DM26 Database Systems

Fall 2005 Project

Department of Mathematics and Computer Science
University of Southern Denmark

October 12, 2005

Introduction

The purpose of this project is to try in practice the process of creating a relational database application. This process includes design in the ER-model, transfer to the relational model, normalization of relations, implementation in a DBMS, and programming of an application controlling user interaction with the database.

The project should be done in groups of two persons (single person groups and three persons groups only by agreement with the lecturer). PostgreSQL should be used as DBMS, and Java with JDBC should be used for programming.

System Requirements

The subject of the project is an electronic cookbook. The idea is to keep information about recipes and kitchen inventory in a system which can suggest recipes, make list of things to buy, and ensure a minimum inventory at all times.

At least the following objects should be modeled in the system:

Recipe Required features: name, list of ingredients, number of persons served, the actions for making the dish, preparation time (divided into the actual working time and the rest), picture of dish (if available), comments on recipe.

Ingredient Required features: name, price.

Menu Required features: list of recipes, comments.

Minimum Inventory Required features: list of ingredients and their allowed minimum amount.

Current Inventory Required features: list of ingredients and their current amount.

Cook Required features: list of recipes mastered.

Guest Required features: List of ingredients disliked.

Several minimum inventories can be envisioned to exist, for use in different situations (e.g. everyday, Christmas time, guests in house), whereas there is only one current inventory (most houses have only one kitchen). A cook can only do recipes which he has mastered. A guest should not be served food with ingredients he dislikes.

Scaling the preparation time for recipes in a realistic way is not trivial. The approach which should be taken here, is for each recipe to store the time when serving one person *and* when serving ten persons. This implicitly defines a linear function $f(x) = ax + b$, which should be used for calculating the time when serving other numbers of persons.[1] This information should be stored for both actual working time and the rest of the preparation time (i.e. store two times two values). When preparing several dishes, e.g. for a menu, sum the actual working times, but take the max of the rest times (assuming these can parallelize). Sum the two values to find an estimate for the total preparation time.

[1]The slope of the function will differ widely among dishes—think of cooking pasta vs. making a salad.

At least the following functionality should be provided by the application:

Find recipes containing a given set of ingredients. Showing the name is enough.

Print recipe scaled to a given number of persons. Show all relevant info, including the price.

Find menus possible to make for a given number of guests while fulfilling some or all of the following constraints: doable with the current inventory, max preparation time, max price, guest list, cook at work.

Write out list of things to buy to be able to make a given recipe or menu, given the current inventory and a number of persons to serve.

Write out list of things to buy to ensure a given minimum inventory.

Additionally, it should be possible to update inventories based on the use of a given recipe, or based on shopping according to a list produced by the system. Additionally, manual updates of amounts of single ingredients should be possible (for freely using or buying ingredients). To limit the amount of programming, no further facilities for updates need to be included in the application (i.e. in the project, you add recipes, ingredients, etc. to your database simply by using `psql`).

The system described above is the minimal required solution. Groups are free to extend the project to include more features (further objects and functionality, such as supermarkets and associated partial list of things to buy, or recipy types such as vegetarian).

Tasks

Your tasks are:

- To develop an appropriate ER-model the system.
- To transfer this to the relational model.
- To ensure that all relations are in BCNF form (if necessary, decomposing relations, which again should lead to a refinement of the ER-model).
- To create these relations in a database in the PostgreSQL DBMS.
- To program (using Java and JDBC) an application controlling the user interaction with the system. The application should provide the functionality described above.

Input and Output

To limit the amount of programming, only a simple, command-line based interface is required for user interaction. For instance, choices by the user can be input by showing a numbered list of alternatives, or by prompting for IDs of recipes, menus, guests, etc. Showing pictures could be done in a simple GUI window, or the byte count of the picture could simply be reported.

Project groups are of course free to make a more elaborate (graphical) user interface (which will be more fun, but no extra credit is given for it).

Test Data

For testing purposes, material from the web can provide the basic info for recipes. The rest can just be made up. A decent amount of test data must be made (at least on the order of 15 recipes and associated ingredients, 4 menus, 2 minimum inventories, 3 guests, and 2 cooks). Sharing data between groups to expand the amount of data is fine.

Formalities

A printed report of 10 to 15 pages should be handed in. Its main aim should be to describe the design choices made during development, the reasoning behind these choices, and the structure of the final solution.[2] Specific items that must also be included in the report are: A diagram of your ER-model, the schemas of your relations, proofs/arguments showing that these are in BCNF form, and a short user manual for the application.

Actual Java code should not be printed out (besides small excerpts in the report text), but rather be handed in using the `aflever` command on the Imada system: Move to the directory containing your code and issue the command `aflever DM26`. This will copy the contents of the directory to a place accessible by the lecturer. Repeated use of the command is possible (later uses overwrite the contents from earlier uses). In the directory, you must for identification purposes have an ASCII file named `names.txt` containing the names of the group members, with one name per line.

You should have a copy of your final database on the PostgreSQL server of the department (at the machine `dbhost`), and your code should work on this copy (i.e. it should be possible to try out your system).

You must hand in the report by

Wednesday, November 30

[2]To prepare for the writing of the report, it is a good idea to keep a log of the discussions within the group and of the work done.