Virginia Tech. **CS 4604 – Introduction to DBMS**
Computer Science **Spring 2015, Prakash**

Homework 2: E/R Models and More SQL

(due February 23rd, 2015, 4:00pm, in class—hard-copy please)

Reminders:

a. Out of 100 points. Contains 4 pages.
b. Rough time-estimates: 4~6 hours.
c. Please type your answers. Illegible handwriting may get no points, at the discretion of the grader. Only drawings may be hand-drawn, as long as they are neat and legible.
d. There could be more than one correct answer. We shall accept them all.
e. Whenever you are making an assumption, please state it clearly.
f. Unless otherwise mentioned, you may use any SQL/RA operator seen in class/in textbook.
g. Unless otherwise specified, assume set-semantics for RA and bag-semantics for SQL.
h. Feel free to use the linear notation for RA and create intermediate views for SQL.
i. **Important:**
   a. For E/R diagrams, use only the style/notation/material given in the lecture slides.
   b. A useful tool for creating E/R diagrams: http://logicnet.dk/DiagramDesigner/. You may have to manually draw-in some things though (like adding proper constraints etc.). There are other such good programs/sites too (like http://creately.com/).
j. Lead TA for this HW: Elaheh Raisi.

Q1. Life without HAVING [20 points]

You are given the following relations:

Take(StudentID, CourseID)
RequiredForGraduation(CourseID)

The Take relation lists IDs of students and IDs of courses taken by the students. (Note that in this problem, we are assuming that each course has a unique ID, as opposed to the scheme we used in class.) The RequiredForGraduation relation lists the courses every student must take to graduate. The following query finds the students who have satisfied all the requirements for graduation.

```
SELECT StudentId
FROM Take AS T, RequiredForGraduation AS R
WHERE T.CourseID = R.CourseID
GROUP BY T.StudentID
HAVING COUNT(T.CourseId) =
        (SELECT COUNT(CourseId) FROM RequiredForGraduation);
```

You hate the HAVING clause and do not see the point of views. Rewrite this query without using views or the HAVING clause.

Q2. Where Art thou? [15 points]

Consider the situation in Exercise 2.8 in your textbook. We repeat it here for your convenience. Although you always wanted to be an artist, you ended up being an expert on databases because you love to cook data and you somehow confused "database" with "data baste". Your old love is still there, however, so you set up a database company, ArtBase that builds a product for art galleries. The core of this product is a database with a schema that captures all the information that galleries need to maintain. Galleries keep information about artists, their names (which are unique), birthplaces, age, and style of art. For each piece of artwork, the artist, the year it was made, its unique title, its type of art (e.g., painting, lithograph, sculpture, photograph), and its price must be stored. Pieces of artwork are also classified into groups of various kinds, for example, portraits, still-lifes, works by Picasso, or works of the 19th century; a given piece may belong to more than one group. Each group is identified by a name (like those just given) that describes the group. Finally, galleries keep information about customers. For each customer, galleries keep that person's unique name, address, total amount of dollars spent in gallery (very important!), and the artists and groups of art that the customer tends to like.

The ER model your DB engineer designed was lost, and he has resigned. So you need to step in:

Q2.1. (10 points) Draw an ER diagram for this database. Make sure to indicate primary keys, cardinality constraints, weak entities (if any), and participation constraints. List any assumptions you make in the process.

Q2.2. (5 points) Translate the ER diagram in Q2.1 into relational database tables (i.e. give the SQL DDL statements). Make sure that the translation captures key constraints (primary keys and foreign keys if applicable) and participation constraints in the ER diagram. Identify constraints, if any, that you are not able to capture.

Q3. Bank DB [20 points]

We want to design a database scheme for consumer banking. A Bank's routing number should be unique. All banks should specify whether they are FDIC members or not. Customers can open different accounts in several banks. Every customer has a unique Social Security ID. Date of birth, name and address of customers are also needed. We assume there are no joint accounts (each account is associated with one customer). Every account is identified with an account number, account type (e.g., savings/checking) and account balance. The banks also support transaction operations on accounts. A transaction can be a withdrawal or a deposit. Transactions are specified with a transaction number, date and amount. In addition, customers can get loans from several banks; a loan has its amount, interest and date. Draw an E/R diagram modeling this scenario. List any assumptions you make in the process.

Q4. Insurance Database [25 points]

We are asked to design a database management system for all information related to VT-Insurance, a life and vehicle insurance company at Virginia Tech. The first step is to organize the information given about insurance. We have collected the following data:

- Every insurance company has its own policies. A Policy has policy_number, term_price, coverage and date.
- There are two types of policies: vehicle_policy, life_policy.
- For life_policy, we have its ID, value, minimum age and maximum age.
- For vehicle_policy, we have its ID, vehicle VIN number, type (new/old), and the driver's driving history,
- Every insurance policy covers one or more vehicles/life and premium payments associated with it. Each payment has its id, due date, and payment amount.
- Vehicle supported by VT-Insurance are only cars or motorcycles.
- Every vehicle has its VIN number, plate, registered state, color, year and model.
- For motorcycles, we should keep track of its weight, and type (standard, cruiser, sport bike, sport touring, dual sport, scooters, etc.)
- For cars, we need to have car type (sedan, coupe, etc.), transmission type (automatic, manual), and size (subcompact, compact, midsize, large).
- For each customer, we have his/her SSN, driving id, name, telephone and address.
- When a vehicle accident happens, a report should be prepared including report number, date, location, and damage cost. Accidents are also associated with a customer; so, driving license should be mentioned in the report.

Please answer the following questions:

Q4.1. (15 points) Draw an ER diagram for this database. Make sure to indicate primary keys, cardinality constraints, weak entities (if any), and participation constraints. List any assumptions you make in the process.
Hint: You may need an ISA hierarchy somewhere.

Q4.2. (10 points) Translate the ER diagram in Q4.1 into relational database tables (i.e. give the SQL DDL statements). Make sure that the translation captures key constraints (primary keys and foreign keys if applicable) and participation constraints in the ER diagram. Identify constraints, if any, that you are not able to capture.

Q5. Permutations via SQL [20 points]

This is a tricky problem designed to help you think out of the box on the use of database programming for solving problems.

SQL is pretty good about letting you do cross products to get all possible pairs (x, y) from two sets of elements with a simple query, for example:

```
SELECT x, y
FROM BigX, BigY;
```

But sometimes you would like to do this sort of thing horizontally instead of vertically. A permutation is an ordered arrangement of elements of a set. For example, if we have the set {1, 2, 3}, the permutations of those elements are (1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), and (3, 2, 1). The rule is that for (n) elements, you have a factorial number (n!) of permutations.

What we would like is a SQL query that returns one permutation per row from a set of the first seven integers (that will give us 5,040 rows).

Assume you are given the following table Elements:

```
CREATE TABLE Elements
(i INTEGER PRIMARY KEY);

INSERT INTO Elements
VALUES (1), (2), (3), (4), (5), (6), (7);
```

Q5.1. (15 points) Write down your SQL query.
Hint: Maybe you can use some NOT INs.

Q5.2. (5 points) The solution you get for the problem when you use an SQL interpreter and RDBMS to solve this problem (e.g. you can use SQLite) i.e. run the create table/insert statements and then your SQL query from Q5.1 above. Copy-paste only the *first 10 rows* you get in the output.

Note: The SQL query may be quite long so you may find it useful to create the query in a text file and use the source command (or equivalent) in your SQL interpreter to read in and execute the query.