**Project Iteration 3**

**Junyu Mou**

**10/22/2020**

Specialization-Generalization relationship:

The first use case for my database is Food nutrient facts recorded. According to the use case， a nutrient dictionary will be established, users insert the nutrient records for the first time and; users can use the history for future records. The use case’s detailed information is:

1. The user explores the internet to find a food’s nutrient facts
2. The user connects the database and insert a line of record
3. The database will record the nutrient facts when the first time the user input it

After learned the specialization-Generalization relationship and look again the use case, the use case can be further detailed and specialized. Something that would be useful for this user case is to have different kinds of food. For example, the food can be further classified into difficult types: meat, vegetable, fruit, nuts etc. Users may be happy to see the food in different groups while they are finding the food nutrient facts from history, because classified history will save users’ time if they can directly find what they need from the specific category. For example, if a user wants to find the nutrient fact for beef, he or she can go to meat group and doesn’t have to try to find it from all foods. Also, classified foods records help users to solve problems such as: I want a fruit, but I don’t want too much calorie, what should I have? The user can go to fruit group and explore the history to find the fruit he or she eaten before with low calorie. I modified the use case as follows:

1. The user explores the internet to find a food’s nutrient facts

2. The user connects the database and insert a line of record

3. The database will record the nutrient facts *into different categories based on types of the food* when the first time the user input it

Second use case is Food eaten tracking use case:

1. User prepares food, and weight each food
2. The database records food name, weight of food
3. The database calculates amount of nutrition and records the result

This use case saves each food records in the database, food records in this use case are every kind of food’s nutrient fact. For example, a user eats one apple as snack between meals, the user weight the apple, insert the weight of the apple and name of food (apple) into the database ass one record. The database is designed for a person who wants to build muscle or lose weight. For people who wants to change their body figure, eating several meals a day is a good habit. A person who wants to gain weight unusual 6, 7 or 8 meals a day to stable the levels of blood glucose. On the one hand, stability of levels of blood glucose keep providing nutrient to muscle in a relatively constant rate that can give muscle enough energy to grow. On the other hand, stability of levels of blood glucose can reduce body fat, because body fat is meanly caused by sudden changes in levels of blood glucose. Users may need to insert their food consumptions records into several meals for one day. For a general meal plan, there are 6 meals for one day: breakfast, extra meal 1, lunch, extra meal 2, dinner and supper. With this plan, user will have the records for them to find out calorie for each meal for a day. I modified the use case as follows:

1. User prepares food, and weight each food
2. The database records food name, weight of food *for the specific meal (breakfast, extra meal 1, lunch, extra meal 2, dinner, supper)*
3. The database calculates amount of nutrition and records the result

The third use case is Nutrition plan and body condition use case:

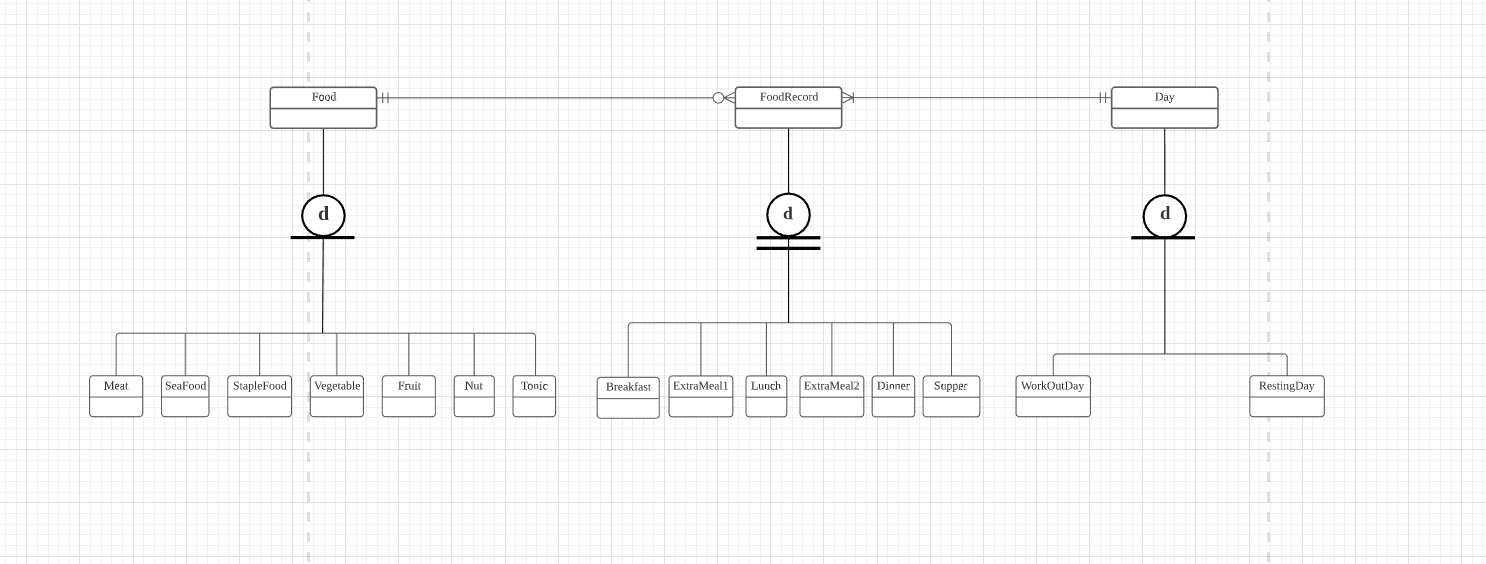
1. The user enter height, body weight, body fat (information can be obtained from specific tool) and date
2. The standard daily nutrition needed is calculated as target (the standard value change with body weight)
3. The total nutrition taken in one day is calculated

For this use case, total nutrition taken for each day is calculated. For the whole exercise period (1 or 2 years), no one can actually train high-intensively every day. Gaining weight or losing is a long-term goal for people. In general, people will have work out days accompanied with one or two resting day. In work out day people are required to eat as planned, but in resting day they can keep the protein target and reduce carbohydrate. And people may need a long period of time to rest, like one month, to restore the body. So, records for days can be classified into records for work out day and records for resting day. I modified the use case as follows:

1. The user enter height, body weight, body fat (information can be obtained from specific tool) and date
2. The standard daily nutrition needed is calculated as target *for different kinds of day: target for work out day and resting day are different, for long resting period the target is the minimum nutrient needed based on body weight* (the standard value change with body weight)
3. The total nutrition taken in one day is calculated and recorded

After modified the use case the structural database rules are modified as follows:

1. Each Food may products many Food records; each food record only has one food.
2. Each day has one or more FoodRecords; each FoodRecord is for one day.
3. A food is a meat, a seafood, a staple food, a vegetable, a fruit, a nut, a tonic or a none of these
4. A day is a work out day, a resting day or none of these
5. A food record is a breakfast record, an extra meal 1 record, a lunch record, an extra meal 2 record, a dinner record or a supper record.

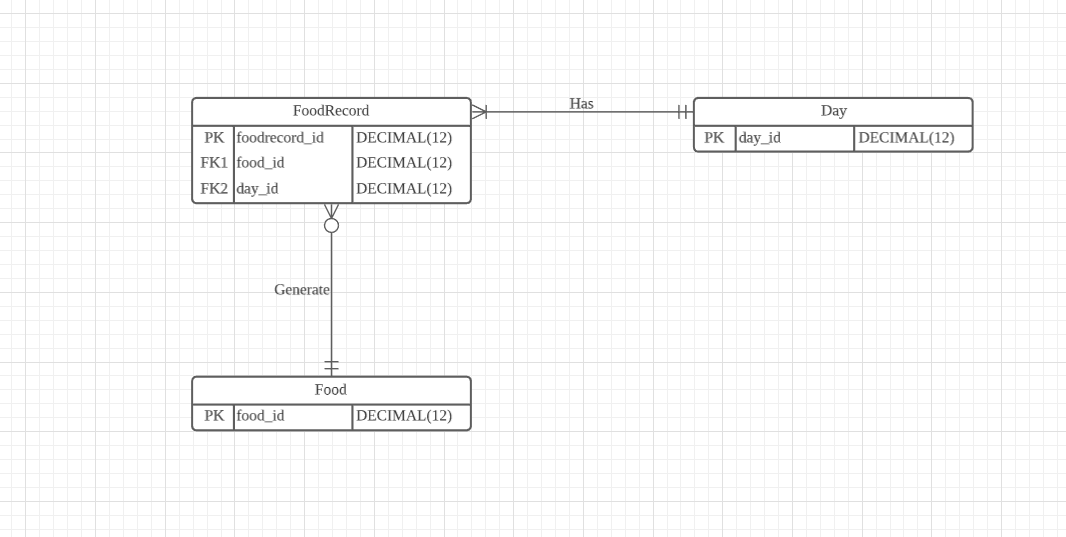


There are several rules about this diagram:

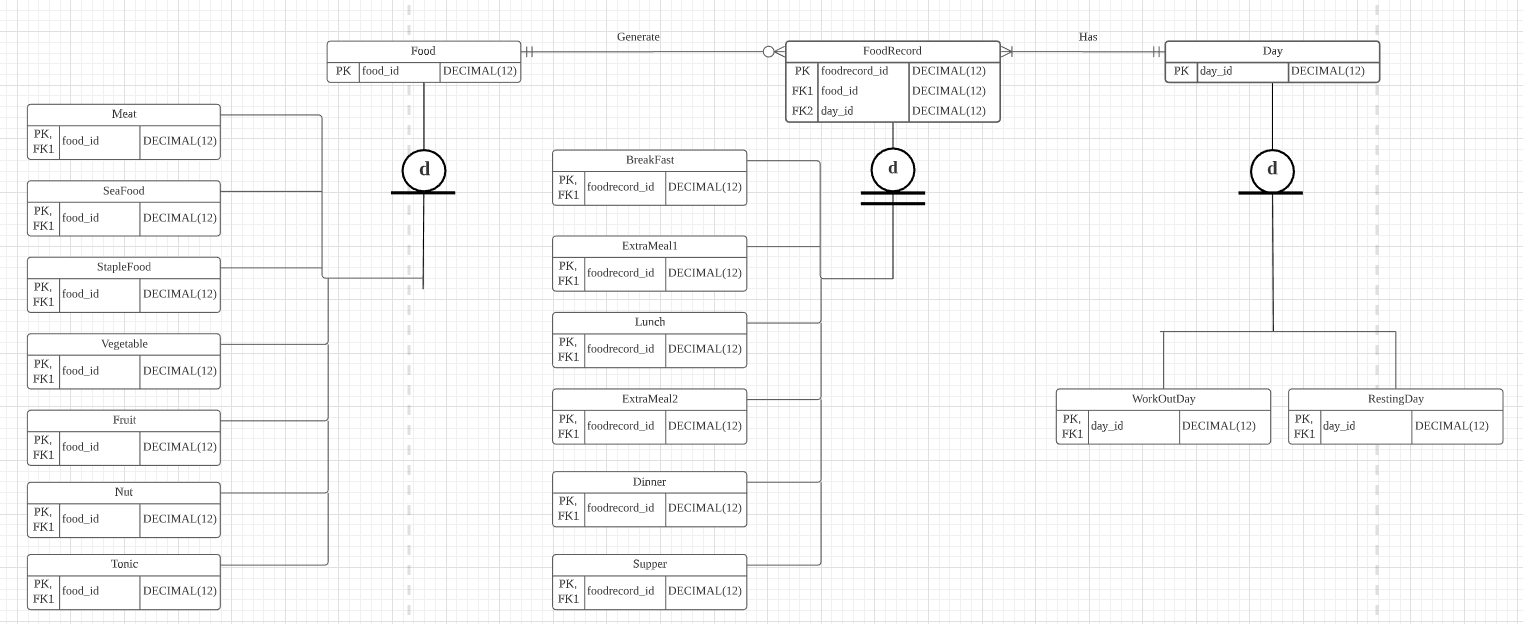
1. One food can only into one category, for example, there is no food can be both vegetable and fruit. The classification of food depends on common sense, someone may say tomato is both vegetable and fruit, this may be scientifically right, but this is no use for food record in this database. The default category for tomato is vegetable. The relationship is disjoint.
2. Some food is not in subtype, for example, a big mac from McDonald’s. A big mac does not belong to any food category. The relation is partial complete.
3. Each one food record must belongs to one of the meals, any thing eaten between breakfast and lunch is food for extra meal 1. The relationship is total complete.
4. No food record for more than one meal. The relationship is disjoint.
5. A day can be a workout day or a resting day or neither. For this database, resting day is still part of the long-term workout plan, but any long period resting is not part of the plan. The relationship is partial complete.
6. A day can’t be both a workout day and a resting day. The relationship is disjoint.

Mapping associative relationships

I used the conceptual ERD to identify the relationships, because it is more direct to detailed the conceptual ERD. The associative relationships in my conceptual ERD are Food/FoodRecord and FoodRecord/Day. The Food/ FoodRrcord relationship is 1: M. One food can generate many food records, but one record can only have one food. The Day/FoodRecord relationship is 1: M. One day can have many food records, but one record can only be in exactly one day. Below are the relationships after further detailed and modification of the conceptual relationships, and I make the primary keys of the DECIMAL (12) datatype to allow for a lot of records.



Mapping Specialization-Generalization Relationships:



Summary and Reflection:

My database is for recording daily diet. Compare nutrition goal and nutrition consumption for users who wants to build muscle. In general, the database records the daily diet, sum the amount of daily nutrition up, records user’s health condition to offer information to user to modify his or her exercise and eating schedule. The database should support a user accessing, and searching information.

This iteration is a good one making the whole project becoming clear and detailed. The physical ERD is based on Oracle Database 11g express edition. I think there might be some difference amount different DBMS, and I don’t want to make information like constrains confusing in this iteration.

After the three iterations, I learned a lot on making a database. I fixed some of my mistakes I make in the earlier iterations, I made the whole project more detailed and precise. I understand the DBMS better, and looking forward to implement the ERD into SQL.