CS4111 Database Project

Group Members: Hao Zeng(hz2759), Yuqing Jin(yj2679)
PostgreSQL account:

'user' : 'hz2759',
'passphrase' : '0703',
'db-server' : '@35.211.155.104/proj1part2'

1.

Implemented Parts:

Overall Database Management System description:
This is a women inner wear shopping platform database management system.
There are 7 entities and 8 relationships in this database. All the entities and relationships are stated as below.

Entity:
user, user_address, user_payment, product, category, shopping_cart, order_detail

Relationship:
user & user_address, user & user_payment, user & shopping_cart, user & order_detail,
order_detail & user_address, order_detail & user_payment, order_detail & shopping_cart,
product & category, product & shopping_cart.

All the functions are presented in the following webpages:

- Home page:

The name and uni of both group members are presented in this page.

Also, when users and administrator have some incorrect operation, it would redirect to this home page sometimes.

- Admin page:

This webpage is for administrators to implement their daily tasks.

Administrators are allowed to search order address and order payment via only order_id or correct only user_id or both order_id and user_id. Only when the input information ID exists in our database, can the administrator obtain the result from the database; or the system would get a null result and redirect to the home page. If admin inputs the order_id with non corresponding user_id, the searching result would be null as well.

Administrators are also allowed to search shopping cart via user_id or shopping cart id and search order detail via order id.

Additionally, administrators are able to add products with product name, inventory(int), unit_price and category_id. We enforce the administrator to input the category id when add new product in order to guarantee each product belongs to exactly one category. Besides, administrators are able to add a category with the category name.

Every time when users or administrators add a new row in tables, they do not need to input an id with it, because our system would directly fetch the maximum id in the existing table and then append new information with id = (max+1).

- User page:

This webpage is to add new users and find registered users in this system. To add a new user, it is required to input not only their user info(name, password, email, phone number), but also their address and payment information. This requirement is to enforce that each user has at least one user_address and one user_payment to be consistent with our ER-diagram and schema.

- Login page:

This webpage is to verify the users' credentials before they navigate to User View webpage so that they cannot see other's information.

- User View page:

This webpage is designed for users to manage their information: find or add user's own addresses and payment, find or update their shopping carts, and find order details.

When finding information, a user can choose whether to only enter their user ID to retrieve all their own information, or to enter both user ID and information ID (for example, shopping cart ID) to retrieve one piece of specific information.

When adding payment information, the expiration date of the card would be verified to ensure that the card would not expire before being added to the database.

Updating users' shopping carts enables users to delete this shopping cart or place an order. If a user chooses to place an order, he/she needs to choose a specific address and payment for the order. The inventory of the ordered product would be changed accordingly.

- Product page:

This page allows users to search products via product name or category name and add product to a new shopping cart.

When a product is being added to a shopping cart, a user needs to enter their user ID, product ID and quantity. Once successfully added, total price would be calculated and all of the user's shopping carts information would be displayed, including shopping cart ID, user ID and total.

Not Implemented Parts:
Compared with the original er-diagram in project1.1, we dropped order item and payment detail entities, in order to make the overall schema more concise and efficient. The attributes of these two entities are merged into order_detail table.

The rest of entities and relationships described in the ER-diagram in project 1.1 are implemented in the database and web pages.

2.

When a user inputs a new payment information, the system would verify the expiration date of the payment method; if the expiration date is in the past, the frontend would pop up a message box with 'Expiration date in the past!' to reject the payment. Also, we have a check constraint in our database schema to reject those payments with past expiration date.

When the user would like to add a product in their shopping cart, the system would check the quantity they entered. The query would fetch the inventory number and compare it with the quantity user entered. If the quantity of this item is beyond its inventory, an error message would pop up to give a warning.

These two pages are both related to adding constraints to the database. We found it interesting to implement two functions in postgreSQL to achieve those constraints because postgreSQL does not allow check constraints to have subquery. Message boxes reporting warning using pymsgbox module in python is also implemented.