By Brendan Edwards
And Brendan Prednis

Data Modeling Exercise

Summary

We want to create a recipe creating/sharing and grocery list app. You'll be planning out what tables we'll need, what information they'll store, and how the data will relate to each other.

Features

- users can sign into the app with their email and password
- users can create recipes with ingredients and instructions
- recipes can be marked as public or private
- users can view other people's recipes
- ingredients from recipes can be added to user's grocery lists
- users can create their own occasions and assign recipes to occasions

BRAINSTORMING LIST

- User information
    - User name
    - Password
    - User id
    - Favorited recipes
- Recipe information
    - Organize by culture
    - Genre of Food
    - Allergy information
        - Boolean is allergy know
    - Ingredients
    - Instructions
    - User Rating
    - Public or private recipe

TABLES

- User table
- Recipe table
- Review/ratings table
- Auth table

Step 3

What relationships should exist among the data ?

- One to Many
    - User favorite recipes
    - Ratings given
- One to One
    - Public or private recipe <-> user information
        - Publisher or reader?
    -
- Many to many
    - Occasions? //NOTE: what does this mean?

STEP 4

Columns

- Auth
    - User_id
        - Type: integer -> to allow scaling within the user base. To make user activity reference quick and readable
    - Password
        - To log in to the user's account
        - Type: Text -> allow for a mix of numbers, letters ,symbols.
    - Email
        - User email to authenticate the users credibility
        - Type: Varchar -> add readability to the email, while limiting input.
    -

- User
    - User_id:
        - Type: integer -> *see auth/user_id
    - First_name:
        - Type: varchar -> can input any name, no character restrictions.
    - Last_name:
        - *Same as first_name*
    - Favorite_recipes
        - Type: varchar -> storing a bunch of recipe_ids in one. (optional: store as json?)

- Recipes
    - Recipe_id:
        - Type: Integer -> easy to store and reference a string of numbers.
    - User_rating:
        - Type: Integer -> allow frontend to determine the universal range of rating / OR: use PostgreSQL int4range type to specify a certain range that can be input for more type safety (Brendan P is a big fan of type safety).
    - Genre:
        - Type: Varchar -> balance between grouping and individuality.
    - Ingredients:

        - Type: Varchar -> split into a list later (front end).
    - Name:
        - Type: Varchar -> allow for any spelling.
    - public/private:
        - Type: Boolean -> it's either or.

- Reviews/ratings
    - Rating_id:
        - Type: integer -> so we can see how many reviews we're getting on each recipe.
    - User_review:
        - Type: varchar -> allow the user to type in freely without character/symbol restrictions.
    - Rating:
        - Type: integer -> (out of 5 stars) number ratings are efficient to work with.
    - User_id:
        - Type: varchar -> displays the user's name that left the review and rating
    - Recipe_id:
        - Type: integer -> allows us to see how many recipes a user has.