<BIPro Project 3>

20180410 유정은
20180475 이시하
20180586 정다연

1. Introduction

인스턴트와 패스트푸드 섭취량이 늘어나며 영양 불균형이 심각한 문제로 떠올랐습니다. 이때, 영양 불균형이 다양한 질병의 원인이 될 수 있기 때문에 건강한 식단에 대한 관심이 커지고 있는 추세입니다. 또한, 개인 맞춤화 시대의 흐름을 따라 개인의 신체 정보, 질병 정보를 기반으로 식단을 평가하고 음식을 추천해주는 서비스의 수요가 있을 것이라고 생각했습니다. 그래서 저희는 개인에게 맞춤화된 식단 평가와 식단 추천 기능을 한데 모은 All-in-one Diet Mate, 줄여서 ADM을 개발했습니다.

ADM은 입력 받은 개인의 나이, 성별, 키, 몸무게, 운동 수준을 바탕으로 계산한 활동 대사량과 권장 섭취량을 입력된 식단에 포함된 영양소양과 비교하여 식단을 평가해줍니다. 그리고 입력된 식단에 특정 영양소가 부족하게 또는 과하게 포함되어 있다면, 어떤 질병에 걸릴 수 있는지 미리 알려줌으로써 예방할 수 있도록 돕는 기능도 있습니다. 마지막으로, 개인이 가지고 있는 질병과 연관된 영양소 부족 또는 과잉이 입력된 식단에서도 동일하게 나타난다면 사용자에게 한번 더 주의를 줌으로써 더욱더 개인 맞춤화된 ADM을 만들었습니다.

2. DataBase Reference

2-1. FoodData Central (https://fdc.nal.usda.gov/download-datasets.html)

Food (음식 id, 음식 이름)
Nutrient (영양소 id, 영양소 unit, 영양소 이름)
Food_Nutrient (음식 id, 영양소 id, 음식 100g에 포함된 영양소량 [g])

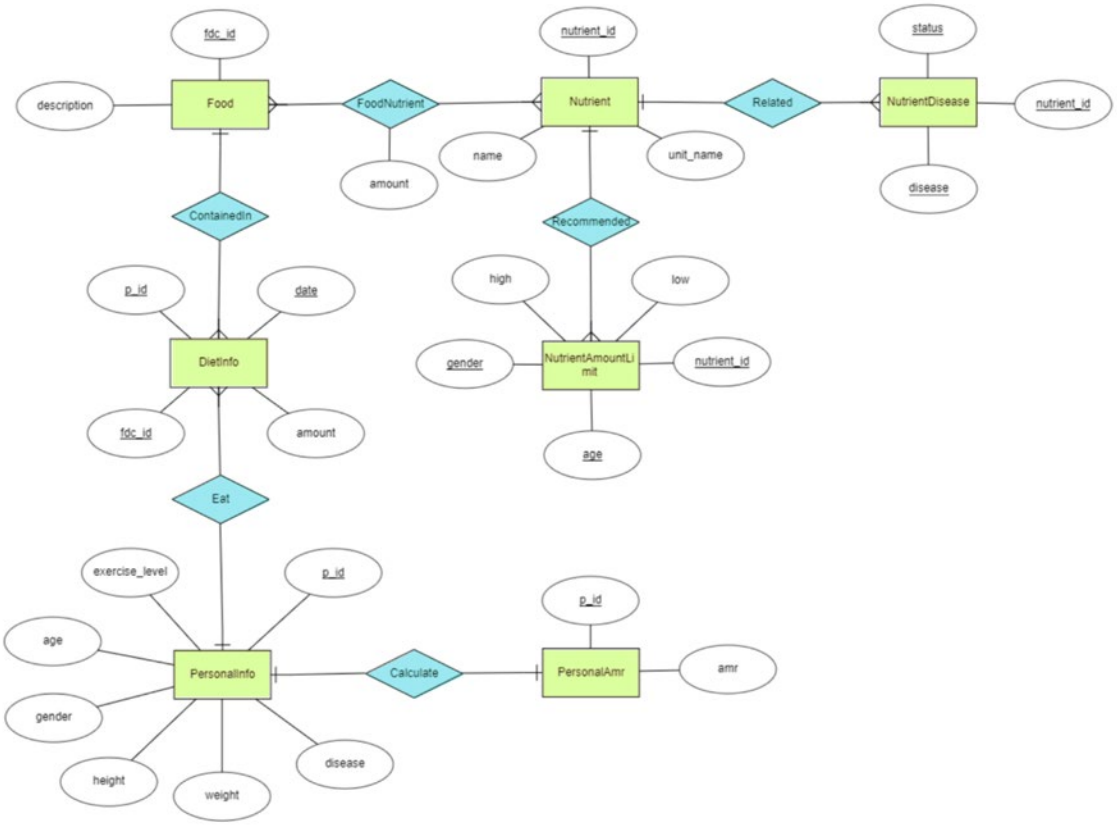
2-2. 2015 한국인 영양소 섭취기준 활용 가이드북, 보건복지부

Recommended_Nutrient (나이대, 성별, 영양소 id, 섭취 하한선[g], 섭취 상한선[g])

2-3. BIPro team 1

직접 의학 정보 검색을 통해 nutrient의 과잉 또는 부족과 연관된 disease를 DataBase로 제작했습니다.
Nutrient_Disease (영양소 id, 상태(over/under), 질병 이름)

3. ER Diagram

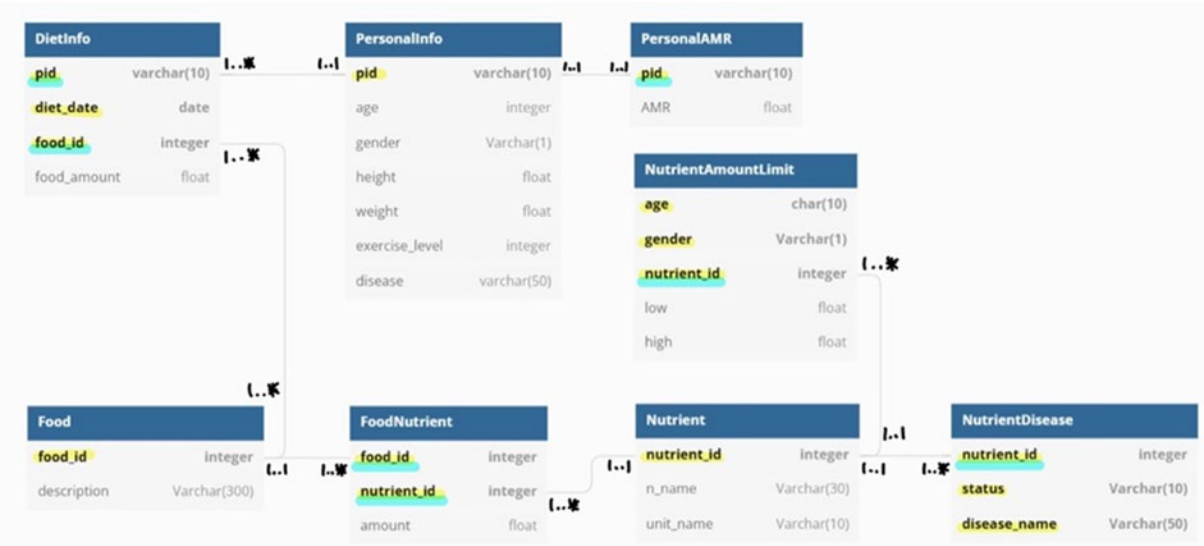


총 7개의 entitiy와 6개의 relationship이 있습니다. 각 entity는 연두색 직사각형, relationship은 파란색 마름모, attribute은 흰색 타원으로 만들었습니다. 또, 각 entity들을 잇는 선에 기호를 통해 multiplicity를 표시했습니다.

Food는 Nutrient를 포함하고 있으므로 Nutrient와 연관되어 있습니다. Food와 Nutrient의 relation인 FoodNutrient에는 특정 음식 100g에 포함되는 영양소 정보가 저장되어 있습니다. 이때 한 음식에 여러 영양소가 포함되고, 한 영양소가 여러 음식에 포함될 수 있으므로 many-to-many입니다.
NutrientDisease는 특정 영양소가 부족하거나 과잉인 경우 발생할 수 있는 질병 정보를 담고 있으므로 Nutrient와 연관되어 있습니다. 한 영양소가 여러 질병의 원인이 될 수 있고, NutrientDisease의 한 row는 한 영양소에 대한 정보만 저장하고 있으므로 many-to-1입니다.
NutrientAmountLimit는 나이, 성별에 따라 한 영양소의 상한 섭취량과 하한 섭취량의 정보를 담고 있으므로 Nutrient와 연관되어 있습니다. 한 영양소의 상한 섭취량과 하한 섭취량이 다양한 연령대와 성별에 따라 다르고, NutrientAmountLimit의 한 row는 한 영양소에 대한 정보만 저장하고 있으므로 many-to-1입니다.
PersonalInfo는 사용자의 id, age, gender, weight, height, exercise_level, disease의 신체 정보와 질병 정보를 담고 있습니다.

DietInfo는 특정 사용자가 특정한 날 먹은 음식에 대한 정보를 담고 있으므로 Food와 PersonalInfo와 연관되어 있습니다. DietInfo의 한 row는 한 사용자와 한 음식에 대한 정보만 저장하고 있으며, 한 음식이 DietInfo의 여러 row에 포함될 수 있고 한 사용자에 대한 정보는 DietInfo의 여러 row에 저장되므로 1-to-many입니다.
PersonalAMR은 특정 사용자의 활동대사량(AMR)을 저장하므로 PersonalInfo와 연관되어 있습니다. 한 사용자는 하나의 AMR과 대응되므로 1-to-1입니다.

4. Relational Schema



Primary key는 굵은 글씨에 노란색 형광펜으로 표시했습니다. 이때 Primary key이면서 Foreign key 인 attribute는 파란색 형광펜으로 표시했습니다. (위 schema에서 Foreign key이면서 Primary key가 아닌 attribute는 없습니다.)

Atomic attribute value로 바꾸기 위해 Multivalue attribute 일 수 있는 NutrientDisease 정보를 아래와 같이 수정했습니다.

ID	status	disease
1089	over	iron overload cardiomyopathy
1089	over	Hemochromatosis
1089	under	iron deficiency anemia
1089	under	fatigue
1095	over	zinc toxicity
1095	over	diarrhea

Partial depenecncy는 없었습니다.
Transitive dependency를 제거하기 위해 PersonalAMR 테이블을 만들었습니다.
이렇게 하여 BCNF를 만족하게 structure를 구성했습니다.

5. SQL DDL

아래와 같이 CREATE TABLE을 통해 필요한 table을 모두 만들었습니다.

CREATE TABLE personal_info(id varchar(10) primary key, age integer, gender char(1), height float, weight float, exercise_level integer, disease varchar(50));

CREATE TABLE personal_amr(id varchar(10) primary key, amr float, foreign key (id) references personal_info(id));

CREATE TABLE food(fdc_id integer primary key, description varchar(300));

CREATE TABLE diet_info(id varchar(10), date date, fdc_id integer, amount float, primary key (id,date,fdc_id), foreign key (id) references personal_info(id), foreign key (fdc_id) references food(fdc_id));

CREATE TABLE nutrient(id integer primary key, name varchar(30), unit_name varchar(10));

CREATE TABLE food_nutrient(fdc_id integer, nutrient_id integer, amount float, primary key (fdc_id,nutrient_id), foreign key (fdc_id) references food(fdc_id), foreign key (nutrient_id) references nutrient(id));

CREATE TABLE recommended_nutrient(age char(10), gender char(1), nutrient_id integer, low float, high float, primary key (age,gender,nutrient_id), foreign key (nutrient_id) references nutrient(id));

CREATE TABLE nutrient_disease(id integer, status varchar(10), disease_name varchar(50), primary key (id,status,disease_name), foreign key (id) references nutrient(id));

6. SQL DML (Query)

5-1. SELECT (column expression) FROM (Table name) WHERE (condition)

- 로그인, 음식 이름 검색 등 db에서 필요한 정보를 가져올 때 사용했습니다.

5-2. INSERT INTO (Table name) VALUES (values)

- 회원가입 정보 입력, 식단 정보 입력 등 DB(Table)에 새로운 row를 추가할 때 사용했습니다.

5-3. UPDATE (Table name) SET (column expression) = (value) WHERE (condition)

- 개인의 신체 정보 수정, 식단의 amount 정보 수정 등 이미 DB에 저장되어 있던 정보를 수정할 때 사용했습니다.

[로그인 및 회원가입]

```
SELECT * FROM personal_info WHERE id=ID

INSERT INTO personal_info
VALUES(id,age,gender,height,weight,exercise_level,disease)

# amr = activity_metabolism_rate(age, gender, height, weight, exercise_level)
INSERT INTO personal_amr VALUES(id, amr)
```

[음식 이름 검색 및 선택]

```
# input: food_name
SELECT * FROM food WHERE description ILIKE '%food_name%'
# Choose a single fdc_id
SELECT * FROM food WHERE fdc_id=fdc_id
# output food information with fdc_id
```

[식단 올리기]

```
# input: id, date, fdc_id, amount
diet_list = SELECT * FROM diet_info WHERE id=id AND date=date

# if fdc_id in fdc_list:
# case 1 : update the amount
# Obtain amount of the food that coresponds to the fdc_id using diet_list
UPDATE diet_info SET (current_amount + amount)
WHERE id=id AND date=date AND fdc_id=fdc_id

# case 2 : replace the amount
UPDATE diet_info SET amount WHERE id=id AND date=date AND fdc_id=fdc_id

# else:
INSERT INTO diet_info VALUES(id,date,fdc_id,amount)
```

[식단 확인하기]

```
# input : id, date
diet_list = SELECT fdc_id,amount FROM diet_info WHERE id=id AND date=date
# Use 'lookup_food' to get the food name of the fdc_id.
```

[개인정보 수정]

```
# input : id, (new value)

# age, gender, height, weight, exercise_level :
# update personal_info, and update personal_amr using the new amr information
# All of this is done in the same manner, and we will only show the age.
UPDATE personal_info SET age=(new_value) WHERE id=id
# calculate new amr
UPDATE personal_amr SET amr=amr WHERE id=id

# disease : Just update the personal_info
UPDATE personal_info SET disease=(new_value) WHERE id=id
```

[영양소 계산]

```
diet_list = SELECT * FROM diet_info WHERE id=id AND date=date

food_nutrient_info = SELECT nutrient_id,amount
FROM food_nutrient WHERE fdc_id=fdc_id
```

[식단 분석]

```
SELECT age,gender,disease FROM personal_info WHERE id=id

SELECT amr FROM personal_amr WHERE id=id

SELECT nutrient_id,low,high FROM recommended_nutrient
WHERE gender=gender AND age=age

SELECT id,name,unit_name FROM nutrient

SELECT * FROM food

SELECT fdc_id,amount FROM food_nutrient
WHERE nutrient_id=nutrient_id ORDER BY amount DESC

SELECT id,nutrient_disease
WHERE disease_name ilike '%disease_name%' AND status='under'

SELECT id,nutrient_disease
WHERE disease_name ilike '%disease_name%' AND status='over'

SELECT disease_name FROM nutrient_disease WHERE id=id AND status = 'under'

SELECT disease_name FROM nutrient_disease WHERE id=id AND status = 'over'
```