Recipe Database Report

Lecture: Databases 2024

Lecturer: Miika Reijonen

Group:

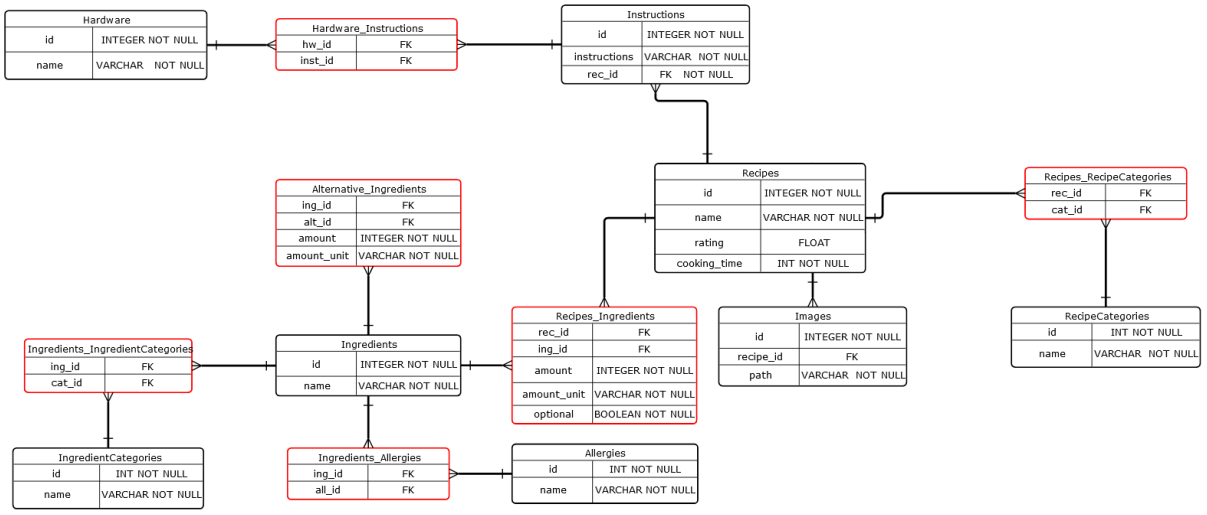
Lukas Schoeck,
Mehmet Akif Tos,
Riccardo Cerimele,
Leon Fischer

Date: 10.04.2024

Introduction

The main goal of this project is to create and execute a relational database for a food recipe management application. The purpose of this application is to offer an efficient and organized system for storing, searching, and recommending recipes based on different factors such as available ingredients, cooking equipment, total cooking time, and optional or alternative ingredient suggestions. The project follows a meticulous progression of steps, from requirement analysis and schema design to database implementation, data insertion, and query development, culminating in the comprehensive documentation of all processes. This report details each phase, the challenges faced, and the steps taken to overcome them.

ER-Diagram



The first step when creating the diagram was identifying the entities: Recipes (which contains the cooking time information and a rating about the difficulty), Ingredients, Hardware, Instructions defining them with IDs and names, and their respectives Categories, in order to guarantee a sort of grouping.

The next step was adding the many-to-many tables regarding the previous tables we defined: Hardware-Instructions, Ingredients-Categories, Recipes-Ingredients, Recipes-Categories. Moreover, we needed to create a table to store alternative Ingredients for better recommendations. Giving such just through the categories table could sometimes be difficult (see chapter Challenges).

We decided to define an Ingredient as “optional” through a boolean column in the many to many Recipes_Ingredients table (which contains for every recipe all of his ingredients, and for every ingredient all of the recipes in which it is used), so that when a user wants to see the ingredients of a recipe, the software will also show the optional ingredients, with the boolean flag turned on.

To add images we added a one-to-one table to the Recipes table, which contains a path to another database with the image files. This approach assures a higher performance and a clear separation of different data in our recipe system.

Concluding the development of the diagram, we decided to include information regarding allergies, linking the ingredients to a new table (Allergies) with a many-to-many Ingredients_Allergies table.

Queries

To demonstrate the functionality of our relational database we create some SQL-Statements executing CRUD-Operations. The data from these queries can be used in the application layer of our recipe app later. In the following we list all the functionality of our queries attached to this assignment. They only serve as examples and queries can be added or deleted at any time, depending on further requirements.

Create:

- create statements for every table visible in the ER-diagram

Insert:

- inserts for every table seen in the ER-diagram (details in attached insert file)

Read:

- Fetch all recipes
- Fetch all recipes with a rating over 4.0
- Fetch all recipes possible with given ingredients from ing_id list
- Fetch a recipe and the corresponding images
- Fetch all hardware needed for a recipe
- Fetch all instructions of a recipe
- Fetch all ingredients of a recipe
- Fetch all allergies of a recipe
- Fetch alternative recipes based on the category
- Fetch all ingredients belonging to an ingredient category
- Fetch categories of an ingredient

Update:

- Update the rating of a recipe
- Update the instructions of a recipe
- Update the used hardware for an instruction
- Update the ingredients of a recipe

Delete:

- Delete a recipe
- Delete an instruction step
- Delete a hardware
- Delete a Recipe_RecipeCategories entry
- Delete all images of a recipe
- Delete an ingredient of a recipe
- Delete an allergy of an ingredient
- Delete an alternative ingredient of an ingredient
- Delete an ingredient category from an ingredient

Challenges

During the design and implementation of the database, we faced several challenges. One of the initial obstacles was determining the best method for including optional and alternative ingredients in the database. We had to model these factors carefully to maintain the database's integrity and avoid unnecessary complexities. In the end we decided to create an extra table for alternative ingredients because e.g. cow milk could have the categories dairy and milk and it is not clear which category to use for searching alternatives. In the milk category there could be soy milk which in this case serves as an alternative but in the dairy category cheese could be found which is unsuitable.

Additionally, we faced a challenge in our schema design when deciding whether a primary key was necessary for the junction tables. After careful deliberation, we ultimately decided to just use composite keys because most of the time the junction tables are only joined via their foreign keys.

Another issue we encountered was deciding how to handle the cooking steps. While separating them into a different table may seem convenient, it may not be practical given the nature of the data. After careful consideration and a discussion with our lecturer we finally decided to separate the steps in a new table to connect each of them with the used hardware in the graphical user interface later on.