Database analysis and design

Draft Tables

This is the initial mock-up of the tables I believe will be essential for the data-driven part of this task.

Users

* Customer
* Admin

Products

* Product details
* Drinks
* Food
* Stock
* Audit table

Order

* Order details

Draft Entities in each Initial table (pre normalisation)

This is the initial mock-up of some of the entities/attributes that I will use within the tables of the database. These fields will most likely be susceptible to change after further analysis of the database design.

Users

* UserID
* Username
* UserType
* Password
* FirstName
* LastName

Products

* ProductID
* ProductType
* Price
* Name
* Calories
* Image
* Details
* Stock/Quantity

Products Audit table

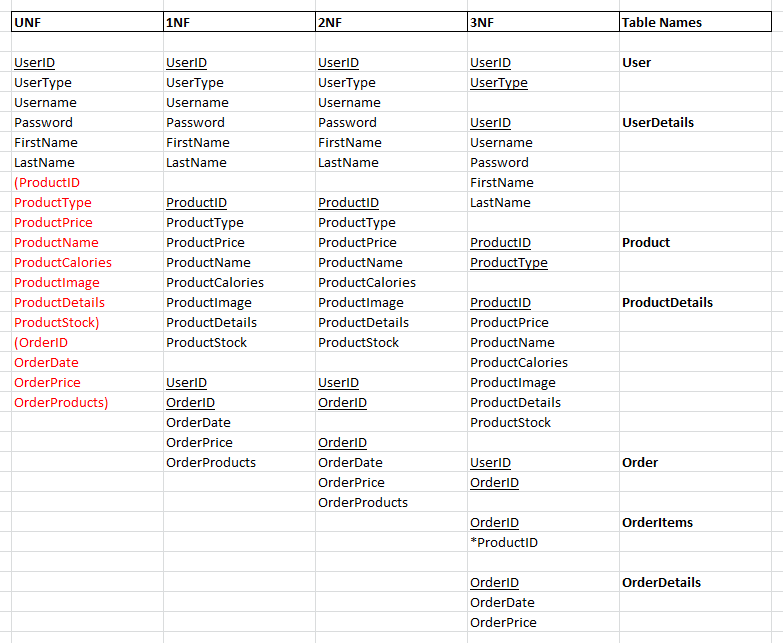
* ProductID
* Date Changed
* UserID
* UserType(Potentially for security reasons)
* Old values
* New values
* Added values
* Deleted values

Orders

* OrderID
* UserID
* ProductID
* OrderDate
* TotalPrice

Draft Normalisation of entities

This is the first attempt of normalising all of the attributes in the database.



Analysis of database design:

I believe that this database design will provide a concrete and easy-to-use data structure to complete all of the application requirements in the task. However, whilst the design is robust, I also believe that it could create potential hindrances to the development process of the application in certain areas, such as with the user details. As I want to have a firm separation between a customer and admin within the data and application, it could be argued that it would be more beneficial to also separate the two user’s data into separate tables. I believe that this would be beneficial as it would make a firm distinction between these user’s, which would therefore help in the development process of this task as I will be able to specify which users will have the ability to access certain parts of the application (such as only allowing the admin’s to access the admin page of the application).

Second iteration of normalisation



Analysis of database design:

In this iteration I have got rid of the ProductStock field as I felt this would be unnecessary for the task at hand and could potentially overcomplicate the application. Additionally, to measure stock of products in a normalised database design would require its own table and could therefore overcomplicate the design.

This iteration also separates the customer and admin users, which I think will prove to be very helpful when creating procedures and triggers based around the data in the table.

To improve on this database design for more functionality related around the admins, I will also implement an audit table that will keep track of changes made to the products via triggers. This will be helpful to keep track and control the changes and additions made to the products table.

Third iteration of normalisation:

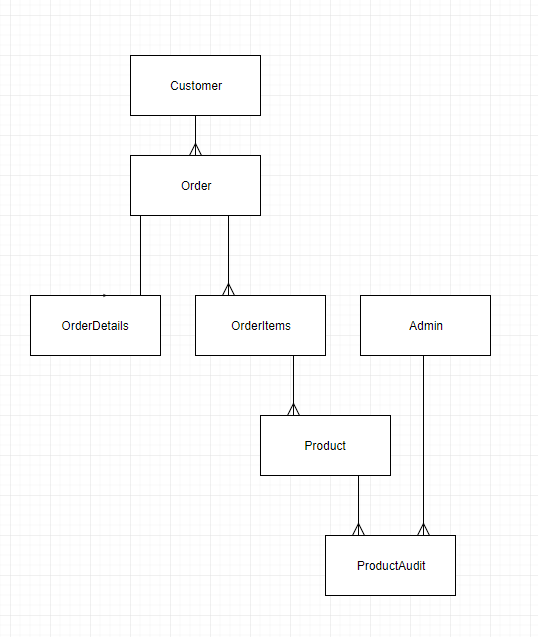


Analysis of database design:

This is the final database design that has implemented an audit table for the products. This will mean that any editing, deleting or additions to the product table will be tracked in the database through the use of triggers in the database. This audit table will track all of the old and new data for each changed entry as well as the date of the change in data and track the admin that changed the data. This design should be robust, however it could potentially run into errors in a scenario where a new product is added as the old data will not exist. In this instance I believe that these fields can simply be left null and only the new fields will be filled out for the row.

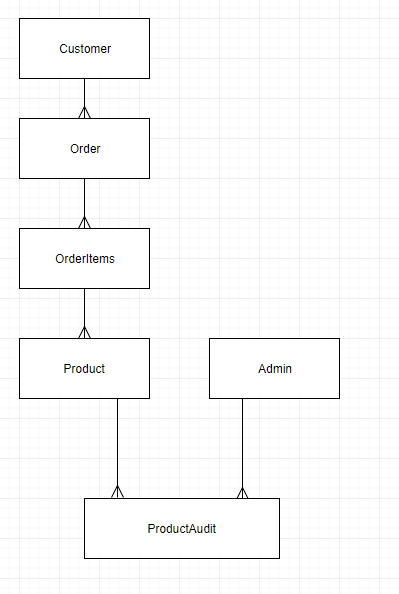
In addition to this, I have added a quantity field in the orderItems table. This field will track how much of the product was purchased in each order and will allow me to implement a quantity selection form for each product in the web application.

Entity Relationship Design:



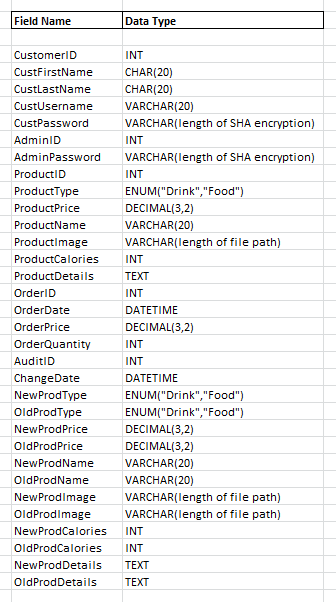
This is the initial ERD of the database and can be seen to have a one-to-one relationship within the database. To resolve this, I will add all of the orderDetails fields into the Order table and remove the orderDetails table.

Resolving one-to-one relationship in ERD:





Datatypes of fields in database: (DATA DICTIONARY)

As a first draft, these are the data types I would like to use for the fields within the database. Specific things such as the ENUM data type and size of each data may be susceptible to change upon creating the database however, I think that these data types will work correctly and fulfil all purposes nicely.

To store the images for the products, I have decided to instead store a string that will contain the file path to the image. This will take up less memory in storing each individual image in a database and therefore be faster and take up less of the databases memory.

