Annie Golladay

Data Modeling Lab 5-26-21

Step 1: Conceptual Posting

Brainstorming Recipe creating/sharing and grocery list app

List:

- User profile,username, email, password, followers and following other accounts, include boards/posts created (like pinterest), boards shared/saved,
- Create Recipe boards/posts include: create your own recipe link, recipe name, ingredients, prep time, instructions, grocery list, category tabs, mark as public or private, checked box to add grocery list item to your personal grocery shopping page, comments section
- Search page: include type of meal, occasion (like wedding, birthday), list of your own saved items(my saved items
- Grocery lists shopping page: list of items from grocery lists recipes that I've saved

Step 2: Table Ideas

1. **User table**: user_id, username, email, password, followers, following, recipe posts created, (recipe_id), recipe posts saved
2. **Newsfeed table**:
    a. following posts
    b. search feature- be able to search accounts
    c. (category_id)- show different categories
    d. comments
3. **Recipe table**: recipe_id,(user_id), recipe name, ingredients, instructions, grocery list items, mark public or private feature, comments
    a. (category_id) - show what category the recipe falls under
    b. saved or shared feature- highlight a heart to save or see if you've saved it already
4. **Search table**:
    a. (Category_id)

    b. (user_id)
    c. (recipe_id)- show recipes all in certain category
    d. search feature
5. **Category table:**
    a. Category_id
    b. (user_id)
    c. (recipe_id)- show recipes all in certain category
6. **Grocery table**:
    a. Grocery_id
    b. (recipe_id) - show what recipe the ingredient is under
    c. (category_id) - show what category the ingredient is under
    d. notes - put any extra notes

Step 3:

Relationships

One-to-One:

- User table to recipe table (1 user_id to 1 specific recipe_id) - 1 user to 1 specific recipe
- Recipe table to User table(1 recipe_id to 1 user_id) - 1 specific recipe to 1 user

One-to-Many:

- User table to recipe table (1 user_id to many different recipes)
- Recipe table to Category table (1 recipe_id to multiple categories)
- Grocery table to recipe table (1 grocery_id to multiple recipe_id)

Many-to-Many:

- Newsfeed table to Category table(many to many search results/categories)
- Category table to Recipe table (many categories to many recipes results)
- Grocery table to category table(many grocery lists to many categories)

Part 2:

Step 1: DB Designer (completed, saved as pdf)

Step 2:

Columns:

User Table

a. User_id: serial primary key
    i. Storing the data so it has a unique id and increments with each one
b. Username: varchar(50) - this data type was chosen because username can be up to 50 characters, so you can come up with your own title
c. Email: varchar(100)- this data type was chosen because email can be up to 100 characters
d. Password: varchar(100)- this data type was chosen because password can be up to 100 characters
e. Followers: integer - show number of followers with integer so you can see who follows you
f. Following: integer- show number of following with integer so you can see your favorites accounts
g. Recipes created: integer - show how many recipes created with a number, to keep track of your creations
h. Recipe_id: integer (foreign key)
    i. User has unique recipes which relates to the recipe table so that the recipes associated with this user so we can see all the recipes under this user
i. Recipes saved: integer - show number of recipes saved, so you can see your favorite recipes and go back to them

Newsfeed Table:

a. (Category_id): integer(foreign key)
    i. Newsfeed relates to the category id, so the newsfeed can list the specific categories associated with the posts that are shown in newsfeed
b. Following posts: text - any text to put in a posted item
c. Search feature: varchar(50) - this data type was chosen because search feature can be up to 50 characters
d. Comments: text - can make comments with as much text as you want

Recipe table:

a. Recipe_id: serial primary key
    i. Storing the data so it has a unique id and increments with each one
b. (user_id): integer (foreign key)
    i. Each recipe will show which user created so it shows each user specific to the recipe, which is why i included the user_id foreign key
c. recipe name: varchar(100) - data type of name of recipe up to 100 characters long
d. Ingredients: text - any text for ingredients
e. Instructions: text - put any text for the instructions on page
f. grocery list items: text - text for any items
g. public feature: boolean - data type chosen so you can answer true or false statement
h. private feature: boolean - data type chosen so you can answer true or false statement
i. Comments: text - make any comments with however much text you want
j. (category_id): integer (foreign key)
    i. Recipes will also show the category id's associated with each recipe and list out each category.
k. saved feature: boolean- data type chosen so you can answer true or false statement
l. shared feature: boolean- data type chosen so you can answer true or false statement

Search table:

a. (Category_id): integer (foreign key)
    i. Search page will show category options which include the id's to better search
b. (user_id): integer (foreign key)
    i. Search page will also show user id's if under the specified recipes
c. (recipe_id): integer (foreign key)
    i. Every search will reveal recipe options will include that unique recipe id
d. search feature: text

Category table:

a. Category_id: serial primary key
    i. Storing the data so it has a unique id and increments with each one
b. (user_id): integer (foreign key)
    i. Category page will also show user id's if under the specified category
c. (recipe_id): integer (foreign key)
    i. category page will also show recipe id's if under the specified categories

Grocery table:

a. Grocery_id: serial primary key
    i. Storing the data so it has a unique id and increments with each one
b. (recipe_id): integer (foreign key)
    i. On grocery list, there will be the unique recipe id to reveal which recipe the ingredient list goes to
c. (category_id) : integer (foreign key)
    i. On grocery list, there will be the unique category id to reveal which category the ingredient list goes to
d. notes - text - add any text for any added notes that you may want for your grocery list

Part 3: Create Tables

https://replit.com/@AnnieGolladay/DataModeling-Lab#main.sql

```
create table user(

    user_id SERIAL PRIMARY KEY,

    username VARCHAR(50),

    email VARCHAR(100),

    password VARCHAR(100),

    followers INTEGER,

    following INTEGER,
```

```
	recipes_created INTEGER,

	recipes_saved INTEGER,

	recipe_id INTEGER NOT NULL REFERENCES recipe(recipe_id)

);



create table newsfeed (

	following_posts text,

	search_feature VARCHAR(255),

	category_id INTEGER NOT NULL REFERENCES category(category_id),

	comments text

);



create table recipe (

	recipe_id SERIAL PRIMARY KEY,
```

```
    user_id INTEGER NOT NULL REFERENCES user(user_id),

    category_id INTEGER NOT NULL REFERENCES category(category_id),

    recipe_name VARCHAR(100),

    ingredients text,

    instructions text,

    grocery_list_items text,

    public_feature boolean,

    private_feature boolean,

    comments text,

    saved_feature boolean,

    Shared_feature boolean

);



create table search (
```

```
	user_id INTEGER NOT NULL REFERENCES user(user_id),

	recipe_id INTEGER NOT NULL REFERENCES recipe(recipe_id),

	category_id INTEGER NOT NULL REFERENCES category(category_id),

	search_feature text

);



create table category (

	category_id INTEGER NOT NULL REFERENCES category(category_id),

	user_id INTEGER NOT NULL REFERENCES user(user_id),

	recipe_id INTEGER NOT NULL REFERENCES recipe(recipe_id)

);



create table grocery (

	grocery_id SERIAL PRIMARY KEY,
```

```
    recipe_id INTEGER NOT NULL REFERENCES recipe(recipe_id),

    category_id INTEGER NOT NULL REFERENCES category(category_id),

    notes text

);
```