cs4111.001–Introduction to Databases, Section 1
Spring 2017
Midterm Exam
Closed Book and Notes
Duration: 75 minutes
Professor Luis Gravano
Thursday, March 9

<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) **Partial key.**
   A minimal set of attributes of a weak entity set that, together with the primary key attributes of the identifying owner entity set, uniquely identify the weak entities.

   (b) **Bad-style attribute-based CHECK constraint.**
   An attribute-based CHECK constraint that has a subquery (or, alternatively, that mentions other attributes or relations beyond the attribute to which it is attached).

   (c) **Trigger.**
   A mechanism that allows a procedure to be automatically invoked by the DBMS in response to specified changes to the database; a trigger generally consists of three parts: an event, a condition, and an action.

   (d) **Union-compatible relations.**
   Two relations are union compatible when (1) they have the same number of attributes and (2) each pair of “corresponding” attributes of the relations have the same domain.

   (e) **Nested query with correlation.**
   A subquery of a SQL query that mentions the “outer” query (i.e., that references elements of the containing query).
2. Consider the following E/R diagram, representing students and their computer accounts at a university:

(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.

i. According to the diagram, can a student not have a computer account at all? Circle one: NO

ii. According to the diagram, can two students own (i.e., share) the same account if they have different values of the since attribute? Circle one: NO

iii. According to the diagram, can two different accounts have the same value for the diskQuota attribute? Circle one: YES
Consider the following SQL schema:

```sql
CREATE TABLE Students (  
    sid INTEGER,  
    sname CHAR (30),  
    PRIMARY KEY (sid));

CREATE TABLE AccountsOwns(  
    login CHAR (20),  
    diskQuota INTEGER,  
    sid INTEGER,  
    since DATE,  
    PRIMARY KEY (login, sid));
```

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.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>Primary key constraint of Accounts is lost.</td>
</tr>
<tr>
<td>Problem 2</td>
<td>Total participation of Students in Owns is not modeled.</td>
</tr>
<tr>
<td>Problem 3</td>
<td>Key constraint of Students in Owns is not modeled.</td>
</tr>
<tr>
<td>Problem 4</td>
<td>Key constraint of Accounts in Owns is not modeled.</td>
</tr>
<tr>
<td>Problem 5</td>
<td>sid in AccountsOwns should be a FOREIGN KEY into Students to avoid ‘‘dangling’’ tuples.</td>
</tr>
</tbody>
</table>
3. (17 points) Consider the following database, consisting of these two relations:

```
CREATE TABLE Students(
    sid INTEGER,
    sname CHAR(30),
    PRIMARY KEY (sid));

CREATE TABLE Enrolled(
    sid INTEGER,
    cid CHAR(20),
    semester CHAR(11),
    grade REAL,
    PRIMARY KEY (sid, cid),
    FOREIGN KEY (sid) REFERENCES Students(sid),
    FOREIGN KEY (cid) REFERENCES Courses(cid));
```

(For brevity, we omit the schema of a third relation in the database, the Courses table, which you do not need to answer this question.) The Students relation stores each student’s id (sid) and name (sname). The Enrolled relation keeps track of each course (cid) in which each student (sid) was enrolled, together with the corresponding semester (semester) when the student took the course and the grade (grade, a real number between 0 and 4) that the student obtained in the course.

Write a SQL query that returns the sid and sname of each student S who satisfies all three conditions below:

(a) S took cs4111 (i.e., a course with cid=“cs4111”), and

(b) S’s grade for cs4111 was the highest grade among the students who took cs4111 that same semester, and

(c) S took cs6111 (i.e., a course with cid=“cs6111”).

**Note 1:** A student satisfies condition (b) even if the student shares the highest grade in cs4111 that semester with other students who took the class at the same time.

**Note 2:** You cannot assume that each class has at least one student with grade 4.0. In other words, the highest grade in a class might well be lower than 4.0.

```
SELECT S.sid, S.sname
FROM Students S, Enrolled E1, Enrolled E2
WHERE S.sid=E1.sid AND E1.cid='cs4111' AND
    S.sid=E2.sid AND E2.cid='cs6111' AND
    E1.grade = (SELECT MAX(E3.grade)
        FROM Enrolled E3
        WHERE E3.cid='cs4111' AND
            E3.semester=E1.semester)
```
4. (15 points) The antisemijoin of relations \( R \) and \( S \), denoted \( R\bowtie S \), is the set of tuples in \( R \) that do not agree with any tuple of \( S \) in the attributes common to \( R \) and \( S \). (Note that this definition is asymmetric, and that the result is a (not necessarily strict) subset of relation \( R \).) Consider a relation \( R \) with attributes \( A \) and \( B \), and a relation \( S \) with attributes \( B \) and \( C \). As an example, consider the following instances of \( R \) and \( S \): \( R \): \[
\begin{array}{cc}
A & B \\
1 & x \\
2 & y
\end{array}
\] and \( S \): \[
\begin{array}{cc}
B & C \\
x & I \\
z & II
\end{array}
\]. Then, for these instances, \( R\bowtie S \) is:
\[
\begin{array}{cc}
A & B \\
2 & y
\end{array}
\]

Give a relational algebra expression that is equivalent to \( R\bowtie S \). You can assume for your answer that \( R \) has attributes \( A \) and \( B \), and \( S \) has attributes \( B \) and \( C \), just as in the example above.

\[
R\bowtie S = R - \Pi_{A,B}(R \bowtie S)
\]
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.

All queries refer to a schema containing two relations:

- \( R(A, B) \) where \( A \) is the primary key and \( B \) is a candidate key
- \( S(A, B) \) 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(R \bowtie S) )</td>
<td>( \Pi_A(R) \bowtie \Pi_A(S) )</td>
<td>NO</td>
</tr>
<tr>
<td>( R \bowtie (R \cap S) )</td>
<td>( R \bowtie S )</td>
<td>YES</td>
</tr>
<tr>
<td>( \Pi_B(R \bowtie S) )</td>
<td>( \Pi_B(R \bowtie \Pi_A(R \cap S)) )</td>
<td>YES</td>
</tr>
</tbody>
</table>