CSE414 Midterm Exam
Spring 2018
May 4, 2018

- Please read all instructions (including these) carefully.
- **This is a closed-book exam. You are allowed one page of note sheets that you can write on both sides.**
- Write your name and UW student number below.
- No electronic devices are allowed, including **cell phones** used merely as watches. Silence your cell phones and place them in your bag.
- Solutions will be graded on correctness and ***clarity***. Each problem has a relatively simple and straightforward solution. Partial solutions will be graded for partial credit.
- There are 10 pages in this exam, not including this one.
- There are 5 questions, each with multiple parts. If you get stuck on a question move on and come back to it later. Do not feel that you need to complete every question on the exam!
- You have 50 minutes to work on the exam.
- Please write your answers in the space provided on the exam, and clearly mark your solutions. You may use the blank pages as scratch paper. **Do not** use any additional scratch paper.
- *Relax. You are here to learn. Good luck!*

By writing your name below, you certify that you have not received any unpermitted aid for this exam, and that you will not disclose the contents of the exam to anyone in the class who has not taken it.

NAME: _____Solutions_____

STUDENT NUMBER: _____

Problem	Points	Problem	Points
1	10	4	15
2	40	5	3
3	32	**Total**	100

Problem 1: Warm up (10 points total)

Select either True or False for each of the following questions. For each question you get 1 points for answering it correctly, -0.5 point for an incorrect answer, and 0 point for no answer. The minimum you will get for this entire problem is 0.

a) 1NF allows storing lists as an attribute.

True False

b) Block-partitioning never results in data skew.

[Ans: technically block-partitioning doesn't introduce data skews, but if the number of tuples is not divisible by the number of servers then some servers will receive one more tuple than others. We decided to give points to either answer.]

True False

c) `R` $\bowtie_{R.a>S.b}$ `S` is an instance of a theta-join.

True False

d) All Datalog programs can be stratified.

True False

e) In `R(x,z) :- S(x,y), T(y,z)`, `R(x,z)` is an extensional predicate.

True False

f) In relational query processing, physical query plans are translated into logical ones by the optimizer.

True False

g) Subqueries can be used in the `SELECT` clause in SQL++, as long as the subquery returns a single value.

True False

h) Using partitioning to scale up a database means cloning the entire dataset multiple times across different machines.

True False

i) Relational data instances cannot be represented as trees.

True False

j) In SQL, aggregates are always processed after grouping operations.

[Ans: aggregates in subqueries are processed prior to grouping operations in the outer queries.]

True False

Problem 2: SQL (40 points total)

We will work with the following schema for sport teams in this exam.

```
Person(pid, name, age, salary, rating) -- pid = person ID
Team(tid, name) -- tid = team ID
MemberOf(pid, tid) -- pid is foreign key to Person, and tid is FK to Team
Games(winnerTid, loserTid) -- games won by winnerTid against loserTid,
                           -- both are foreign keys to Team
```

You can assume none of the tables contains `NULL` values.

a) (10 points) Write a SQL query that returns the names of those who are currently members of multiple teams. Call the resulting column `name`.

```
SELECT p.name AS name
FROM Person AS p, Team AS t, MemberOf AS m
WHERE p.pid = m.pid AND M.tid = t.tid
GROUP BY p.pid, p.name
HAVING count(*) > 1
```

b) (10 points) Write a SQL query that returns the team name(s) that has the highest total salary across all team members. Call the resulting column `name`.

```
WITH ( SELECT t.tid, t.name AS name, Sum(p.salary) AS total_salary
       FROM Team AS t, Person AS p, Member AS m
       WHERE t.tid = m.tid AND p.pid = m.pid
       GROUP BY t.tid, t.name ) AS X

SELECT x1.name AS name
FROM X AS x1, X AS x2
WHERE x1.total_salary = max(x2.total_salary)
```

Schema repeated here for your reference:

```
Person(pid, name, age, salary, rating) -- pid = person ID
Team(tid, name) -- tid = team ID
```

```
MemberOf(pid, tid) -- pid is foreign key to Person, and tid is FK to Team
Games(winnerTid, loserTid) -- games won by winnerTid against loserTid, both are FKs to Team
```

c) (5 points) Given the following contents of `Person`:

pid	name	age	salary	rating
1	"A"	30	50,000	3
2	"B"	24	10,000	3
3	"C"	21	15,000	2

What does the following query return? Write out the tuples in the resulting relation.

```
SELECT P1.rating as rating, AVG(P1.age) as avgAge
FROM Person P1
WHERE P1.age > 21
GROUP BY P1.rating
HAVING 1 < (SELECT COUNT(*)
            FROM Person P2
            WHERE P1.rating = P2.rating AND P2.age >= 21)
```

rating	avgAge
3	27

Schema repeated here for your reference:

```
Person(pid, name, age, salary, rating) -- pid = person ID
Team(tid, name) -- tid = team ID
MemberOf(pid, tid)
Games(winnerTid, loserTid) -- games won by winnerTid against loserTid
```

d) (10 points) Rewrite the `Answer` relation following relational algebra query into SQL **without using `GROUP BY`**.

```
T1(rating,sal) = GroupBy[rating, avg(salary)->sal](Person)
Answer(r,sal) = Rename[r,sal](T1)
```

```
SELECT DISTINCT P2.rating as r,
                (SELECT AVG(t.salary)
                 FROM (SELECT P1.salary AS salary
                       FROM Person AS P1
                       WHERE P1.rating = P2.rating) AS t)
FROM Person AS P2
```

Schema repeated here for your reference:

```
Person(pid, name, age, salary, rating) -- pid = person ID
Team(tid, name) -- tid = team ID
MemberOf(pid, tid) -- pid is foreign key to Person, and tid is FK to Team
Games(winnerTid, loserTid) -- games won by winnerTid against loserTid, both are FKs to Team
```

e) (5 points) Are the following two queries equivalent? If so, write "Yes" below. Otherwise, write "No," and make up the contents of `Person` such that, when run on the two queries, they will return different results. Write the contents in a table form, for instance,

pid	name	age	salary	rating
1	"A"	30	50,000	3

(not an answer). Make sure your data satisfies the key constraints!

```
SELECT P1.name
FROM Person P1
WHERE NOT EXISTS (SELECT *
                  FROM Person P2
                  WHERE P2.age < 21 AND P1.rating <= P2.rating)
```

```
SELECT P1.name
FROM Person P1
WHERE P1.rating > ANY (SELECT P2.rating
                       FROM Person P2
                       WHERE P2.age < 21)
```

No, these queries are not equivalent. Example data:

pid	name	age	salary	rating
1	Alex	27	1000	3
2	Mike	20	500	2
3	Julie	18	600	4
4	Alice	25	1900	5

Q1 will return only Alice whereas Q2 will return Alice, Julie, and Alex.

Problem 3: Datalog (32 points total)

Same schema as before, repeated here for your reference.

```
Person(pid, name, age, salary, rating) -- pid = person ID
Team(tid, name) -- tid = team ID
MemberOf(pid, tid) -- pid is foreign key to Person, and tid is FK to Team
Games(winnerTid, loserTid) -- games won by winnerTid against loserTid,
                           -- both are foreign keys to Team
```

You can assume none of the tables contains `NULL` values.

a) (10 points) Write a safe Datalog query that finds all teams that have never lost any games. Return the team's ID and name in the relation `Q3a(tid, name)`.

```
Lost(x, y) :- Team(x, y), Games(_, x).
Q3a(x, y) :- Team(x, y), !Lost(x, y).
```

b) (10 points) Write a safe Datalog query that find the teams that each of those teams in a) have directly or indirectly beaten. Return the results in the relation `Q3b(winnerTid, loserTid)`. For instance, if `A` is in `Q3a` and `A` beats `B`, and `B` beats `C`, then `Q3b` should contain `(A,B)` and `(A,C)`. You can use the `Q3a` relation without recomputing it.

```
Q3b(x, a) :- Q3a(x, y), Games(x, a).
Q3b(x, a) :- Q3b(x, b), Games(b, a).
```

Schema repeated here for your reference:

```
Person(pid, name, age, salary, rating) -- pid = person ID
Team(tid, name) -- tid = team ID
MemberOf(pid, tid) -- pid is foreign key to Person, and tid is FK to Team
Games(winnerTid, loserTid) -- games won by winnerTid against loserTid, both are FKs to Team
```

c) (4 points each, 12 points total) Are the following relational algebra or SQL queries equivalent? If so, write "Yes" below. Otherwise, write "No," and make up the contents of `Person` and `MemberOf` such that, when run on the two queries, they will return different results.

Write the contents in a table form, for instance, `Person`:

pid	name	age	salary	rating
1	"A"	30	50,000	3

(not an answer). Make sure your data satisfies the key constraints!

Q1:
```
SELECT *
    FROM Person as P, MemberOf as M
    WHERE P.age < 21 AND M.tid > 40 AND P.pid = M.pid
```

Q2: $\sigma_{\text{age<21}}(\sigma_{\text{tid>40}}(\text{MemberOf}) \bowtie_{\text{pid=pid}} \text{Person})$

Yes

Q1: $\sigma_{\text{salary<10k}}(\sigma_{\text{age<21}}(\text{Person}) \bowtie_{\text{pid=pid}} (\sigma_{\text{tid>40}}(\text{MemberOf})))$

Q2: $\sigma_{\text{salary<10k}}(\text{Person} \bowtie_{\text{pid=pid}} (\sigma_{\text{tid>40}}(\text{MemberOf})))$

No.

Person

pid	name	age	salary	rating
1	Alex	22	9000	4

MemberOf

pid	tid
1	45

Q1 will return nothing while Q2 will return the first tuple.

Schema repeated here for your reference:

```
Person(pid, name, age, salary, rating) -- pid = person ID
Team(tid, name) -- tid = team ID
MemberOf(pid, tid) -- pid is foreign key to Person, and tid is FK to Team
Games(winnerTid, loserTid) -- games won by winnerTid against loserTid, both are FKs to Team
```

Q1: $\gamma_{\text{tid, max(salary)->ms}}((\text{Person} \bowtie_{\text{pid=pid}} \text{MemberOf}) \bowtie_{\text{tid=tid}} \text{Team})$

Q2: $\gamma_{\text{tid, max(salary)->ms}}(\text{Person} \bowtie_{\text{pid=pid}} \text{MemberOf})$

Yes.

Problem 4: SQL++ (15 points total)

Same schema as before, repeated here for your reference.

```
Person(pid, name, age, salary, rating) -- pid = person ID
Team(tid, name) -- tid = team ID
MemberOf(pid, tid) -- pid is foreign key to Person, and tid is FK to Team
Games(winnerTid, loserTid) -- games won by winnerTid against loserTid,
                           -- both are FKs to Team
```

You can assume none of the tables contains `NULL` values.

a) (10 points) Assume we now have the information in four JSON documents with the following schema (quotation marks omitted for clarity):

```
Person = [ {pid:1, name:P1, age:30, salary:10k, rating:3.5},
           {pid:2, name:P2, age:24, salary:5k, rating:4.0}, … ]
Team = [ {tid:1, name:T1}, {tid:2, name:T2}, … ]
MemberOf = [ {pid:1, tid:1}, {pid:1, tid:2}, … ]
Games = [ {winnerTid:1, loserTid:2}, {winnerTid:3, loserTid:4}, … ]
```

Write a SQL++ query that combines data from `MemberOf` and `Person` as part of the new $\texttt{Team}_{new}$ document with the following schema:

```
Team_new = [ {tid:1, name:T1,
              members:[ {pid:1, name:P1, age:30, salary:10k, rating:3.5},
                        {pid:2, name:P2, age:24, salary:5k, rating:4.0}, … ]},
             {tid:2, name:T2, Members:[ … ]}, … ]
```

```
SELECT t.tid, t.name, (SELECT p.pid, p.name, p.age, p.salary, p.rating
                       FROM MemberOf AS m, Person AS p
                       WHERE m.pid = p.pid AND m.tid = t.tid)
FROM Team AS t
```

b) (5 points) Given the following contents of $Team_{new}$:

```
Team_new = [ {tid: 1, name: T1, members: [ {pid: 1, name: P1},
                                           {pid: 2, name: P2},
                                           {pid: 3, name: P3} ] },
             {tid: 2, name: T2, members: [ {pid: 3, name: P3},
                                           {pid: 2, name: P2},
                                           {pid: 4, name: P4} ] },
             {tid: 3, name: T3, members: [ {pid: 3, name: P3},
                                           {pid: 2, name: P2},
                                           {pid: 4, name: P4} ] } ]
```

What does the following query return?

```
SELECT DISTINCT m1.pid
FROM Team_new AS t1, t1.members AS m1, Team_new AS t2, t2.members as m2
WHERE m1.pid = m2.pid AND t1.tid <> t2.tid
ORDER BY m1.pid
```

```
[ {pid: 2}, {pid: 3}, {pid: 4} ]
```

Problem 5: Trivia! (3 points total)

Name 3 of the course staff below their picture (first name is fine).

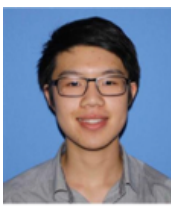
Kodiak

Jack

Boyan

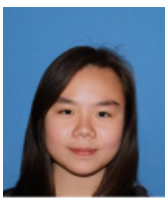
Vineeth

Ying



Jonathan

Alvin



Cindy