cs4111–Introduction to Databases
Fall 2010
Midterm Exam
Closed Book and Notes
Duration: 75 minutes
Professor Luis Gravano
Tuesday, October 19

<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><strong>Total</strong></td>
<td><strong>75</strong></td>
<td><strong>61</strong></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) Correlated nested query.
A query that both (1) is inside another query and (2) includes references to the outer query (e.g., by referring to attributes of the outer query).

(b) “Good-style” attribute-based check constraint.
A constraint associated and declared with an attribute of a relation (i.e., an attribute-based CHECK constraint) such that it does not include any subqueries.

(c) Partial key.
The set of attributes of a weak entity set that uniquely identify a weak entity for a given owner entity.

(d) Descriptive attribute.
An attribute of a relationship set, used to record information about the relationship.

(e) 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.
2. Consider the following E/R diagram, representing customers, their accounts at a bank, and the checks that are issued from 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 Checks to Belongs-To, the line that connects Accounts to Owns, and the line that connects Customers to Owns are all bold and that all three have arrowheads. The box around Checks and the diamond around Belongs-To are both bold as well. The line that connects Accounts to Belongs-To is not bold and does not have an arrowhead.

i. According to the diagram, can two different checks share the same value of both the number and the amount attributes? Circle one: YES

ii. According to the diagram, is it true that an account might have no checks? Circle one: YES

iii. According to the diagram, is it true that a customer can own two accounts if the accounts have different associated values of the account_type attribute? Circle one: NO
Consider the following SQL schema:

```sql
CREATE TABLE ChecksBelongsTo (  
    number INTEGER,  
    amount REAL,  
    date_issued DATE,  
    aid CHAR(30),  
    PRIMARY KEY (number, aid),  
    FOREIGN KEY (aid) REFERENCES AccountsOwns(aid));

CREATE TABLE AccountsOwns (  
    aid CHAR(30),  
    balance REAL,  
    account_type CHAR(20),  
    ssn CHAR(11),  
    PRIMARY KEY (aid),  
    UNIQUE (aid, ssn),  
    FOREIGN KEY (ssn) REFERENCES CustomersOwns(ssn));

CREATE TABLE CustomersOwns (  
    ssn CHAR(11),  
    name CHAR(30),  
    account_type CHAR(20),  
    aid CHAR(30),  
    PRIMARY KEY (ssn),  
    FOREIGN KEY (aid) REFERENCES AccountsOwns(aid));
```

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: | A Check weak entity does not get automatically deleted if its owning entity is deleted. |
| Problem 2: | The total participation of Accounts in Owns is not enforced in AccountsOwns (ssn could be NULL). |
| Problem 3: | The total participation of Customers in Owns is not enforced in CustomersOwns (aid could be NULL). |
| Problem 4: | AccountsOwns can cause the violation of the key constraint of Customers in Owns: the same ssn can be associated with multiple Accounts in AccountsOwns. |
| Problem 5: | CustomersOwns can cause the violation of the key constraint of Accounts in Owns: the same aid can be associated with multiple Customers in CustomersOwns. |
| Problem 6: | A relationship instance of Owns, involving one Account and one Customer, might have two different values of the account_type descriptive attribute, one in AccountsOwns and another in CustomersOwns. |
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),
    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.

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

(a) S took cs4111 (i.e., a course with cid = “cs4111”), and
(b) No more than 100 students took cs4111 in the same semester in which S took this class.

Here are some of the possible correct answers:

```
SELECT S.sid, S.sname
FROM Students S, Enrolled E
WHERE S.sid=E.sid AND E.cid='cs4111' AND
100>=(SELECT COUNT(*)
    FROM Enrolled E1
    WHERE E1.cid='cs4111' AND E1.semester=E.semester)
```

```
SELECT S.sid, S.sname
FROM Students S, Enrolled E
WHERE S.sid=E.sid AND E.cid='cs4111' AND
E.cid IN (SELECT E2.cid
    FROM Enrolled E2
    WHERE E2.semester=E.semester
    GROUP by E2.cid
    HAVING COUNT(*)<=100)
```

```
SELECT S.sid, S.sname
FROM Students S, Enrolled E
WHERE S.sid=E.sid AND
(E.cid, E.semester) IN (SELECT E2.cid, E2.semester
    FROM Enrolled E2
    WHERE E2.cid='cs4111'
    GROUP BY E2.cid, E2.semester
    HAVING COUNT(*)<=100)
```
4. **(15 points)** The *full outerjoin* of relations $R$ and $S$, denoted $R \bowtie S$, is a variation of the natural join $R \bowt S$ whose result is logically specified as follows:

- Start with the results of the natural join $R \bowt S$.
- Take all tuples of $R$ that did not match with any tuple of $S$, pad the tuples with a special value `null` for all “missing” attributes from $S$, and add the resulting tuples to the result of $R \bowtie S$.
- Take all tuples of $S$ that did not match with any tuple of $R$, pad the tuples with a special value `null` for all “missing” attributes from $R$, and add the resulting tuples to the result of $R \bowtie S$.

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:

$$\begin{array}{ccc} A & B & C \\ 1 & x & I \\ 2 & y & null \\ null & z & II \end{array}$$

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. Also, for your solution you can assume that you can use a *constant* table $N$ with a single attribute $C$ and a single tuple `(null)`: $N$: $\begin{array}{c} C \\ null \end{array}$.

$$R \bowtie S = (R \bowt S) \cup 
((R - \Pi_{A,B}(R \bowt S)) \times N) \cup
(\rho((1 \rightarrow A), N) \times (S - \Pi_{B,C}(R \bowt S)))$$

Note that there is an alternative answer that is also correct and does not use the renaming operator on $N$; this operator is not strictly necessary, because attribute names do not matter to determine when two relations are *union compatible*, and the union of two relations inherits the attribute names from the first relation (see textbook discussion on page 104):

$$R \bowtie S = (R \bowt S) \cup 
((R - \Pi_{A,B}(R \bowt S)) \times N) \cup
(N \times (S - \Pi_{B,C}(R \bowt S)))$$
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>$\sigma_{A=5}(\Pi_{A}(R))$</td>
<td>$\Pi_{A}(\sigma_{A=5}(R))$</td>
<td>YES</td>
</tr>
<tr>
<td>$\sigma_{A=5}(\Pi_{A}(R))$</td>
<td>$\sigma_{A=5}(\Pi_{A}(R \bowtie S))$</td>
<td>NO</td>
</tr>
<tr>
<td>$R \bowtie \Pi_{A}(S)$</td>
<td>$R \bowtie (\Pi_{A}(S) \times \Pi_{B}(R))$</td>
<td>YES</td>
</tr>
<tr>
<td>SELECT DISTINCT R.A, R.B</td>
<td>SELECT DISTINCT R.A, R.B</td>
<td>NO</td>
</tr>
<tr>
<td>FROM R, S</td>
<td>FROM R</td>
<td>(Think of the case $S=\emptyset$.)</td>
</tr>
<tr>
<td>SELECT R.A, S.B</td>
<td>SELECT DISTINCT R.A, S.B</td>
<td>YES</td>
</tr>
<tr>
<td>FROM R, S</td>
<td>FROM R, S</td>
<td></td>
</tr>
<tr>
<td>WHERE R.A=S.A</td>
<td>WHERE R.A=S.A</td>
<td></td>
</tr>
</tbody>
</table>