Homework 4: Database Design

Objectives: To translate from entity-relationship diagrams to a relational database, and to understand functional dependencies and normal forms.

Assignment tools:
- Pen and paper or any drawing tools you prefer (e.g., PowerPoint, draw.io).
- SQLite, Azure SQL Server, or any other relational database.

Assignment Details

Part 1: Geography E/R Diagram (10 points)

Design an E/R diagram for geography that contains the following kinds of objects or entities together with the listed attributes.

Model the relationships between the objects with edges. Note that edges between entities can be labeled with constraints. Make sure to choose the correct primary key(s) based on the information below.

Entities
- countries (with attributes): name, area, population, GDP ("gross domestic product")
  - a country's name uniquely identifies the country within all countries
- cities: name, population, longitude, latitude
  - a city is uniquely identified by its (longitude, latitude) (not by name, ex: there are 41 different cities and towns named Springfield in the US!)
- rivers: name, length
- seas: name, max depth
- rivers and seas are uniquely identified within the set of all water entities by their name (e.g., "Ganges" would be a unique water entity)

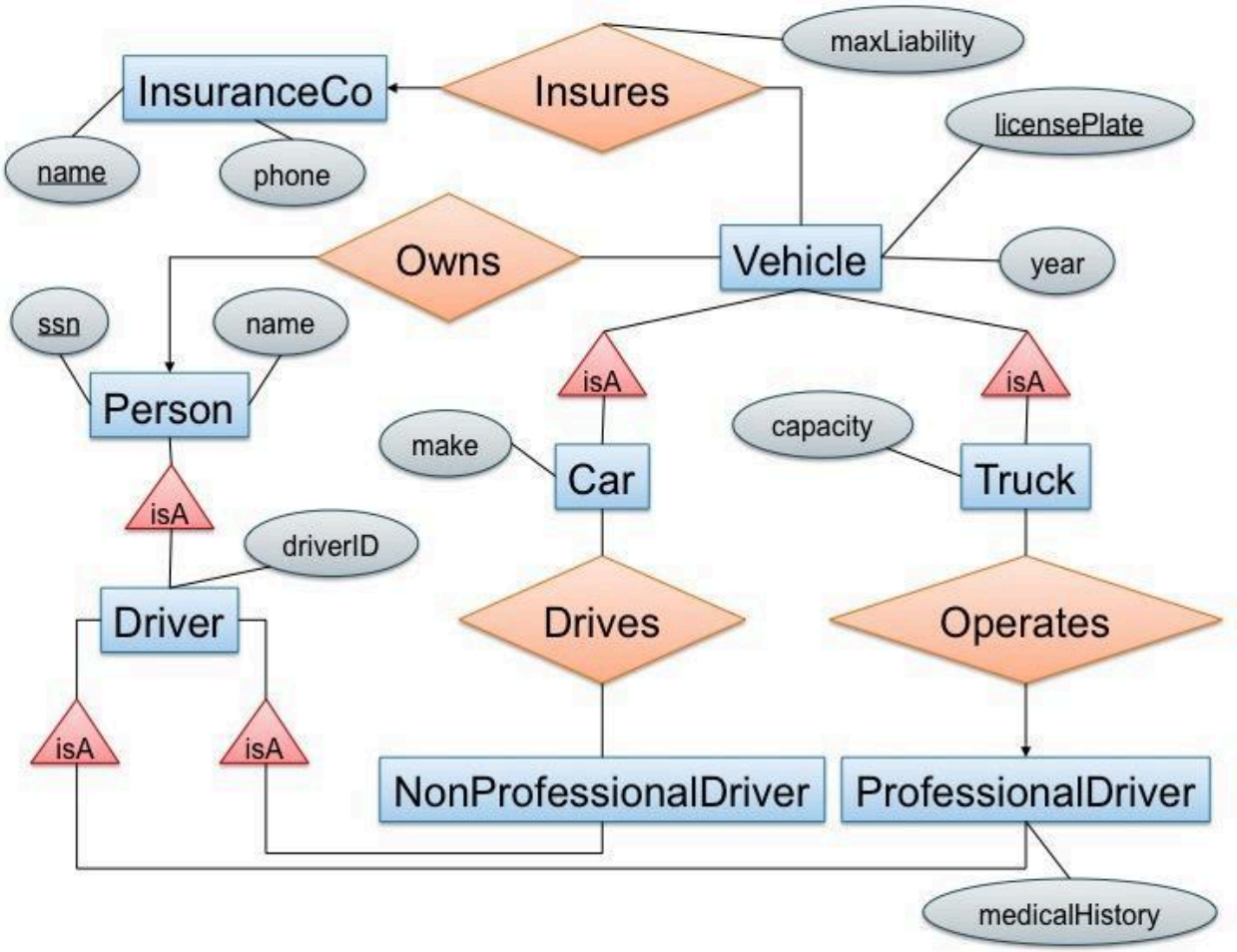
Relationships:
- each city belongs to exactly one country
- each river crosses one or several countries
- each country can be crossed by zero, one, or several rivers
- each river ends in either a river or a sea

You can draw your diagrams on paper and scan them, take *quality* pictures of your drawn diagram then use https://imagetopdf.com to convert your image to a pdf or use your favorite drawing tools such as Powerpoint, Keynote, or draw.io. (FYI: Google Slides lacks a few shapes that you might need such as rounded arrows... You can use a crescent and a line, or simply type a constraint label such as "=1".)

Name your file **geography.pdf**.

Part 2: E/R to Schema (20 points)

Consider the following E/R diagram:



- ***License plate*** **can have letters and numbers**
- ***driverID*** **and** ***Social Security*** **contain only numbers (integers)**
- ***maxLiability*** **is a real number**
- ***year***, ***phone***, ***capacity*** **are integers**
- **everything else is a text.**

a. (10 points) Translate the diagram above by writing the SQL CREATE TABLE statements to represent this E/R diagram. Include all key constraints; you should specify both primary and foreign keys. Make sure that your statements are syntactically correct (you might want to check them using SQLite, Azure SQL Server, or another relational database).

b. (5 points) Which relation in your relational schema represents the relationship "insures" in the E/R diagram? Why is that your representation?

c. (5 points) Compare the representation of the relationships "drives" and "operates" in your schema, and explain why they are different.

Write your answers in a SQL file named **driving.sql**. Write your answers to questions b and c as *SQL comments* in the same file.

Part 3: Functional Dependency Theory (35 points)

Consider the following two relational schemas and sets of functional dependencies:

a. (10 points) R(A,B,C,D,E) with functional dependencies D → B, CE → A.
b. (10 points) S(A,B,C,D,E) with functional dependencies A → E, BC → A, DE → B.

For each of the above schemas, decompose it into BCNF. Show all of your work and explain, at each step, which dependency violations you are correcting. Make sure you describe each step in your decomposition steps.

Consider a relation with schema R(A,B,C,D) and an unknown set of functional dependencies. For each closed attribute set below, give a *set of functional dependencies* that is consistent with it. (You don't have to specify all of the trivial dependencies that already exist like A -> A, AB -> AB, and so on. You are still welcome to specify them, if necessary.)

A set of attributes X is called "closed" (with respect to a given set of functional dependencies) if $X^+ = X$.

c. (5 points) All subsets of {A,B,C,D} are closed.
d. (5 points) The only closed subsets of {A,B,C,D} are {} and {A,B,C,D}.
e. (5 points) The only closed subsets of {A,B,C,D} are {}, {B,C}, and {A,B,C,D}.

Write your answers in a text file named **theory.txt**.

Part 4: Mr. Frumble Relationship Discovery & Normalization (35 points)

Mr. Frumble (a great character for small kids who always gets into trouble) designed a simple database to record projected monthly sales in his small store. He never took a database class, so he came up with the following schema:

Sales(name, discount, month, price)

He inserted his data into a database, then realized that there was something wrong with it: it was difficult to update. He hires you as a consultant to fix his data management problems. He gives you this file mrFrumbleData.txt and says: "Fix it for me!". Help him by normalizing his database.

Unfortunately, you cannot sit down and talk to Mr. Frumble to find out what functional dependencies make sense in his business. Instead, you will reverse engineer the functional dependencies from his data instance.

a. (2 points) Create a table in the database and load the data from the provided file into that table; use SQLite or any other relational DBMS of your choosing.
   Turn in your create-table statement. Turning in the data import statements is optional.

b. (10 points) Find all non-trivial functional dependencies in the database. This is a reverse engineering task, so expect to proceed in a trial-and-error fashion. Search first for the simple dependencies, say name → discount then try the more complex ones, like name, discount → month, as needed. To check each functional dependency you have to write a SQL query. Your challenge is to craft this SQL query for every candidate functional dependency that you check, such that:
   a. the query's answer is always short (say: no more than ten lines - remember that 0 results can be instructive as well)
   b. you can determine whether the FD holds or not by looking at the query's answer.

   Try to be clever in order not to check too many dependencies, but don't miss potential relevant dependencies. For example, if you have A → B and C → D, you do not need to check AC → BD as well.
   In your answers, provide a SQL query for each functional dependency you find along with a comment describing the functional dependency. Please also include a SQL query for at least one functional dependency that does not exist in the dataset along with a comment mentioning that the functional dependency does not exist. Remember, short SQL queries are preferred.

c. (11 points) Decompose the table into Boyce-Codd Normal Form (BCNF), then write the Create Table commands for the normalized schema. Create keys and foreign keys where appropriate. For this question turn in the SQL commands for creating the tables.

d. (12 points) Write SQL queries to populate your normalized database with the data from the table you created in 4.1. Comment the size of the tables after loading them (obtained by running SELECT COUNT(*) FROM Table).

Write your answers in a SQL file named **frumble.sql**. Use SQL comments for all non-SQL-code answers.

Submission Instructions

The files you will need to submit to Gradescope:

- geography.pdf
- driving.sql
- theory.txt
- frumble.sql

Points may be deducted for incorrect file names/types, or for .sql files that are not executable.

Submit your answers to Gradescope.