CSE 414 Midterm Exam

May 6, 2013

Name ________________________________

Question 1	/ 42
Question 2	/ 40
Question 3	/ 18
Total	/ 100

The exam is open textbook but otherwise you may not use any materials other than one sheet of paper with hand-written notes on one or both sides. No computers, electronics devices, phones of the smart or not-so-smart variety, telegraphs, telepathy, smoke signals, or other contraptions permitted.

The exam lasts 50 min. Please budget your time so you get to all questions.

Please wait to turn the page until everyone has their exam and you are told to begin.

Relax. You are here to learn.

Question 1. (42 points) SQL queries and indexes. We have a database that stores information about a simplified twitter application. When a user creates an account, he or she is assigned a user id (uid) and is added to the User table. Users may create tweets. Each of these is assigned a unique tweet id (tid) and is stored in the Tweet table. A Tweet may be posted as many times as desired, and each posting is recorded in the Post table (and a tweet can be posted by anyone, not just the user who created it). Finally, a user can follow other users and that information is recorded in the Follow table. For example, if user A is following user B (i.e., A is a follower of B), then a row with id of user A as uid and the id of user B as followsuid is added to the Follow table.

User(<u>uid</u>, name)
Tweet(<u>tid</u>, content)
Post(<u>uid</u>, <u>tid</u>, <u>time</u>)
Follow(<u>uid</u>, <u>followsuid</u>)

- The underlined attribute(s) represent the primary key for each relation.
- Post.uid is a foreign key that references User.uid.
- Post.tid is a foreign key that references Tweet.tid.
- Follow.uid is a foreign key that references User.uid.
- Follow.followsuid is a foreign key that references User.uid.

The queries you write must be proper SQL that would be accepted by SQL Server and any other SQL implementation. You should not use incorrect SQL, even if sqlite might produce an apparently correct answer from the buggy SQL.

(a) (16 points) Write a SQL query that retrieves the names of users who have posted more than 5 tweets that are at least 100 characters long. (Hint: If s is a string attribute, length(s) will return its length.)

(continued next page)

Question 1. (cont)

User(uid, name)
Tweet(tid, content)
Post(uid, tid, time)
Follow(uid, followsuid)

(b) (16 points) An active user is one who is following more than 50 users. Write a SQL query that retrieves the names of users who are being followed by one or more active users.

(continued next page)

Question 1. (cont)

User(uid, name)
Tweet(tid, content)
Post(uid, tid, time)
Follow(uid, followsuid)

(c) (10 points) Suggest **two indexes** that would be most likely to speed up execution of the queries in your answers to the previous parts of this question. Give a brief justification for your answer.

Question 2. (40 points) Relational algebra and query plans. Consider the following schema for the relations R, S, and T to answer part (a), (b) and (c).

R(a, b, c)
S(d, e)
T(f, g, h)

(a) (16 points) Draw a tree giving a relational algebra query plan corresponding to the SQL query below. (Recall that the main relational algebra operators are $\bowtie$, join; σ, select; Π, project; γ, grouping and aggregation; δ, duplicate elimination; and –, difference or subtract.)

```
SELECT R.a, count(R.b)
FROM R, S, T
WHERE R.c = S.d AND S.e = T.f AND T.h > 3
GROUP BY R.a
HAVING max(R.b) > 2;
```

Question 2 (cont.) R(a, b, c)
S(d, e)
T(f, g, h)

(b) (16 points) Draw a tree giving a relational algebra query plan corresponding to the SQL query below.

```
SELECT R.c
FROM R, S
WHERE R.c = S.d AND NOT EXISTS (SELECT *
                                FROM T
                                WHERE S.d = T.f);
```

(c) (8 points) Suppose relations R, S, and T contain the following data:

a	b	c
A	3	1
B	4	2
C	1	3

d	e
1	1
1	4
2	3
3	1
4	2

f	g	h
4	5	4
3	1	5
1	2	1

What output is produced by the query in part (b) when it is executed using this data?

Question 3. (18 points) Query implementation. For each of the following, either circle the right answer (true/false, etc.) or give a short answer to the question, as appropriate. You do not need to provide an explanation for your answer unless the question asks for one.

(a) (2 points) A relation can have clustered indexes on two or more attributes if this will speed up queries.

True / False

(b) (2 points) Which type of index is more suitable for speeding up queries that involve both exact matches (a=v) and range queries (a>v1 and a<v2)? (circle)

hash index / B+-tree index

(c) (2 points) Two measures of a relation R are the number of blocks B(R) and number of tuples T(R). For typical tables holding textual data like credit card transactions or the Twitter database from problem 1, what is the expected relationship between the sizes of B(R) and T(R)? (circle the correct choice)

- B(R) is normally significantly smaller than T(R)
- B(R) and T(R) are normally about the same
- B(R) is normally significantly larger than T(R)

(d) (2 points) Once a database engine translates a SQL query to a relational algebra logical query plan, that choice determines how the physical query will be executed.

True / False

(continued next page)

Question 3. (cont) (e) (5 points) When scanning a table to perform a join, the database can either do a full scan of the table or use an index to directly access tuples as needed. Will indexed access always be cheaper than a full scan? Answer yes or no and give a brief explanation supporting your answer.

(f) (5 points) Suppose we execute a join operation $\mathbf{R} \bowtie_{R.c = S.d} \mathbf{S}$ using a hash join. Assuming that at least one of relations **R** or **S** will fit entirely in main memory, and that no suitable indexes are available to use, what is the estimated cost of the hash join? Give your answer in terms of quantities like B(R), T(R), etc., and give a brief explanation supporting your answer (i.e., how does a hash join access the data?).