CSCE-608 Database Systems

Fall 2020

Project # 1

Name: Sun Yul Lee

UIN: 824007724

Description

The database application that I would work with is the online retail application. The main process of the application would be similar to the other online retail websites. At first, the user can log-in to the application or register new account to the application. There would be the option to choose type of user between customer or supplier so that the user can tell the application whether the user is a customer or supplier. When the user clicks ‘register’ button, the application will show the screen for submitting the information of the user. Based on the option that the user chooses, the information will be stored in the proper table. The application can notice the user if all data are not provided. Also, the application will not store the information and give notice to the user if the user provides user id which is already in the table. After the registration finishes, the application will show the log-in screen. When the user clicks ‘login’ button, the application will show different main screen based on the user type. As a customer, the user can shop products, see order history, edit user information, and edit bank account information. In the main screen, there will be a button for displaying categories, a panel for searching, a button for displaying account information, and a button for displaying shopping cart. When the user clicks the button for displaying categories, a panel with buttons for each category will be displayed on the left side of the main screen. The user can click category button to see all products in the category. The user can do the same process by selecting one category in the dropdown list in the panel for searching and click the search button without searching word. The user can search products which have a name containing searching word. When the user clicks a product in the list, a screen displaying the product information will be displayed. In the screen, the user can choose the quantity of the product and add the product in the cart. When the user clicks account button, a screen displaying account information will be displayed. In the screen, the user can see the account information, bank accounts, and order history. Also, the user can edit account information and add, edit or delete bank accounts. When the user clicks cart button, a screen displaying cart information will be displayed. In the screen, the user can see the items in the cart, remove items in the cart, choose which bank account for payment, and make a payment. The user cannot make payment if there is no item in the cart or there is no bank account. If there is no bank account, the user can add bank account in the screen without going to the account information screen. As a supplier, the user can edit or delete products, search products, and edit the user information. Most of the components will be similar to the components in the main screen for customer. But there would be no button for displaying cart. Unlike a main screen for a customer, a main screen will show a list of products that the supplier provides. The search button and the category bar will work the same. In the screen, the user can delete products. When the user clicks a product in the list, a screen displaying the product information will be displayed. In the screen, the user can edit the product information. When the user clicks account information button, a screen displaying account information will be displayed. Unlike a user as a customer, the user can see the user information only. In the screen, the user can edit account information. When the window is closed, a window for log-in will be displayed. The application will be closed when the window for log-in is closed.

E/R Diagram

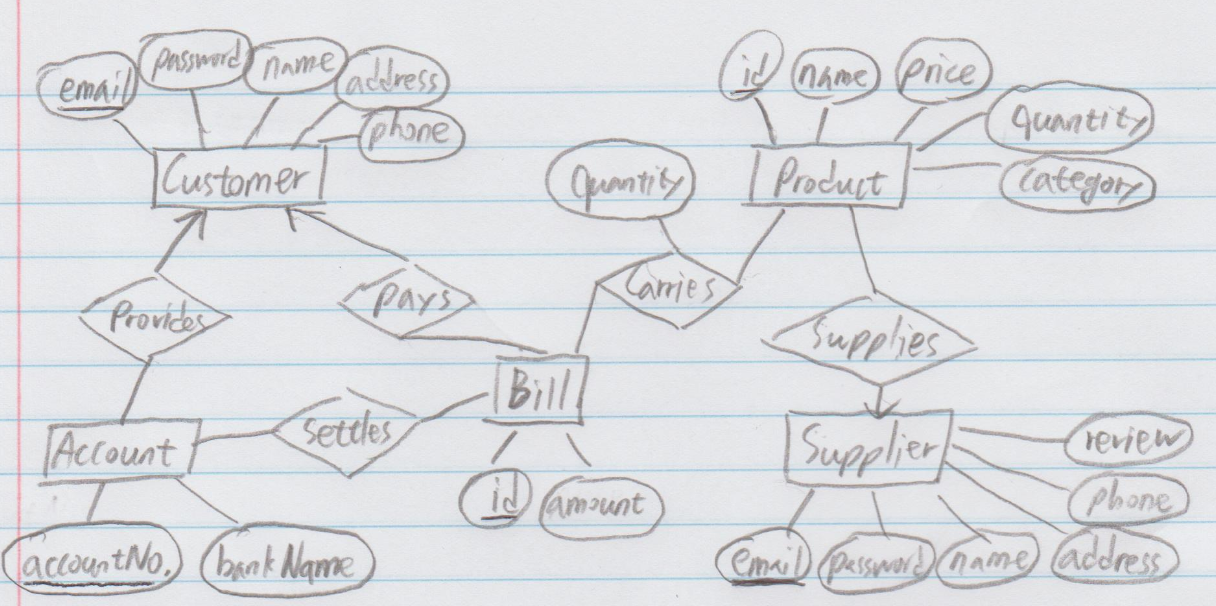


Figure 1. E/R Diagram

Relational Schema

The original E/R Diagram is shown above. In the diagram, the relation provides, pays, and supplies have a many-one relationship. With the relation, I combined the relation Account and Provides, Bill and Pays, and Product and Supplies. The new E/R Diagram is shown below.

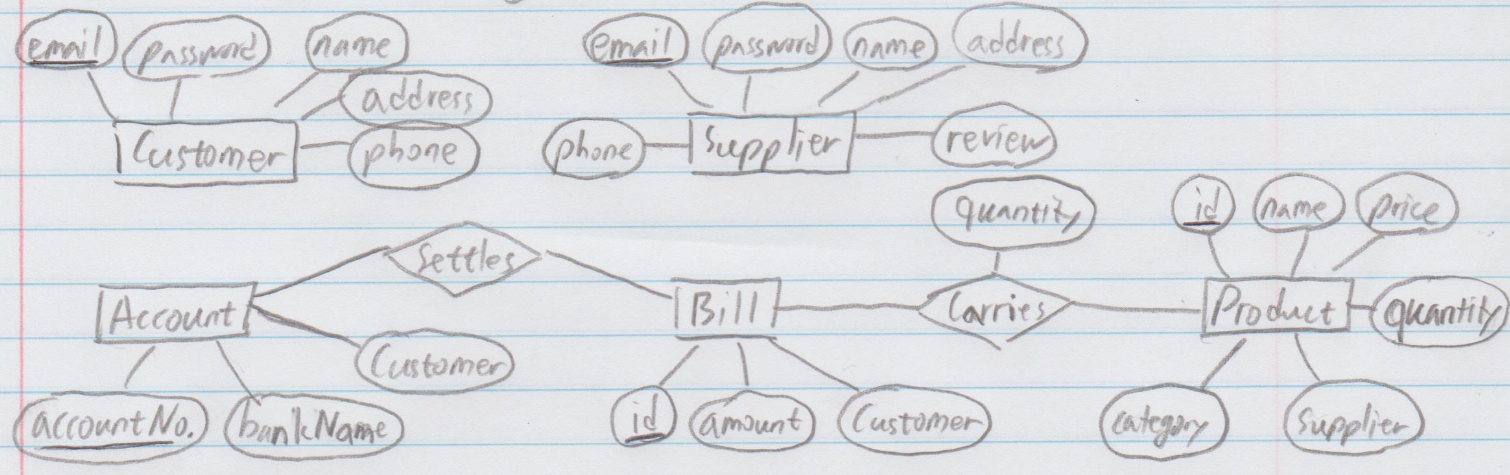


Figure 2. New E/R Diagram

With the new E/R Diagram, the set of relations are produced like below.

* Customer (email, password, name, address, phone)

FD’s: {email -> password, name, address, phone}

* Supplier (email, password, name, address, phone, review)

FD’s: {email -> password, name, address, phone, review}

* Account (accountNo., bankName, customer)

FD’s: {accountNo. -> bankName, customer}

* Product (id, name, price, quantity, category, supplier)

FD’s: {id -> name, price, quantity, category, supplier}

* Bill (id, amount, customer)

FD’s: {id -> amount, customer}

* Carries (bill, product, quantity)

FD’s: {(bill, product) -> quantity}

* Settles (bill, account)

FD’s: {bill -> account}

For the relation Customer, the key attribute is email and FD’s = {email -> password, name, address, phone}.

Since every nontrivial FD has email which is a key and thus, also a superkey on the left side, the relation is in BCNF.

For the relation Supplier, the key attribute is email and FD’s = {email -> password, name, address, phone, review}.

Since every nontrivial FD has email which is a key and thus, also a superkey on the left side, the relation is in BCNF.

For the relation Product, the key attribute is id and FD’s = {id -> name, price, quantity, category, supplier}.

Since every nontrivial FD has id which is a key and thus, also a superkey on the left side, the relation is in BCNF.

For the relation Bill, the key attribute is id and FD’s = {id -> amount, customer}.

Since every nontrivial FD has id which is a key and thus, also a superkey on the left side, the relation is in BCNF.

For the relation Carries, the key attribute is (bill, product) and FD’s = {(bill, product) -> quantity}.

Since every nontrivial FD has (bill, product) which is a key and thus, also a superkey on the left side, the relation is in BCNF.

For the relation Settles the key attribute is bill and FD’s = {bill -> account}.

Since every nontrivial FD has email which is a key and thus, also a superkey on the left side, the relation is in BCNF.

All the relations pass the BCNF test. The relations pass the BCNF test because all FD in the relations are in the format that the key attribute determines the other attributes.

Creating Database using SQL

I used python to create data and insert the data into database. The database application that I chose to use is Microsoft SQL Server Management Studio 18. First, I created database named ‘ProjectDB’. Then I created each table for the relation in the database. To create data for the relation ‘Customers’ and ‘Suppliers’, I used iteration index for every attribute on each tuple. To create data for the relation ‘Products’, I randomly choose the value for price, quantity, and category. I assigned two products to the same supplier. To create data for the relation ‘Accounts’, I used iteration index for attribute ‘accountNo.’ and ‘customer’. I randomly assigned the bank name in the list to set the value for attribute ‘bankName’. To create data for the relation ‘Carries’, I first generate random numbers for the attribute ‘quantity’ and used iteration index to set values for attribute ‘bill’ and ‘product’. To Create data for the relation ‘Bills’, I used iteration index to set values for attribute ‘bill’. I assigned three products for each bill. The values for attribute ‘amount’ were calculated using the price and the quantity of the products in the bill. The values for attribute ‘customer’ was selected randomly from the list of customer emails. To create data for the relation ‘Settles’, I used iteration index to set values for attribute ‘bill’. The values of the attribute ‘account’ were set to the value of account id of the account which is owned by the customer who has the bill. The program for creating database also creates csv files for the relations. The program is ‘Create\_database.py’.

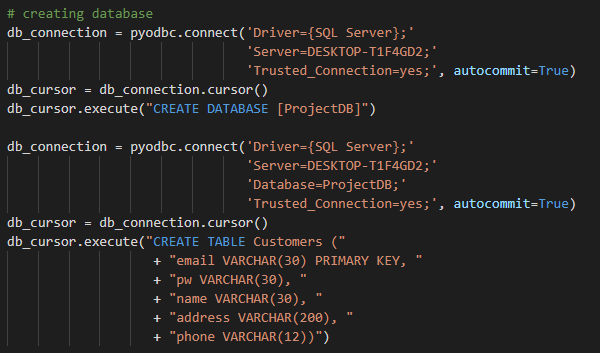


Figure 3. Microsoft SQL Connection string and SQL Command string

for creating database and table

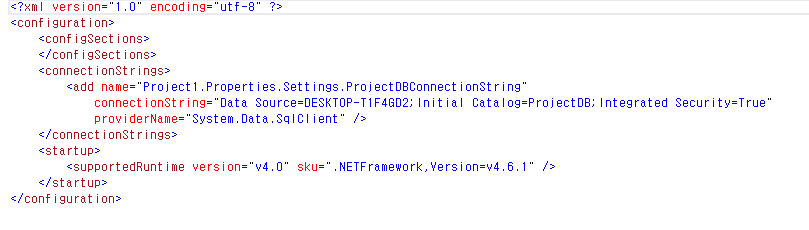


Figure 4. Microsoft SQL Connection string in the App.Config file of the C# program

Developing a user interface

I used Microsoft Visual Studio to develop a user interface in c#. When the user run the program, the login screen will first show up. In the screen, the user can choose the type of user between Customer and Supplier. The user can login to the system by providing valid email address and password and click login button. If the user is first time to use the application, user can register a new account by clicking blue ‘Sign In’ link-label. If the user clicks the link-label, a screen for registration will be displayed. In the screen, the user can provide user information and create a new user account. When the user logs in to the application as a customer, the user can use the application as other online retail application. The user can search products, add products to cart, edit user information, add, edit, or remove bank account, and make a payment. When the user logs in to the application as a supplier, the user can search the user’s products, add new products, edit products, remove products, and edit user information. When the user closes the main screen, the login screen will be displayed. If the user closes the login screen, the application will be closed.

Functions

* Login
  + The program generates SQL query to check whether the table Customers or Suppliers has a tuple with the provided email and password.
* Registration
  + The program generates SQL query to check whether the table Customers or Suppliers has a tuple with the provided email. If the table has the tuple, it does not add new tuple to the table. If the table does not have the tuple, the program generates SQL query to add new tuple to the table.
* Loading a list of products
  + The program generates SQL query to get all tuples in the table Products. If one of the categories is selected, the program generates SQL query to get all tuples in the table Products where the attribute ‘category’ is the same as the selected category.
* Search
  + The program generates SQL query to get all tuples in the table Products where the attribute ‘name’ contains search word. If there is no search word, the program gets all tuples. It also checks attribute ‘category’ if one of the categories is selected.
* Display product information
  + The program generates SQL query to get one tuple in the table Products where the attribute ‘id’ is the same as the id of selected product in the list.
* Add product to cart
  + The program generates SQL query to add new tuple to the table Carries. Then it generates SQL query to update attribute ‘quantity’ of a tuple with the same product id in the table Products. And it generates SQL query to update attribute ‘amount’ of a tuple with the same bill id in the table Bills.
* Edit product information
  + The program generates SQL query to update selected tuple in the table Products. It also checks whether the provided key value is duplicated or not.
* Remove product
  + The program generates SQL query to remove selected tuple in the table Products.
* Display user information
  + The program generates SQL query to get values of the tuple in the table Customers or Suppliers where the attribute ‘id’ is the same as the id of current user.
* Display bank accounts
  + The program generates SQL query to get tuples of the table Accounts where the attribute ‘customer\_id’ is the same as the id of current user.
* Add bank account
  + The program generates SQL query to add new tuple to the table Accounts. It also checks whether the provided key value is duplicated or not.
* Edit bank account
  + The program generates SQL query to update selected tuple in the table Accounts. It also checks whether the provided key value is duplicated or not.
* Remove bank account
  + The program generates SQL query to delete selected tuple in the table Accounts.
* Display order history
  + The program generates SQL query to check table Bills and Settles. It gets tuples if the attribute ‘customer\_id’ is the same as the id of the user and table Settles has tuples with the bill id. Then the program generates SQL query to check table Settles, Products, and Carries to get tuples to gather name, price, and quantity of the products in orders and account number which settled the payment.
* Display cart
  + The program generates SQL query to check table Carries, Products, and Bills to get tuples to gather id, name, price, quantity of the products in the cart and amount of bill.
* Remove product from cart
  + The program generates SQL query to delete selected tuple in table Carries. Then it updates attribute ‘quantity’ of the tuple with the same product\_id in table Products.
* Make payment
  + The program generates SQL query to add new tuple in table Settles.

Discussion

It is my first time to develop a database application. I started the development process from the bottom. Creating an E/R Diagram for the database application, I realized that there are many factors that I need to consider. At first, I tried to develop a web application. However, I have no experience in developing a web application and very little knowledge of network programming. Since I have experience in developing a Winform application with Microsoft Visual Studio, I decided to develop the application using Microsoft Visual Studio with C#. As developing the application, I learned how to create a database and table in Microsoft SQL using python. I learned how to connect the application with the database and how to interact with the database by working on connecting the database with the application. At the beginning of the development, I got a lot of errors caused by the wrong SQL query command. As I work on the project, with the knowledge that I learned from the lectures, I get used to handling SQL queries to interact with a database in C#. Overall, through the project, I earned experience in developing database application; I learned the process, constraints, and difficulties of database application.