TDA357/DIT620 Databases lab assignment

Last updated: January 20, 2017

Contents

1 Introduction

1.1 Purpose

The purpose of this assignment is for you to get hands-on experience with designing, constructing and using a database for a real-world domain. You will see all aspects of database creation, from understanding the domain to using the final database from external applications.

The final result of this assignment is a playable computer game. The game is set in a world with towns and cities, which are interconnected by roads. From time to time, money is awarded to the first player that reaches a certain city or town. Players can build roads and hotels. Traveling across a road costs money, and so does staying at a hotel. More details about the game-play of this game will be provided in one of the later tasks.

1.2 Structure

This assignment consists of four distinct tasks:

Task 0: Getting started This task gets you set up to successfully complete this lab assignment by finding a lab partner, registering in Fire and getting PostgreSQL credentials.

Task 1: Designing the database schema From the domain description, you will create an E/R diagram which correctly models the domain model and translate it into a relational database schema.

Task 2: Constructing the database From the E/R diagram and relational database schema, you will implement the database schema in the PostgreSQL database engine.

Task 3: Writing a front end application Based on a working database implementation in PostgreSQL, you will write a front end application in Java using JDBC that communicates with the database.

Contrary to previous lab assignments for this course, a task does not build on the solution of the previous task.
For instance, do not assume that the database schema you design in Task 1, will be the database schema you implement in Task 2.

To make it easier to grade, we will select a good solution for a given task and ask all teams to start from that.
For instance, after the deadline for Task 1, we select a good solution for Task 1 and distribute it to everyone. Task 2 will then consist of implementing the database schema we provide, instead of your own.

1.3 Assignment submission and deadlines

To pass the programming assignment, you must pass all four tasks described in this document. You will do the assignment in groups of two.

You must submit your solutions through the Fire reporting system. You must submit your groups solutions to each task by the given deadline. Each task will list what files to submit to Fire.

Each task has a first deadline and last deadline. You should submit a solution by the first deadline, even if it is incomplete. Not submitting a solution by the first deadline means you fail that task (and thus the lab).

After submission, your assignment will be graded (pass or reject) and you will receive comments on your solution. If your submission is rejected, you are allowed to refine your solution and re-submit it until the last deadline. After the last deadline, no more solutions for that task are accepted.

To reduce the TAs workload for grading lab assignments, some tasks will be automatically graded. This means that your Fire submission is processed using an automated tool which runs a collection of tests on your submission. This automated tool will report any problems found with your submission, just like a human grader would. Since this is a new way of grading for this course, we expect that there may be some issues along the way. Please give us some leeway on this.

The advantage of automated grading is that you receive feedback on your submissions a lot faster.

2 Task 0: Getting started

Deadline for first submission:	**Sunday 22 January 2017**
Deadline for last resubmission:	**Sunday 12 March 2017**

2.1 Your job

Find a lab partner This lab should be completed in teams of 2. Exceptions can be made for extraordinary reasons, after consulting with the examiner. Do not wait until the last minute: the sooner you find a lab partner, the sooner you can start working on Task 1.

Register in the Fire system We use the Fire system to handle submissions and grading of the tasks in this lab assignment. You need to sign up to Fire in order to register your lab team, and later, submit your solutions for the different tasks. Read the separate notes on registering in the Fire system and submitting assignment tasks [3].

Get your PostgreSQL credentials Each lab team has its own PostgreSQL username and password. These credentials are distributed through the Fire system, after you have registered your lab team and submitted Task 0 in Fire (send in an empty file). You will need these credentials for Tasks 2 and 3. Read the separate notes on requesting your groups PostgreSQL account [3].

2.2 What to submit

Just an empty file. You will receive the PostgreSQL credentials as a reply.

3 Task 1: Designing the database schema

Deadline for first submission:	**Sunday 5 February 2017**
Deadline for last resubmission:	**Sunday 12 March 2017**

3.1 Domain description

The domain to model is that of a fictional board game.

Countries Countries consist of areas which are either cities or towns. Countries have unique names.

Areas Areas can be interconnected by roads. There can be many roads between 2 areas. Areas have names that are only unique in the country they are located in. Areas also have a population size associated with them.

Roads Roads can be owned by a person, or be a public road (not owned). Roads have a roadtax associated with them.

Hotels Hotels have names, but they are not unique in any way. Hotels also have a price per night.

People A Person is always a citizen of a country. People travel around, but are always located in a certain city or town (an area). People can own hotels, which are located in cities (not towns). A person can own multiple roads, but cannot own more than 1 road between 2 areas. A person can own maximum 1 hotel in a city. Persons have a name and a personnummer. A person's name is not unique, but the personnummer is unique in their country. Each person can also speak several languages and have a budget.

3.2 Your job

Your first task is to design the database that your application will use. The goal of this task and the next is to reach a correct database schema that could later be implemented in PostgreSQL.

E/R diagram You are to create an E/R diagram that correctly models the domain described above. Hint: if your diagram does not contain (at least) one weak entity, (at least) one ISA relationship, and (at least) one many-to-at-most-one relationship, you have done something wrong. You can use any tool you like for this task, as long as you hand in your solution as an image in one of the formats .png, .jpg, .gif or .pdf. The tool Dia [1] is available on the school computer system, has a mode for ER diagrams, and can export diagrams to image files, but using any other tool is also OK.

Relational schema When your E/R diagram is complete, you should translate it, using the (mostly) mechanical translation rules [2], into a database schema consisting of a set of relations, complete with column names, keys and references.

Functional dependencies Identify the functional dependencies that you expect should hold for the domain. Do this starting from the domain description. Do not only look for functional dependencies in your relation schemas! If you do, then any potential errors in your schema would not be caught, nor will you find any extra constraints not already captured by the relational structure! At least one such constraint exists in the domain. You should always look for the functional dependencies in the domain description. Remember to check for functional dependencies with multiple attributes on the left-hand side of the arrow.

Update based on constraints Update your relational schema with the extra constraints discovered from your functional dependencies. Alternatively, argue why your schema should not capture the constraint (see e.g. the difference between 3NF and BCNF for why not all constraints can be captured). Further, do the same for any constraints needed to handle cyclic relationships in your diagram.

3.3 What to submit

You should submit 4 files. Files with a ".txt" suffix are text files.

diagram.X your E/R diagram, where .X is one of .png, .jpg, .gif or .pdf.

schema1.txt the database schema that you get from translating the diagram into tables. Clearly mark keys (e.g. _key_). If any attribute can be null, indicate so by writing (nullable) or similar next to it. If you have made any non-obvious choices, when choosing which translation approach to use (e.g. ER or Null) include a comment about why you decided on using it.

fds.txt the full list of functional dependencies that you have identified for the domain.

schema2.txt the database schema that you get after updating your schema1.txt with any added constraints found because of the functional dependency analysis.

4 Task 2: Constructing the database

Deadline for first submission:	**Sunday 19 February 2017**
Deadline for last resubmission:	**Sunday 12 March 2017**

4.1 Description

to be determined

4.2 Your job

to be determined

4.3 What to submit

to be determined

5 Task 3: Writing a front end application

Deadline for first submission:	**Sunday 5 March 2017**
Deadline for last resubmission:	**Sunday 12 March 2017**

5.1 Description

to be determined

5.2 Your job

to be determined

5.3 What to submit

to be determined

References

[1] Dia. http://www.cse.chalmers.se/edu/course/TDA357/VT2017/lab/dia.html.

[2] ER diagram translation rules. http://www.cse.chalmers.se/edu/course/TDA357/VT2017/_downloads/Translation.pdf.

[3] How to use Fire, submit solutions and request PostgreSQL credentials. http://www.cse.chalmers.se/edu/course/TDA357/VT2017/lab/submission.html.