterators, r and s, reach the end of the table.

6. (1 point) What is the problem we run into when we forget to write line (i)?
   A. Error when joining the tables using SMJ
   B. Too many records in the resulting table after running SMJ
   C. Too few records in the resulting table after running SMJ
   D. Nothing wrong

7. (1 point) What is the problem we run into when we forget to write line (ii)?
   A. Error when joining the tables using SMJ
   B. Too many records in the resulting table after running SMJ
   C. Too few records in the resulting table after running SMJ
   D. Nothing wrong

8. (1 point) What is the problem we run into when we forget to write line (iii)?
   A. Error when joining the tables using SMJ
   B. Too many records in the resulting table after running SMJ
   C. Too few records in the resulting table after running SMJ
   D. Nothing wrong

3 Perpendicular Query Processing (17 points)

1. (1 point) What type of join helps reduce network costs when we have two tables with drastically different sizes?
2. (1 point) What partitioning scheme is generally best if we only care about ensuring that each machine does an equal amount of work?
3. (1 point) Which type of parallelism is most likely better if we want to scale up an application with millions of users making simple queries?
   A. Interquery
   B. Intraquery
4. (1 point) Bushy Tree and Pipeline parallelism are subtypes of which class?
   A. Interoperator
   B. Intraoperator

For questions 5-8 we are trying to do a parallel sort on the Students table. The assumptions are:

- $[Students] = 1500$
- We have 5 identical machines (m1, m2, m3, m4, m5) that have B = 10
- All of the pages start on machine 1
- Each page is 1KB
- We are able to partition perfectly so that each machine gets the same number of pages

5. (1 point) What partitioning scheme will we use?
   A. Range
   B. Hash
   C. Round Robin
   D. Other
6. (2 points) How many disk I/Os will machine 1 have to do during the initial partitioning?
7. (2 points) What is the network cost of this sorting operation? Remember that we do not need to re-aggregate the data on m1. We can leave data pages on all 5 machines.
8. (2 points) How many I/Os will each machine need to do after the initial partitioning phase over the machines? Assume that the "conquer" phase of sorting is streamed from the network, so that each machine does not incur an I/O right after data is received from the network.

For questions 9-11 assume that we have hash partitioned the Classes(cid, enrollment) table over three machines on the cid column. The page counts are as follows:

- m1 has 1000 pages
- m2 has 1500 pages
- m3 has 500 pages

Assume 1 I/O takes 1 ms and assume that there is no network or CPU cost.

9. (2 points) How many I/Os will it take to calculate the MAX of the enrollment column?
10. (2 points) How long will it take to calculate the MAX of the enrollment column?
11. (2 points) How many I/Os will need to be done in the worst case to find how many students are enrolled in the class with cid=1?

4 Query Pessimization (17 points)

1. (1 point) True or False: The lowest-cost order in which to join tables is always a left-deep order.
2. (1 point) True or False: During optimization, the Selinger optimizer will calculate the estimated I/O cost for all possible left-deep orders.
3. (1 point) True or False: In any given pass, the number of subplans that the pass returns is always more than the previous pass.

Suppose you are executing the first pass of the Selinger optimizer, and you are currently doing calculations for scanning some table `T` with integer columns `a` and `b`, and 5000 rows over 100 pages.

Suppose the query you are optimizing includes the predicate `T.a > 50`. All predicates will be pushed into the scan when possible.

Suppose we have an unclustered index on column `a` of height 0 (it is just a single node). This index tells us that column `a` has 20 distinct values, the smallest being 1 and the largest being 100.

4. (1 point) What is the cost, in I/Os, of a full table scan on `T`?
5. (1 point) What is the cost, in I/Os, of an index scan on `T` over the index for `a`?
6. (1 point) Now suppose the predicate is `T.a > 50 AND T.b < 40`. There is no index on `b`.

   What is the cost, in I/Os, of an index scan on `T` over the index for `a`?
7. (1 point) Now suppose there is also an index on `b`. It is a height 0 clustered index, and it tells us there are 50 distinct values, the smallest being 30 and the largest being 129.

   What is the cost, in I/Os, of an index scan on `T` over the index for `b`, given the same predicate in the previous problem?
8. (1 point) What is the size, in pages, of the output of this scan?

Now suppose you are executing pass 3 of the Selinger optimizer for a different query:

```
SELECT COUNT(*)
FROM A, B, C
WHERE A.x = B.x AND B.y = C.y
GROUP BY A.x;
```

In order to do this, you will need the results from pass 2 of the Selinger optimizer. So, you ask your friend to run the optimizer up to pass 2 and give you the results.

Your friend runs pass 2 and hands you the following list of subplans. It looks funny... Oh no! They forgot to prune the results. That was the whole point of dynamic programming :(

9. (9 points) Not all of the following subplans should have been returned by pass 2 of the Selinger optimizer. For each of the subplans, mark True if it belongs in the result, and False if it does not.

   You will need to look through all the plans before making your selections!

   A. Plan: A $\bowtie$ B (Block Nested Loop Join). Output order: None.
   Cost: 35,000 I/Os. Output size: 1040 pages.

   B. Plan: B $\bowtie$ C (Block Nested Loop Join). Output order: None.
   Cost: 25,000 I/Os. Output size: 550 pages.

   C. Plan: C $\bowtie$ A (Block Nested Loop Join). Output order: None.
   Cost: 7,500 I/Os. Output size: 3850 pages.

   D. Plan: A $\bowtie$ C (Block Nested Loop Join). Output order: None.
   Cost: 6,500 I/Os. Output size: 3850 pages.

   E. Plan: B $\bowtie$ A (Block Nested Loop Join). Output order: None.
   Cost: 40,000 I/Os. Output size: 1040 pages.

   F. Plan: C $\bowtie$ B (Block Nested Loop Join). Output order: None.
   Cost: 20,000 I/Os. Output size: 550 pages.

   G. Plan: A $\bowtie$ B (Sort-Merge Join). Output order: `x`
   Cost: 50,000 I/Os. Output size: 1040 pages.

   H. Plan: A $\bowtie$ C (Sort-Merge Join). Output order: `x`
   Cost: 10,000 I/Os. Output size: 3850 pages.

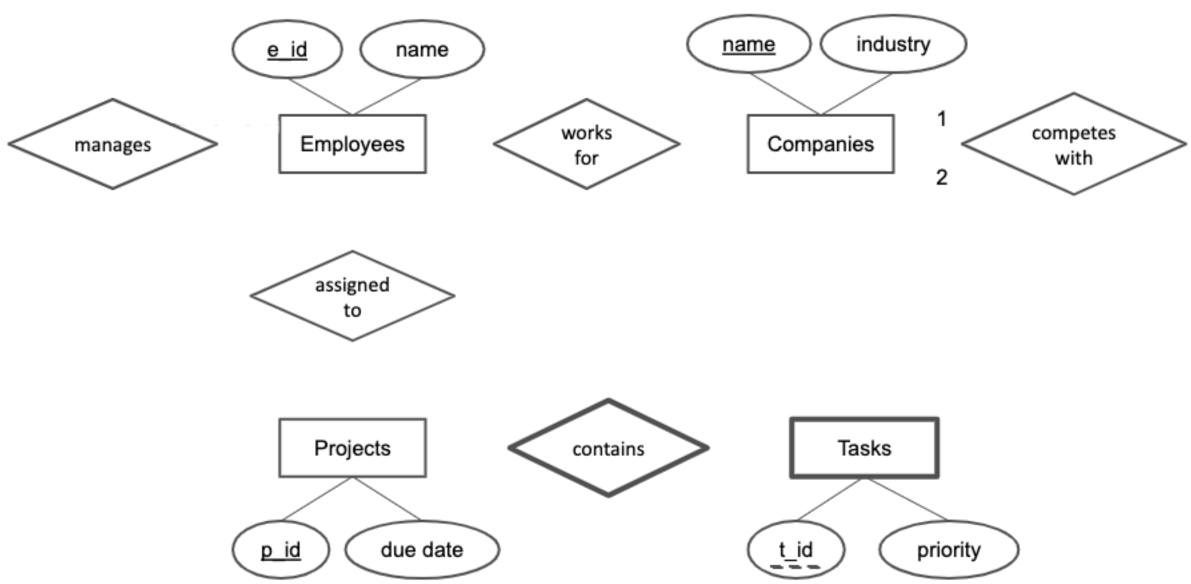
   I. Plan: B $\bowtie$ C (Sort-Merge Join). Output order: `y`
   Cost: 12,500 I/Os. Output size: 550 pages.

5 DB Design Doc (14 points)

Welcome to Corporate America! Every employee here works as hard as they can every day to complete as many tasks as possible. Total Compensation (TC) is everything in this world and the fastest way to a high TC is to become a 100000x employee.

Assumptions:

- The same entity cannot exist in a relationship with itself when considering recursive relationships.
- A company has to have at least one employee.
- Not all companies have competition but competition that occurs between companies emulates real world competition. That is, a company can have multiple competitors.
- Every employee has one manager and must work for one company.
- Not all employees work on projects but every project must have at least one employee assigned to it.
- All projects contain at least one task that must be finished by the due date.



1. (1 point) What type of edge should be drawn at 1?
   A. thin line
   B. bold line
   C. thin arrow
   D. bold arrow

2. (1 point) What type of edge should be drawn at 2?
   A. thin line
   B. bold line
   C. thin arrow
   D. bold arrow
3. (1 point) What type of edge should be drawn between **Companies** and **works for**?
   A. thin line
   B. bold line
   C. thin arrow
   D. bold arrow
4. (1 point) What type of edge should be drawn between **Employees** and **works for**?
   A. thin line
   B. bold line
   C. thin arrow
   D. bold arrow
5. (1 point) What type of edge should be drawn between **Employees** and **manages**?
   A. thin line
   B. bold line
   C. thin arrow
   D. bold arrow
6. (1 point) What type of edge should be drawn between **Employees** and **assigned to**?
   A. thin line
   B. bold line
   C. thin arrow
   D. bold arrow
7. (1 point) What type of edge should be drawn between **assigned to** and **Projects**?
   A. thin line
   B. bold line
   C. thin arrow
   D. bold arrow
8. (1 point) What type of edge should be drawn between **Projects** and **contains**?
   A. thin line
   B. bold line
   C. thin arrow
   D. bold arrow
9. (1 point) What type of edge should be drawn between **contains** and **Tasks**?
   A. thin line
   B. bold line
   C. thin arrow
   D. bold arrow

10. (1 point) What is the relationship between **Projects** and **Tasks**?
    A. one-to-one
    B. one-to-many
    C. many-to-one
    D. many-to-many

11. (2 points) Decompose R = SFLADHG into BCNF in the order of the following FDs:
    $FL \rightarrow DH$, $GL \rightarrow A$, $S \rightarrow F$, $H \rightarrow GA$
    Which of the following tables are included in the final decomposition?
    A. SLG
    B. HGA
    C. GLA
    D. FLDH
    E. HLGA

12. (1 point) Which of the following could possibly be primary keys of the relation? (There may be zero or more correct answers.)
    A. SH
    B. SL
    C. SG
    D. SD

13. (1 point) Is the decomposition dependency preserving?
    A. True
    B. False

14. (0.0001 points) The weekly posts on Piazza have included the phrase "Go Bears" and "Go" what other animal?

6 The Last Straw (14 points)

1. (5 points) Which of the following statements are true? **There may be zero, one, or more than one correct answer.**
    - A. Under strict two-phase locking, once a transaction releases a lock, it must release all other locks.
    - B. Holding a SIX lock prevents other transactions from reading or writing at a lower level.
    - C. We cannot prevent cascading aborts by using strict two-phase locking.
    - D. Two operations are always conflicting if both of them are write.
    - E. A waits-for graph is used to figure out if a transaction schedule is serializable.

2. (2 points) Mark the transactions involved in the cycle of conflicting operations. If the schedule is conflict serializable, mark none.

T1	R(A)				W(C)	W(D)						
T2		W(B)								R(D)		W(E)
T3			R(A)	W(A)							R(B)	
T4							R(E)	R(C)	R(A)			

    - A. T1
    - B. T2
    - C. T3
    - D. T4
    - E. None

3. (1 point) If we remove the write to E in T2, would this schedule be conflict serializable? If not, mark the transactions involved in the cycle of conflicting operations. Otherwise, mark none.
    - A. T1
    - B. T2
    - C. T3
    - D. T4
    - E. None

Jenny, Jiayue, Jasmine, and Jeremy want to go try a bunch of different boba drinks together, but they were only able to get one straw per drink. To make sure they don't bump heads while drinking boba with one another, they decide to use locks. Multiple people can pick up a drink at one time, but only one person may drink from a certain drink at a time. In other words, one must obtain a shared lock to pick up the drink. To take a sip, one must obtain an exclusive lock on that drink. Thus, the people are the transactions in this problem (people and transaction are used interchangeably).

Let's say Jenny (J1), Jiayue (J2), Jasmine (J3), and Jeremy (J4) buy a black milk tea (B), a matcha latte (M), jasmine milk tea (J), and a hojicha latte (H).

Person/Time	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
J1	S(H)							X(M)		
J2			S(M)							X(J)
J3		S(J)				X(H)			X(B)	
J4				S(M)	S(H)		X(H)			

4. (1 point) Which people/transactions are in the granted set for M at timestep 4 (including the lock request at this timestep)?
   - A. J1
   - B. J2
   - C. J3
   - D. J4
5. (1 point) Which people/transactions are in the wait queue for H at timestep 7 (including the lock request at this timestep)?
   - A. J1
   - B. J2
   - C. J3
   - D. J4
6. (2 points) At timestep 10 (including the lock request at this timestep), we run deadlock detection. Mark the people/transactions involved in the deadlock. If there is no deadlock, mark 'No Deadlock'.
   - A. J1
   - B. J2
   - C. J3
   - D. J4
   - E. No Deadlock
7. (2 points) If we instead decide to use the wait-die policy for deadlock avoidance, which of the following would be 'aborted' and have to do their boba tasting routine over again based on the order of lock acquisitions? If a transaction is aborted, assume it doesn't restart until after timestep 10. If a transaction is waiting, assume that it doesn't continue to try to acquire other locks.
   - A. J1
   - B. J2
   - C. J3
   - D. J4

7 Scratching Post (0 points)