CSE 414 15sp Midterm Exam Sample Solution

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.

```
SELECT A.aname
FROM Attraction A, Has H, Requires R
WHERE A.aid = H.aid
  AND A.rid = R.rid
  AND R.min_age >= 18
GROUP BY A.aid, A.aname
HAVING COUNT(H.pid) >= 3;
```

(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.

```
SELECT P.pname, P.location
FROM Park P, Attraction A, Requires R, Has H
WHERE P.pid = H.pid AND A.aid = H.aid AND R.rid = A.rid AND R.min_age <= 12
GROUP BY P.pid, P.pname, P.location
HAVING COUNT(A.aid) >= ALL (SELECT COUNT(A.aid)
                            FROM Park P, Attraction A, Requires R, Has H
                                WHERE P.pid = H.pid AND A.aid = H.aid AND R.rid = A.rid
                                    AND R.min_age <= 12
                                GROUP BY P.pid);
```

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 (
                            SELECT *
                            FROM Attraction A2, Has H2
                            WHERE H2.pid = P.pid AND H2.aid = A2.aid
                                AND A2.aname = 'Snow White'
);
```

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.

The attribute P.pname in the SELECT clause does not appear in the GROUP BY clause as either an attribute name or the named result of an aggregate function.

The easiest fix is to add P.pname to the GROUP BY clause.

(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.

$$\pi_{pname}(\sigma_{cnt>=10}(\gamma_{pid,pname,count(*)->cnt}(((\sigma_{min\_height<48}(R) \bowtie_{rid} A) \bowtie_{aid} H) \bowtie_{pid} P))$$

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.

```
SELECT P.pname
FROM Park P, Has H, (SELECT DISTINCT  A.aid
                     FROM Attraction A
                     WHERE A.aname = 'Gravitrion' OR A.aname = 'Ring of Fire') as A
WHERE A.aid = H.aid AND P.pid = H.pid;
```

Other queries that returned the proper result and corresponded reasonably closely to the given relational algebra expression received full credit.

(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.

The query returns the names of all Parks that have an Attraction named 'Gravitron' or an Attraction named 'Ring of Fire'.

(continued next page)

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) **Attraction(aname)**

(ii) **Has(aid,pid)**

(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.

For this particular query, an index on Attraction(aname) is likely to be very selective and allow us to narrow down the Attraction records to only the ones that we need.

A two-level index on Has(aid,pid) will allow us to directly find all of the pid values associated with each of the selected attractions. Since the Park table is likely to be relatively small compared to Has and Attraction, it will be less expensive to scan that sequentially once we know the desired pid values.

There are some other reasonable choices for indexes and some that are not quite as effective but could still significantly speed up the query. Answers were judged on the usefulness of the proposed indexes and the quality of the reasoning justifying the choices.

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

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

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.

The total cost is the cost to read each relation once: B(X) + B(Y) = 40 + 100 = 140

(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.

The minimum space needed is the amount needed for a hash table to hold the smaller relation (X) plus one block to hold tuples from the other relation as they are read during the join. So:

M = min(B(X), B(Y)) + 1 = 41

(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.

The cost to read Y sequentially is B(Y) = 100.

The cost to read the selected tuples one at a time using the index is T(Y) / V(Y,q) = 1500/20 = 75.

So it is somewhat cheaper to use the index to read the desired tuples in this case.