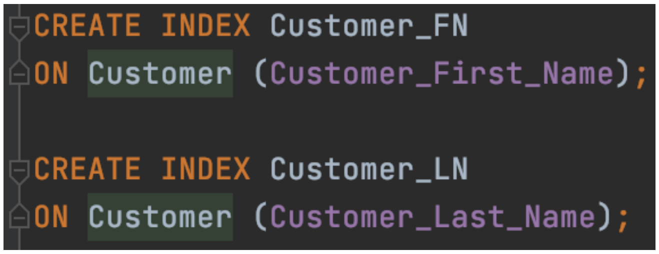
Retail

By Jaewon Park, Alberto Ng

CPSC 408 - 02

Our project mainly focuses on the retail business, especially retail database application, which is used by the retail store administrators. Since the data is truncated into different CSV files and has a lack of consolidated view, we decided to create an efficient database with data that is scattered. From solving these issues, we will be able to efficiently fetch data and be able to successfully modify the information by adding, deleting, and updating. Some of the features of our project are viewing tables, editing customer tables, and ability to search specific attributes by filter.

In this project, the task for the back-end is to design database schema based on data provided, normalize tables and create using mysql, maintain referential integrity to keep consistency, and create indexes to make searches more efficient. Index is a pointer to data in a table and used to locate data without having to search every row in a database. Index is used to quickly find values of rows and columns and improves the speed of operations in a table (our index examples below).



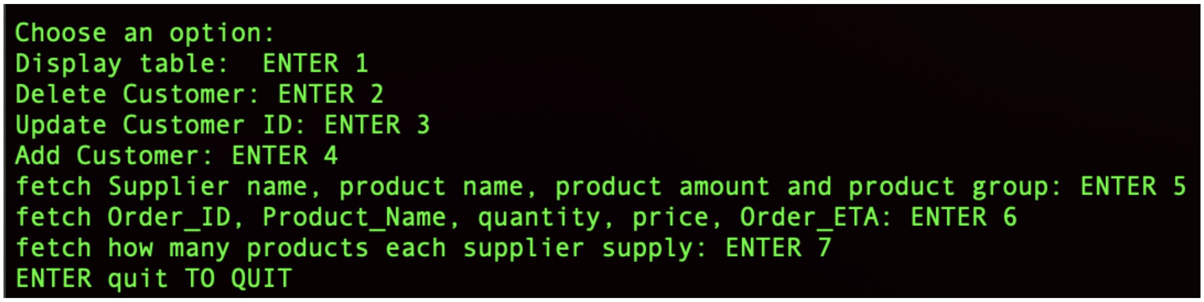
```
CREATE INDEX Customer_FN
ON Customer (Customer_First_Name);

CREATE INDEX Customer_LN
ON Customer (Customer_Last_Name);
```

Along with that, the front-end will be processed by using a python application for the end-users to interact with. The end-user for this application is not customers, but managers or related employees.

Some examples of interactions between our application and the end-users are asking users if they want to search by any features. The response would then be either yes or no, if it's "yes", users will be asked what feature and what the condition is. If "no", users will be continuing to the next prompt. The next question will ask users how many records they would like to see, either partial or all. If the response is "partial", the system will prompt the users how many records they desire to see. But if the choice is "all", the system will display all records to users.

In terms of the final results of our project, we are able to successfully modify the information by adding, deleting, and updating. The screenshot below shows all of the options in the main page users are able to choose from, by entering 1 through 7 as well as "quit" to exit. After executing any of those possible options, the system will ask if the user wants to continue the same procedures or return back to the main page.



```
Choose an option:
Display table:  ENTER 1
Delete Customer: ENTER 2
Update Customer ID: ENTER 3
Add Customer: ENTER 4
fetch Supplier name, product name, product amount and product group: ENTER 5
fetch Order_ID, Product_Name, quantity, price, Order_ETA: ENTER 6
fetch how many products each supplier supply: ENTER 7
ENTER quit TO QUIT
```

If the user enters "1" to display all tables, the system will ask the user to choose one option out of product, customer, card, payment, orders, shopping cart, product detail, product group, and supplier. Once the user chooses one, the system will ask if the user wants to search by attributes or not, and the given options will be in "y" or "n". If the user enters "y", then it will lead to the page where users can search by an attribute by their choice. However, if the user enters "n", the system will then ask to choose one of two options: view all by entering "all" or partial by entering "p". While "all" shows all the list of attributes, “partial” option will ask the user to choose a number of tables they prefer to see. For instance, entering “2” will display the first two records and the system will ask if the user wants to view another table or go back to the main page. One thing to note is the Customer table will not show the Soft_Delete attribute, while other tables are showing all attributes.

If the user enters “2” to delete customer data, the system will list all attributes from the Customer table that the user can choose from. Available attributes include customer’s ID, first name, last name, phone number, email, and type, only one option can be chosen. For example, if the user decides to delete a customer's last name, the response has to match with one of the listed last names in the table in order to successfully delete (soft delete).

If the user enters “3” to update customer ID, the system will ask users to enter the customer ID that needs to be updated. Just like option 2, deleting customer data, if the user’s response does not match with the one in the table, then it will count as a user input error. Once the user inputs the existing customer ID and the input of a new ID for an update, the system will ask if the user wants to update more or go back to the main page.

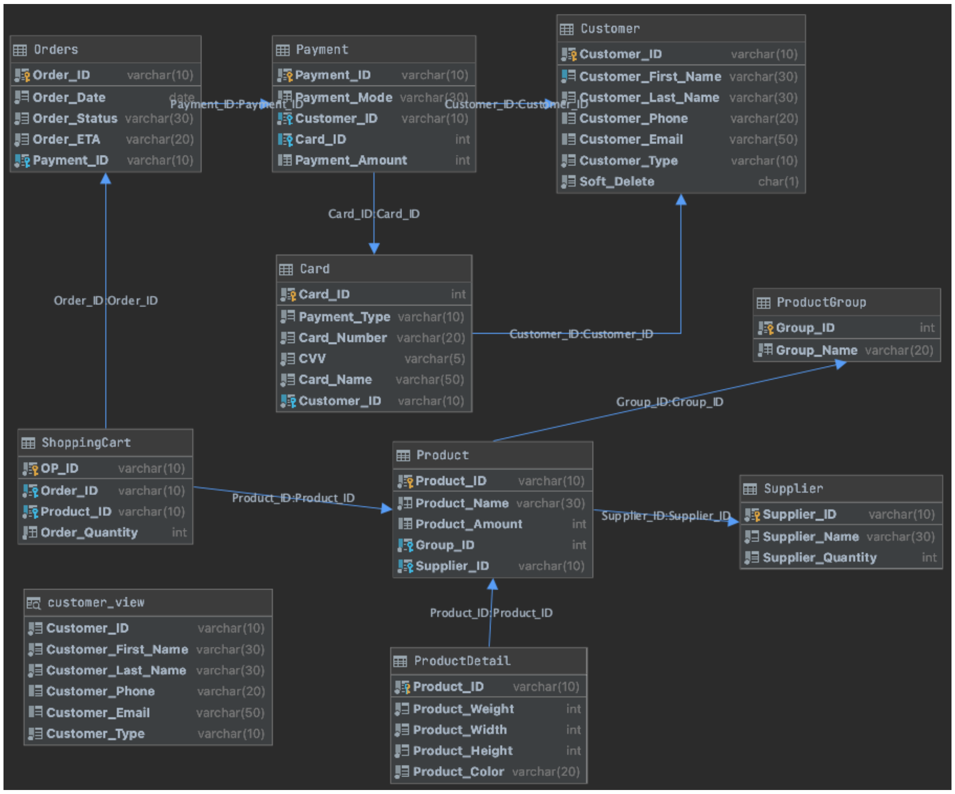
If the user enters “4” to add a new customer, the system will provide all attributes under the Customer table. The user will, then, add information of a new customer, so it could be inserted into the Customer table. After successfully adding, the system will ask if the user wants to add another customer or go back to the main page.

If the user enters “5” to fetch supplier’s name, product name, product amount, and product group, the system will provide all stored data of supplier’s name and its product name with the product type and the amount of product they have.

If the user enters “6”, the system will show all stored data of the order ID, product name, quantity, price, and order ETA. Just like option 5, the system will automatically go back to the main page after successfully performing the task.

If the user enters “7”, the system will fetch how many products each supplier supplies by showing the supplier’s name, number of products they have, and the group name. Just like option 5 and 7, the system will automatically go back to the main page after successfully performing the task.

Here is our schema diagram and it has 10 tables in our database after reading in the csv files. Each table is appropriately connected to the other table(s) and it represents the relationship between those tables. The diagram also displays what attributes are in the tables as well as how they will be connected to each other.



Orders	
Order_ID	varchar(10)
Order_Date	date
Order_Status	varchar(30)
Order_ETA	varchar(20)
Payment_ID	varchar(10)

Payment	
Payment_ID	varchar(10)
Payment_Mode	varchar(30)
Customer_ID	varchar(10)
Card_ID	int
Payment_Amount	int

Customer	
Customer_ID	varchar(10)
Customer_First_Name	varchar(30)
Customer_Last_Name	varchar(30)
Customer_Phone	varchar(20)
Customer_Email	varchar(50)
Customer_Type	varchar(10)
Soft_Delete	char(1)

Card	
Card_ID	int
Payment_Type	varchar(10)
Card_Number	varchar(20)
CVV	varchar(5)
Card_Name	varchar(50)
Customer_ID	varchar(10)

ProductGroup	
Group_ID	int
Group_Name	varchar(20)

ShoppingCart	
OP_ID	varchar(10)
Order_ID	varchar(10)
Product_ID	varchar(10)
Order_Quantity	int

Product	
Product_ID	varchar(10)
Product_Name	varchar(30)
Product_Amount	int
Group_ID	int
Supplier_ID	varchar(10)

Supplier	
Supplier_ID	varchar(10)
Supplier_Name	varchar(30)
Supplier_Quantity	int

customer_view	
Customer_ID	varchar(10)
Customer_First_Name	varchar(30)
Customer_Last_Name	varchar(30)
Customer_Phone	varchar(20)
Customer_Email	varchar(50)
Customer_Type	varchar(10)

ProductDetail	
Product_ID	varchar(10)
Product_Weight	int
Product_Width	int
Product_Height	int
Product_Color	varchar(20)