Smart Cookbook

Tyler Pelton

Khoi Le

# Application Background

This project is a comprehensive cookbook, which allows users to create recipes, diets, ingredients, and cooking utensils. The user is able to specify which ingredients are in each recipe, as well as which utensils are used in each recipe, and the diet the recipe is suitable for. The application allows the user to search for recipes by ingredient names, diet names, or with options to exclude an ingredient, or search for multiple ingredients.

# Data Description

Ingredients is a relation that stores a name as the primary key, the shelf life, and the calorie density of a cooking ingredient

Ingredients:

* name – Text, primary key
* shelf life - Integer
* calorie\_density - Integer

Recipes is a relation that stores an ID as the primary key, the name, the total calories of the recipe, and text instructions of how to cook the recipe.

Recipes:

* ID – Integer, primary key
* Name - Text
* total\_calories - Integer
* instructions - Text

Contains is a relation that stores ingredient names, retrieved from ingredients, and a recipe, retrieved from recipes

Contains:

* ingredient name – Text, foreign key referencing ingredients
* recipeId – Integer, foreign key referencing recipe

Diets is a relation that stores name of the diet (vegetarian,etc)

Diets:

* name – Text, primary key

Suitable is a relation that links a diet name to a recipe id, both being foreign keys

Suitable:

* diet\_name – Text, foreign key referencing diet
* recipe\_id – Integer, foreign key referencing recipes

Utensils is a relation that stores the name of the utensils (Oven, Pot, etc)

Utensils:

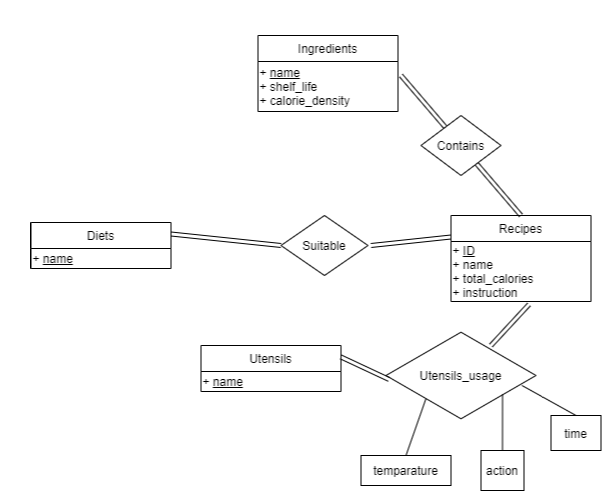
* name – Text, primary key

Utensil\_usage

Utensil\_usage:

* recipe\_id – Integer, foreign key referencing recipes
* utensil\_name – Text, foreign key referencing utensil
* temperature – Integer
* time - Integer
* action - Text

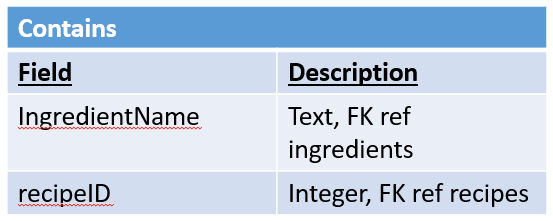
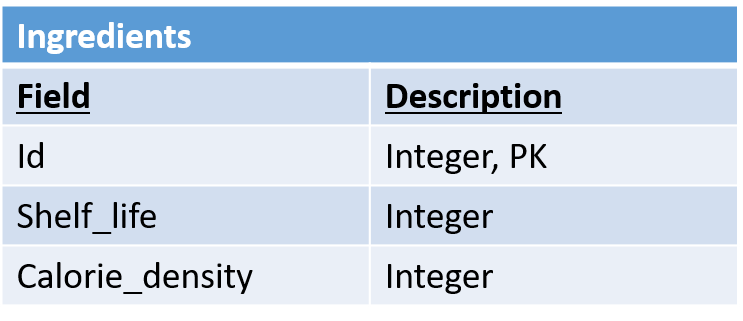
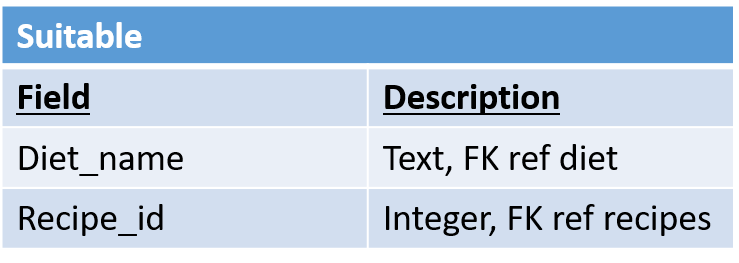
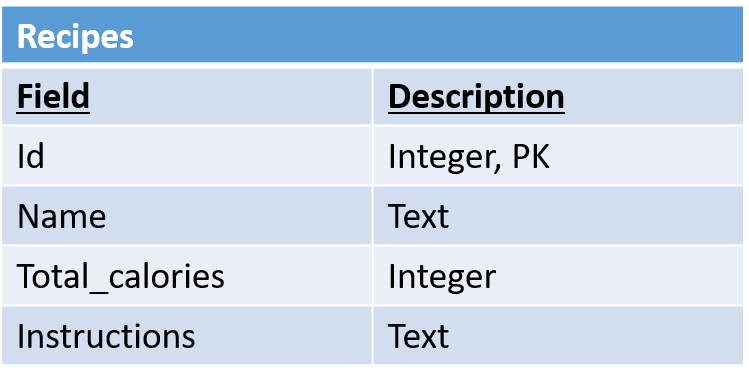
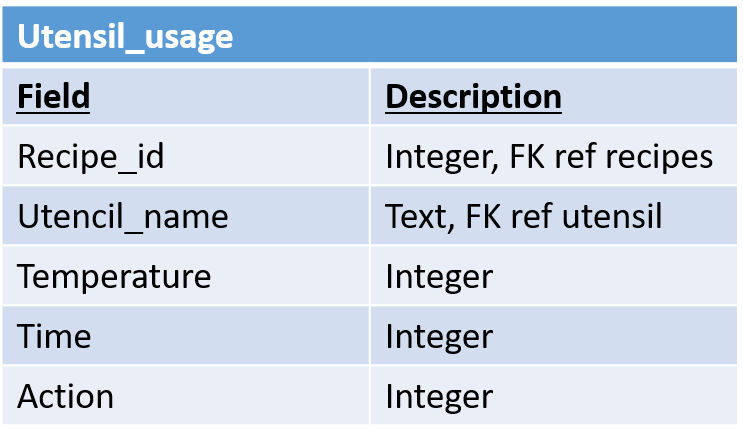
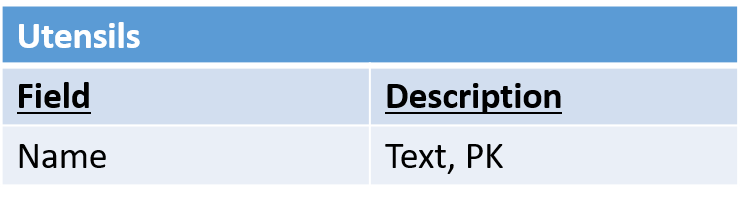
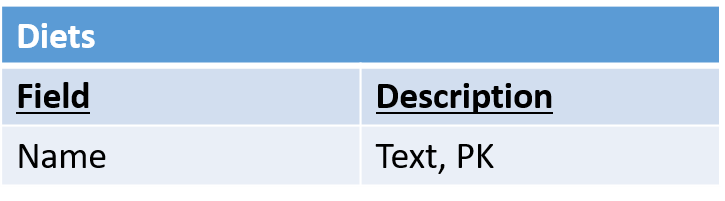
# E/R Diagram



# Functional Dependencies

* **Ingredients**
  + {name -> shelf life, calorie\_density}
* **Recipes**
  + {ID -> name, total\_calories, instructions}
* **Contains**
  + None, many to many relation
* **Diets**
  + None, only one entry in table
* **Suitable**
  + None, many to many relation
* **Utensils**
  + None, only one entry
* **Utensil\_usage**
  + {recipe id, utensil name -> temperature, time, action}

# Relation Schemas



## Create Table Statements:

CREATE TABLE IF NOT EXISTS ingredients (name TEXT PRIMARY KEY, shelf\_life INTEGER, calorie\_density INTEGER)

CREATE TABLE IF NOT EXISTS recipes (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, total\_calories INTEGER, instruction TEXT)

CREATE TABLE IF NOT EXISTS diets (name TEXT PRIMARY KEY)

CREATE TABLE IF NOT EXISTS utensils (name TEXT PRIMARY KEY)

CREATE TABLE IF NOT EXISTS contains (ingredientName TEXT, recipesID INTEGER, FOREIGN KEY (ingredientName) REFERENCES ingredients(name), FOREIGN KEY (recipesID) REFERENCES recipes(id))

CREATE TABLE IF NOT EXISTS suitable (dietName TEXT, recipesID INTEGER, FOREIGN KEY (dietName) REFERENCES diets(name), FOREIGN KEY (recipesID) REFERENCES recipes(id))

CREATE TABLE IF NOT EXISTS utensils\_usage (utensilName TEXT, recipesID INTEGER, time INTEGER, action TEXT, temperature INTEGER, FOREIGN KEY (utensilName) REFERENCES utensils(name), FOREIGN KEY (recipesID) REFERENCES recipes(id))

# Example Queries

## Search for the name of vegetarian recipes:

### SQL:

SELECT r.name  
FROM recipes r, diets d, suitable s  
WHERE r.id = s.id AND d.name = s.name AND d.name = “vegetarian”

### RA:



### TRC:

{t | (∃ s)(s ∈ recipes ∧ t[name] = s[name] ∧ (∃ u) (u ∈ diets ∧ (∃ v) (v ∈ suitable ∧ u[name] = v[diet\_name] ∧ u[name] = “vegetarian”)))}

## Search for the name of recipes that contain chicken and peas

### SQL:

SELECT r.name

FROM recipes r, contains c1, contains c2, ingredients i1, ingredients i2  
WHERE r.id = c1.id AND r.id = c2.id AND i1.name = “Chicken” AND i2.name = “Peas”  
AND c1.ingredient\_name = i1.name AND c2.ingredient\_name = i2.name

### RA:



### TRC:

∈ ∧ ∃

{t | (∃ s) (s ∈ recipes ∧ t[name] = s[name] ∧ (∃ u1) (u1 ∈ contains ∧ (∃ u1) (u2 ∈ contains ∧ (∃ v1) (v1 ∈ ingredients ∧ (∃ v2) (v2 ∈ ingredients ∧ v1[name] = “Chicken” ∧ v2[name] = “Peas” ∧ u1[ingredient\_name] = v1[name] ∧ u2[ingredient\_name] = v2[name] ∧ s[id] = u1[recipeID] ∧ s[id] = u2[recipeID] )))))}

## Search for the name of recipes that contain chicken but not fish

### SQL:

SELECT r.name

FROM recipes r, contains c, ingredients i

WHERE r.id = c.recipesID AND i.name = “chicken” AND i.name = c.ingredientName

EXCEPT SELECT r2.name

FROM recipes r2, contains c2, ingredients i2

WHERE r2.id = c2.recipesID AND c2.ingredientName = i2.name AND i2.name = “fish”

### RA:



### TRC:

{t | (∃ s) (s ∈ recipes ∧ t[name] = s[name] ∧ (∃ u1) (u1 ∈ contains ∧ (∃ v1) (v1 ∈ ingredients ∧ s[id] = u1[recipeID] ∧ (∃ u2) (u2 ∈ contains ∧ s[id] = u2[recipeID] ^ v1[name] = “chicken” ∧ u1[ingredientName] = v1[name] ∧ (∀  v2) (v2 ∈ ingredients ∧ v2[name] <> “fish” ∧ v2[name] = u2[ingredientName] ))))}

# Implementation

## OS:

This program will run on any OS that supports Python3

## DBMS:

This program uses Sqlite3 in Python 3 as the DBMS, with a command line interface

## Programming Language:

This program was written in Python 3.

# Role and Contributions

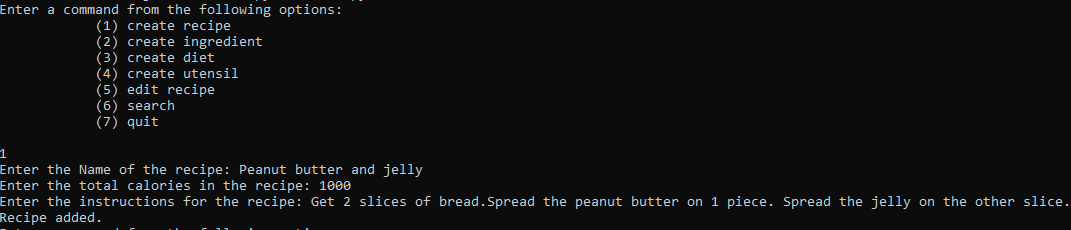
## Tyler Pelton

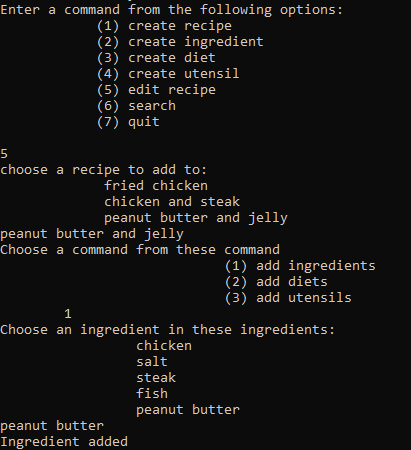
* Typing Documents
* Command line interface flow
* Some create commands
* Search command
* SQL, TRC Queries

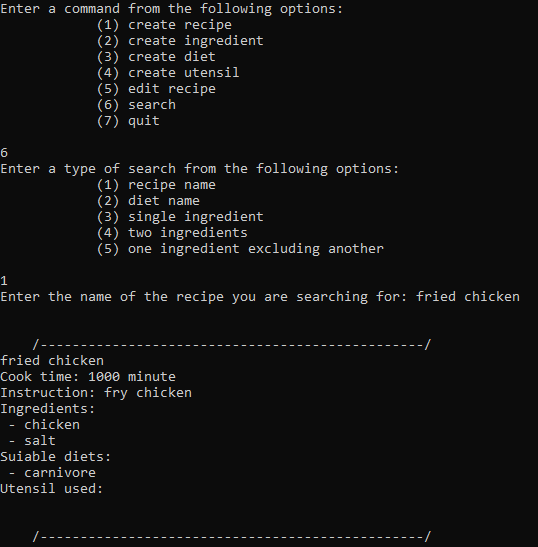
## Khoi Le

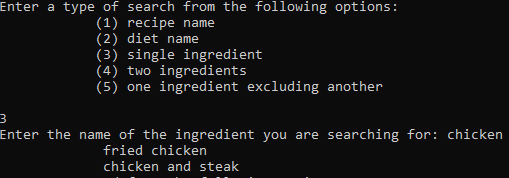
* E/R Diagram creation
* Base Framework for SQLite setup, create table statements
* Some create commands
* Edit/show recipe framework
* RA Queries

# Screenshots









https://media.discordapp.net/attachments/494986872309940272/705503060301971546/unknown.png