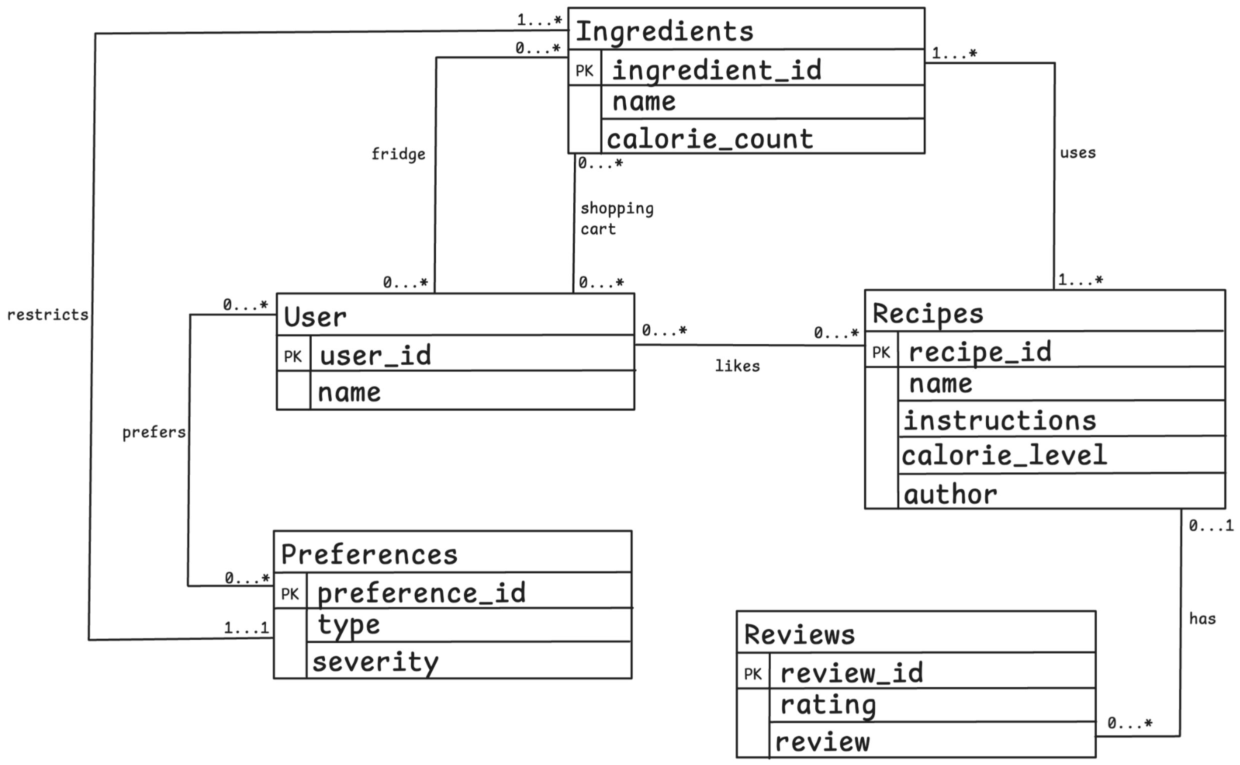
Pantry Pal Database Design

Team 101: nehasv2, pawudu2, alizain2, manasv2

UML



Entities

- *Users*
  This entity represents a user of Pantry Pal.
  - user_id: unique identifier for each user. This is the primary key for the entity.
  - name: stores the user's name to personalize their experience on the app.

  This must be its own entity because it has more than one attribute and is involved in multiple many-many relationships. For example, multiple users can own the same ingredients, or like the same recipes.
- *Recipes*
  This entity represents a Recipe that will be returned to the user after the appropriate filters are applied.
  - recipe_id: unique identifier for each recipe
  - name: name of the dish
  - instructions: step-by-step instructions for how to make the dish. This will be displayed if a user decides they want to make the recipe.
  - calorie_level: rough level for how many calories are in the dish. This field helps fit search criteria for 'caloric intake' queries.

    - author: the name of the person who wrote the recipe. This field helps fit search criteria for ‘similar recipes’ queries.

  It is its own entity because it has more than one attribute and is involved in multiple many-many relationships. For example, multiple recipes can have the same ingredients, or be liked by the same user.
- *Preferences*
  Preferences represent the type of food restriction that is possible. These include low sugar preferences, peanut allergies, or likes/dislikes.
    - preference_id: unique identifier for the preference
    - type: the specific type of preference. E.g: peanut allergy
    - Severity: high or low. Specifies whether to still show results that disobey the preference or avoid them all together when querying.

  Preferences needs to be its own entity because it is part of a many-many relationship: a user can have multiple preferences.
- *Reviews*
  Reviews represent the reviews that were left on recipes in the past.
    - review_id: unique identifier for the review
    - rating: rating the user gave the recipe out of 5
    - review: text review of the recipe

  This is its own entity because it is in a many-many relationship: each recipe can have multiple reviews.
- *Ingredients*
  This entity represents the meta-data of each ingredient
    - ingredient_id: the unique identifier for the ingredient
    - name: the name of the ingredient
    - calorie_count: average number of calories for a dose of the ingredient

  This must be an entity because it has more than one attribute and it has many-many relationships. For example, a recipe can have multiple ingredients.

<u>Relations</u>

- *Fridge*
  (Users - Ingredients, many-many)
  Encodes the relationship between a user and what ingredients they already have at home. This is many-many because multiple users can have the same ingredients at home, and each user can have multiple ingredients.
- *Shopping Cart*
  (Users - Ingredients, many-many)
  This encodes the relationship between a user and the ingredients they want/ need to buy in order to make the recipes on their current meal plan. This also must be many-many

because multiple users can be missing the same ingredients, and each user could be in need of multiple ingredients.

- *Ingredients List*
  (Recipes - Ingredients, many-many)
  This relation encodes the ingredients that are needed to make a recipe. It is many-many because multiple recipes could use the same ingredients, and each recipe uses multiple ingredients.
- *Recipe reviews*
  (Recipes - Reviews, one-many)
  Models the relationship between a recipe and its reviews. Each recipe can have multiple reviews (that are specific to that recipe), so this relation is one-many.
- *User preferences*
  (Users - Preferences, many-many)
  This relation models the preferences of a user. When a user selects certain preferences, they are encoded in this relation. It must be many-many because each user can select multiple preferences and each preference can be selected by multiple users.
- *Dietary Restrictions*
  (Preferences - Ingredients, -many)
  Relates the preference type to the ingredients it restricts. For example, a peanut allergy would restrict peanuts. This is a one-many relationship because each preference can restrict multiple ingredients.

Normalization: This set is already normalized because each entity's primary key already has an unique id for each element to handle functional dependencies. For example the entity ingredients has an ingredient_id which will make sure the name and calorie count is always unique taking away this dependency. All attributes contain atomic values and there are no repeating groups in any of the tables. This also applies for the rest of the entities with having an id making sure to keep all attributes unique to each other and constraint enough to avoid duplication and fix collision and deletion cases.

Conversion to Relational Dataset:
Relation: Users(user_id: varchar(100) [PK], name: varchar(100))

Relation: Ingredients(ingredient_id: varchar(100) [PK], name: varchar(100), calorie_count: int)

Relation: Recipes(recipe_id: varchar(100) [PK], name: varchar(100), instructions: varchar(10000), calorie_level: int, author: varchar(100))

Relation: Preferences(preferences_id: varchar(100) [PK], type: varchar(100), severity: varchar(10))

Relation: Reviews(review_id: varchar(100) [PK], rating: int, review: varchar(1000))

User and Recipes: This is a many to many relationship.
Relation: User_Recipe((user_id,recipe_id) : [PK], user_id : varchar(100)[FK to Users.user_id], recipe_id: varchar(100) [FK to Recipes.recipe_id])

User and Preferences: This is a many to many relationship.
Relation: User_Preferences((user_id,recipe_id): [PK], user_id: varchar(100) [FK to Users.user_id], preference_id: varchar(100) [FK to Preferences.preference_id])

User and Ingredients: This is a many to many relationship.
Relation:User_Ingredients((user_id,ingredients_id): [PK], user_id: varchar(100) [FK to Users.user_id], ingredients_id: varchar(100) [FK to Ingredients.ingredients_id])

User and Ingredients: This is a many to many relationship.
Relation: User_Preferences_Fridge((user_id,ingredients_id): [PK], user_id: varchar(100) [FK to Users.user_id], ingredients_id: varchar(100) [FK to Ingredients.ingredients_id])

Preferences and Ingredients: This is a One to One relationship.
Relation:Preferences_Ingredients((preference_id,ingredients_id): [PK], preference_id: varchar(100) [FK to Preference.preference_id], ingredients_id: varchar(100) [FK to Ingredients.ingredients_id])

Recipes and Reviews : This is a zero to one to zero or more relationship.
Relation:Recipes_Reviews((recipe_id,review_id): [PK],recipe_id: varchar(100) [FK toRecipes.recipe_id],review_id: varchar(100) [FK to Reviews.review_id])