**Structural Database Rules**

The first use case is Food nutrient facts recorded:

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

(It’s like building a food dictionary, so the information will be recorded; users don’t have to google it every time when they eat it.)

After seeing the first use case, there is an entity called Food. The Food entity has several attributes: FoodName, Carbohydrate, Protain, Fat and Calorie.

The 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 is for calculating the actural nutrient based on the Food and weight of the food. There is also an entity for this use case, this entity can be called ActualNutritionRecords, it has several attributes: FoodName (connect to the Food nutrient facts recorded use case), weight, and attributes for nutrition eaten. The first structural database rule comes from combination of Food nutrient facts recorded use case and Food eaten tracking use case: *Each Food may generate many Food records; each food record is only generated by one food.* This structural database rule indicates that for each day a food may be eaten for zero-time, one time or many times. So, each food for a day may have no records, one records or many records. And one record ca only has one food’s information.

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
4. the difference between target daily nutrition and total nutrition eaten is calculated (nutrition eaten minus target, if the result is negative then it means the task is not completed).

From this use case there is an entity named Day which means each day’s nutrition facts. It has attributes like: date, personal information (e.g. BodyWeight), nutrition target information (e.g. ProteinTarget) and nutrition calculated from records (e.g. FatEatenTotal). So, the second structural database rule is: *Each day has one or more FoodRecords; each FoodRecord is for one day.* This structural database rule indicates that each day can have at least one nutrition record; it is not reasonable to keep a Day without any food records. In general, only one nutrition record for one day is not passible (I think no one eat one meal with only one gradient for one day) but I still keep the relationship for any special demand.

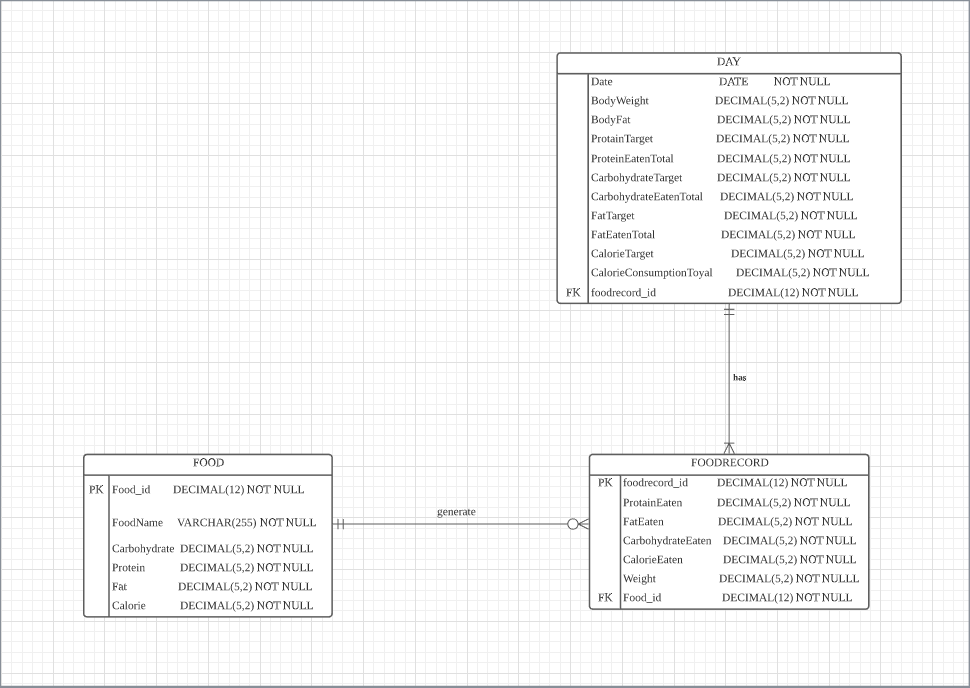
From the three use cases I have thus far, I have two structural database rules:

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.

**Entity-Relationship Diagramming**

1.Each Food may generate many Food records; each food record is only generated by one food.

2.Each day has one or more FoodRecords; each FoodRecord is for one day.



Here is the ERD I came up with for structural database rules, using crow’s foot symbol since it’s easier to be understood. The FoodRecord entity is associated with both Food and Day entity, it is in both structural database rules. The circle combined with the crow’s foot symbol indicates that the relationship is optional and plural representing that Food may generate many FoodRecords. From the perspective of Day, I use a bar followed by the Crow’s foot symbol to indicate the relationship is mandatory-plural to Day. From the perspective of FoodRecord, I use a bar and another bar to indicate the relationship is mandatory-singular to FoodRecord.

**Summary and Reflection**

My database is for users to recording their daily diet which can give them information on health management. The database can record each food the user eats, record the food history, calculate nutrition and offer data such as nutrition goal.

For this iteration, I reviewed the use case from the last iteration and drive the structural database rules and ERD from use cases. The structural database rules and ERD make it more clear about the whole project. For the iteration 1, the idea and plan were just some course without structured, in this iteration I learned how to structure the coarse information into something can be worked with.

The process of making structural database rules and ERD is fine, it is an opportunity to think as a real designer. The outcome is easier than I thought, the ERD and structural database rules should be more complicated in my imagination. During the iteration, I solve problems about the database design, things are getting clear.