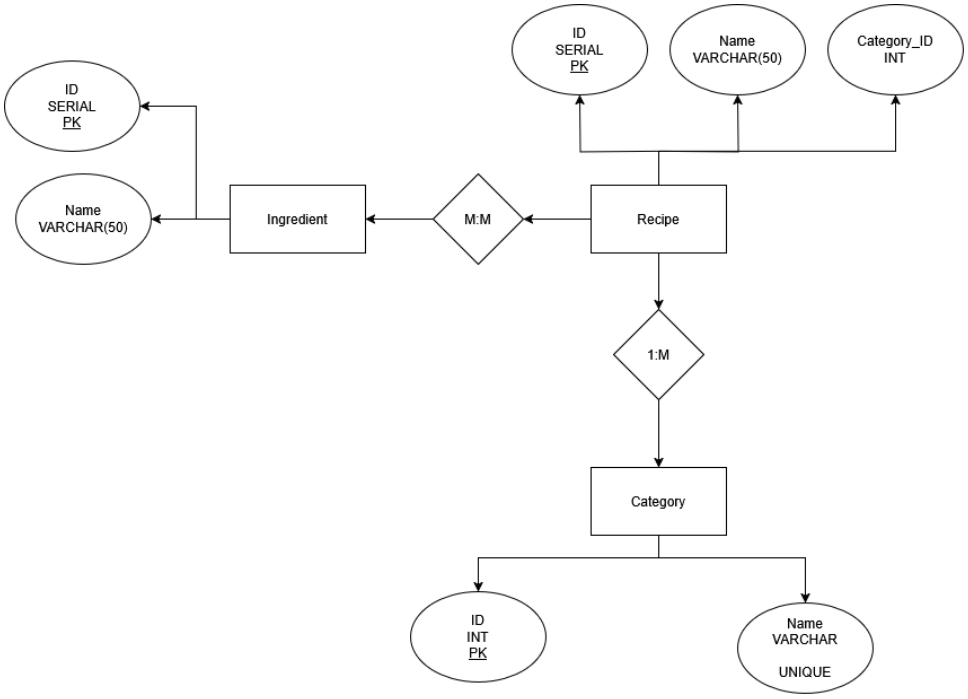
Jere Mikkonen, Jesse Pirinen, Rasmus Puromäki ja Eetu Hokkanen OTMI24SP
https://github.com/jere-bot/SQL-Loppuprojekti

Step 1: Database Design

Schema Overview

- **High-Level Description**: Give a concise explanation of your schema's purpose (e.g., to store recipes, ingredients, categories, etc.).

  Teimme 3 eri taulukkoa Recipe, Category ja ingredient joille annoimme suhteet M:M 1:N. Reseptissä voi olla monta ainesosaa ja ainesosa voi kuulua moneen reseptiin. Reseptillä voi olla yksi kategoria, mutta kategorialla voi olla monta reseptiä.



Entities and Relationships

- **List of Entities**: Describe each table (e.g., Recipe, Ingredient), including their main attributes.

  Recipe: Atribuutit on ID, Name ja Categoty_ID. Recipe tarkoitus on tallettaa itse reseptin nimi, Primary Key ja kertoa mihin ketegoriaan se kuuluu.

  Ingredient: Atribuutit on ID ja Name. Ingredient tarkoitus on tallentaa ainesosan nimi jonka maksimi 50 merkkiä ja sen Primary Key eli ID.

  Category: Atribuutit on ID ja Name. Tarkoituksena on tallentaa kategorian nimi esimerkiksi juoma ja sitten asettaa ID.

- **Relationship Descriptions**: Explain the relationships (one-to-many, many-to-many, etc.) and how they are represented (junction tables, foreign keys, etc.).

  Recipe ja Ingredient on M:M, koska reseptillä voi olla monta ainesosaa ja ainesosa voi olla monessa reseptissä.

  Recipe ja Category on 1:N, koska Reseptillä voi olla 1 kategoria ja kategorialla voi olla monta reseptiä.

Normalization & Constraints

- **Normalization Level**: State the level of normalization (e.g., 3NF) you aimed for and why.
  1NF, koska projekti on melko pieni ja tarkoituksena oli pyrkiä pitämään tietokanta melko yksinkertaisena.

- **Constraints**: Discuss your use of primary keys, foreign keys, NOT NULL, UNIQUE, etc.

  Resepti ja Ingredient taulun välissä M:M suhteessa käytettiin ID yhdistämistä, mikä helpotti huomattavasti taulujen kanssa. Nimien kanssa käytettiin UNIQUE komentoa ja ID luomisessa käytettiin SERIAL komentoa.

Design Choices & Rationale

- **Reasoning**: Justify **why** you structured the schema the way you did. For instance, "We used a junction table for Recipe-Ingredient because it's a many-to-many relationship."

  Koska mielestämme oli loogista antaa resepteille tälläiset ominaisuudet.

- **Alternatives Considered**: Note any alternative designs you evaluated and why you chose not to implement them.
  Mietimme esimerkiksi valmistusohjeiden lisäämistä

Step 2: Database Implementation

Table Creation

- **SQL Scripts Overview**: Provide or reference your CREATE TABLE statements.

  Nämä löytyy "create.sql" projektitiedostosta.

- **Explanation of Key Fields**: For each table, briefly explain the most important columns and their data types.

  Category – taulussa tärkeää oli kentät ID ja Nimi

ID – SERIAL PK

Name – VARCHAR(50) UNIQUE

Recipe – taulussa tärkeää oli:

ID – SERIAL PK

Name – VARCHAR(50)

Category_ID INT

Ingredient – taulussa tärkeää oli:

ID - SERIAL PRIMARY KEY,

Name - VARCHAR(50) UNIQUE

- **Constraints**: Show how you implemented the constraints (e.g., PRIMARY KEY, FOREIGN KEY, etc.) in SQL.

  Primary Keyksi laitoimme taulujen ID kolumnin.

Data Insertion

- **Sample Data**: Summarize the sample data you inserted. For example, 5 ingredients, 3 recipes, multiple categories, etc.

  Laitoimme sample dataksi 3 eri reseptiä, joissa oli 3-5 ainesosaa ja jokainen kuului yhteen kolmesta kategoriasta (Pääruoka, Jälkiruoka, Välipala).

- **INSERT Statements**: Provide or reference your data insertion scripts (INSERT INTO ...).

  Löytyy projektin “Insert.sql” - tiedostosta.

Validation & Testing

- **Basic Queries**: Document a few test queries you ran using psql or another tool (e.g. pgAdmin) to confirm your data was inserted correctly.

  Select * from ingredients

  Select * from recipe

- **Results**: Summarize the outcome (e.g., “Query shows 3 recipes in the Recipe table. Each has multiple entries in RecipeIngredient. No foreign key violations.”)

Step 3: .NET Core Console Application Enhancement

EF Core Configuration

- **Connection String**: Describe where and how you manage the database connection string (e.g., appsettings .json, environment variables).

  Connection string on contextin konstruktori.

- **DbContext**: Summarize your e.g. RecipeDbContext setup, how you map entities.

  Käytimme EF-corea contextin luomiseen.

  CRUD Operations: Explain your approach for Create, Read, Update, and Delete methods (e.g., adding new recipes, listing all recipes).

- **Advanced Features**: List any advanced features such as searching by multiple ingredients or retrieving recipes by category.

Luominen – Käytimme entiteettien luomiseen contextin add-methodia.
Poistaminen – Käytimme entiteettien poistamiseen contextin remove-methodia.
Listaaminen – Käytimme entiteettien listaamiseen LINQ-kyselyjä.
Päivittäminen – Etsii reseptin ja sen ainesosat LINQ-kyselyllä.

Advanced Queries & Methods

- **LINQ Queries**: Show or summarize the LINQ queries used for more complex requirements (e.g., “Fetch recipes containing all specified ingredients”).
- **Performance Considerations**: If relevant, mention any indexes or optimizations you added.

```
var newIngredients = ingredientNames
    .Where(name => !existingIngredients.Any(i => i.Name.ToLower() == name))
    .Select(name => new Ingredient { Name = name })
    .ToList();
```

Challenges & Lessons Learned

Suunnittele Database hyvin ennen kuin alat toteuttamaan mitään.

Conclusion

- **Project Summary**: Recap the final state of the project, including major accomplishments (e.g., fully functioning console app integrated with your Postgres database).

  Onnistuimme tekemään visualstudio console sovelluksen, joka tallentaa ja voi hakea tietoa tietokannasta. Projekti itsestään oli yllättävän mukava tehdä, mutta olisi voinut olla kiva tehdä tämän tyylisiä harjoituksia enemmän, jossa tarvitsee yhdistää visual studio ja tietokanta toisiinsa. Tällä taidolla kuitenkin tulee olemaan jotakin hyötyä tulevaisuuden projekteissa.

- **Future Enhancements**: Suggest any next steps or improvements you would make if you had more time (e.g., add user authentication, implement rating systems, or expand the domain).

  No tietenkin tällaiseen sovellukseen voi lisätä vaikka Kuinka paljon kaikkia eri toimintoja, mutta suurimpana asiana tulee mieleen valmistusohjeet ja sovelluksen käytettävyyden parantaminen.

Instructions for Use

1. **Fill Out Each Section**: Provide clear, concise, and **original** explanations.
2. **Include Screenshots or Snippets**: If it helps clarify a point (e.g., menu output from the console app, partial code snippets).
3. **Maintain Professional Formatting**: Use consistent headers, bullet points, and references.