CS 6083 – Final Exam

Instructions. This is a 150-minute exam, with a total of 54 points available.

- Do not open this test booklet until you are told to do so.

- Write your name, student ID, and seat number on this cover sheet.

- Write your name *at the bottom of each page.*

- Please write your answers neatly *in the box that is provided directly below each question.*

- If you run out of space in a box, use one of the overflow boxes on the last three pages. Put a statement inside the original box indicating which overflow box you are using for this question.

- **Do not** write on the margins or the backs of pages.

- Use a pen, not a pencil.

- The exam is closed book and closed notes, with one page of notes per student allowed. Please place all phones, tablets, and watches inside your bag, and do not access them during the exam. Switch off your phone.

- Good luck!

Name: _____

Student ID: _____

Problem 1. (16 Points)

Suppose we have a relational schema that models student organizations (clubs) at a university, their members, events held by these clubs, and who attends these events. Clubs are uniquely identified by a name (e.g., Debate Club, or French Student Association). Students are identified by their student ID, and also have a name, a gender, a major, and an email address. Students become members of a club on a semester basis, so that a student may be a member in Spring 2018, but maybe not in Fall 2018. Each club has a membership fee. Clubs organize events. Each event has a unique eid and a name (such as Chess Club Christmas Party or Fishing Trip). Each event may require a fee, but the fee is usually lower for members than for non-members. Each event is organized by one club, so there are no joint events between two clubs. The database consists of the following six relations, with primary keys underlined:

Student (<u>sid</u>, name, gender, major, email)
Club (<u>cname</u>, description)
ClubFee (<u>cname, semester</u>, fee)
Membership (<u>cname, sid, semester</u>)
Event (<u>eid</u>, ename, semester, cname, edate, memberfee, nonmemberfee)
Registration (<u>sid, eid</u>, asmember, attended)

Note that asmember and attended are Boolean attributes.

(a) (2 points) Identify suitable foreign keys for this schema.

(b) (4 points) Sketch an ER diagram that is consistent with this schema and all the stated conditions. Identify any weak entities and the mapping cardinalities of the relationships.

(c) (6 points) Write SQL statements for the following queries:

(i) List the names of any clubs that did not have any female members during Spring 2018.

(ii) Output the names of all students who attended at least one event organized by the Chess Club during Spring 2018 as non-members.

(iii) Output the student IDs of all students who were not members of the Chess Club during Spring 2018, but who attended enough events such that they would have saved money if they had joined the club (because the member rates for the events were so much cheaper than the non-member rates that it makes up for the semester membership fee).

(d) (4 points) Write statements in Relational Algebra for the first two queries.

(i) List the names of any clubs that did not have any female members during Spring 2018.

(ii) Output the names of all students who attended at least one event organized by the Chess Club during Spring 2018 as non-members.

Problem 2. (18 points total)

Consider the following schema keeping track of twitter users, their tweets, and whom they follow:

> User(<u>uid</u>, screenname, uname, ucity, ucountry);
> Tweet(<u>tid</u>, ttitle, ttext, uid, tdate, ttime);
> Follows(<u>uid, fuid</u>, ftimedate);

In this simplified schema, we do not model retweets, and each tweet is simply a message with a title and text. In the Follows table, the fuid specifies a user that follows the user specified by the uid, and there is a timestamp specifying when they started to follow. For simplicity, we assume that once a user follows another user, they will continue to do so forever.

Consider the following two queries:

> SELECT U.screenname
> FROM User U, Tweet T
> WHERE U.uid = T.uid and U.ucity = Hoboken and T.tdate = July 6, 2017

> SELECT U.uid, U.screenname
> FROM User U, Follows F, Tweet T, User U2
> WHERE U.uid = T.uid and U.uid = F.fuid and U2.uid = F.uid
>   and T.tdate = November 11, 2015 and U2.screenname = Joe Schmoe

(a) (2 points) In one sentence each, describe what question the query answers.

(b) (2 points) Transform each SQL query into an equivalent expression in relational algebra.

In the following, assume that there are 500 million users, of which 10000 live in Hoboken. There are 50 billion tweets over a period of 1000 days, or on average about 50 million per day. There at 5 billion records in the Follows table. Finally, Joe Schmoe has only ten followers.

For simplicity, assume that all records are of size 100 bytes, and any IDs and timestamps are 12 bytes. Any index entries are of size 24 bytes, and any index structures have a node size of 4 KB and a height of 4 (the root, the leaf level, and two levels in between). There are 2 GB of main memory available for query processing, and you are given a disk with 100 MB/s transfer rate and $10ms$ latency (seek time plus average rotational latency).

(c) (6 points) Consider the first query, and estimate its running time under the latency-transfer-rate model of disk performance, for the following three cases:

(i) (2 points) There are no index structures available.

(ii) (2 points) There is a sparse clustered B+-tree index on tdate in Tweet that is used by the query.

(iii) (2 points) There is an unclustered B+-tree index on uid in Tweet that is used by the query to perform an index-based join.

(d) (8 points) Now consider the second query.

(i) (3 points) Draw an optimized query evaluation plan for this query, assuming no index structures are available. For each join, show the join algorithm that is being used by the plan.

(ii) (3 points) Estimate the running time of the query evaluation plan under the latency-transfer-rate model of disk performance.

(iii) (2 points) Suppose you can create one index structure to make this query faster. Which index structure would you choose? In one sentence, why this one?

Problem 3. (12 points total) In each of the following questions, **circle** the correct answer **for each statement**.

(a) (3 points) Which of the following statements about join algorithms are true, and which are false?

- It is always fastest to use an index-based join when an index on the join attribute is available.
  TRUE   FALSE

- Index-based joins and blocked nested-loop joins are the most commonly used join algorithms in database applications.
  TRUE   FALSE

- Hash-based joins are preferable to other methods when both inputs are very small.
  TRUE   FALSE

- Clustered indexes can never be faster than unclustered indexes for index-based joins.
  TRUE   FALSE

- Sort-based joins are preferable for outer joins.
  TRUE   FALSE

(b) (3 points) Which of the following statements about index structures are true, and which are false?

- Using an index for an inequality selection is always better than scanning the table.
  TRUE   FALSE

- For equality selections, clustered and unclustered B^+-trees always have the same performance.
  TRUE   FALSE

- Clustered B^+-trees are usually better than unclustered B^+-trees for inequality selections.
  TRUE   FALSE

- Hash indexes are particularly useful for inequality (range) selections.
  TRUE   FALSE

- Composite index structures combine ideas from hash indexes and B^+-trees.
  TRUE   FALSE

(c) (3 points) Which of the following statements about database design theory are true, and which are false?

- The right side of any functional dependency must contain a candidate key.
  TRUE   FALSE

- Given a set of functional dependencies F, there always exists a canonical cover of F.
  TRUE   FALSE

- Some schemas cannot be transformed into BCNF.
  TRUE   FALSE

- Every schema can be transformed into 3NF, and the resulting schema is dependency-preserving.
  TRUE   FALSE

- Any schema that is in BCNF is also in 3NF.
  TRUE   FALSE

(d) (3 points) Which of the following statements about concurrency control and serializability are true, and which are false?
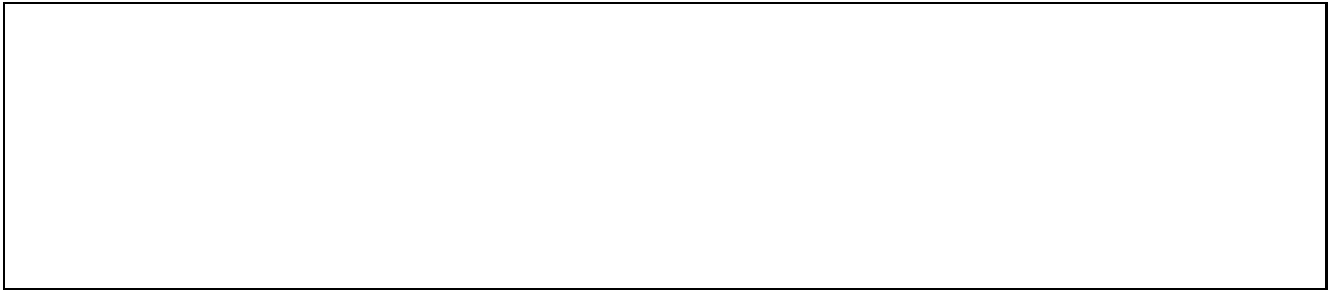
- Every conflict-serializable schedule avoids cascading aborts.
  TRUE   FALSE

- 2PL guarantees that there are no deadlocks.
  TRUE   FALSE

- Every schedule generated by 2PL is conflict-serializable.
  TRUE   FALSE

- For every conflict-serializable schedule, there exists an equivalent serial schedule.
  TRUE   FALSE

- There exist schedules that are serializable but not conflict-serializable.
  TRUE   FALSE

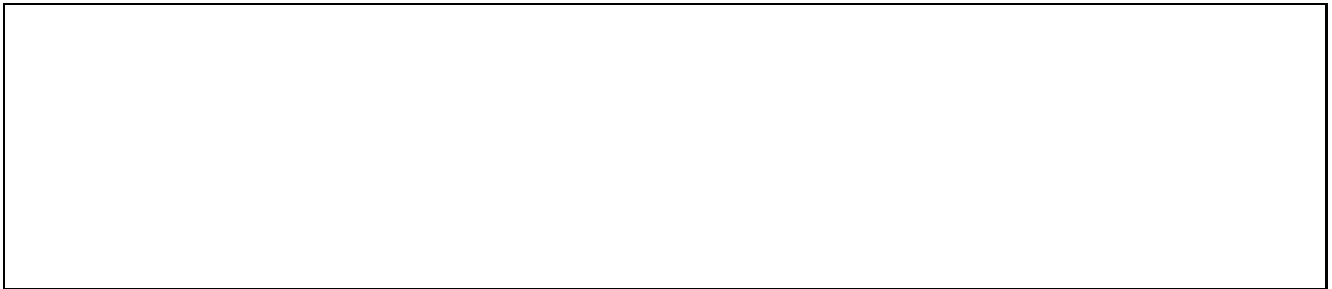**Problem 4. (8 points total)** Consider the schedule shown on the NEXT page.

(a) (2 points) Show the complete conflict graph for this schedule.

(b) (2 points) Is the schedule conflict serializable? Yes or no? Also explain why or why not in one sentence.

(c) (2 points) Could this schedule be generated by a 2PL protocol? Yes or no? Also explain why or why not in one sentence.

(d) (2 points) Is the schedule recoverable? Is it cascadeless? Yes or no? Also explain why or why not in one sentence.

ADDITIONAL ANSWER BOXES IF YOU RUN OUT OF SPACE

A

B

C
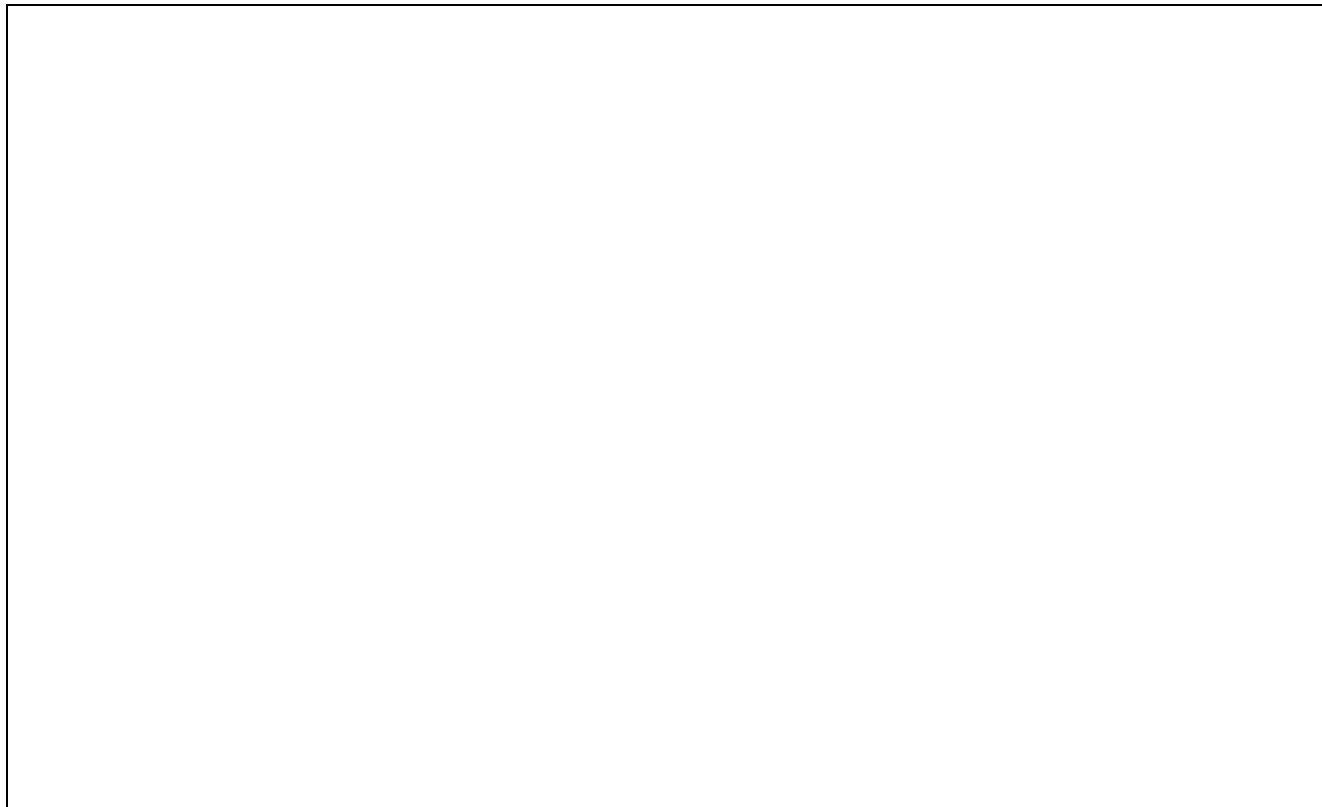
D

ADDITIONAL ANSWER BOXES IF YOU RUN OUT OF SPACE

E

F

ADDITIONAL ANSWER BOXES IF YOU RUN OUT OF SPACE

G

H