CIS 560, Fall 2011 **Name:_____**
Exam 1 – 75 minutes

This test consists of questions in five categories. The number of points for each question is shown below.

- Read all questions carefully before starting to answer them.
- Write all your answers on the space provided in the exam paper.
- The order of the questions is arbitrary, so the difficulty may vary from question to question. Don't get stuck by insisting on doing them in order.
- Show your work. Correct answers without justification will not receive full credit. However, also be concise. Excessively verbose answers may be penalized.
- Clearly state any simplifying assumptions you may make when answering a questions.
- **Be sure to write your name on the test paper.**

Question	1	2	3	4	5	total
Points	10	20	20	30	20	100
Your points						

I. True/False Questions [10 questions: you get 1 point for each correct answer; you lose 0.5 points for each incorrect answer; you get 0 points for questions that you don't answer]:

True **False**

A relation R that never contains more than one tuple is in Boyce-Codd Normal Form (BCNF).

True **False**

The SQL UNION operation eliminates duplicates.

True **False**

FALSE AND UNKNOWN = FALSE.

True **False**

A relation can only have one key, but may have multiple primary keys.

True **False**

In a relation R(X,Y,Z,T), if XY →YZ then X→Z.

True **False**

Schema refinement using the 3rd normal form (3NF) is dependency preserving.

True **False**

The NULL value approach in translating subclass relationship in the E/R model always saves space.

True **False**

In the WHERE clause of an SQL query, the condition 'Smith' = NULL is evaluated to be FALSE.

True **False**

Let's say you examined all the tuples in a relation at a particular time, and found that no two tuples have the same value for a particular attribute A. You can designate A to be a key for the relation.

True **False**

If we define a foreign key in relation R, the DBMS checks the foreign-key constraint whenever a tuple in R is deleted.

II. SQL Queries [20 points]

Consider a database with the following schema. This database records information about chefs (including their name and their rating), and the dishes they prepare. For each dish, the database records its name and its popularity score.

```
Chef(cid, name, rating)          /* cid is a chef's unique identifier */
Dish(did, name, popularity)      /* did is a dish's unique identifier */
Prepares(cid, did)               /* cid references Chef.cid and did references Dish.did */
```

Sample solutions for SQL queries shown below (some of them are solutions I selected from your exams). Other solutions are possible.

a. [5 points] Write an SQL query that lists the names of the dishes prepared by only one chef. Your query should return a list of dish names ordered alphabetically.

```
SELECT    d.name
FROM dish d, prepares p
WHERE     d.did=p.did
GROUP BY d.did, d.name
HAVING    COUNT(p.cid)=1
ORDER BY d.name
```

```
SELECT d.name
FROM dish d
WHERE
  (SELECT COUNT(cid) FROM prepares WHERE d.did=prepares.did) = 1
ORDER BY name;
```

```
SELECT a.d
FROM (SELECT count(prepares.did) as c, dish.name as d
      FROM  dish, prepares
      WHERE dish.did=prepares.did
      GROUP BY dish.did) as a
WHERE a.c = 1
OREDER BY name;
```

```
SELECT d.name
FROM dish AS d
WHERE d.did IN (SELECT p.did FROM prepares p
                GROUP BY p.did HAVING COUNT(*)=1)
ORDER BY d.name
```

b. [7 points] Write an SQL query that finds the names of the chefs with above average rating and the names of the dishes with above average popularity that they prepare. Your query should return pairs of highly-rated chef name and popular dish name.

```
SELECT  c.name, d.name
FROM    chef c, dish d, prepares p
WHERE  p.cid = c.cid AND p.did = d.did
       AND c.rating > (SELECT AVG(c2.rating) FROM chef c2)
       AND d.popularity > (SELECT AVG(d2.popularity) FROM dish d2)
```

```
SELECT  c.name, d.name
FROM    chef c, dish d, prepares p
WHERE  p.cid = c.cid AND p.did = d.did
       AND c.rating > (SELECT SUM(rating)/COUNT(cid) FROM chef)
   AND d.popularity > (SELECT SUM(popularity)/COUNT(did) FROM dish)
```

```
select chef.name, dish.name
from chef, dish, prepares,
    (select AVG(rating) as x1 from chef) as    x,
    (select AVG(popularity) as y1 from dish) as y
where chef.cid=prepares.cid and prepares.did=dish.did
      and chef.rating>x1 and dish.popularity>y1
```

c. [8 points] Write an SQL query that finds, for each chef, the most popular dish that he or she prepares. Your query should output the chef's name along with the name of his/her most popular dish. If there is more than one most popular dish, your query should output them all.

```
SELECT   c.name, d.name
FROM     chef c, dish d, prepares p
WHERE    p.cid = c.cid AND p.did = d.did AND
         d.popularity >= ALL (SELECT d2.popularity
                              FROM dish d2, prepares p2
                              WHERE p2.cid = p.cid AND p2.did=d2.did)
```

```
SELECT     c.name, d.name
FROM  chef c, dish d, prepares p
WHERE      p.cid = c.cid AND p.did = d.did AND
           d.popularity = (SELECT MAX(d2.popularity)
                           FROM dish d2, prepares p2
                           WHERE p2.cid = p.cid AND p2.did=d2.did)
```

```
select chef.name, dish.name
from (select prepares.cid as cid, MAX(dish.popularity) as maxpop
from prepares, dish
where prepares.did=dish.did
group by prepares.cid) as subq,
chef,dish,prepares
where chef.cid=subq.cid and prepares.cid=chef.cid and
prepares.did=dish.did and dish.popularity=subq.maxpop
```

III. Constraints, Triggers, Assertions (20 points)

a) [5 points] What is referential integrity? Explain using an example.

If a table R contains a foreign key referencing another relation T, referential integrity requires every such reference to be valid, i.e., the entry being referenced must actually

exist in T. For example, given an Accounts table that references a Customers table through customerSSN, then for each customerSSN in the Accounts table, there must be a tuple in the Customers table with that SSN.

b) [5 points] Explain "**on delete set null**" in the context of referential integrity constraints.

In the above example, if a customer row is deleted, the corresponding accounts will have their customer SSN set to null.

c) [5 points] Describe a situation where an attribute-based constraint with the keyword CHECK is violated.

This question was meant to ask about situations when the DB is in a state where the CHECK constraint is violated. The answer to this question is below. However, most students thought of situations where the check condition is violated on an insert or update, and as a consequence the insert/update is rejected. Both types of solutions received full credit.

CHECK is not checked if the database modification does not change the attribute with which the constraint is associated, e.g. foreign key constraint. Suppose we have two tables Department and Employees, and DepartmentName is a foreign key in Employees. Modifications on the DepartmentName of the Department table are not reflected in the Employees table.

See also page 12 in lecture 6.
Another example is shown in Example 7.7 (page 321 in the textbook).

d) [5 points] Explain the main reason that we sometimes prefer to use triggers rather than assertions.

Triggers allow users to specify when the check occurs, whereas assertions are checked for every INSERT/DELETE/UPDATE.

IV. E/R Diagram, Relational Schema, Basic SQL Statements [30 points]

a) [2 points] Explain the difference between a weak and a strong entity set.

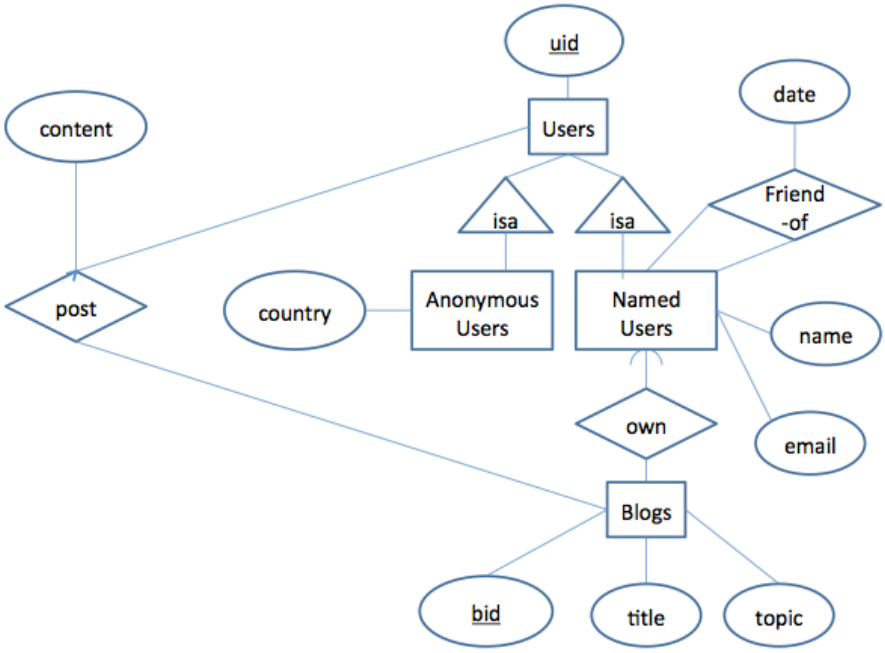
A strong entity set has a primary key. All tuples in the set are distinguishable by that key. A weak entity set has no primary key unless attributes of the strong entity set on which it depends are included.

b) [10 points] Draw an E/R diagram describing the following blog domain:

- Users have a single attribute uid.
- AnonymousUsers are a type of Users with attribute country.
- NamedUsers are a type of Users with attributes name and email.
- Blogs have attributes bid, title, and topic.
- A NamedUser can be a friend-of zero or more other NamedUsers. Each friend-of relationship has an associated start date.
- A NamedUser can own many Blogs, but each blog is owned by exactly one user.

- A User can post to zero or more Blogs. Each post has a given content. Many users can post to a blog.

Your answer should consist of an E/R diagram, which includes entity sets, attributes and relationships. If you feel that you must make any additional assumptions, please state them clearly in your solution. Remember to indicate the key for each entity set, as well as the multiplicity of each relationship using the appropriate notation.



c) [10 points] Translate your E/R diagram to a relational schema. Minimize the number of relations your solution has; merge relations where appropriate. Specify the keys and foreign keys of each relation in your schema. In translating a subclass hierarchy, use the E/R style translation.

Users(uid)

AnonymousUsers(uid, country)
AnonymousUsers.uid is foreign key referencing Users.uid

NamedUsers(uid, name, email)
NamedUsers.uid is foreign key referencing Users.uid

Friends(uid1, uid2, date)
uid1 and uid2 are foreign keys referencing Users.uid

Blogs(bid, title, topic, uid)
Blogs.uid is foreign key referencing Users.uid

Post(uid,bid,content)
Post.uid is foreign key referencing Users.uid and Post.bid is foreign key referencing Blogs.bid

Note: We can see that Users table only holds the uid and not other information => we can drop that table.

d) [5 points] Choose ONE relation from your relational schema and write the SQL statement to create the corresponding table. (You may make any reasonable choice of data types. Remember to include any constraints that follow from the description of the schema or your E/R diagram, including primary key and foreign key constraints.)

Answers will vary based on the relation you choose. An example:

```
CREATE Table Blogs(
    bid INT PRIMARY KEY,
    title VARCHAR (50),
    topic VARCHAR (25),
    uid INT REFERENCES Users(uid));
```

e) [3 points] You have just registered as a NamedUser. Write an SQL statement to add yourself to the database.

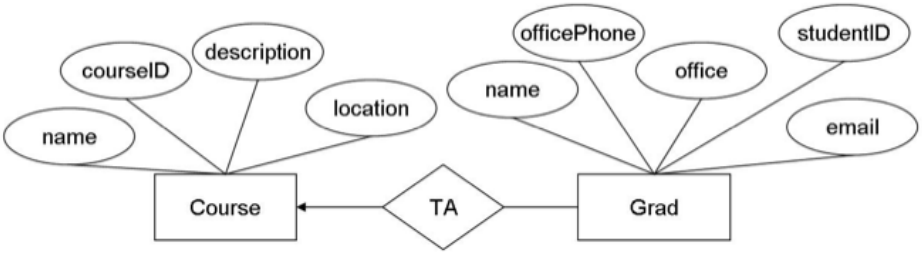
Given that we are using the E/R style to convert a subclass hierarchy, we first need to insert ourselves into Users and then into NamedUsers. I could add myself to the database using the following two statements:

```
INSERT INTO Users Values (1579);
INSERT INTO NamedUsers Values(1579, 'Doina', 'dcaragea@ksu.edu');
```

Note: If you drop the Users table in point c), then you don't need to insert into Users first.

V. Functional Dependencies, Keys and Normalization [20 points]

a) [4 points] Consider the following ER diagram, which describes graduate students (Grad) and courses (Course) they serve as TAs.



For each of the following statements, write a functional dependency that best captures the statement.

(FD1) The studentID of each graduate student uniquely identifies the student.

studentID -> name, email, office, officePhone

Note: You receive full credit if the attributes from the Course relation are included on the right hand side (with studentID on the left hand side), due to the many to one relationship between Grad and Course.

(FD2) No two offices have the same phone number (officePhone).

officePhone -> office

(FD3) No two courses have the same courseID.

courseID -> name, description, location

(FD4) If two courses have the same course name, their course descriptions are the same.

name -> description

b) [3 points] Using the definition of functional dependencies, prove that any two-attribute relation is in BCNF.

See Example 3.17 in the Textbook (page 89).

c) [3 points] Give a simple example illustrating that a dependency-preserving decomposition into BCNF may not always be possible.

Consider the relation R(A,B,C) with the FDs A->B and C->B. We have A+=AB, so we can decompose R based on A into R1(A,B) and R2(A,C). The FD C->B is not preserved in either of the decomposed relations.

See also example on page 7 in Lecture 10.

d) [3 points] Consider the relation R(A, B, C, D, E, F), with the following functional dependencies:

AB → C, C → D, F → B, D → A.

List all the keys that contain F. Do not list superkeys that are not (minimal) keys.

AEF, CEF, DEF

e) [7 points] Decompose the above relation R into a collection of Boyce-Codd Normal Form (BCNF) relations. Show your intermediate steps; for each step show the relation that you are decomposing and the violation of BCNF that you are using during that decomposition step. Indicate clearly your final result: the relations, their attributed and their keys. Does your decomposition preserve functional dependencies?

Some possible solutions – there are more.

Solution 1:

AB+=ABCD=/=ABCDEF

Break R(ABCDEF) into R1(ABCD) and R2(ABEF).

For R1(ABCD), C+=ACD =/=ABCD, which means R1 is not in BCNF.

Decompose R1(ABCD) into R11(ACD) and R12(CB).

R12(CB) is in BCNF.

For R11(ACD), we have D+=AD=/=ACD, which means R12(ACD) can be decomposed further into R111(AD) and R112(DC).

For R2(ABEF), we have F+=BF, which means R2 is not in BCNF. We can decompose R2 into R21(BF) and R22(FAE).

Thus, the final relations are R12(CB), R111(AD), R112(DC), R21(BF), R22(FAE), with the keys underlined.

The FD AB->C is lost, so this decomposition does not preserve all the FDs.

Solution 2:

C+=CDA, so we can decompose R into R1(CDA) and R2(CBEF).

D+=DA, so we can further decompose R1 into R11(DA) and R12(DC).

F+=FB, so we can further decompose R2 into R21(FB) and R22(FCE)

The final relations are R11(DA), R12(DC), R21(FB), R22(FCE), with the keys underlined.

The FD AB->C is lost, so this decomposition does not preserve all the FDs.

Solution 3:

F+=FB, so we can decompose R into R1(FB) and R2(ACDEF).

D+=DA, so we can decompose R2 into R21(AD) and R22(CDEF).

C+=CD, so we can decompose R221(CD) and R222(CEF).

The final relations are R1(FB), R21(AD), R221(CD), R222(CEF).

The FD AB->C is lost, so this decomposition does not preserve all the FDs.