CSCI 126 Project Report

Juan Marquez and Yubo Zhou

05/10/2023

INTRODUCTION

The objective of the project is to create a prototype of a relational database management system (RDBMS) associated with food, nutrition, and people. By utilizing this RDBMS, we can answer some questions such as how to create a healthy diet, choose a reasonable amount of food, and figure out the essential nutrition for certain age groups of people. We believe that promoting living a healthy life is significant. The domains of the data in this project include food, food category, nutrition of the food, people, and essential nutrition of people.

METHODS

The method to construct this project follows a regular pathway to build up an RDBMS, including formulating functional dependencies of the domains, constructing the schema, creating a UML diagram for the relationship, fixing violations of the normalization, creating the DBMS, and utilizing the DBMS to answer our questions.

UML DIAGRAMS

According to the UML diagrams (Figure 1), R1:People and R2: people_daily_nutrition is one-to-one relationship; R1: people and R6: people_daily_calorie is one-to-one relationship; R2: people_daily_nutrition and R3: food is one-to-many relationship; R3: food and R4: food_nutrition is one-to-many relationship; R3: food and R5: food_group is a many-to-one relationship.

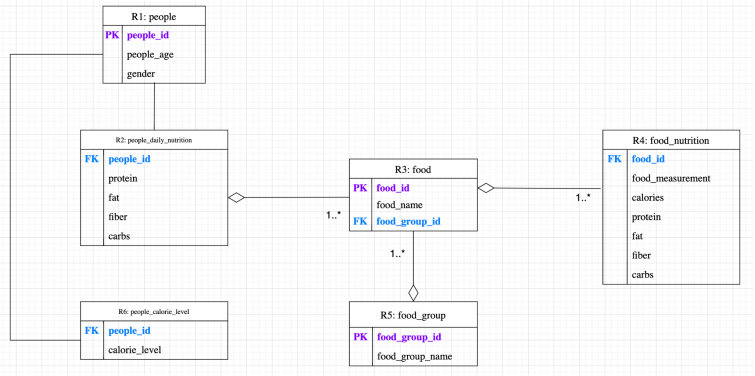

Figure 1 UML Diagram

SCHEMAS

```
people (
      people_id int,
      age   int,
      gender varchar(255),
      primary key(people_id)

);

people_daily_nutrition(
      people_id int,
      protein int,
      fat int,
      fiber int,
      carbs int,
      foreign key (people_id)
            references people(people_id)
);

food(
      food_id int,
      food_name varchar(400),
      food_group_id int,
      primary key(food_id),
      foreign key (food_group_id)
      references
      food_group(food_group_id)
);
```

```
food_nutrition(
      food_id int,
      food_measurement int,
      measurement_unit varchar(400),
      calorie int,
      protein int,
      fat int,
      fiber int,
      carbs int,
      foreign key (food_id) references food(food_id)
);



food_group(
      food_group_id int,
    food_group_name varchar(400),
    primary key(food_group_id)
);

people_calorie_level (
      people_id int,
      calorie_leve int,
      foreign key (people_id) references
      people(people_id)
);
```

A total of 6 domains were added and they each work hand to hand with data management. The first table, people, will contain each person's id, age, and gender. People_daily_nutrition will contain the people_id from the people table and the different nutrition levels which are protein, fat, fiber, and carbs. People_calorie_level will contain the people's id and their calorie level. In the food table, it'll contain different foods and ids for each one. It'll also contain a food group id, which connects each food item with a category. Food nutrition will contain the nutritional information of each food item. Each item is referenced with the food id and we'll be storing the food measurement, the measurement unit, calorie amount, protein, fat, fiber, and carbs amount. The final table is the food group, which contains a unique key for each food group. This id is referenced in the food table.

FUNCTIONAL DEPENDENCY & NORMALIZATION

FDs:

People_ID -> Age_Group, Gender

People_ID -> Protein, Fat, Fiber, Carbs

People_ID-> Calorie_level

Food_ID -> Food_Name, Food_Group_ID

Food_ID, Food_Measurement -> Food_Calorie, Protein, Fat, Fiber, Carbs

Food_Group_ID -> Food_Group_Name

Normalization

```
people (
      people_id int,
      age   int,
      gender varchar(255),
      primary key(people_id)
);
```

```
people_daily_nutrition(
      people_id int,
      protein int,
      fat int,
      fiber int,
      carbs int,
      foreign key (people id)
            references people(people_id)
);
```

```
people_calorie_level (
     people_id int,
     calorie_leve int,
     foreign key (people_id) references
     people(people_id)
);
```

The table of people should follow the 3NF, BCNF, and 4NF because it does not contain transitive dependency, no non-prime attributes determine the prime attribute, and no multiple-valued dependency.

The table of people_daily_nutrition should follow the 3NF, BCNF, and 4NF because it does not contain transitive dependency, no non-prime attributes determine the prime attribute, and no multiple-valued dependency.

The table of people_calorie_level should follow the 3NF, BCNF, and 4NF because it does not contain transitive dependency, no non-prime attributes determine the prime attribute, and no multiple-valued dependency.

```
food(
      food_id int,
      food_name varchar(400),
      food_group_id int,
      primary key(food_id),
      foreign key (food_group_id)
      references
      food_group(food_group_id)
);
```

The table of people_daily_nutrition should follow the 3NF, BCNF, and 4NF because it does not contain transitive dependency, no non-prime attributes determine the prime attribute, and no multiple-valued dependency.

```
food_nutrition(
      food_id int,
      food_measurement int,
      measurement_unit varchar(400),
      calorie int,
      protein int,
      fat int,
      fiber int,
      carbs int,
      foreign key (food_id) references food(food_id)
);
```

The table food_nutrition should follow 3NF, BNCF, and 4NF because there's no transitive dependency, no non-prime attribute determines a prime attribute, and there's no multiple-valued dependency. Unlike the other tables, food nutrition contains a super key, which would be food_id and food_measurement.

```
food_group(
      food_group_id int,
    food group name varchar(400),
    primary key(food_group_id)
);
```

The table of food_group should follow 3NF, BNCF, and 4NF because transitive dependency does not exist, there are no non-prime attributes that determine the prime attribute, and no multiple-valued dependencies.

CONSTRAINTS

While developing our tables for our data, we had to specify rules for some attributes. In our first table, People, we specified that people_id should be a primary key. The reason is that each person will have a unique identification number which will make it easy to reference them. That primary key is referenced in tables people_daily_calorie_nutrition and people_calorie_level as Foreign Keys. The reason why they are foreign keys is that we can easily reference data from one table to another. Therefore, allowing our tables to be connected by reference.

Other mentions of primary keys in the database would be in the Food table, where food_id is a primary key, and in the Food_Group table, where the attribute food_group_id is a primary key. Food_group_id would be a foreign key in the Food table and the

attribute food_id would be a foreign key in the food_nutrition table.

TRIGGERS

The Trigger (Figure 2) will activate before any data is inserted into the attributes. It'll first check if the id value does not exist. If it returns true, the maximum food_id value is returned and incremented by 1. The newly created food_id value is assigned to the newly inserted food item and added to the table.

While building the subqueries, we encountered an issue when using the insert into the statement in mySQL. The user would be required to insert a food identification number. What if the user does not know the identification number of the last item added? To fix that, a Trigger had to be implemented to avoid the repetition of food id numbers or the adding of out-of-range values.

Another issue when implementing the trigger came out when the triggers were created in an empty table. The data could not be imported to the empty table if triggers before the INSERTION operation existed. To fix that issue, the data had to be imported before the triggers were created.

```
-- Create Triggers for the table before Insertion Operation
-- Trigger for assigning a new primary key for a new tuple for the table "food"
delimiter //
CREATE TRIGGER food_id_creator
BEFORE INSERT ON food
FOR EACH ROW
BEGIN
    DECLARE newID INT;
    IF NOT EXISTS(SELECT food_id FROM Food WHERE Food_ID = new.Food_ID) THEN
        SELECT MAX(food_ID)+1 INTO newID from Food;
        SET NEW.Food_ID = newID;
    END IF;
END;//
delimiter ;
-- -- --
```

Figure 2. Trigger

IMPLEMENTATION

While working on the project, five tools were used to construct our database. This includes kaggle.com, USDA.gov, Microsoft Excel, mySQL Workbench, and diagrams.net. For the data collection, Kaggle was used to collect our food nutrition dataset. For information on people and their nutritional data, USDA.gov provided a collection of data for all ages and gender. To organize the data from both sources, Excel allowed us to fix small syntax issues. Diagrams.net served as a great tool to design UML diagrams and organize the different datasets used. The tool heavily relied on was mySQL Workbench because merging tables based on certain restrictions or returning specified data is possible without having to do it manually.

mySQL

Using mySQL, we developed four queries: Aggregation, Subquery, Insert, and Update. Each of the queries performs and displays a modification that's done to the tables. By using these, it can be seen how the tables work and how the tables interact with each other.

In the first query, the aggregation type used is the count function, which returns an amount from the specified attribute or table. We used the count function to count the amount of food items from each food category and we return that value and each of the categories.

```
13    -- Aggregation Query
14    -- Return the number of food items grouped by the food group name
15 •  SELECT
16    count(food_id) AS the_food_items_count,
17    food_group_name
18    FROM food NATURAL JOIN food_group
19    GROUP BY food_group_name
20    ORDER BY count(food_id) asc;
21
22
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

the_food_items_count	food_group_name
3	Vegetables A-E
3	Vegetables F-P
3	Vegetables R-Z
3	Fruits A-F
3	Fruits G-P
3	Fruits R-Z
8	Dairy Products
8	Fats, Oils, Shortenings
8	Meat
8	Seafood
8	Grains

Figure 3. Aggregation

Next, a subquery (Figure 4) is implemented with the five tables. The tables, people, people_daily_nutrition, and people_calorie_level are natural joined together. This creates a whole table new table which is crossed joined with another table that is formed from the natural join of food and food_nutrition. Using the newly formed table, we return a subquery that contains the daily amount of beef a 28-year-old must consume.

```
1   -- Subquery
2   -- Return the essentailly daily amount of beef if a person is 28 years old.
3   SELECT
4   age,
5   gender,
6   food_name,
7   people_daily_nutrition.protein * (food_nutrition.food_measurement/food_nutrition.protein) as amount,
8   measurement_unit
9   FROM  ( people NATURAL JOIN people_daily_nutrition NATURAL JOIN people_calorie_level) CROSS JOIN (food NATURAL JOIN food_nutrition)
10  WHERE age = 28 AND food_name = 'beef';
11
12
13  -- Aggregation Query
14  -- Return the number of food items grouped by the food group name
15  SELECT
16  count(food_id) AS the_food_items_count,
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

age	gender	food_name	amount	measurement_unit
28	Female	Beef	6.0000	oz.
28	Male	Beef	7.3043	oz.

Figure 4. Subquery

The Insert Figure and Update queries (Figure 5) are implemented to work with each other. In the insertion query, we insert a new food item into the food table. As mentioned before, the Trigger will act upon the newly inserted data and it'll create a new identification number for that food. The update query is used to change something from a table or database. In this case, it's used to delete the newly added item. The recently created identification number is used to reference and delete the food item. Once deleted, it's verified that the food item we inserted no longer exists in the table.

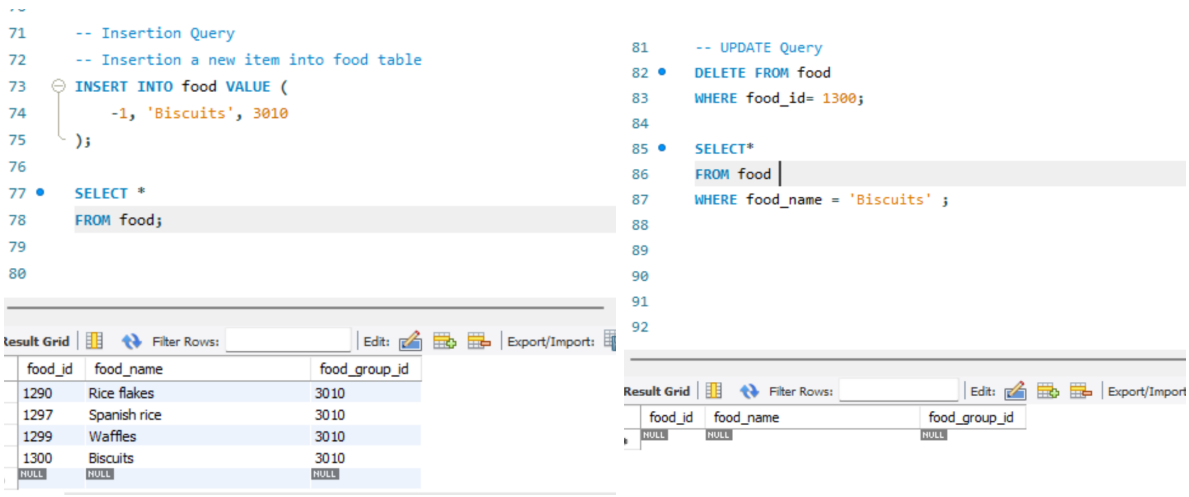

Figure 5. Insertion & Update

RESULTS

As mentioned beforehand, the questions we hope to answer are:

- How to create a healthy diet?
- How to choose a reasonable amount of food?
- How can we figure out the essential nutrition for certain age groups?

CREATING A HEALTHY DIET

To answer this question, the recommended calorie of the age group between 19 and 25 was chosen as the reference to create a diet of the day. The food choices are salmon as the protein source, oatmeal as the fiber source, rice as the carbohydrate source, banana, and ice cream as supplementary sources. After running the queries to calculate the necessary amount of each food (Figure 6), to meet the essential 2800 calories per day requires 9.9 oz of salmon, 6.8 cups of oatmeal, 1.3 cups of rice, 1 banana, and 1 cup of ice cream. Compared with the reference of daily nutrition of that age group, the breakdown of the nutrients of the diet did return a reasonable range of protein, fat, and fiber. However, the amount of carbohydrates was much higher than expected.

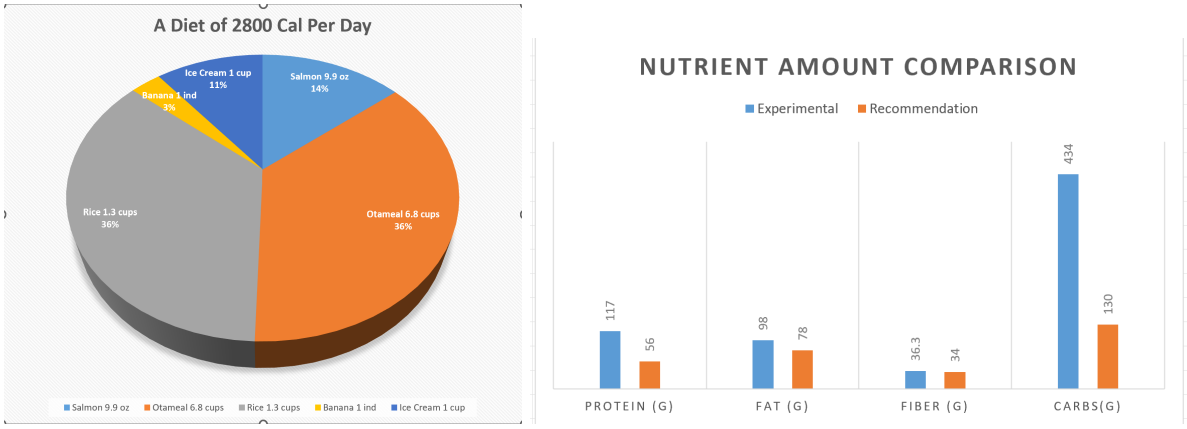

Figure 6. Food and Nutrient Amount

CHOOSING A REASONABLE AMOUNT OF FOOD

Since the database includes the reference to people's daily nutrition and the details of food nutrients, it can be utilized to choose a reasonable amount of food for certain age groups. To testify this function, a query was run to return the amount of beef if a 28-year-old choose it as the protein source for the daily supply. The amount of beef returned 6.0 oz for the female and 7.3 oz for the male, which is in a reasonable range.

FIGURING OUT THE ESSENTIAL NUTRITION FOR CERTAIN AGE GROUPS

It's given that not all age groups eat the same amount of food. This is due to the amount of calories they can consume. The younger the person is, the fewer calories they consume, and the older the person is, the more calories they will consume. Figuring out the essential nutrition for a specified age group will require to have knowledge of their calorie intake. Using mySQL, one can return the calorie amount for a specified age group.

For instance, let's say we want to know the calorie amount ages 19 to 25 consume on a daily basis. Using mySQL, we see that the amount of calories they can consume is 2,800.

CONCLUSION

The project successfully constructed a prototype of a relational database management system (RDBMS) associated with food, nutrition, and people. The process of developing the project provided an opportunity for us to utilize the knowledge that we learned from the course and strengthen our problem-solving ability. The project included research for the topic, design of the database, implementation, and fix and modification. The final version of the RDBMS ran as expected and provided proper answers to the questions that

we brought up.

For the future improvement and application of this project, it can be added more features by including more data domains related to food and nutrition, such as providing a function to check the nutrition of popular items on some popular restaurants' menus.

REFERENCES

Kaggle-Nutritional Facts for most common foods

<https://www.kaggle.com/datasets/niharika41298/nutrition-details-for-most-common-foods>

USDA- Estimated Calorie Needs per Day by Age, Gender, and Physical Activity Level.

<https://www.fns.usda.gov/estimated-calorie-needs-day-age-gender-and-physical-activity-level>