Team 100 - Group sudo\_rm\_rf

PT1 - Stage 2

Relational schema:

User(UserID:INT [PK],

Email:VARCHAR(50),

Password: SOME KIND OF HASH IDK)

Result(ResultID: INT [PK],

UserID:INT [FK to User.UserID], CreationDate:DATETIME,

ResultName:VARCHAR(50))

InteractionPair(dfID1: VARCHAR(6) [FK to DrugFoodIDs.FoodDrugID], dfID2: VARCHAR(6) [FK to DrugFoodIDs.FoodDrugID], ResultID:INT [FK to Result.ResultID],

(dfID1, dfID2, ResultID) [PK], InteractionResult: INT)

DrugFoodIDs(FoodDrugID: VARCHAR(6) [PK],

FoodDrugName: VARCHAR(50))

DrugBrand(ReferenceDrug: VARCHAR(50) [PK] [FK to DrugFoodIDs.FoodDrugName], BrandedName: VARCHAR(50))

IDtoInteraction(FoodDrugID: VARCHAR(6) [PK],

ReferenceDrug: VARCHAR(50) [PK])

Relationships:

* User - Result
  + Description: One user can create many sets of results, each with their own unique set of medications/diets which contain results from different drug food/drug drug interactions.
  + Cardinality: One-to-Many
* InteractionPair - Result
  + Description:
    - One drug-drug/drug-food/drug-herb interaction pair should correspond to exactly one result since this represents interactions with medications which users input.
  + Cardinality: One-to-One
* DrugFoodIDs - DrugBrand
  + Description:
    - We can use the Brand Name of drugs to figure out its active ingredient, for instance Tylenol is actually the drug acetaminophen.
  + Cardinality:
    - Many to one, as there could be many branded names for a given drug
* DrugFoodIDs - InteractionPair
  + Description:
    - The dfID1 and dfID2 listed in the InteractionPair table correspond to drug/food names and, in the case of drugs, possible branded names. There is also the possibility of linking additional information to the drug/food IDs in InteractionPair such as caloric content or chemical formulas.
  + Cardinality: Many-to-Many
    - Each drug/food can be used in multiple DDI/DFI interaction pairs and each DDI/DFI interaction pair can have multiple drug/foods linked.

Appended Section:

In response to comments made by CA, the following section is dedicated to addressing questions raised in those comments.

User: We model User as an individual user of the system. This allows for easier retrieval of user information and better gives feedback to the queries made by users.

Result: We model result based on an input of drugs and food consumed or to be consumed by individual user. We model Result as an entity rather than an attribute of the User because we may be able to discover population pattern of drug-food usage and interaction, which is a desirable functionality beyond serving individuals.

InteractionPair: We model interaction pair as the result of interaction between a particular type of food or drug. We model it as an entity because (i) It serves as a knowledge base that opens up further possibilities of new functionalities and design, (ii) Due to possible many-to-one relationship between drug/food name and drug/food ID, we may need to perform certain join operations before returning results. Therefore, we need an intermediate platform like InteractionPair.

DrugFoodIDs: It maps the common names of drugs and food to a distinct set of codes (IDs). It can be seen as a new set of vocabulary for this database. We model it as an entity because there are only drug name to ID and food name to ID databases, but no instance of drug and food ID database. We need this entity to unify drug and food for further operations.

DrugBrand: It maps branded names to a distinct drug name in DrugFoodIDs table. We model it as an entity instead of an attribute of DrugFoodIDs because (i) we can obtain information of the manufacturers, price of drug, which offers more potential functionalities to serve users to choose drugs with more reasonable price from a trusted manufacturer, (ii) we can infer drug components and compositions from certain chemical names of drugs that help us better classify drugs based on chemical characteristics.

BCNF Normalization

To ensure that this schema adheres to BCNF we must make sure that for every functional dependency X->Y, X is a superkey.

DrugBrand

* Reference Drug
* BrandName

(ReferenceDrug, BrandName) -> DrugName

In this situation DrugID is the superkey in this table and associates BrandedName. There are no other dependencies.

DrugFoodIDs

* FoodDrugID
* FoodDrugName

FoodDrugID is a primary key that uniquely identifies each FoodDrugName. Given that you can determine FoodDrugName through the FoodDrugID, FoodDrugID is a superkey.

User

* UserID
* Email
* Password

UserId -> Email, Password

UserID is a superkey that identifies Email and password. There are no other functional dependencies.

InteractionPair

* dfID1
* dfID2
* ResultID
* InteractionResult

The primary key is (dfID1,dfID2, resultID) which identifies each interaction and the result. The rest of the result is associated with a resultID which handles the other dependencies.

Result

* ResultID
* UserId

CreationDate

* ResultName

The primary key is ResultID which is a primary key that identifies all of the other values. ResultID is a superkey and can identify every other attribute.

Since all of these different tables contain a superkey and all other values are dependent on these superkeys they comply with BCNF.