cs4111–Introduction to Databases
Fall 2010
Midterm Exam

Closed Book and Notes

Duration: 75 minutes

Professor Luis Gravano

Tuesday, October 19

Your Name	

Problem	Points	Score
1	10	
2	18	
3	17	
4	15	
5	15	
Total	75	

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.

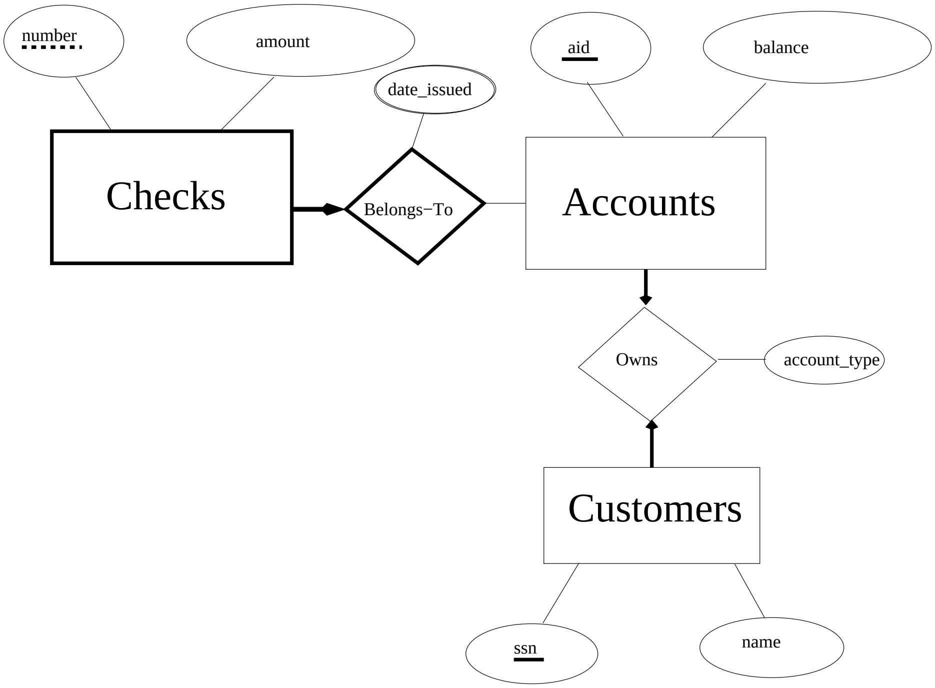
   (b) "Good-style" attribute-based check constraint.

   (c) Partial key.

   (d) Descriptive attribute.

   (e) Trigger.

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 NO

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

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: YES NO

(b) **(12 points)** Consider the following SQL schema:

```
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 <u>not</u> 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:	
Problem 2:	
Problem 3:	
Problem 4:	

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.

4. **(15 points)** The *full outerjoin* of relations R and S, denoted R ⟗ S, is a variation of the natural join $R \bowtie S$ whose result is logically specified as follows:

- Start with the results of the natural join $R \bowtie S$.
- 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 ⟗ 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 ⟗ 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: and S: . Then, for these instances, R ⟗ S is: . Give a **relational algebra** expression that is equivalent to R ⟗ 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: .

A	B
1	x
2	y

B	C
x	I
z	II

A	B	C
1	x	I
2	y	*null*
null	z	II

C
null

R ⟗ 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.

Query 1	**Query 2**	**Equivalent? (YES/NO)**
$\sigma_{A=5}(\Pi_A(R))$	$\Pi_A(\sigma_{A=5}(R))$	
$\sigma_{A=5}(\Pi_A(R))$	$\sigma_{A=5}(\Pi_A(R \bowtie S))$	
$R \bowtie \Pi_A(S)$	$R \bowtie (\Pi_A(S) \times \Pi_B(R))$	

`SELECT DISTINCT R.A, R.B`
`FROM R, S`
	`SELECT DISTINCT R.A, R.B`
	`FROM R`	

`SELECT R.A, S.B`
`FROM R, S`
`WHERE R.A=S.A`
	`SELECT DISTINCT R.A, S.B`
	`FROM R, S`
	`WHERE R.A=S.A`	