You should receive a double-sided answer sheet and a 13-page exam. Mark your name and login on the front of the answer sheet, and in the blanks above. For each question, place only your final answer on the answer sheet—do not show work or formulas there. You may use the backs of the questions for scratch paper, but do not tear off any pages. We will ask you to turn in your question sheets as well as your answers.

I. Query Processing [16 points]
You and your friends have a brilliant idea for a startup: you decide to build a social-media site, FishBook, which allows users to share their pet fish with the world.

Low on funds, you start building your application on an old ten-year-old computer your friend’s parents threw out last year. After much difficulty, you succeed in installing a database management system (DBMS) on this computer, which uses a System R (Selinger) optimizer:

- Page size: 32 KB
- Allocated memory for DBMS: 500 pages (about 16MB)

1. (2 points) You decide to test out your shiny new system by running some performance tests on it. What is the maximum amount of data your system can aggregate using external hashing in three passes, in pages, assuming the hash functions partitions data uniformly? You can write the final answer as an arithmetic expression.

You start building your web application. Each user can post descriptions of their pet fish to this application. Here’s part of your database schema:

Users(uid int, username text, ...)
Fish(fid int, ownerid int, age_in_days int, description text, ...)

where uid and fid are primary keys, assigned as sequential integers in ascending order starting from 0. Fish.ownerid is a foreign key to Users.

Overnight, your application goes viral. Thousands of users have signed up and have posted their pets on your site:

- Users: 10000 tuples, 1000 pages
- Fish: 50000 tuples, 10000 pages

For the following questions, assume the values are distributed uniformly and independently.
I. Query Processing, cont.

Your startup co-founder wants some material for marketing, so you produce some example data from your database:

```
SELECT U.username, F.description
FROM Users U, Fish F
WHERE F.fid > 45000
AND U.uid = F.ownerid;
```

2.

a. (3 points) What is the estimated I/O cost of the query plan above, assuming it uses a Sort-Merge join? Use the optimization (described in class) where we merge-join during the final sort pass. Ignore the effects of the buffer pool and disk locality. Do not count the cost of write your final output. Assume that we use quicksort for in-memory sorting.

b. (2 points) How many tuples are estimated to be returned by this query?

3. (2 points) You decide to speed up your application by adding indexes to your database. Consider two possible indexes:
   - #1: A clustered alternative 1 index on Users.uid, or
   - #2: An unclustered alternative 2 index on Fish.ownerid

   For each of the following queries, which of the above indexes will be more useful in speeding up these queries on expectation? Answer #1 or #2 for each of the following:
   a. Given a user ID, compute the average age of his/her fish.
   b. Given a fish ID, look up the username of the user who owns this fish.
   c. Count how many fish some user has.
   d. Find the usernames of the users who own the 100 oldest fish.
I. Query Processing, cont.

Users might find it easier to become acquainted with each others’ pet fish if they can also see pictures of them. You add to your database storage for these pictures:

\[ \text{Pictures}(\text{pid}, \text{fishid}, \text{image blob}, \ldots) \]

where `pid` is a primary key, `Pictures.fishid` is a foreign key to Fish and a `blob` is the image data associated with a picture.

4. (3 points) A part of your site needs to display the number of pictures there are of each pet. Which of the following will decrease the I/O cost of this query? Circle all that apply.
   i. Sorting Pictures by `fishid` on disk
   ii. Running a compression algorithm on fish image data before inserting the tuples into Pictures
   iii. An unclustered index on `Pictures.pid`

You decide to add a species-recognition feature to your application. Using state-of-the-art machine learning algorithms, you write a classifier for these fish images. You add this to your DBMS as a custom user-defined function (UDF), `classify(blob)`.

5. (4 points) Consider the following query:

```sql
SELECT U.username, count(F.fid)
FROM Users U, Fish F, Pictures P
WHERE U.uid = F.ownerid
AND F.fid = P.fishid
AND classify(P.image) = 'clownfish'
GROUP BY U.uid
ORDER BY count(F.fid) DESC;
```

Assume only heap files are used for storage. Write down the letters of all joins that produce interesting orders for this query.

   a. Sort-merge(Users, Fish)
   b. Sort-merge(Fish, Users)
   c. Chunk-Nested-Loops(Users, Fish)
   d. Chunk-Nested-Loops(Fish, Users)
   e. Sort-merge(Pictures, Fish)
   f. Sort-merge(Fish, Pictures)
   g. Chunk-Nested-Loops(Pictures, Fish)
   h. Chunk-Nested-Loops(Fish, Pictures)
II. Indexing / Storage [17 points]

1. (5 points) True/False
   a. The pin count for a buffer pool frame reflects the number of times the page it holds has been referenced since it was last loaded into memory.
   b. If pages are always fully packed, then inserting or deleting a tuple in a sorted file of N pages will require \( \frac{1}{2} N \) I/Os on average.
   c. The slotted page format allows tuples to be moved to a different location on the same page without changing the record ID.
   d. For a lookup on a primary key, a clustered B+ tree provides better spatial locality than an unclustered B+ tree.
   e. In a tree-structured index, the fanout is large in order to keep the height of the tree fairly shallow.

You are working as a database administrator (DBA) at ClikClak, an anonymous text message forum app. Their data is stored in a single table with the following schema:

\[
\text{Post (postid int, userid int, pdate int, content text)}
\]

where postid is a primary key for Post. Post takes up \( N = 10,000 \) pages on disk.

The Post table is partitioned across three servers based on the data’s access pattern.

<table>
<thead>
<tr>
<th>Server ID</th>
<th>Read/Write Ratio</th>
<th>% of N</th>
<th>Access Pattern Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>80/20</td>
<td>20%</td>
<td>sequential scans in chronological order</td>
</tr>
<tr>
<td>S2</td>
<td>40/60</td>
<td>60%</td>
<td>posts made by a user, shown on their profile</td>
</tr>
<tr>
<td>S3</td>
<td>99/1</td>
<td>20%</td>
<td>queries for random specific posts no temporal locality</td>
</tr>
</tbody>
</table>

2. (12 points) For each server, fill in the blank on your answer sheet to choose parameters that best optimize performance for that server’s access pattern. Don’t ignore the maintenance cost of insertions/updates when considering performance.

2a. Select a buffer replacement policy:
   A. Least Recently Used
   B. Most Recently Used
   C. Random
   D. Any of the above

2b. Select a file layout and index:
   A. HeapFile, Unclustered B+ tree on postid
   B. HeapFile, Clustered B+ tree on userid
   C. SortedFile on date, Unclustered B+ tree on date
   D. SortedFile on date, No index
   E. SortedFile on userid, Unclustered B+ tree on content
   F. SortedFile on userid, No index
III. Database Design [18 pts]

The CS186 staff is designing a database to store information about students who are currently taking the course.
For each student, the staff wants to keep track of the one optional discussion section to which the student is assigned—students may choose to be assigned to no discussion section if they like. The discussion section is composed of a unique course control number (ccn) for that section, and the name of a TA. Students will also do assignments—all of which have differing weights. For each assignment, the staff wants to store the number of points the student earned and the number of slip days they have used on that assignment. If a student does not turn in an assignment, they will be recorded in the database as having done the assignment but will receive zero points. Assume there are many assignments, and all assignments are released to the class to be done.

1. **(3.5 points)** To implement the database, a TA decides to create an ER diagram. Complete the ER diagram on the answer sheet for the Student, Assignment, and Discussion entities by underlining the primary keys and connecting the given entity and relationship sets using the appropriate lines and/or arrows. If bolding a line/arrow, be sure to clearly make it bold.

2. **(1.5 points)** The given ER diagram is missing the proper notation for its weak entity. Complete the notation for the weak entity in the diagram on the answer sheet.

3. **(2 points)** The staff decides the assignments will be difficult, so students should work in teams of two. Each team should have one student who is the “team lead”. Make the additions to the ER diagram on the answer sheet to do so. Assume that there are an even number of students. The fewer entities and relationships you use, the better. You should not need to add more than one entity and/or relationship.

4. **(2 points)** For each of the following, select whether the addition would best be included in the ER diagram as an attribute (A), entity (E), or needs more information (NMI).
   a. Student’s primary contact email
   b. Student’s additional emails
   c. Student’s address with city, street, etc.
   d. Student’s classes
III. Database Design, Cont.

The staff decides that they need to store more information about each assignment. The TA who redesigns things ignores the ER diagram, and simply designs a relational schema with functional dependencies. Thus, treat the following parts separately from the ER diagram above.

Along with the id of the student who did the assignment, the assignment name, weight, and number of slip days used, they also want to store data for all of the individual questions. For each question, they want to track the total number of points it is worth, how many points the student received on that question, and the student's response.

The TA proposes that they should use the relation INWSQTPR, where each attribute is abbreviated by the capitalized version of the underlined, bolded character of the corresponding attribute in the description above.

They imagine the table would look like the one to the right. However, the TA realizes that they can improve their design by finding some functional dependencies (FDs) from the data model, then decomposing the relation into BCNF.

5. (2 points) What benefits would the decomposed schema give over using the original relation? Mark all of the following that will always apply.
   a. Reduce space required to store the data
   b. Avoid cartesian products
   c. Speed up queries
   d. Avoid update anomalies

6. a. (2 points) Currently, the FDs that the TA can think of are \( \{N \rightarrow W, IN \rightarrow S\} \).
   Decompose INWSQTPR into BCNF given these functional dependencies.
   b. (1 points) Is the decomposition dependency-preserving based on the FDs above? If not, what relation(s) could you add to make it dependency preserving? Hint: You do not need to compute \( F^+ \) to answer this question!

7. (4 points) The TA needs some help determining the rest of the functional dependencies for this assignment data model. Currently, the FDs he has are \( \{N \rightarrow W, IN \rightarrow S\} \), but he is still missing some useful functional dependencies that will allow him to break the relation into more fundamental components. Write those FDs below. Do not include any trivial dependencies or augmentations of the FDs already listed. (Hint: No more than 4 FDs).
IV. Concurrency [19 points]

1. (5 points) True/False
   a. SIX locks are compatible with IX locks.
   b. Strict 2PL guarantees conflict serializability but does not prevent cascading aborts.
   c. A schedule is conflict serializable if and only if its dependency graph is acyclic.
   d. All view serializable schedules avoid cascading aborts.
   e. Wound-wait and wait-die both favor older transactions over newer ones; however, they favor old transactions in different ways.

2. (2 points) Consider the following sequence of transactions:

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td>R(A)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W(A)</td>
<td>R(B)</td>
<td></td>
</tr>
<tr>
<td>R(B)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W(B)</td>
<td></td>
<td>R(C)</td>
</tr>
<tr>
<td>W(A)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1 COMMIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1 END</td>
<td></td>
<td>W(C)</td>
</tr>
<tr>
<td>T2 COMMIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2 END</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3 COMMIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3 END</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Does this schedule conform to 2PL?

Does this schedule conform to strict 2PL?
IV. Concurrency

3. **(4 points)** Write the serial schedule that is conflict equivalent to this schedule. If this schedule is not conflict serializable, write “None”.

<table>
<thead>
<tr>
<th>T1</th>
<th>R(B)</th>
<th></th>
<th></th>
<th>R(F)</th>
<th>W(F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2</td>
<td></td>
<td>R(E)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td></td>
<td>R(E)</td>
<td>W(E)</td>
<td>R(F)</td>
<td>W(F)</td>
</tr>
<tr>
<td>T4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>W(B)</td>
</tr>
</tbody>
</table>

4. **(8 points)** Under a TO-MVCC scheme, the following actions happen, in the given order. Assume that all objects A-E were originally written at TS0. Write down the letters of the actions that would cause aborts. If no actions would cause abort, write down the letter corresponding to “No writes fail”.

- a. TS10 reads A
- b. TS15 reads B
- c. TS10 reads B
- d. TS20 reads C
- e. TS10 writes A
- f. TS20 writes A
- g. TS15 reads A
- h. TS20 reads B
- i. TS10 writes B
- j. TS15 reads D
- k. TS20 writes D
- l. TS5 writes D
- m. TS1 writes E
- n. TS15 reads E
- o. TS20 writes E
- p. No writes fail
V. Recovery [22 points]

Consider the log below. Some of the information for UPDATE and CLR log records is omitted for brevity. Adjacent log records are spaced exactly 10 LSN’s apart. The system crashes after the last log record is written.

<table>
<thead>
<tr>
<th>LSN</th>
<th>Record</th>
<th>prevLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>UPDATE: T1 writes P1</td>
<td>null</td>
</tr>
<tr>
<td>10</td>
<td>BEGIN_CHECKPOINT</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>UPDATE: T1 writes P2</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>END_CHECKPOINT</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>UPDATE: T2 writes P3</td>
<td>null</td>
</tr>
<tr>
<td>50</td>
<td>UPDATE: T2 writes P4</td>
<td>40</td>
</tr>
<tr>
<td>60</td>
<td>ABORT: T1</td>
<td>20</td>
</tr>
<tr>
<td>70</td>
<td>CLR: T1 LSN 20</td>
<td>60</td>
</tr>
<tr>
<td>80</td>
<td>??</td>
<td>??</td>
</tr>
<tr>
<td>90</td>
<td>END: T1</td>
<td>80</td>
</tr>
<tr>
<td>100</td>
<td>UPDATE: T3 writes P3</td>
<td>null</td>
</tr>
<tr>
<td>110</td>
<td>ABORT: T2</td>
<td>50</td>
</tr>
<tr>
<td>120</td>
<td>CLR: T2 LSN 50</td>
<td>110</td>
</tr>
</tbody>
</table>

At checkpoint time, the dirty page table is empty, and the transaction table is as follows:

<table>
<thead>
<tr>
<th>XID</th>
<th>Status</th>
<th>lastLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Running</td>
<td>0</td>
</tr>
</tbody>
</table>

1. **(2 points)** What is the missing log record with LSN 80? Please follow the same format as the records above.

The system comes back up after the crash and starts performing recovery.

2. **(4 points)** Fill in the transaction table as it appears at the end of analysis. You may not need to use all rows in the table.

3. **(4 points)** Which pages are in the dirty page table at the end of analysis? You may not need to use all rows in the table.

4. **(4 points)** What actions will occur during the REDO phase? List the LSN’s on the answer sheet in the order that they are redone, separated by commas. No new log records will be written during REDO.

5. **(4 points)** What are the records written during the UNDO phase? Start at LSN 200. The difference between the LSN’s of two adjacent log records should be 10. You may not need to use all rows in the table.
V. Recovery, cont.

6. **(4 points)** Imagine a version of CS186 where you have to implement ARIES as a class project. Your partner is responsible for the code managing the dirty page table, and his code updates the recLSN in the dirty page table to reflect each update to a page, regardless of when the page was brought into the buffer pool. What bugs might you see after recovery? Circle all that apply.
   a. Some writes of committed transactions would be lost.
   b. Some writes of aborted transactions would be visible in the database.
   c. Some transactions that should have been aborted will be committed.
   d. The system tries to commit or abort a transaction that is not in the transaction table.

VI. SQL [15 points]
A group of college dropouts decide that they can take on Uber and start their own ridesharing service called Super.

Super’s core business depends on PostgreSQL and primarily relies on 3 tables shown below. Assume that records in the Trip table are only created and committed once the actual trip has been completed. Assume the distance stored is in miles. The passenger_rating of a trip is the rating that the passenger received from the driver, and the driver_rating of a trip is the rating that the driver received from the passenger. Assume every driver in the database has completed at least one trip. Ratings are on a scale from 1.0 to 5.0 and can be NULL.

As the Super business grows, they realize the need for analysts with SQL knowledge to help them make better business decisions. They consult you in order to answer their questions regarding their Super business.
VI. SQL, cont.

Passenger(pid int, first_name text NOT NULL, last_name text NOT NULL)

Driver(did int, first_name text NOT NULL, last_name text NOT NULL)

Trip(tid int,
id int references Passenger(pid) NOT NULL,
pid int references Driver(did) NOT NULL,
start_time timestamp NOT NULL,
end_time timestamp NOT NULL,
distance decimal NOT NULL,
passenger_rating decimal,
driver_rating decimal)

1. (3 points) You hypothesize that some months of the year are more popular than others, perhaps due to weather or special events like holidays. To assess this, you want to know how many trips were taken in each month, independent of year. You and your team define a Trip’s start_time to indicate which month the trip belongs to.

   CREATE VIEW num_trips_by_month AS
     SELECT EXTRACT(MONTH FROM ___________________) AS month,
                __________________ AS num_trips
           FROM __________________
      GROUP BY month;

Note: EXTRACT(MONTH FROM _____) is a PostgreSQL function that extracts the numeric month (e.g. January = 1) out of a timestamp or interval.

2. (3 points) You want to prepare your staff next year to improve heavily on the poorest performing month(s), independent of year. Which month(s) had the minimum number of trips? (Use the view created in Q1)

   SELECT month
   FROM num_trips_by_month NTM JOIN
   (SELECT __________________ AS min_column
    FROM ________________________) MIN_TABLE
   ON _________________________;
VI. SQL, cont.

3. (3 points) Which drivers have a perfect 5.0 average rating from all their trips that received driver ratings? (Return just the unique did)

```sql
SELECT T1.did
FROM Trip AS T1
WHERE ____________________ (  
    SELECT __________________  
    FROM Trip AS T2  
    WHERE ___________________________ ;  
) ;
```

b. (1.5 points) When executing this query, you find that this query runs very slowly. What about the structure of this query may cause it to execute so slowly? Circle all statements that could apply.
   A. There are no indices built on the Trip table  
   B. The query optimizer optimizes each SELECT block in the query independently  
   C. The query will take $O(n^2)$ of work, where $n$ is the size of Trip table and the table is very large

c. (3 points) You attempt re-writing this same query with the hope of speeding it up.

```sql
SELECT T.did  
FROM Trip as T  
GROUP BY ___________________________  
HAVING ___________________________ ;
```

4. (1.5 points) You hypothesize that drivers and passengers with the same first name get along better (have better ratings) than drivers and passengers that don’t share any commonalities.

You want to eyeball this data yourself. So you have your associate write a query that returns a single row with 2 columns. The first column is the average `driver_rating` of all trips that have driver ratings where the driver and passenger also had the same `first_name`. The second column is the average `driver_rating` of all trips that have driver ratings where the driver and passenger had distinct `first_name`'s.

Read the following three queries, and circle the ones that yield the desired result on your answer sheet. Notice that the passenger and driver ratings can be NULL, and that SQL aggregate functions (COUNT, SUM, AVG, etc) are defined to ignore NULL values in their calculations.
VI. SQL, cont.

**Query A:**

```sql
SELECT SAME_NAME.rating AS same_name,
      DIFF_NAME.rating AS diff_name
FROM (SELECT AVG(driver_rating) AS rating
      FROM Passenger P, Driver D, Trip T
      WHERE P.pid = T.pid AND D.did = T.did
      AND P.first_name = D.first_name) AS SAME_NAME,
      (SELECT AVG(driver_rating) AS rating
       FROM Passenger P, Driver D, Trip T
       WHERE P.pid = T.pid AND D.did = T.did
       AND P.first_name <> D.first_name) AS DIFF_NAME;
```

**Query B:**

```sql
SELECT SAME_NAME.rating AS same_name,
      DIFF_NAME.rating AS diff_name
FROM (SELECT SUM(driver_rating) / COUNT(*) AS rating
      FROM Passenger P, Driver D, Trip T
      WHERE P.pid = T.pid AND D.did = T.did
      AND P.first_name = D.first_name) AS SAME_NAME,
      (SELECT SUM(driver_rating) / COUNT(*) AS rating
       FROM Passenger P, Driver D, Trip T
       WHERE P.pid = T.pid AND D.did = T.did
       AND P.first_name <> D.first_name) AS DIFF_NAME;
```

**Query C:**

```sql
SELECT AVG(TS.driver_rating) AS same_name,
      AVG(TD.driver_rating) AS diff_name
FROM Passenger PS, Driver DS, Trip TS,
     Passenger PD, Driver DD, Trip TD
WHERE PS.pid = TS.pid AND DS.did = TS.did AND
      PD.pid = TD.pid AND DD.did = TD.did AND
      PS.first_name = DS.first_name AND
      PD.first_name <> DD.first_name;
```
I. Query Processing
1. ___________________________ pages
2a. ___________________________ I/Os
2b. ___________________________ tuples
3a. #1 / #2
3b. #1 / #2
3c. #1 / #2
3d. #1 / #2
4. i. / ii. / iii.
5. ___________________________

II. Indexing / Storage
1a. True / False
1b. True / False
1c. True / False
2a. S1 _____ S2 _____ S3 _____
2b. S1 _____ S2 _____ S3 _____

III. Database Design
1-3. Add to the diagram below.
4. a. A / E / NMI
   b. A / E / NMI
   c. A / E / NMI
   d. A / E / NMI
5. a / b / c / d
6a. ___________________________
6b. Yes / No ____________________
7. ___________________________

IV. Concurrency
1a. True / False
1b. True / False
1c. True / False
1d. True / False
2a. Yes / No
2b. Yes / No
3. ___________________________
4. ___________________________
V. Recovery

1. LSN | Record | prevLSN
      |        |        
      |        |        

2. XID | Status | lastLSN
      |        |        
      |        |        

3. PID | recLSN
      |        
      |        

4. ______________________________________________________________________

5. LSN | Record | prevLSN
      |        |        
      |        |        
| 200   |        |        |
| 210   |        |        |
| 220   |        |        |
| 230   |        |        |
| 240   |        |        |
| 250   |        |        |

6. A / B / C / D

VI. SQL

1. CREATE VIEW num_trips_by_month AS
   SELECT EXTRACT(MONTH FROM
                   _______________) AS month,
                   _______________ AS num_trips
   FROM ________________
   GROUP BY month;

2. SELECT month
   FROM num_trips_by_month NTM JOIN
       (SELECT ____________ AS min_column
        FROM ________________ ) MIN_TABLE
   ON ________________;

3a. SELECT T1.did
    FROM Trip AS T1
    WHERE ________________ (SELECT ________________
                              FROM Trip AS T2
                              WHERE ________________
                              ________________);

3b. Circle statements that apply:
    A / B / C

3c. SELECT T.did
    FROM Trip AS T
    GROUP BY ________________
    HAVING ________________;

4. Circle the queries that produce the desired outcome: A B C