CSE 344 Midterm Exam

February 9, 2015

Name: _____Solutions_____

Question 1	/ 10
Question 2	/ 39
Question 3	/ 16
Question 4	/ 28
Question 5	/ 12
Total	/ 105

The exam is closed everything except for 1 letter-size sheet of notes. No books, computers, electronics devices, phones of the smart or not-so-smart variety, telegraphs, telepathy, tattoos, mirrors, smoke signals, or other contraptions permitted. By putting your name on this exam, you are certifying that you did not give or receive any unpermitted aid in the exam.

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.

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
Natural Join	⋈
Group By	γ
Set Difference	–
Duplicate Elimination	δ
Renaming	ϱ

Question 1. (10 points, 1 point each) Warm up – True or false (circle one).

T F XML documents must have a predefined schema.

T F Given relational schema R(A, B, C), having a single index on (A, B) always provide the same speedup on queries as having two separate indexes on A and B.

T F Nested loop join always returns the same results as a merge join.
// They are just different implementations of join

T F Queries with a subquery that uses `sum` cannot be unnested.

T F XML attributes are unordered.

T F If A = Unknown, B = False, C = True, then NOT((A → B) ∧ C) ∧ C = False when evaluated in SQL.

T F You can select an attribute that is not in a group by or an aggregate.

T F There can be multiple unclustered indexes for a given relation.

T F Non-recursive Datalog without negation is as expressive as relational calculus.

T F A left outer join can always be rewritten using a right outer join.
// R loj S = S roj R ☺

Question 2. At the racetrack (39 points).

The International Sled Dog (Husky) Racing Association (ISDRA) uses the following tables to store information about previous sled races[1]:

```
Dogs    (did integer, name string, age integer)
Mushers (mid integer, name string) -- mushers are drivers of sleds

Races   (mid integer, did integer, raceNumber int)
-- mid and did are foreign keys to Mushers and Dogs, respectively
```

Sample tuples from the tables:

```
Dogs:    (1, "Harry"), (2, "Dubs")
Mushers: (10, "Mary Shields"), (11, "Rick Swenson")
Races:   (10, 1, 1), (11, 2, 1)
```

Write a SQL statement that computes each of the following:

a) Show the SQL statements for creating the Races table (5 points).

```
CREATE TABLE Races (mid INTEGER REFERENCES Mushers,
                    did INTEGER REFERENCES Dogs,
                    raceNumber INTEGER,
                    PRIMARY KEY(mid, did, raceNumber));
```

b) Find the unique set of musher names who have used both "Dubs" and "Harry" in races before (7 points).

```
SELECT DISTINCT M.name
FROM   Mushers M, Racers R1, Dogs D1, Races R2, Dogs D2
WHERE  M.mid = R1.mid AND R1.did = D1.did AND
       M.mid = R2.mid AND R2.did = D2.did AND
       D1.name = "Dubs" AND D2.name = "Harry"
```

[1] This is a made-up fact.
[2] Unfortunately, this is also a false fact…

Problem 2, continued

(Schema repeated here for your reference)

```
Dogs    (did integer, name string, age integer)
Mushers (mid integer, name string) -- mushers are drivers of sleds

Races   (mid integer, did integer, raceNumber int)
-- mid and did are foreign keys to Mushers and Dogs, respectively
```

c) Find the number of times that each dog has participated in races. Only show the dogs that have participated in at least 1 race. The resulting table should have schema (`name string, raceCount integer`) (7 points).

```
SELECT   D.name AS name, COUNT(*) AS raceCount
FROM     Dogs D, Races R
WHERE    R.did = D.did
GROUP BY D.did, D.name
```

d) Find the names of mushers who have raced with all the dogs in the past.

```
SELECT M.name
FROM   Mushers M
WHERE  NOT EXISTS (SELECT D.did
                   FROM   Dogs D
                   WHERE  NOT EXISTS (SELECT R.did
                                      FROM  Races R
                                      WHERE R.did = D.did AND R.mid = M.mid))
```

Problem 2, continued

(Schema repeated here for your reference)

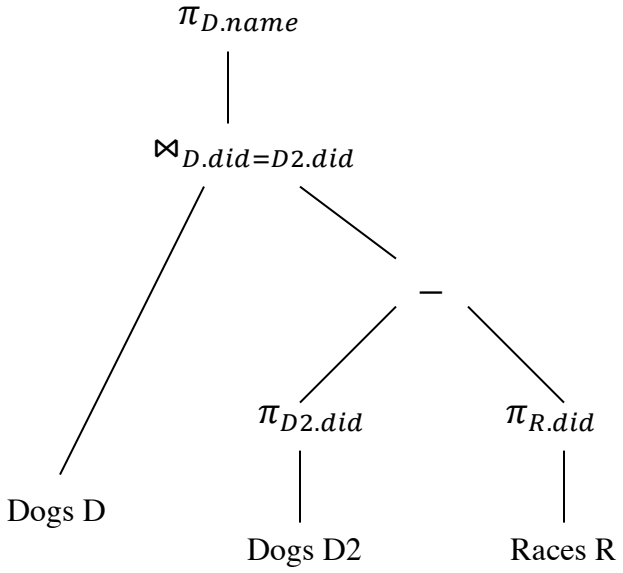
```
Dogs    (did integer, name string, age integer)
Mushers (mid integer, name string) -- mushers are drivers of the sled

Races   (mid integer, did integer, raceNumber int)
-- mid and did are foreign keys to Mushers and Dogs, respectively
```

e) Find the names of dogs that have never participated in any races (7 points).

```
SELECT D.name
FROM   Dogs D
WHERE  D.did NOT IN (SELECT did FROM Races)
```

f) Draw the relational algebra tree for the query in e) (6 points).



Question 3. To index or not to index (16 points, 4 points each).

The ISDRA wants to create some indexes for the following queries.[2] For each of the queries, circle the top **ONE** index that would be **most useful** in speeding up processing of that query (there can be more than one answer, but only circle one. No credit will be given if multiple entries are circled). Assume that there are no existing indexes and all data in the tables are not clustered. Ignore the time needed to construct the indexes. Below are the relations for your reference.

```
Dogs    (did integer, name string, age integer)
Mushers (mid integer, name string) -- mushers are drivers of sleds
Races   (mid integer, did integer, raceNumber integer)
-- mid and did are foreign keys to Mushers and Dogs, respectively
```

a)
```
SELECT * FROM Dogs D, Races R WHERE D.did = R.did
```

Dogs(did)	Dogs(name)	Dogs(did, name)	
Mushers(mid)	Mushers(name)	Mushers(mid, name)	Mushers(name, mid)
Races(mid)	Races(did)	Races(raceNumber)	
Races(mid, did)	Races(did, mid)	Races(raceNumber, mid)	
Races(mid, raceNumber)	Races(did, raceNumber)	Races(raceNumber, did)	
Races(mid, did, raceNumber)	Races(did, mid, raceNumber)	Races(raceNumber, mid, did)	

b)
```
SELECT M.name FROM Mushers M WHERE M.mid > 10
```

Dogs(did)	Dogs(name)	Dogs(did, name)	
Mushers(mid) **(2 points)**	Mushers(name)	Mushers(mid, name)	Mushers(name, mid)
Races(mid)	Races(did)	Races(raceNumber)	
Races(mid, did)	Races(did, mid)	Races(raceNumber, mid)	
Races(mid, raceNumber)	Races(did, raceNumber)	Races(raceNumber, did)	
Races(mid, did, raceNumber)	Races(did, mid, raceNumber)	Races(raceNumber, mid, did)	

c)
```
SELECT   R.raceNumber, R.mid, COUNT(*) FROM Races R
  GROUP BY R.raceNumber, R.mid HAVING R.raceNumber > 10
```

Dogs(did)	Dogs(name)	Dogs(did, name)	
Mushers(mid)	Mushers(name)	Mushers(mid, name)	Mushers(name, mid)
Races(mid)	Races(did)	Races(raceNumber) **(2 points)**	
Races(mid, did)	Races(did, mid)	Races(raceNumber, mid)	
Races(mid, raceNumber)	Races(did, raceNumber)	Races(raceNumber, did)	
Races(mid, did, raceNumber)	Races(did, mid, raceNumber)	Races(raceNumber, mid, did)	

d)
```
SELECT D.name FROM Dogs D WHERE 42 > D.age AND D.age > 21
```

Hashtable index on Dogs(age)	B-tree index on Dogs(age) **(2 points)**
Hashtable index on Dogs(name)	B-tree index on Dogs(name)
Hashtable index on Dogs(age, name)	B-tree index on Dogs(age, name)
Hashtable index on Dogs(name, age)	B-tree index on Dogs(name, age)

[2] Unfortunately, this is also a false fact…

Question 4. Fun with languages (28 points, 7 points each)

The ISDRA maintains bookstores with the following schema:

```
Books   (bid integer, name string, author string, genre string)
Stores  (sid integer, name, city)
Catalog (sid, bid)
-- means that store sid sells book bid
-- sid and bid are foreign keys to Stores and Books, respectively
```

Sample tuples from the tables:

```
Books:   (1, “Running Dog”, “Don Delillo”, “fiction”)
Stores:  (15, “Top Dawgs”, “Seattle”)
Catalog: (1, 15)
```

No fact is null in the relations and you can assume set semantics for all the questions below. You can use = and ≠ to compare strings. Please only write safe queries.

a) Write a Datalog+negation query to find all stores in Seattle that only sell fiction.

```
NonFiction(bid) ← Books(bid,n,a,g), g ≠ “fiction”
NonAns(sid)     ← Stores(sid,_,_), Catalog(sid, bid), NonFiction(bid)
Ans(n)          ← Stores(sid, n, “Seattle”), NOT NonAns(sid)
```

b) Write a relational calculus query to find the names of all the books that are sold in a store that sells at least one fiction.

```
FictionStore(s)= ∃b, n, a, c. Catalog(s, b) ∧ Books(b, n, a, “fiction”) ∧
                              Stores(s, n, c)

Ans(n)         = ∃s, b, a, g. Catalog(s, b) ∧ FictionStores(s) ∧
                              Books(b, n, a, g)
```

Question 4 (continued).

(Schema repeated for your reference)

```
Books    (bid integer, name string, author string, genre string)
Stores   (sid integer, name, city)
Catalog  (sid, bid)
-- means that store sid sells book bid
-- sid and bid are foreign keys to Stores and Books, respectively
```

c) Write a Datalog+negation query that find all the stores that only sell books by a single author.

```
NonAns (s) ← Catalog(s, b1), Catalog(s, b2),
             Books(b1, _, a1, _), Books(b2, _, a2, _), a1 ≠ a2
Ans(n)     ← Stores(s, n, _), NOT NonAns(s)

-- We also accepted answers that compute stores that sell books that are written
-- by a single author.
```

d) Write the same query as in c) but in relational calculus.

```
Ans(n) = ∃a, s, c, g. Stores(s, n, c) ∧
                      (∀b. Catalog(s, b) → Books(b, n, a, g))
```

Question 5. Are they the same? (12 points, 4 points each).

Consider the following two relational algebra expressions Q1 and Q2 on relation R(A, B). Both attributes are integers.

a) If Q1 and Q2 are equivalent, write "equivalent." Otherwise, write "not equivalent" and construct a small instance on R where the above queries return different answers.

Q1: $\pi_A(\sigma_{A=42 \wedge B \neq C}(R \bowtie \varrho_{A,C}(R)))$
Q2: $\sigma_{A=42}(\pi_A(R))$

Not equivalent.
R(A,B) = (42,42)

b) If Q1 and Q2 are equivalent, write "equivalent." Otherwise, write "not equivalent" and construct a small instance on R where the above queries return different answers.

Q1: $\delta(\sigma_{B=1}(\pi_B(R \times \varrho_{B,C}(R)))$
Q2: $\pi_B(\sigma_{A=1}(R))$

Not equivalent.
R(A,B) = { (2,1), (2,1) }

c) Now consider these three SQL queries on relations R(A, B) and S(X, Y). All attributes are integers.

Q1:
```
SELECT DISTINCT R.B
FROM R
WHERE R.A < 3 AND
      R.B NOT >
       ALL (SELECT S.X
            FROM S
            WHERE S.Y > 8);
```

Q2:
```
SELECT DISTINCT R.B
FROM R
WHERE R.A < 3 AND
      R.B <
       ANY (SELECT S.X
            FROM S
            WHERE S.Y > 8);
```

Q3:
```
SELECT DISTINCT R.B
FROM R
WHERE R.A < 3 AND
      R.B NOT >=
       ALL (SELECT S.X
            FROM S
            WHERE S.Y > 8);
```

Is Q1 equivalent to Q2 and / or Q3, or neither? (Just state the answer, no justification needed.)

Neither.
// You have to clearly state that Q1 is not equivalent to Q2, and that Q1 and also not equivalent to Q3
// ambiguous statements such as Q1 ≠ Q2 = Q3 do not get full points.

END OF EXAM