University of Toronto
CSC343, Fall 2017

Assignment 3

Due: Wednesday, December 6 before 8pm!

IMPORTANT: Read the Piazza discussion board for any updates regarding this assignment. We will provide a summary of key clarifications, and it is required reading. Check it regularly for updates.

Learning Goals

By the end of this assignment you should be able to:

- identify tradeoffs that must be made when designing a schema in the relational model, and make reasonable choices,
- express a schema in SQL's Data Definition Language,
- formally reason about functional dependencies,
- appreciate differences between what DDL can express, what an XML DTD can express, and what a JSON schema can express, and
- recognize differences in the strengths of the relational model and the semi-structured model for representing a domain.

Part 1: Informal Relational Design

In class, we are in the middle of learning about functional dependencies and how they are used to design relational schemas in a principled fashion. After that, we will learn how to use Entity-Relationship diagrams to model a domain and come up with a draft schema which can be normalized according to those principles. By the end of term you will be ready to put all of this together, but in the meanwhile, it is instructive to go through the process of designing a schema informally.

The domain

Suppose we want to build a system for running quizzes online. There will be a large bank of questions on various topics. A course instructor will pick questions from these to create a quiz. The answers of students in the class will be recorded so that grades can be assigned according to a marking scheme.

The following features must be supported:

- Each student has a unique student ID, which is a 10-digit number, and a first and last name, and is in zero or more classes.
- A class has a room (*e.g., "room 366"*), grade (*e.g.,* "grade 5"), teacher (*e.g.*, "Miss Nyers"), and one or more students who are in that class.
- There can be multiple classes for the same grade.
- A room can have two classes in it (for example, if we have a grade 2-3 split class), but never more than one teacher.
- Our question bank includes three types of question: true-false, multiple choice, and numeric. Numeric questions can only have integer answers. (We won't handle floats.)

- Questions are identified by their unique question ID.
- All questions have question text (*e.g.*, "What is the capital city of Saskatchewan?") and a single correct answer.
- A multiple choice question has answer options (*e.g.*, "Saskatoon"), and there are always at least two. The correct answer must be one of the options.
- An incorrect answer to a multiple-choice or numeric question may have one hint associated with it. Correct answers do not have hints.
- For numeric questions, each hint is specific to a given range of values into which an incorrect answer may fall. The range will be specified by its lower (inclusive) bound and its upper (non-inclusive) bound. For instance, if the range for a hint is from 3 to 7, it is a hint associated with an answer x such that $3 \leq x < 7$.
- A quiz has a unique ID, a title, a due date and time, one or more questions from the question bank, and a class to which it is assigned.
- The instructor can choose whether or not students should be shown a hint (if there is one in the test bank) when they give a wrong answer to a question. This is a single flag for the whole quiz rather than one per question.
- Each question on a quiz has a weight: an integer that indicates how much a correct answer contributes to the student's total score on the quiz. The same question could occur in multiple different quizzes with different weights.
- The database records student responses to the quiz.
- Only a student in the class that was assigned a quiz can answer questions on that quiz.
- A student may not have answered all the questions. It's even possible that they answered none.

Don't get distracted by wondering whether students can answer a question multiple times, when hints will be given, or other aspects of taking the quiz. Your only responsibility is to record the relevant data.

Define a schema

Your first task is to construct a relational schema for our domain, expressed in DDL. Write your schema in a file called `schema.ddl`.

As you know, there are many possible schemas that satisfy these properties. We aren't following a formal design process for Part 1, so instead follow as many of these general principles as you can when choosing among options:

- If a constraint given above in the domain description can be expressed without assertions or triggers, it should be enforced by your relational schema.
- Avoid redundancy.
- Avoid designing your schema in such a way that there are attributes that can be null.
- Wherever an attribute *cannot* be null (according to the domain description), add a `NOT NULL` constraint.

You may find there is tension between some of these principles. Where that occurs, use your judgment to make a tradeoff.

To facilitate repeated importing of the schema as you correct and revise it, begin your DDL file with our standard three lines:

```
drop schema if exists quizschema cascade;
create schema quizschema;
set search_path to quizschema;
```

Document your choices

At the top of your DDL file, include a comment that answers these questions:

1. What constraints from the domain could not be enforced?
2. What constraints that could have been enforced were not enforced? Why not?

Instance and queries

Once you have defined your schema, create a file called `data.sql` that inserts data into your database that represents the small dataset defined informally in file `Quiz-data.txt`. You may find it instructive to consider this data as you are working on the design.

Then write queries to do the following:

1. Report the full name and student number of all students in the database.
2. For all questions in the database, report the question ID, question text, and the number of hints associated with it. For True-False questions, report `NULL` as the number of hints (since True-False questions cannot have hints).
3. Compute the grade and total score on quiz Pr1-220310 for every student in the grade 8 class in room 120 with Mr Higgins. Report the student number, last name, and total grade.
4. For every student in the grade 8 class in room 120 with Mr Higgins, and every question from quiz Pr1-220310 that they did not answer, report the student ID, the question ID, and the question text.
5. For each question on quiz Pr1-220310, report the number of students in the grade 8 class in room 120 with Mr Higgins who got the question right, the number who got it wrong, and the number who did not answer it.

We will not be autotesting your queries, so you have latitude regarding details like attribute types and output format. Make good choices.

Write your queries in files called `q1.sql` through `q5.sql`. Download file `runner.txt`, which has commands to import each query one at a time. Once all your queries are working, start postgreSQL, import `runner.txt`, and cut and paste your entire interaction with the postgreSQL shell into a plain text file called `demo.txt`.

What to hand in for Part 1

Hand in plain text files `schema.ddl`, `data.sql`, `q1.sql` through `q5.sql`, and `demo.txt`. These must be plain text files, and you must include the demo file, or you will get zero for this part of the assignment. If you are unsure about this, please talk to an instructor during office hours.

Thought questions

These questions will deepen your appreciation of issues concerning design, efficiency, and expressive power. They are for your learning, not for marks. Although you won't hand them in, feel free to discuss them with us in class or office hours.

1. Suppose we allowed you to be less strict in following the design principles listed above. Describe one compromise you would make differently.
   (a) How would the schema be different?
   (b) What would be the benefits of this new schema?
   (c) What would be lost in this new schema?

(d) Why would you make this different compromise?

2. At the end of the course we will discuss the semi-structured data model, and will see two examples of it: XML and JSON. Think about these questions:

   (a) What aspects of the data were awkward to express in SQL? Would they be easier in JSON or in XML?

   (b) Are there constraints that could not be expressed in SQL but that can be expressed in JSON or in and XML DTD?

   (c) Are there constraints that you *could* express in SQL that cannot be expressed in JSON or in and XML DTD?

Part 2: Functional Dependencies, Decompositions, and Normal Forms

1. Consider a relation V with attributes $LMNOPQRST$ and functional dependencies W.
$$W = \{\ LPR \to Q, \quad LR \to ST, \quad M \to LO, \quad MR \to N\ \}$$

   (a) State which of the given FDs violate BCNF.

   (b) Employ the BCNF decomposition algorithm to obtain a lossless and redundancy-preventing decomposition of relation R into a collection of relations that are in BCNF. Make sure it is clear which relations are in the final decomposition, and don't forget to project the dependencies onto each relation in that final decomposition. Because there are choice points in the algorithm, there may be more than one correct answer. List the final relations in alphabetical order (order the attributes alphabetically within a relation, and order the relations alphabetically).

2. Consider a relation P with attributes $ABCDEFGH$ and functional dependencies T.
$$T = \{\ AB \to CD, \quad ACDE \to BF, \quad B \to ACD, \quad CD \to AF, \quad CDE \to FG, \quad EB \to D\ \}$$

   (a) Compute a minimal basis for T. In your final answer, put the FDs into alphabetical order. Within a single FD, this means stating an FD as $XY \to A$, not as $YX \to A$. Also, list the FDs in alphabetical order ascending according to the left-hand side, then by the right-hand side. This means, $WX \to A$ comes before $WXZ \to A$ which comes before $WXZ \to B$.

   (b) Using your minimal basis from the last subquestion, compute all keys for P.

   (c) Employ the 3NF synthesis algorithm to obtain a lossless and dependency-preserving decomposition of relation P into a collection of relations that are in 3NF. Do not "over normalize". This means that you should combine all FDs with the same left-hand side to create a single relation. If your schema includes one relation that is a subset of another, remove the smaller one.

   (d) Does your schema allow redundancy? Explain how you know that it does or does not.

Show all of your steps so that we can give part marks where appropriate. There are no marks for simply a correct answer. If you take any shortcuts, you must explain why they are justified.

What to hand in for Part 2

Type your answers up using LaTeX or Word. Hand in your typed answers, in a single pdf file called part2.pdf.

Final Thoughts

Declare your group now: Well before the due date, declare your team (even if you are working solo) on MarkUs. It is impossible to do so during the grace period, even if you have grace points you are going to use or an extension.

Submission: Check that you have submitted the correct version of your files by downloading it from MarkUs; new files will not be accepted after the due date.

Some parting advice: It will be tempting to divide the assignment up with your partner. Remember that both of you probably want to answer all the questions on the final exam.