CSE 344 Midterm

Wednesday, Oct. 31st, 2018, 1:30-2:20

Name: __________________________

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Total:</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

- This exam is CLOSED book and CLOSED devices.
- You are allowed ONE letter-size page with notes (both sides).
- You have 50 minutes;
- Answer the easy questions before you spend too much time on the more difficult ones.
- Good luck!
1 SQL

1. (50 points)

A large software company maintains the following database about its projects and developers

\[
\text{Project} (\text{pid, name, startYear}) \\
\text{Developer} (\text{did, name, hireYear}) \\
\text{WorksOn} (\text{pid, did, year})
\]

Each project has a name and a start year. Each developer has a name and the year when she/he was hired. \text{WorksOn} records whether a developer worked on a project in a given year. Note that a developer may work on the same project in multiple years, and a project may have multiple developers in any given year.

(a) (5 points) Write the sequence of SQL statements necessary to create the tables above. Assume that \text{name} is a TEXT type, and all other attributes are integers. Include all keys or foreign keys declarations.

\textbf{Solution:}

DROP TABLE IF EXISTS WorksOn;
DROP TABLE IF EXISTS Project;
DROP TABLE IF EXISTS Developer;

CREATE TABLE Project(pid INT PRIMARY KEY, name TEXT, startYear INT);
CREATE TABLE Developer(did INT PRIMARY KEY, name TEXT, hireYear INT);
CREATE TABLE WorksOn(pid INT REFERENCES Project, did INT REFERENCES Developer, year INT);
(b) (5 points) Write a SQL query that returns for each project and for each year the total number of developers who worked on that project in that year. Your query should return the project’s ID, name, year, and number of developers, sorted by year in increasing order, and, within each year, sorted in decreasing order of the number of developers. (Years in which no developers worked need not be included as “0 developers.”)

Solution:
```
SELECT X.pid, X.name, Y.year, count(Y.pid) AS cnt
FROM Project AS X, WorksOn AS Y
WHERE X.pid = Y.pid
GROUP BY X.pid, X.name, Y.year
ORDER BY Y.year, cnt desc;
```
(c) (10 points) A project is called inactive if no developer worked on it since 2010 (including and after 2010). Write a SQL query to find all inactive projects. You should return their IDs and names. (A project is active if a developer worked on it in 2010.)

Solution: Solution 1:

```
SELECT X.pid, X.name
FROM Project AS X
WHERE NOT EXISTS (SELECT *
                   FROM WorksOn AS Y
                   WHERE X.pid = Y.pid and Y.year >= 2010);
```

-1 point for unnecessary joins
-3 points for missing correlation in the subquery X.pid = Y.pid

Solution 2:

```
SELECT X.pid, X.name
FROM Project X left outer join WorksOn Y
     ON X.pid = Y.pid and Y.year >= 2010
GROUP BY X.pid, X.name
HAVING count(Y.pid) = 0;
```

Solution 3 (imperfect because it misses projects without any developers):

```
SELECT X.pid, X.name
FROM Project X, WorksOn Y
WHERE X.pid = Y.pid
GROUP BY X.pid, X.name
HAVING 2010 > max(Y.year);
```

-1 point because incomplete answer

Solution 4

```
SELECT X.pid, X.name
FROM Project X
WHERE 2010 > ALL (SELECT Y.year from WorksOn Y WHERE Y.pid = X.pid);
```
(d) (10 points) For each year since 1990 (including and after 1990), return the project(s) that the most developers worked on during that year. Return the year and project ID. In case of a tie (if multiple projects had the maximum number of developers) return all of them.

Solution:

-- Solution 1: Universal Quantifier Approach with ALL
SELECT W.year, W.pid
FROM WorksOn AS W
WHERE W.year >= 1990
GROUP BY W.year, W.pid
HAVING count(*) >= ALL (SELECT count(*)
FROM WorksOn AS W2
WHERE W2.year = W.year
GROUP BY W.pid);

-- Solution 2: Witness Approach
WITH A AS (SELECT W.year AS year, W.pid AS pid, count(*) AS count
FROM WorksOn AS W
WHERE W.year >= 1990
GROUP BY W.year, W.pid),
B AS (SELECT A.year AS year, max(A.count) AS count
FROM A
GROUP BY A.year)
SELECT A.year, A.pid
FROM A, B
WHERE A.year = B.year AND A.count = B.count;

-- Solution 3: variant on the Witness
WITH A ... -- same as above
SELECT x.year, x.pid
FROM A x, A y
WHERE x.year = y.year
GROUP BY x.year, x.pid, x.count
HAVING x.count = max(y.count);
-5 points for answers containing the expression max(count(*))
-4 points for an expression like HAVING x.count = max(x.count) (same x)
(e) (10 points) 'SystemX' is the oldest project of the company. Write a SQL query that returns all developers who worked every year on 'SystemX', from when it started until 2015. Your query should return the developers’ ID and name.

Solution:

```sql
-- Solution 1
SELECT Z.did, Z.name
FROM Project AS X, WorksOn AS Y, Developer AS Z
WHERE X.pid = Y.pid AND Y.did = Z.did
  AND X.startYear <= Y.year
  AND Y.year <= 2015
  AND X.name = 'SystemX'
GROUP BY Z.did, Z.name, X.startYear
HAVING count(*) = 2015 - X.startYear + 1;

-- Solution 2 (imperfect, but got full credit where present)
SELECT Z.did, Z.name
FROM Developer Z
WHERE NOT EXISTS ( -- find a year where Z didn’t work on SystemX
  SELECT *
  FROM WorksOn Y -- proxy for the set of all years (which we don’t have)
  WHERE NOT EXISTS ( -- check if Z worked that year on SystemX
    SELECT * FROM WorksOn Y2, Project X
    WHERE Y.year = Y2.year and Y2.pid = X.pid and X.name = 'SystemX'));

A solution without any negation got 2-3 points max.
A solution with only 1 negation got 5 points max.
1 point off for unnecessary join
(f) For each question below indicate whether the two SQL queries are equivalent. Assume that the database does not contain any NULL values.

i. (1 point) Are Q1 and Q2 equivalent?

Q1: SELECT DISTINCT Z.did, Z.name
     FROM Project X1, Project X2, Workson Y1, Workson Y2, Developer Z
     WHERE X1.pid = Y1.pid AND Y1.did = Z.did
     AND X2.pid = Y2.pid AND Y2.did = Z.did
     AND X1.startYear < 2010 AND Y1.year > 2015
     AND X2.startYear < 2012 AND Y2.year > 2018;

Q2: SELECT DISTINCT Z.did, Z.name
     FROM Project X, Workson Y, Developer Z
     WHERE X.pid = Y.pid AND Y.did = Z.did
     AND X.startYear < 2010 AND Y.year > 2018;

i. ____ No ____

Yes/No:

ii. (1 point) Are Q3 and Q4 equivalent?

Q3: SELECT DISTINCT Z.did, Z.name
     FROM Project X1, Project X2, Workson Y1, Workson Y2, Developer Z
     WHERE X1.pid = Y1.pid AND Y1.did = Z.did
     AND X2.pid = Y2.pid AND Y2.did = Z.did
     AND X1.startYear < 2010 AND Y1.year > 2015
     AND X2.startYear < 2012 AND Y2.year > 2018;

Q4: SELECT DISTINCT Z.did, Z.name
     FROM Project X, Workson Y, Developer Z
     WHERE X.pid = Y.pid AND Y.did = Z.did
     AND X.startYear < 2010 AND Y.year > 2018;

ii. ____ Yes ____

Yes/No:
iii. (1 point) Are Q5 and Q6 equivalent?

Q5: SELECT X.did, count(*)
    FROM WorksOn X
    GROUP BY X.did;

Q6: SELECT X.did, (SELECT count(*)
    FROM WorksOn Y
    WHERE X.did = Y.did)
    FROM WorksOn X;

iii. Yes

iv. (1 point) Are Q7 and Q8 equivalent?

Q7: SELECT X.did, count(*)
    FROM WorksOn X
    WHERE X.year > 2010
    GROUP BY X.did;

Q8: SELECT X.did, (SELECT count(*)
    FROM WorksOn y
    WHERE X.did = y.did AND y.year > 2010)
    FROM WorksOn X;

iv. No

Yes/No:
v. (1 point) Are Q9 and Q10 equivalent?

Q9: SELECT X.name, Z.name
    FROM Developer X
    LEFT OUTER JOIN
    WorksOn Y ON X.did = Y.did
    LEFT OUTER JOIN
    Project Z ON Y.pid = Z.pid;

Q10: SELECT X.name, Z.name
    FROM Developer X
    JOIN
    WorksOn Y ON X.did = Y.did
    LEFT OUTER JOIN
    Project Z ON Y.pid = Z.pid;

v. Yes/No: No

vi. (1 point) Are Q11 and Q12 equivalent?

Q11: SELECT X.name, Z.name
    FROM Developer X
    LEFT OUTER JOIN
    WorksOn Y ON X.did = Y.did
    LEFT OUTER JOIN
    Project Z ON Y.pid = Z.pid;

Q12: SELECT X.name, Z.name
    FROM Developer X
    LEFT OUTER JOIN
    WorksOn Y on X.did = Y.did
    JOIN
    Project Z on Y.pid = Z.pid;

vi. Yes/No: Yes
(g) Consider the following database instance:

<table>
<thead>
<tr>
<th>Project</th>
<th>WorksOn</th>
<th>Developer</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>name</td>
<td>startYear</td>
</tr>
<tr>
<td>10</td>
<td>SystemX</td>
<td>NULL</td>
</tr>
<tr>
<td>20</td>
<td>SystemY</td>
<td>2016</td>
</tr>
</tbody>
</table>

Solution:

```sql
INSERT INTO Developer VALUES (1, 'Alice', 2015);
INSERT INTO Developer VALUES (2, 'Bob', NULL);
INSERT INTO Project VALUES (10, 'SystemX', NULL);
INSERT INTO Project VALUES (20, 'SystemY', 2016);
INSERT INTO WorksOn VALUES (10, 1, NULL);
INSERT INTO WorksOn VALUES (20, 1, 2015);
INSERT INTO WorksOn VALUES (20, 2, NULL);
```

Indicate for each query below what answers it returns; write "empty" if the answer is the emptyset.

i. (1 point) What does query Q1 return?

```sql
SELECT X.name AS proj, Z.name AS dev
FROM Project X, WorksOn Y, Developer Z
WHERE X.pid = Y.pid AND Y.did = Z.did
    AND ((Z.hireyear = 2015 AND Y.year = 2015) OR X.startyear = 2016);
```

Solution:

```
proj | dev
--- | ---
SystemY | Alice
SystemY | Bob
```

ii. (1 point) What does query Q2 return?

```sql
SELECT X.name AS proj, Z.name AS dev
FROM Project X, WorksOn Y, Developer Z
WHERE X.pid = Y.pid AND Y.did = Z.did
    AND (Z.hireyear = 2015 AND Y.year = 2015) OR X.startyear = 2016);
```

Solution:

```
proj | dev
--- | ---
SystemY | Alice
```
iii. (1 point) What does query Q3 return?:

SELECT X.name AS proj, Z.name AS dev
FROM Project X, Workson Y, Developer Z
WHERE X.pid = Y.pid AND Y.did = Z.did
AND (Z.hireyear = 2015 OR (Y.year = 2015 AND NOT(X.startyear = 2016)));

Solution:

<table>
<thead>
<tr>
<th>proj</th>
<th>dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>SystemX</td>
<td>Alice</td>
</tr>
<tr>
<td>SystemY</td>
<td>Alice</td>
</tr>
</tbody>
</table>

iv. (1 point) What does query Q4 return?:

SELECT X.name AS proj, Z.name AS dev
FROM Project X, Workson Y, Developer Z
WHERE X.pid = Y.pid AND Y.did = Z.did
AND (NOT(Z.hireyear = 2015) OR (Y.year = 2015 AND X.startyear = 2016));

Solution:

<table>
<thead>
<tr>
<th>proj</th>
<th>dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>SystemY</td>
<td>Alice</td>
</tr>
</tbody>
</table>
2 Relational Algebra

2. (25 points)

Consider the same relational schema as before, including the key/foreign key constraints:

- Project(pid, name, startYear)
- Developer(did, name, hireYear)
- WorksOn(pid, did, year)

(a) (5 points) Write a Relational Algebra expression in the form of a logical query plan (i.e., draw a tree) that is equivalent to the SQL query below. Your query plan does not have to be necessarily “optimal”: however, points will be taken off for overly complex solutions.

Hint: to avoid renaming, use aliases in the query plan, like this

SELECT X.did, X.name, count(*)
FROM Developer X, WorksOn y, Project Z
WHERE X.did = y.did and y.pid = Z.pid
    AND y.year < Z.startYear + 2
GROUP BY X.did, X.name, X.hireYear
HAVING X.hireYear + 10 < max(startYear);

Solution:

\[\pi_{\text{budget}}(\text{WorksOn} \bowtie_{\text{x.pid}=\text{y.pid}} \text{Project} \bowtie_{\text{x.pid}=?})\]
(b) i. (2 points) Which of the following is the most accurate English interpretation of the SQL query below?

```
SELECT X.did
FROM Developer X
WHERE NOT EXISTS
  (SELECT *
   FROM Project Z
   WHERE NOT EXISTS
     (SELECT *
      FROM WorksOn Y
      WHERE X.did = Y.did AND Y.pid = Z.pid
      AND Y.year = 2015));
```

Developers that...
A: in 2015, didn’t work on any projects at all
B: in 2015, didn’t work on at least one of the projects
C: in 2015, worked on every single project
D: in 2015, worked on at least one project

i. ______ C ______

A/B/C/D:
ii. (10 points) Write a Relational Algebra expression in the form of a logical query plan (i.e., draw a tree) that is equivalent to the SQL query in the previous question (reproduced below). Your query plan does not have to be necessarily “optimal”: however, points will be taken off for overly complex solutions.

```
SELECT X.did
FROM Developer X
WHERE NOT EXISTS
  (SELECT *
   FROM Project Z
   WHERE NOT EXISTS
     (SELECT *
      FROM WorksOn Y
      WHERE X.did = Y.did AND Y.pid = Z.pid
      AND Y.year = 2015));
```

**Solution:**

\[
\pi_{X_.did}(\rho_{X_3}(Developer)) - \pi_{X_.did}(A - B)
\]

Where A is

\[
\pi_{Y_2.pid,Y_2.did}(\rho_{Y_2}(WorksOn))
\]

Where B is

\[
\pi_{Z.pid,X_.did}(\sigma_{Y_.year=2015}(\rho_{X_3}(Developer) \bowtie_{(X_.did=Y_.did)}\rho_{Y_3}(WorksOn) \bowtie_{(Y_.pid=Z_.pid)}\rho_{Z_3}(Project)))
\]
(c) Assume the database schema in Question 1 (also shown in the top right corner); assume the key and foreign keys defined there, and that the database instance does not contain NULL’s. Answer the questions below, assuming all expressions have set semantics.

i. (2 points) The notation \(|S|\) means the cardinality of a set \(S\) (number of tuples in \(S\)). Does the following always hold?

\[ |\text{Developer} \bowtie_{\text{did} = \text{did}} \sigma_{\text{year} < 2015} (\text{WorksOn})| \leq |\text{Developer}| \]

i. ____ No ____

Yes/No:

ii. (2 points) Does the following always hold?

\[ |\text{Developer} \bowtie_{\text{did} = \text{did}} \sigma_{\text{year} < 2015} (\text{WorksOn})| \leq |\text{WorksOn}| \]

ii. ____ Yes ____

Yes/No:

iii. (2 points) Does the following always hold?

\[ |\sigma_{\text{hireYear} < 2015} (\text{Developer}) \bowtie_{\text{did} = \text{did} \land \text{hireYear} = \text{year}} \text{WorksOn}| \leq |\sigma_{\text{year} < 2015} (\text{WorksOn})| \]

iii. ____ Yes ____

Yes/No:

iv. (2 points) Does the following always hold?

\[ |\text{Developer} \bowtie_{\text{did} = \text{did} \land \text{hireYear} = \text{year}} \sigma_{\text{year} < 2015} (\text{WorksOn})| \leq |\sigma_{\text{year} < 2015} (\text{Developer})| \]

iv. ____ No ____

Yes/No:
3 Datalog

Project(pid, name, startYear)
Developer(did, name, hireYear)
WorksOn(pid, did, year)

3. (15 points)

Answer the questions below.

(a) (5 points) Write a datalog program that returns the id and name of all employees who never worked on the project "SystemX".

Solution:

\[
\text{nonAnswer}(\text{did}) :- \text{Developer}(\text{did}, \_, \_), \text{WorksOn}(\text{pid}, \text{did}, \_), \text{Project}(\text{pid}, "\text{SystemX}", \_)
\]

\[
\text{answer}(\text{did}, \text{name}) :- \text{Developer}(\text{did}, \text{name}), ! \text{nonAnswer}(\text{did})
\]
(b) (10 points) Two developers are “pals” if they worked together on a common project during the same year, or if they have a common pal. Write a datalog program to find the number of common pals of "Alice" and "Bob". For the aggregate, use the Souffle syntax.

Solution:

\[
\text{pal}(x,y) :- \text{WorksOn}(z,x,w), \text{WorksOn}(z,y,w), x \neq y \\
\text{pal}(x,y) :- \text{pal}(x,z), \text{pal}(z,y), x \neq y \\
\text{common}(x) :- \text{Developer}(aid,"Alice",:) \land \text{Developer}(bid,"Bob",:), \text{pal}(aid,x), \text{pal}(bid,x) \\
\text{answer}(c) :- c = \text{count}\{\text{common}(x)\} \\
\]

-- No points removed if 'x \neq y' is missing.
4 Miscellaneous

4. (10 points)

For each statement below, indicate whether it is true or false.

(a) (1 point) Physical data independence means that the logical representation of the data is independent of its physical representation in memory and on disk.

True/False:

(b) (1 point) If the primary key of a relation consists of the attributes $A, B$, then no record can have $A = B$.

True/False:

(c) (1 point) If the attribute $K$ of a relation $R$ is a foreign key to some $S$ then every value of that attribute in $R$ must be the value of some key in $S$.

True/False:

(d) (1 point) First normal form means that all relations in the database are flat.

True/False:

(e) (1 point) All queries expressible in Relational Algebra are monotone.

True/False:
(f) (1 point) The following Datalog rule is safe:
Q(name, year) :- Developer(did, name, _), !WorksOn(_, did, year)

(f) false

True/False:

(g) (1 point) The following Datalog rule is safe:
Q(name, year) :- Project(_, name, year), year < 2015

(g) true

True/False:

(h) (1 point) If two relations have no duplicate tuples (i.e. are sets), then their join (i.e., a relation resulting from joining the two relations with any join condition) does not have duplicate tuples either.

(h) true

True/False:

(i) (1 point) The left outer join returns at most as many tuples as the inner join.

(i) false

True/False:

(j) (1 point) This relational algebra expression is monotone: $\sigma_{\text{not}(\text{year}=2015)}(\text{WorksOn})$.

(j) true

True/False: