Homework 4 | Database Design

Updates made to the assignment after its release are highlighted in red.

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

Due date: Tuesday, Oct 24th @ 10:00pm

***Median completion time (23sp):* 5 hours**

Resources

For this assignment, you will need:

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

Problem Set

Part 2: Geography E/R Diagram (10 points)

Draw an E/R diagram for a mapping application. Your diagram must contain the following list of entities, along with their listed attributes. Your diagram should also contain the relationships described below, modeled as edges between entities; 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`: name, area, population, GDP ("gross domestic product")
  - a country's name uniquely identifies the country
- `cities`: name, population, longitude, latitude
  - a city is uniquely identified by its (longitude, latitude), not by name!
  - eg: 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 by their name *within the set of all water entities*

    - e.g., if there were a river named "Ganges" would be a unique water entity, no other river or sea could have that name

Relationships

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

If you choose to draw your diagrams on paper, take *high quality pictures* and then use https://imagetopdf.com to convert the image to a pdf.

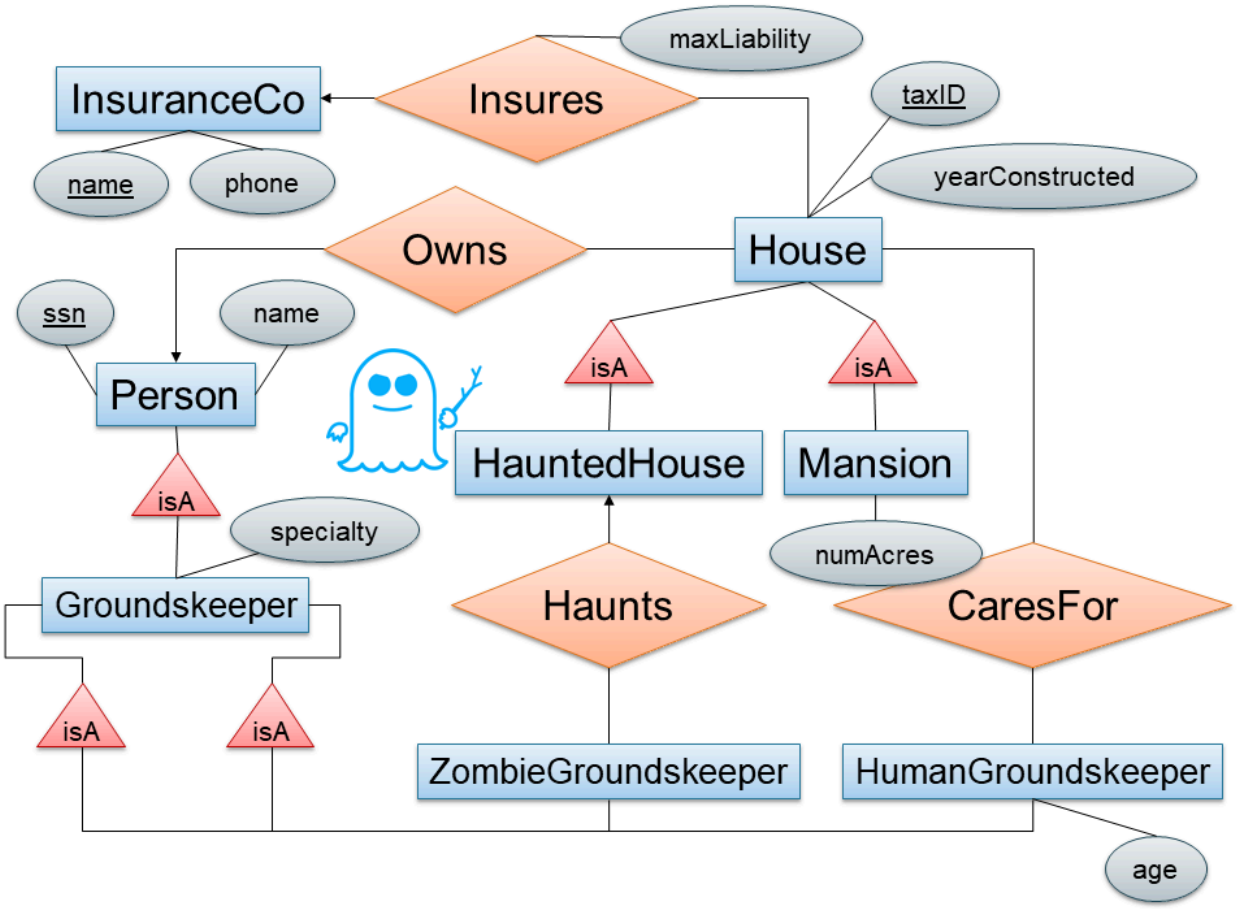
Alternatively, you can draw your diagrams using a favorite drawing tool, such as PowerPoint, Keynote, or draw.io. We do not recommend Google Slides because it lacks a few shapes that you might need (eg, rounded arrows), but you can work around those omissions if you are committed (eg, you can use a crescent and a line, or simply type a constraint label such as "=1".)

Name your file **geography.pdf** and upload to Gradescope.

Part 3: E/R to Schema (20 points)

Consider the following E/R diagram. All attributes in this diagram are alphabetic strings, with the following exceptions:

- `taxID` and `specialty` have letters and digits
- `ssn` only contains digits
    - In other words, can be represented as integers
- `maxLiability` is a real (floating point) number
- `yearConstructed`, `phone`, `numAcres`, `age` are integers



1. (10 points)
   Translate the above E/R diagram into a series of SQL `CREATE TABLE` statements, including all key constraints (ie, primary and foreign keys). We will run these statements, so ensure that the statements are syntactically correct (ie, check them using SQLite, SQLServer, or another relational database).

2. (5 points)
   Which relation in your schema represents the relationship `insures` in the E/R diagram? Why is that your representation?

3. (5 points)
   Compare the representation of the relationships `haunts` and `caresFor` in your schema, and explain why they are different.

~~*Submit your answers in a SQL file named* ***halloween.sql****; your answers to questions b and c should be SQL comments in the same file.*~~

Part 4: Functional Dependency Theory (35 points)

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

1. (10 points)
   R(A,B,C,D,E) with functional dependencies D → B, CE → A.

2. (10 points)
   S(A,B,C,D,E) with functional dependencies A → E, BC → A, DE → B.

Decompose both of the above schemas into BCNF. At each step of the decomposition, explain which bad FD you are correcting. For each table in your final answer, please note which (if any) of the original FDs still apply to that table.

A set of attributes X is called *closed* with respect to a given set of functional dependencies if $X^+ = X$. Consider a relation with schema **R**(A,B,C,D) and a to-be-determined set of functional dependencies. For the following scenarios, give a *set of functional dependencies* that is consistent with it.

3. (5 points)
   All subsets of {A,B,C,D} are closed.

4. (5 points)
   The only closed subsets of {A,B,C,D} are:
   - {}
   - {A,B,C,D}

5. (5 points)
   The only closed subsets of {A,B,C,D} are:
   - {}
   - {A,C}
   - {A,B,C,D}

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

Part 5: 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 the 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 quickly realized it was difficult to update. Fortunately, he has hired *you* to fix his data management problems! He gives you his file mrFrumbleData.csv and says: "Fix it for me!".

You decide to help Mr. Frumble by normalizing his database. Unfortunately, he has already dashed off (*hats don't chase themselves, doncha know?*) and you are unable to ask him what functional dependencies make sense in his business. Instead, you will reverse engineer them from his data instance.

1. Create a table in a DBMS of your choosing, and load Mr. Frumble's data into that table; use SQLite or any other relational DBMS of your choosing.

   Submit your `CREATE TABLE` statement. Do not include the data import statements.

2. Find all non-trivial functional dependencies in his schema.

   This is a *reverse engineering task*, so expect to proceed in a trial and error fashion. We recommend first searching for simple dependencies (eg, `name → discount`), and only searching for more-complex dependencies (eg, `name, discount → month`) as needed. For each candidate FD you consider, you will need to write a SQL query to verify whether it does or does not hold.

   A few notes:
   a. Your query's answer should always be short
      i. A good rule of thumb is no more than ten results
      ii. Remember that 0 results can be instructive as well
   b. You should be able to determine whether a candidate FD does or does not hold by examining the query's answer. Don't miss relevant dependencies but, if you can, try to minimize the number of queries you write.
      i. For example, if you have queries demonstrating `A → B` and `C → D`, you do not need to write a query for `AC → BD` (but don't forget to find all non-trivial FDs!).
   c. Write a SQL query for each functional dependency you find, along with a comment describing the FD.
   d. Also write 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.
   e. Not only should your query's results be short, the queries themselves should be short.

   Submit one SQL query for each FD you identify. For example: if you find 3 functional dependencies, submit 4 FDs and 4 queries. You should submit 3 queries showing that

*your 3 FDs hold over this instance, and a fourth FD along with a query demonstrating how it does **not** hold on this instance.*

3. Decompose Mr. Frumble's single table into Boyce-Codd Normal Form (BCNF), and create SQL tables for the decomposed schema. Create keys and foreign keys where appropriate.

   Submit the SQL commands for creating the tables.

4. Submit the SQL queries that *load data* into your new tables. In a SQL comment, count the size of *each of your new* tables after they've been loaded (ie, the result of `SELECT COUNT(*) FROM MyNewTable`).

~~*Submit your answers in a SQL file named **frumble.sql**.*~~

Part 6: Reflection (5 points)

Please reflect on your personal experience with this project. Specifically:

1. (1 point) What is one thing that you *learned* while doing this assignment?
2. (1 point) What is one thing that *surprised you* while doing this assignment?
3. (1 point) What is one *question that you still have* after doing this assignment?
4. (2 points) Recall the dataset you selected in our first lecture (eg, a grocery shopping list).
   - Draw an ER diagram for it; you do not need to submit it
   - Instead, answer the following question: Is there an attribute which is naturally a primary key for this dataset? If there is, what is it? If there isn't, how might you synthesize or enforce a primary key for your table?

The remaining questions are completely optional, but help us understand if we are creating appropriately-challenging questions.

- How many hours did it take you to finish this assignment, including time to set up your computer (if necessary)?
- How many of those hours did you feel were valuable and/or contributed to your learning?
- Did you collaborate with other students on this assignment? If so, approximately how many people did you collaborate with? Do not include yourself in the count.

~~*Save your answer to these reflection and feedback questions in a file called **hw4-writeup.txt** in the submission directory.*~~

Submission Instructions

You should answer the questions on Gradescope under HW4. You can resubmit an arbitrary number of times until the due date, which will save your progress without submitting all answers. ~~*As noted within the problem set, you will need to submit the following files all in the same directory:*~~

- ~~*geography.pdf*~~
- ~~*halloween.sql*~~
- ~~*theory.txt*~~
- ~~*frumble.sql*~~
- ~~*hw4-writeup.txt*~~

~~*Whew, that was a lot of text. Let's finish with a quick piece of trivia! Did you know that the opening lines of the Aeneid are:*~~

> ~~*Arma virumque cano, Troiae qui primus ab oris*~~
> ~~*Italiam fato profugus Laviniaque venit*~~
> ~~*litora multum ille et terris iactatus et alto*~~
> ~~*vi **Gradescopae compilationae errorarum***~~

~~*Which loosely translates to:*~~

> ~~*Warfare! and a man at war! Of these I sing. A man who fled from Troy*~~
> ~~*to Italy and Lavinia, a fugitive driven by fate.*~~
> ~~*A man who was flung from lands and the deepest of seas*~~
> ~~*by the power of Gradescope compilation errors*~~

~~*Clearly, the ancient Romans respected Gradescope autograders' need for **filenames that exactly match the homework spec**. Let us all learn from Aeneas' epic journey.*~~