cs4111.001–Introduction to Databases
Spring 2016
Midterm Exam
Closed Book and Notes
Duration: 75 minutes

Professor Luis Gravano
Thursday, March 10

<table>
<thead>
<tr>
<th>Problem</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>75</td>
<td></td>
</tr>
</tbody>
</table>
1. **(10 points)** In at most two short sentences each, explain the meaning of the following terms as they relate to database systems:

(a) SQL aggregate operators.
An aggregate is an operation in SQL (e.g., MIN, AVG, etc.) for combining numerical values.

(b) Renaming operator.
A relational algebra operator that allows us to assign names to relations and their attributes.

(c) Outerjoin.
A variant of natural join in which tuples from one relation that do not match any tuples of the other relation can still appear in the query result, with missing attribute values padded with NULLs.

(d) Identifying relationship set.
A relationship set that helps define a weak entity set in an ER diagram.

(e) Assertion.
A powerful SQL construct to define arbitrary constraints in a database.
2. Consider the following E/R diagram, representing students and their computer accounts at a university, as well as the sponsor of each account:

(a) (6 points) For the next 3 items, you will get 2 points for each correct answer, -1 points for each incorrect answer, and 0 points for each answer left blank.

Note that the line that connects Students to Owns and the line that connects Accounts to Owns are both bold and that both have arrowheads. The line that connects Sponsors to Owns is not bold and it does not have an arrowhead.

i. According to the diagram, can a student change the login of his or her account? Circle one: YES

ii. According to the diagram, can an account have two sponsors? Circle one: NO

iii. According to the diagram, is it true that a student can own two accounts as long as they have different sponsors? Circle one: NO
(b) **(12 points)** Consider the following SQL schema:

```sql
CREATE TABLE StudentsOwnsAccounts(
    sid INTEGER,
    sname CHAR(30),
    login CHAR(20),
    diskQuota INTEGER,
    pid INTEGER,
    PRIMARY KEY (sid),
    UNIQUE (pid),
    FOREIGN KEY (pid) REFERENCES Sponsors(pid));

CREATE TABLE Sponsors(
    pid INTEGER,
    pname CHAR(30),
    PRIMARY KEY (pid),
    FOREIGN KEY (pid) REFERENCES StudentsOwnsAccounts(pid));
```

List and briefly explain the four most important problems that you find with this schema, in terms of how well it models the E/R diagram above. **Do not base your answer on comparing this schema with other possible schemas.** Instead, just compare the schema against the E/R diagram to identify what is not captured from the diagram, etc. in the schema. You will be graded on the importance of the problems that you identify.

| Problem 1: | The foreign key in Sponsors does not refer to the primary key of StudentsOwnsAccounts. |
| Problem 2: | The total participation of Students in Owns is not enforced in StudentsOwnsAccounts (login and pid could be NULL). |
| Problem 3: | A “relationship instance” for Owns might involve just two “entities” (login or pid might be NULL). |
| Problem 4: | The primary key constraint for Accounts is lost (i.e., multiple accounts might have the same login value). |
| Problem 5: | Each sponsor might sponsor at most one student-account pair (i.e., a spurious key constraint of Sponsors in Owns is introduced by the UNIQUE (pid) constraint in StudentsOwnsAccounts). |
3. **(17 points)** Consider the following database, consisting of two relations, whose schemas are:

```sql
CREATE TABLE Employees(
    ssn CHAR(11),
    name CHAR(30),
    salary REAL,
    PRIMARY KEY (ssn));

CREATE TABLE Manages(
    employee_ssn CHAR(11),
    manager_ssn CHAR(11),
    PRIMARY KEY (employee_ssn),
    FOREIGN KEY (employee_ssn) REFERENCES Employees(ssn),
    FOREIGN KEY (manager_ssn) REFERENCES Employees(ssn));
```

The `Manages` relation keeps track of the manager (`manager_ssn`) of each employee (`employee_ssn`), if any. Note that managers are also regular employees, as indicated in the last line of the `CREATE TABLE` statement for the `Manages` table.

Write a **SQL query** that returns the `ssn` and `name` of each manager together with the average salary of the employees that s/he manages.

```sql
SELECT M.manager_ssn, E2.name, AVG(E1.salary)
FROM Manages M, Employees E1, Employees E2
WHERE M.employee_ssn=E1.ssn AND M.manager_ssn=E2.ssn
GROUP BY M.manager_ssn, E2.name
```
4. **(15 points)** Consider the relational schema in Problem 3, with relations Employees and Manages. We are interested in detecting “cycles” in the management. For simplicity, we will focus on cycles of length two. Write a relational algebra expression that returns the ssn of each employee \( e \) such that \( e \) is the manager of \( e \)’s manager. (So suppose that there is an employee \( m \) whose manager is employee \( e \) and, additionally, that \( m \) is \( e \)’s manager; then your relational algebra expression should return \( e \)’s ssn as well as, of course, also \( m \)’s ssn.)

\[
\rho(M_1(employee \_ssn \rightarrow m_1 \_employee \_ssn, manager \_ssn \rightarrow m_1 \_manager \_ssn), Manages) \\
\rho(M_2(employee \_ssn \rightarrow m_2 \_employee \_ssn, manager \_ssn \rightarrow m_2 \_manager \_ssn), Manages) \\
\Pi_{m_1 \_employee \_ssn}(M_1 \land_{m_1 \_manager \_ssn=m_2 \_employee \_ssn/m_1 \_employee \_ssn=m_2 \_manager \_ssn} M_2)
\]
5. **(15 points)** On this problem, you will get 3 points for each correct answer, -1.5 points for each incorrect answer, and 0 points for each answer left blank.

Each row of the following table shows two queries. In the blank third column of the table write “YES” if the two queries are equivalent, and “NO” if they are not equivalent. Two queries are equivalent if they always return exactly the same answer on all databases with the schema below.

All queries refer to a schema containing two relations:

- \( R(A, B, C) \) where \( \{A, B\} \) is the primary key and \( C \) is a candidate key
- \( S(A, B, C) \) where \( A \) is the primary key (and only key)

You may assume that the relations do not contain *null* values, but do not make any other assumptions about the relations.

<table>
<thead>
<tr>
<th>Query 1</th>
<th>Query 2</th>
<th>Equivalent? (YES/NO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Pi_{A,B}(R) \bowtie \Pi_{A,B}(S) )</td>
<td>( \Pi_{A,B}(R \bowtie S) )</td>
<td>NO</td>
</tr>
<tr>
<td>( R - S )</td>
<td>( R - (R \bowtie S) )</td>
<td>YES</td>
</tr>
<tr>
<td>( (\Pi_{A,B}(R) \times \Pi_C(R)) \bowtie R )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \text{SELECT MAX}(R.A), R.B, R.C )</td>
<td>( \text{SELECT DISTINCT} R.A, R.B, R.C )</td>
<td>YES</td>
</tr>
<tr>
<td>\text{FROM } R</td>
<td>\text{FROM } R</td>
<td></td>
</tr>
<tr>
<td>\text{GROUP BY R.B, R.C}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \text{SELECT S.A, AVG(S.B), AVG(S.C)} )</td>
<td>( \text{SELECT S.A, AVG(S.B), AVG(S.C)} )</td>
<td>YES</td>
</tr>
<tr>
<td>\text{FROM } S</td>
<td>\text{FROM } S</td>
<td></td>
</tr>
<tr>
<td>\text{GROUP BY S.A}</td>
<td>\text{GROUP BY S.A}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>\text{HAVING COUNT(*)}=1</td>
<td></td>
</tr>
</tbody>
</table>