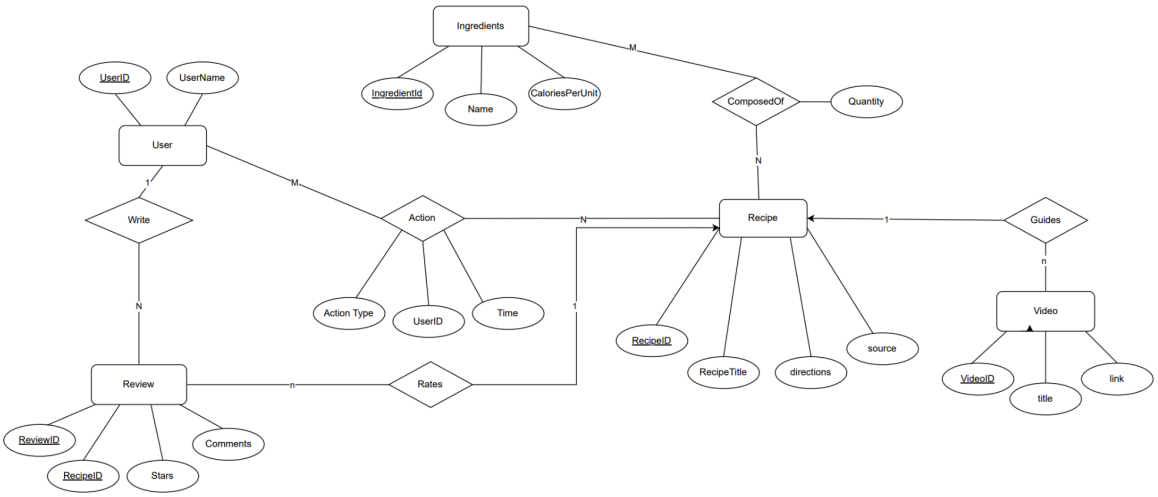
Conceptual and Logical Database Design - BiteMatch

Students:

Zhihao Cheng	zhihao11
Jiajun Huo	jiajunh6
Yiwen Zhang	yiwen19
Yiting Wang	yiitingw7

Introduction



Entities and Relationship and Their Assumptions

Entities

User

Users are central to this system as they perform actions like interacting with recipes and writing reviews. In this table, each user is uniquely identified by `UserID` with attribute `UserName`, which is decided by user. When a user signs up our product, a record will be inserted into this table.

Recipe

Recipes table is another central feature of the system, containing all the recipes users will be able to see. Recipe is complex and cannot be an attribute; it has the recipe name, directions which is a paragraph of how to make it, and the source of the recipe.

Review

Review is a feature that allows the users to review one recipe. It cannot be an attribute to the recipes because a single recipe can have multiple reviews (one per user), storing reviews as a single attribute in the Recipe table would create a multi-valued attribute (violating 1NF). Therefore, reviews need their own table.

Video

Recipe Video provides links to instructional videos associated with a specific recipe. Each video provides visual and verbal guidance on how to prepare the dish described in the corresponding recipe. This is to help users better understand the preparation process.

Ingredients

Ingredients provide detailed nutritional and categorical information about each ingredient in the recipe. `IngredientID` is a unique identifier for each ingredient to ensure distinct records in the database. `Name` is the descriptive name of the ingredient, making it recognizable. `CaloriesPerUnit` is the amount of calories per a specific unit of the ingredient, useful for nutritional calculations.

Relationships

Action

Users can interact with recipes in the feed through the following actions (e.g. Skip, Save, Maybe). Each recipe in the feed can receive these actions from multiple users. This means User and Recipe are a many-to-many relationship. This allows the system to Personalize the feed based on user preferences.

In terms of product design, and to avoid redundancy, when the user clicks “Save” on a recipe, it will be added to the Saved Collection, and the recipe will no longer be recommended in the future. When the user clicks “Skip”, it will be removed from the user’s recommendation stream, which means that the recipe will not be recommended in the future. At this point, only if the user clicks `Maybe` will it be possible to get the recommendation of the recipe again in the future.

Write

Users can write their reviews on a certain recipe, while a recipe can receive reviews from many users. This means it is a one-to-many relationship. To simplify complexity, the product is not allowed a user to submit another review on a recipe if there is one that exists.

ComposedOf

Typically, a recipe includes multiple ingredients, and an ingredient can be used in multiple recipes, making this a many-to-many relationship.

Guides

Guides is a many to one relationship between Video and Recipes. Each dish in the recipe matches one or more instructional videos, so that users can have options of videos to watch.

Normalization of the Schema (3NF)

Our schema is already normalized and follows 3NF. Here's the verification:

1.1NF (No repeating groups, atomic attributes)
All attributes are atomic (no multi-valued fields). Any many-to-many relationships (e.g., Recipe ↔ Ingredients) are broken out into separate tables (ComposedOf), ensuring no multi-valued attributes appear in the same row.

2. 2NF (No partial dependencies)
Every non-key attribute fully depends on the entire primary key. Example: In Review(ReviewID, RecipeID, UserID, Stars, Comments), all attributes depend only on ReviewID.

3. 3NF (No transitive dependencies)
Every non-key attribute depends only on the primary key.
Example: In Recipe(RecipeID, RecipeTitle, Directions, Source), all attributes are directly dependent on RecipeID, with no indirect dependencies.

Relational Schema

```
User(
  UserID: INT [PK],
  Username: VARCHAR(255)
);

Review(
  ReviewID: INT [PK],
  RecipeID: INT [FK to Recipe.RecipeID],
  UserID: INT [FK to User.UserID],
  Stars: INT,
  Comments: VARCHAR(1000)
);

Action(
  ActionType: VARCHAR(255),
  Time: TIMESTAMP,
  UserID: INT [FK to User.UserID],
  RecipeID: INT [FK to Recipe.RecipeID]
);

Recipe(
  RecipeID: INT [PK],
  RecipeTitle: VARCHAR(255),
  Directions: VARCHAR(1000) ,
  Source: VARCHAR(255)
);
```

```
Ingredients(
  IngredientID: INT [PK],
  Name: VARCHAR(255),
  CaloriesPerUnit: DECIMAL
);

ComposedOf(
  RecipeID: INT [FK to Recipe.RecipeID],
  IngredientID: INT [FK to Ingredients.IngredientID],
  Quantity: VARCHAR(255)
);

Video(
  VideoID: INT [PK],
  Title: VARCHAR(255),
  link: VARCHAR(255),
  RecipeID: INT [FK to Recipe.RecipeID]
);
```