Internet Fridge

Ben Crabbe

May 1, 2015

1 User Accounts

I decided one of the most useful features would be to have users that can own items so that each person that uses the fridge can keep track of whats theirs. I have gone with the assumption that the people sharing the fridge are friends and therefore users don't need passwords. I did not want a situation where users had to enter a password every time they use the fridge since this would be annoying.

A users table in the database which contains the users name, which must be unique, and an ID. I decided to have an integer primary key rather than using the name (even though it is required to be unique and not null) since the PK will have to appear a lot in other tables and it is more storage efficient to have small integers rather than possibly long names.

The active user can be set by calling selectUser request, the ID of the user selected will then be stored in a variable. users can be created with createUser

2 Products and Inventory

I would like users to be able to just scan an item to place it on the inventory. I tried Outpan for getting the product details automatically, but after trying a number of EAN numbers from my own fridge I found none with any info on outpan. So instead I have users enter product info manually, this would be asked for if a product that has not previously been seen is scanned.

Products have their EAN as their primary key since we wouldn't want to store more than one instance of the same product anyway. Things I deem necessary to store about products are the name, the number of recommended days to use by after opening, the storage requirements and the average stock. This last value is the mean number of these products stocked since the fridge has been in use. Storage requirements have their own table so that users can select the one thats applicable easily, this also reduces the storage size.

When adding an item to the inventory the users can decide wether to add a use by date. I did not want to require this since I want the system to deal with frozen items as well, in which case use by is irrelevant, also If you were putting a big shop in the fridge it may get tiresome. Users can enter a use by date for a inventory item later using the addUseByToInventoryItem request.

Users can set a borrowingMessage for each item in inventory so that when the inventory is listed it says 'you can help yourself', 'please do not use', 'use but please replace', or 'use but do not finish'. these are stored in the borrowingMessages table and one is selected by users for ease of use.

When an inventory item is opened users should call inventoryItemOpened this records the current date in the item tuple. Using this in conjunction with the daysToUseByAfterOpening field in the product table as well as the use by date in the inventory table can be used to find expired products with a listExpiredItems request.

When an item is declared finished by a request to inventoryItemFinished the details are moved over to a usedItems table. I decided to have separate tables for current items and used items due to storage size requirements. The list of usedItems could get very big so we want to store as little as necessary. What is stored for used items is dateAdded, dateFinished, productEAN, ownerID. I store the dates so that we can calculate the mean stock of each product which is used by a generateShoppingList request to tell you the items you do not currently have which you have had in the past, and to order them by the average stock.

I decided not to store quantities even though they may have been useful since no one is going to want to measure how much milk they have left every time they make a cup of tea etc its just impractical.. maybe if the fridge had scales built into the shelves?

I have created tables for recipes and product-recipe table which I never got time to implement.

3 Evalutation

It should be possible to add additional attributes to the relations at will since i have used self.db.row_factory = sql.Row which allows name addressing the results of executes i.e. row["ID"] rather than row[8] which would be vulnerable if someone added another attribute above it.

I have used minimal storage to be as efficient as possible without compromising features.

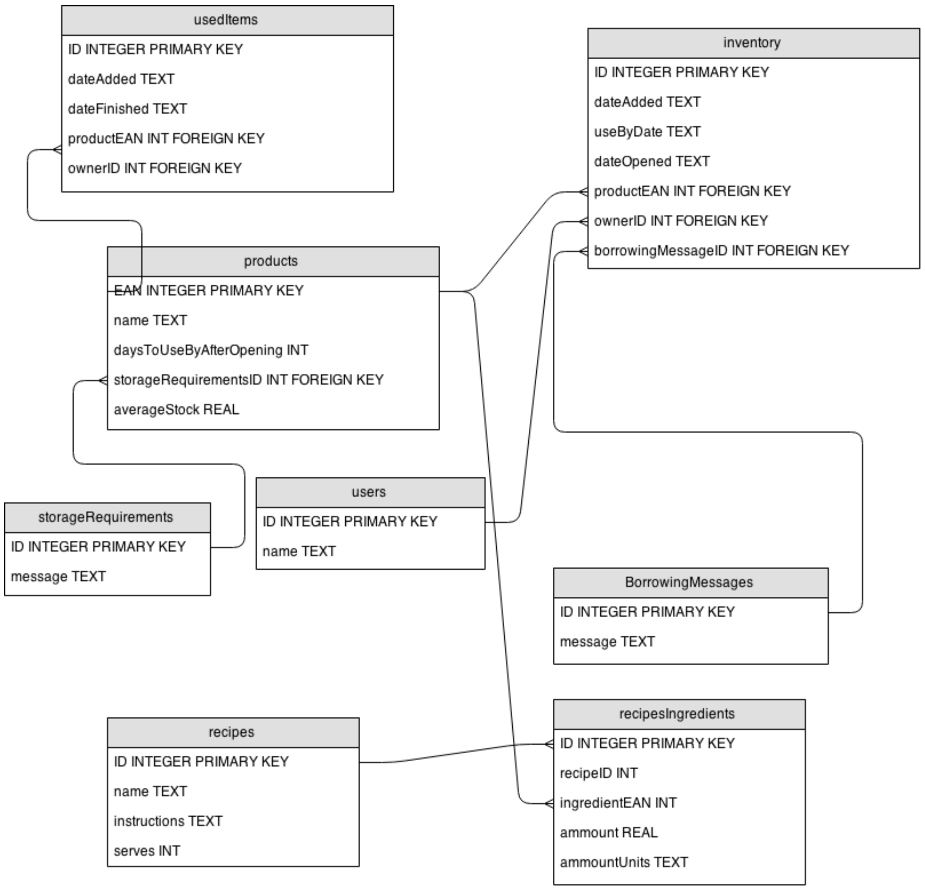

Figure 1: ER diagram