Nate Warner CS 466 Fall 2024

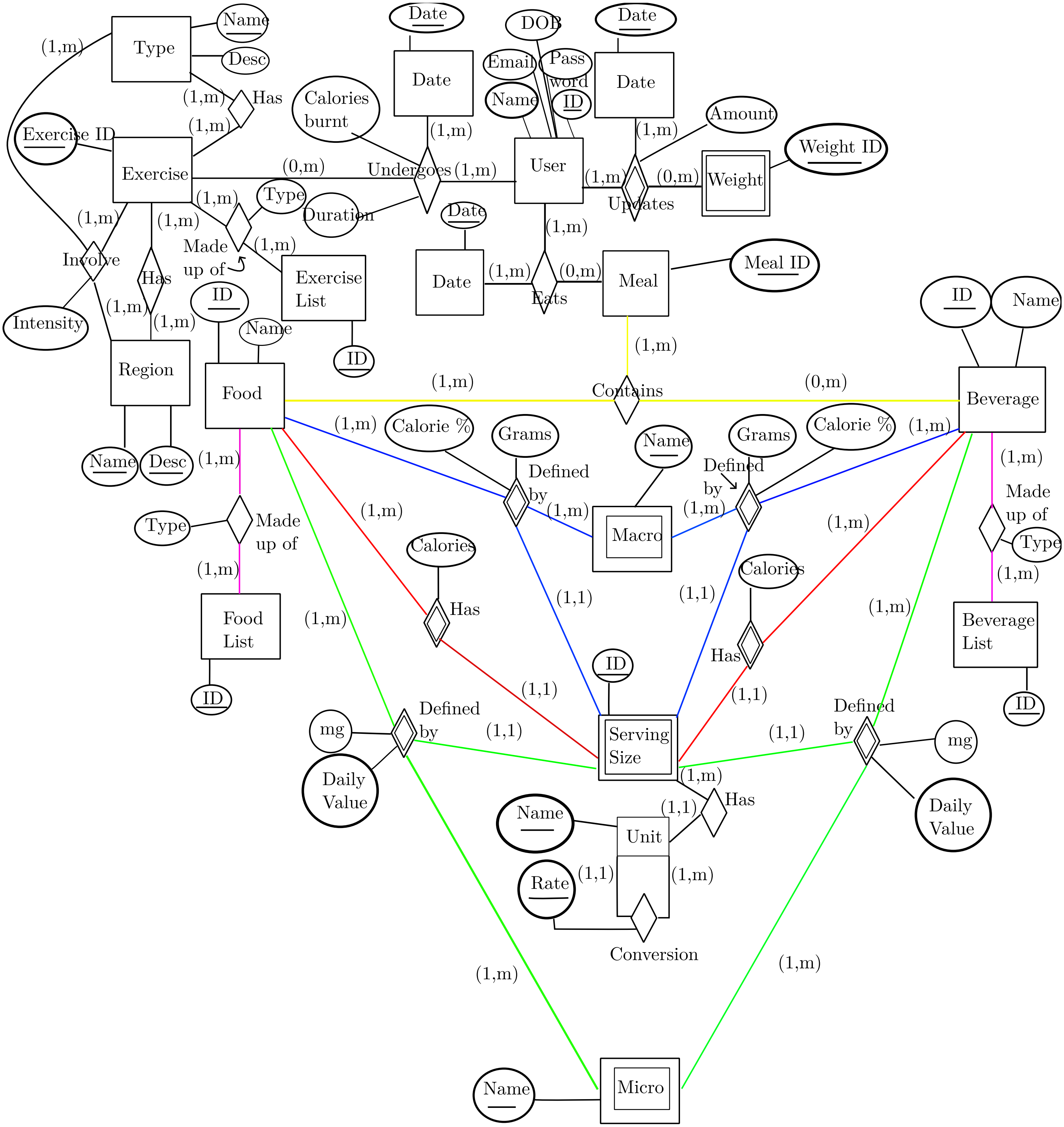
Pset 2 - Due: Friday, September 13

1. You are being employed by a company that offers a fitness tracking service. They are working on a phone app that will allow the user to track what they eat, as well as when/how they work out. Another employee will be designing the user interface, but you are responsible for designing the database. Design an ER diagram to fulfill this goal, making sure to meet all of the requirements. All entities must have an appropriate identifier specified. If a surrogate key is used, explain why a natural key was not appropriate. In the interests of saving space, attributes that are not part of an identifier may be omitted from the diagram, but they should be included and explained in that portion of your submission.

- Every user will have an account, and all of their meals and workouts will be linked to this account. An account is for a single user.
- The user will update their weight periodically. This data must be retained so that progress can be tracked over time.
- The serving size will be some number of some unit (grams, ounces, Tbsp, cups, lbs, etc.). There will be information stored for conversion between different unit types.
- There will need to be a database of foods/beverages. Each of these will have information on serving size, calories per serving, and grams per serving of each of the macronutrients (protein, fat, carbohydrates).
- It should be possible to store information on the quantities of micronutrients or chemicals (i.e. vitamin D, caffeine) that are present in a given food or beverage in significant amounts. Recommended daily values for any of these should be stored, when applicable.
- Each time the user eats, a record of who ate how many servings of what and when they ate it is stored.
- The app needs to allow a user to track their workouts. This includes the type of the workout, its intensity, and its duration.
- When a user tracks their workout, a record is created of who did what type of workout, when, and for how long.

Note:-

If you are not sure what needs to be included for foods/drinks/etc., a good place to look for example data would be the nutritional facts on food packaging.


Type
Name
Desc
(1,m)
Has
(1,m)
(1,m)
Exercise ID
Exercise
Date
Date
Calories burnt
(1,m)
Undergoes
(0,m)
(1,m)
DOB
Email
Pass word
Name
ID
User
Date
Date
(1,m)
Amount
Updates
(1,m)
(0,m)
Weight
Weight ID
(1,m)
Type
Duration
(1,m)
(1,m)
Made up of
(1,m)
Exercise List
ID
Involve
Has
(1,m)
(1,m)
Intensity
Region
Name
Desc
Date
Date
(1,m)
Eats
(1,m)
(0,m)
Meal
Meal ID
ID
Name
Food
(1,m)
Contains
(1,m)
(0,m)
ID
Name
Beverage
(1,m)
Calorie %
Grams
Defined by
(1,m)
Name
Macro
Grams
Calorie %
Defined by
(1,m)
(1,m)
(1,m)
(1,m)
Made up of
Type
(1,m)
Food List
ID
(1,m)
Calories
Has
(1,1)
(1,1)
Calories
Has
(1,m)
(1,m)
Made up of
Type
(1,m)
Beverage List
ID
(1,m)
(1,1)
ID
Serving Size
(1,1)
mg
Defined by
Daily Value
(1,1)
(1,1)
Defined by
mg
Daily Value
(1,m)
Has
(1,1)
Name
Unit
Rate
(1,1)
(1,m)
Conversion
(1,m)
(1,m)
Name
Micro

1 The entities and their attributes

Note:-

All date entities include the Month/day/year, time, and timezone

1. **User**: User has six attributes
   - **Name**: The users first and last name, cannot be used as a natural key because names are not unique
   - **User ID**: The users account ID, this is a surrogate key used to uniquely identify each user
   - **Email**: The users email
   - **Password**: The password that the user uses along with their email to login to the ≈
   - **Date of birth (DOB)**: The users date of birth
2. **All date entities (3)**: All date entites have a natural key *date*, dates are unique, ie 9/15/2024 is only associated with one specific day
3. **Weight**: Weight has one attribute: Weight ID, used as a surrogate key
4. **Exercise**: Exercise has one attribute: Exercise ID used as a surrogate key
5. **Type (exercise type)**: Exercise type has two attributes
   - **Name**: The name of the exercise type (cardio, strength, etc). This can be used as a natural key because these names are unique
   - **Description**: Description of the exercise type
6. **Region**: Region has two attributes
   - **Name**: The name of the region, this can be used as a natural key because these names are unique. There is only one place on your body associated with ”arms”, etc
   - **Description**: Description of the region, what mucles are in that region that are focused in the exercise etc...
7. **Exercise list**: Exercise has one attribute: ID, which is used as a surrogate key. This lists could have unique names, but its safer to just give them an ID.
8. **Meal**: Meal has one attribute: Meal ID, used as a surrogate key. A natural key might work in this case but its safer to have an ID instead, there could be two meals with the same name (maybe they are made differently, have different recipes etc.)
9. **Food**: Food has two attributes
   - **Name**: Name of the food
   - **ID**: ID of the food, used as a surrogate key. Name might work as a natural key but its not guaranteed that the names are unique
10. **Beverage**: Beverage has two attributes
    - **Name**: Name of the beverage
    - **ID**: ID of the beverage, used as a surrogate key. Name might work as a natural key but its not guaranteed that the names are unique
11. **Macro**: Macro has one attribute: name, used as a natural key. Macronutrient names are unique
12. **Food list**: Food list has one attribute ID, used as a surrogate key. Name could work as a natural key but its not guaranteed that the names are unique.

13. **Beverage list**: Beverage list has one attribute ID, used as a surrogate key. Name could work as a natural key but its not guaranteed that the names are unique.
14. **Serving size**: Serving size has one attribute ID, used as a surrogate key. No good choice of natural key here.
15. **Unit**: Unit has one attribute: Name, used as a natural key. Unit names are unique
16. **Micro**: Micro has one attribute: name, used as a natural key. Micro names are unique

2 Weak entities

There are four weak entities in this conceptual model.

- **Serving size**: Serving sizes cannot exist in our database without being tied to a food or a beverage
- **Macro**: Macros cannot exist in our database without being tied to a food, and a serving size, or a beverage and a serving size.
- **Micro**: Micros cannot exist in our database without being tied to a food, and a serving size, or a beverage and a serving size.
- **Weight**: Weights cannot exist in our database without being tied to a user.

3 Entities

- **User**: The user entity is cruical. The app is focused around giving users the ability to track their fitness journey. Thus, an entitiy to store each user is critical to the application.
- **All date entities (3)**: The data entities are used in conjunction with a couple entities to add another degree to the relationship. This is needed in some cases in order for the user to be able to track achievements throughout time.
- **Weight**: The weight entity is needed so the user can update and track their weight through time.
- **Exercise**: The exercise entity is needed so the user can track various exercises completed through time.
- **Type (exercise type)**: The type entity is used with the exercise entity to track the types of exercises the user completes
- **Region**: The region entity is used with the exercise entity to track the regions of the body they exercise through time.
- **Exercise list**: The exercise list serves as a way for the user to find common exercise options.
- **Meal**: The meal entity is needed to track the meals that a user consumes through time. To track calories, macros, micros, etc.
- **Food**: The food entity is needed to define what makes up a users meals, and so they have accesss to the foods serving size, macros, micros, etc.
- **Beverage**: The beverage entity is needed to define what makes up a users meals, and so they have accesss to the beverages serving size, macros, micros, etc.
- **Macro**: The macro entity is used to list the macros defined by each food and serving size, so the user can track these things and know exactly what they're eating and getting in terms of nutrients, daily values, etc.
- **Micro**: Same concept as macro entity
- **Food list**: The food list serves as a list that the users can access to get common foods, new foods, etc.
- **Beverage list**: Same concpt as food list
- **Serving size**: The serving size entitiy is needed so that the user can know how much of a particular food they need to eat and how many calories are in each meal.
- **Unit**: The unit entity is used to define the units associated with each serving

4 Entities, Relationships, Cardinalities

- **User, weight, and date connected via ”updates”**: A user updates their weight on a specific date and time, so they can track weight loss. This is a many to many to many relationship. A user may update their weight at a specific time many times or no times. A weight and date can be associated with many users, but at least one user. Since weight is a weak entity, it must be associated with at least one user to exist in the database. A user may maintain the same weight (and still update) through one to many days.

- **User, exercise, and date connected via ”undergoes”**: A user undergoes an exercise on a specific date. This is a many to many to many relationship. If we know the user and the date, there can be many exercises, there also could be no exercises, a user can just use the app on a specific day to only track meals. If we know the date and the exercise, there can be many users. If we know the exercise and the user, they can be many dates.

- **Exercise and type connected via ”has”**: An exercise has a specific type. This is a many to many relationship. If we know the exercise, they can be many types. For example, a ”Burpee” could be classified under both ”Cardio” and ”Strength.” If we know the type, there can be many exercises.

- **Exercise and region connected via ”has”**: An exercise has a region. This is a many to many relationship. If we know the exercise, there could be many regions. For example, a specific exercise could involve more than one muscle group. If we know the region, there can be many exercises.

- **Region, exercise, and type connected via ”involve”**: A region involves a type of exercise. This is a many to many to many relationship. If we know the region and exercise, there can be many types. If we know the region and type, there can be many exercises. If we know the exercise and type, there can be many regions.

- **Exercise and exercise list connected via ”made up of”**: An exercise list is made up of exercises. This is a many to many relationshhip. If we know the exercise, there can be many lists, there should be multiple lists of exercises in our database. If there were just one master list of exercises, it woud then be a one to many relationship. If we know the list, there can be many exercises. A list is made up of at least one exercise, an exercise is in at least one list.

- **User, meal, and date connected via ”eats”**: A user eats a meal on a specific date. This is a many to many to many relationship. If we know the user and the meal, there can be many dates. If we know the date and the meal, there can be many users. If we know the user and the date, there can be many meals, there also could be zero meals, a user may not log any meals on a specific date.

- **Meal, food, and beverage connected via ”contains”**: A meal contains food and beverages. This is a many to many to many relationship. If we know the food and the beverage, there can be many meals. If we know the meal and the beverage, there can be many foods. If we know the meal and the food, there can be many beverages. There also could be no beverages.

- **Food and food list connected via ”made up of”**: A food list is made up of food. This is a many to many relationship. There are many food lists in our database, and they are defined by the type. If we know the food, there can by many lists. If we know the list, there can be many foods

- **Beverage and beverage list connected via ”made up of”**: A beverage list is made up of beverages. This is a many to many relationship. There are many beverage lists in our database, and they are defined by the type. If we know the beverage, there can by many lists. If we know the list, there can be many beverages

- **Food and serving size connected via ”has”**: A food has a serving size. This is a one to many relationship. If we know the food, there is only one serving size. If we know the serving size, there can be many foods.

- **Beverage and serving size connected via ”has”**: A beverage has a serving size. This is a one to many relationship. If we know the beverage, there is only one serving size. If we know the serving size, there can be many beverages.

- **Food, macro, and serving size connected via ”defined by”**: A macro is defined by the food and the serving size. This is a one to many to many relationship. If we know the food and the macro, there is one serving size. If we know serving size and the macro, there can be many foods. If we know the food and the serving size, there can be many macros.

- **Beverage, macro, and serving size connected via ”defined by”**: A macro is defined by the beverage and the serving size. This is a one to many to many relationship. If we know the beverage and the macro, there is one serving size. If we know serving size and the macro, there can be many beverages. If we know the beverages and the serving size, there can be many macros.
- **Food, micro, and serving size connected via ”defined by”**: A micro is defined by the food and the serving size. This is a one to many to many relationship. If we know the food and the micro, there is one serving size. If we know serving size and the micro, there can be many foods. If we know the food and the serving size, there can be many micros.
- **Beverage, micro, and serving size connected via ”defined by”**: A micro is defined by the food and the serving size. This is a one to many to many relationship. If we know the food and the micro, there is one serving size. If we know serving size and the micro, there can be many foods. If we know the food and the serving size, there can be many micros.
- **Serving size and unit connected via ”has”**: A serving size has a unit. This is a one to many relationship. If we know the serving size, there is usually only one unit. If we know the unit, there can be many serving sizes.
- **Unit related recursively via ”Conversion”**: A unit has a conversion with a unit. This is a one to many relationship. If we know the unit that is going to be converted, there can be many units to convert to. If we know the unit to convert to, there is only one unit that we need to convert.

5 Intersection data

- **User, weight, and date connected via ”updates”**: When these entities come together, a record is stored for ”amount”
- **Region, exercise, and type connected via ”involve”**: When these entities come together, a record is stored for ”intensity”
- **User, exercise, and date connected via ”undergoes”**: When these entities come together, a record is stored for ”calories burnt” and ”duration”
- **Exercise and exercise list connected via ”made up of”**: When these entities come together, a record is stored for ”type”
- **Food and food list connected via ”made up of”**: When these entities come together, a record is stored for ”type”
- **Beverage and beverage list connected via ”made up of”**: When these entities come together, a record is stored for ”type”
- **Food and serving size connected via ”has”**: When these entities come together, a record is stored for ”calories”
- **Beverage and serving size connected via ”has”**: When these entities come together, a record is stored for ”calories”
- **Food, macro, and serving size connected via ”defined by”**: When these entities come together, a record is stored for ”grams” and ”calorie %”
- **Beverage, macro, and serving size connected via ”defined by”**: When these entities come together, a record is stored for ”grams” and ”calorie %”
- **Food, micro, and serving size connected via ”defined by”**: When these entities come together, a record is stored for ”mg” and ”daily value”
- **Beverage, micro, and serving size connected via ”defined by”**: When these entities come together, a record is stored for ”mg” and ”daily value”