CSE 414 Midterm Exam

May 4, 2015

Name ______________________________

Question 1	/ 25
Question 2	/ 10
Question 3	/ 20
Question 4	/ 30
Question 5	/ 15
Total	/ 100

The exam is closed everything but otherwise you may not use any other references. No books, computers, electronics devices, phones of the smart or not-so-smart variety, telegraphs, telepathy, tattoos, mirrors, smoke signals, banana watches, or other contraptions permitted.

The exam lasts 50 minutes. 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.

Reference Information

This information may be useful during the exam. Feel free to use it or not as you wish. You can remove this page from the exam if that is convenient.

Reference for SQL Syntax

Outer Joins

```
-- left outer join with two selections:
select *
from R left outer join S on R.x=55 and R.y=S.z and S.u=99
```

The UNION Operation

```
select R.k from R union select S.k from S
```

The CASE Statement

```
select R.name, (case when R.rating=1 then 'like it'
                 when R.rating=0 then 'do not like it'
                 when R.rating is null then 'do not know'
                 else 'unknown' end)
                 as a_rating
from R;
```

The WITH Statement

Note: with is not supported in sqlite, but it is supported SQL Server and in postgres.

```
with T as (select * from R where R.K>10)
select * from T where T.K<20
```

Reference for Relational Algebra

Name	Symbol
Selection	σ
Projection	π
Join	⋈
Group By	γ
Set Difference	–
Duplicate Elimination	δ
Renaming	ρ
Sort	τ

Several questions on this exam deal with a database that contains information about amusement parks, attractions (rides), requirements for visitors, and which parks have which attractions. There are four tables:

Park(pid, pname, location)
Attraction(aid, aname, rid)
Requires (rid, min_age, min_height)
Has (pid, aid)

The underlined attributes are keys for each relation. The tables contain the following information:

- *Park* describes an amusement park giving its unique id (*pid*, an integer) and the park name and location (both strings).
- *Attraction* describes each attraction that might be found in one or more parks. Attribute *aid* is a unique integer id, *aname* is a string giving the attraction name, and *rid* is a foreign key into the Requires relation giving the minimum age and height requirements needed to use the attraction.
- *Requires* gives a set of minimum visitor requirements that apply to one or more attractions. Each set of requirements has a unique integer *rid*, and the two other attributes are integers giving the minimum age and height (in inches) needed to meet this set of requirements.
- *Has* is the inventory of which parks have which attractions. *pid* and *aid* are integers that are foreign keys to *Park* and *Attraction* respectively.

You should assume that all attributes in all relations are *not NULL* and you do not need to worry about NULL values in the tables.

The next page contains some sample data for each of these tables. The data may be useful in understanding how the information is stored in the tables, although the data is not used directly in any of the questions.

You may remove this page and the next from the test for reference if that is convenient.

Some example data as it could appear in the tables. Schemas repeated for reference.

Park(pid, pname, location)
Attraction(aid, aname, rid)
Requires (rid, min_age, min_height)
Has (pid, aid)

Park

pid	pname	location
001	Disneyland	Anaheim, CA
002	Walt Disney World	Orlando, FL
003	Disney's California Adventure	Anaheim, CA
004	Six Flags Great Adventure	Jackson, NJ
005	Wild Waves	Federal Way, WA
…	…	…

Attraction

aid	aname	rid
001	It's A Small World	001
002	Matterhorn Mountain	014
003	Aladdin	105
004	Wave Pool	003
…	…	…

Requires

rid	min_age	min_height
001	2	24
002	18	50
003	12	32
…	…	…

Has

pid	aid
001	001
001	002
005	004
…	…

Park(pid, pname, location) Requires (rid, min_age, min_height)
Attraction(aid, aname, rid) Has (pid, aid)

Question 1. (25 points) SQL queries. Write SQL queries to retrieve the requested information from the database tables described previously. The queries you write must be proper SQL that would be accepted by SQL Server or any other standard SQL implementation. You should not use incorrect SQL, even if sqlite or some other system might produce some sort of answer from the buggy SQL.

(a) (10 points) List the names of all attractions that are found in at least three different parks and that require customers to be adults. A person is considered an adult if they are at least 18 years old. The names can be listed in any order.

(continued next page)

Park(pid, pname, location)	Requires (rid, min_age, min_height)
Attraction(aid, aname, rid)	Has (pid, aid)

Question 1. (cont.) (b) (15 points) List the names and locations of the parks that have the most attractions for visitors age 12 and under. If more than one park is tied for this maximum number, list all such parks and their locations. The results can be listed in any order.

Question 2. (10 points) Complete this query by adding an appropriate subquery so it produces the following result: List the names of all parks that have an attraction named ‘Aladdin’ but do not have one named ‘Snow White’.

```
SELECT P.pname
FROM Parks P, Attraction A, Has H
WHERE P.pid = H.pid AND H.aid = A.aid AND A.aname = ‘Aladdin’
   AND NOT EXISTS (




);
```

Question 3. (20 points) We tried to execute the following query and discovered it would not work because it was not valid SQL:

```
SELECT P.pname
FROM Park P, Attraction A, Has H, (SELECT rid FROM Requires WHERE min_height < 48) as R
WHERE A.rid = R.rid AND A.aid = H.aid AND P.pid = H.pid
GROUP BY P.pid
HAVING count(*) >= 10;
```

(a) (6 points) Explain what the problem is and how to fix it. You need to explain exactly what makes this an *illegal* SQL command – i.e., what standard SQL rule(s) or restriction(s) are violated (it is not just a trivial punctuation error). Then explain or show how to fix the bug(s) to get a SQL command that does the same thing as the original one, but does it legally. You ***may not*** rewrite the SQL code beyond that, even if there is a simpler or more elegant way to produce the same answer. If there is more than one way to fix the problem(s), pick one that seems most appropriate to match the intent of the original SQL.

(b) (14 points) Once you have repaired the query in part (a), give a relational algebra expression as either a tree or an equation that is equivalent to the repaired query.

Park(pid, pname, location)	Requires (rid, min_age, min_height)
Attraction(aid, aname, rid)	Has (pid, aid)

Question 4. (30 points) One of the summer interns was working on some queries for us when they left abruptly. On the empty desk we found the following mysterious relational algebra expression:

$$\pi_{pname}\ (\ (\delta_{aid}(\sigma_{aname='Gravitron'}(A) \cup \sigma_{aname='Ring\ of\ Fire'}(A)\ ) \bowtie_{aid} H) \bowtie_{pid} P)$$

We're quite sure that A, H, and P refer to the Attraction, Has, and Park tables respectively. But we don't know what the query does.

(a) (12 points) Translate the above relational algebra expression to SQL. Your SQL does not have to exactly mimic the structure of the relational algebra – just be sure it is a straightforward translation that produces the same results.

(b) (6 points) Give a brief English description of the result returned by this query. You should describe the data or values produced by the query, not give a transliteration or narration of the SQL or relational algebra operations and how they are executed – i.e., an answer that says things like "first join this to that then group by these then sort by those …" will not receive much credit. The answer is supposed to describe "what" the query produces, not "how" it is done.

(continued next page)

Park(pid, pname, location)	Requires (rid, min_age, min_height)
Attraction(aid, aname, rid)	Has (pid, aid)

Question 4. (cont.) Suppose that *no* indexes have been defined on any of the tables in our database. If you were allowed to create ***no more than two*** indexes to speed up the relational algebra query given at the beginning of this question, which one or two indexes would you create and why? Yes, you can have an index involving two or more attributes (e.g., Table(attr1,attr2, …)) if that is useful. You should assume that the relational algebra plan is not modified further, but will be executed as written.

And yes, it may well be that three or more indexes would really be needed to make this query run quickly, but the point of the question is for you to pick and justify the two that would be *most* useful.

(c) (6 points) List the one or two indexes you would pick:

(i)

(ii)

(d) (6 points) Give a short justification for your choice(s) of index(es). In your justification you might need to make some assumptions based on a reasonable understanding of what typical data might look like (the relative numbers of Attractions vs. Parks, for example), which attribute values are likely to be unique or duplicated, and so forth. Please keep your answer short and to the point.

Question 5. (15 points) Suppose we have two relations X and Y with the following characteristics:

Schemas: X(<u>a</u>, b), Y(<u>p</u>, q) (i.e., primary keys are X(<u>a</u>) and Y(<u>p</u>))

Value frequency estimates for non-keys: V(X,b) = 200, V(Y,q) = 20

Sizes: B(X) = 40, T(X) = 1000, B(Y) = 100, T(Y) = 1500

Clustering: the data in both X and Y are physically clustered on primary keys.

(a) (7 points) Suppose that we implement a simple join between X and Y using a one-pass *hash join*. Assume that no indexes are used for any attributes of either table.

(i) What is the total cost of doing this hash join on these tables? Give your answer in terms of appropriate quantities involving blocks or tuples (e.g., B(X), T(Y)) or whatever is appropriate, then simplify the answer by substituting the actual numbers given above to get your cost estimate.

(ii) What is the minimum number of main memory blocks (M) needed to do this 1-pass hash join? Give a brief explanation for your answer.

(b) (8 points) Now suppose we want to access all the tuples in Y where Y.q=*v*, for some specific value *v* (i.e., $\sigma_{q=v}$). Assume that we have an index on Y.q, but remember that Y is clustered on primary key Y.p. What is the fastest way to access the desired tuples? Should we do a sequential scan of Y or should we use the index on Y.q to retrieve the data? As before, give an analysis using B(…), T(…), V(…, …) or other appropriate quantities, then substitute in the actual numbers to get costs and justify your final answer.