Clothing Store Database - Group 6

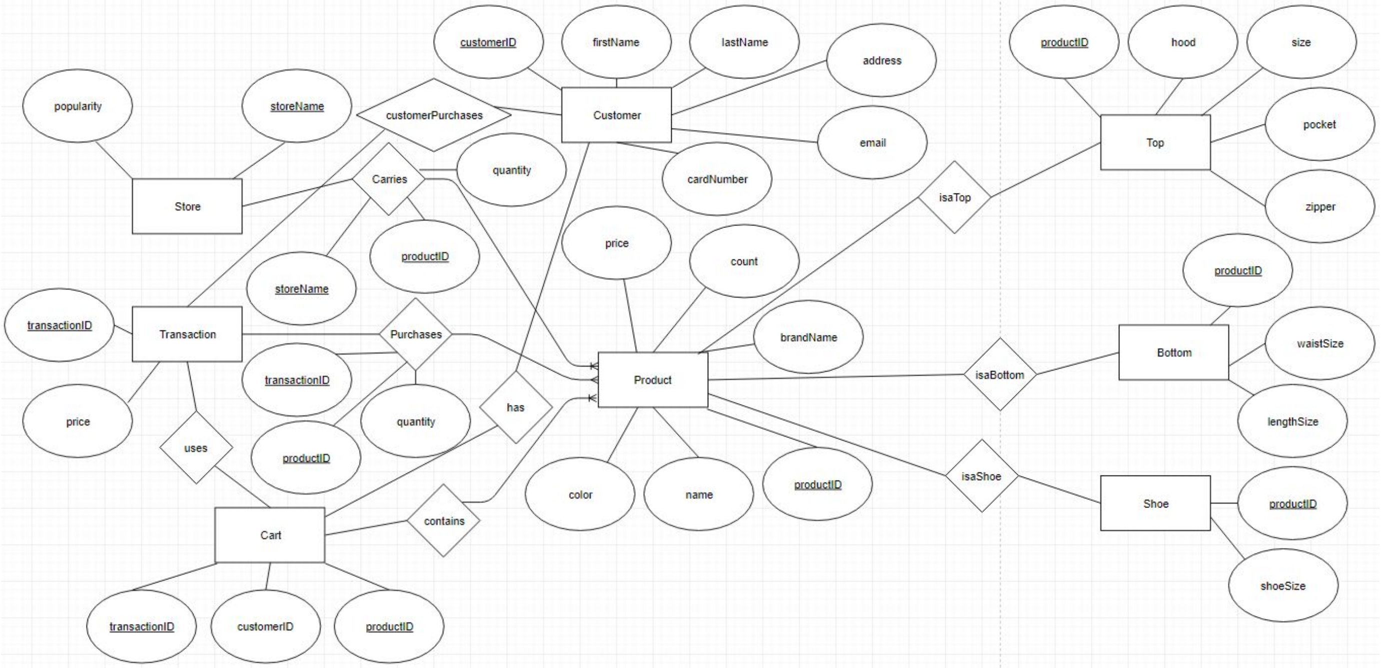
Minh An Cao, Alexander Nguyen, Ricky Singh

Project Milestone 1

Diagram tool used: draw.io
Database: MySQL

1) Textual description of the project (Milestone 0)
   Our application will be a database that will allow users to buy clothing from different stores and brands by choosing different parts of the outfit, such as shirts and pants for different brands.

2) E/R Diagram



3) Description of Entities
   Description of what each entity represents

Entity name	**Description**

Store	The store is the seller of products.
Customer	Customer entity will contain information needed for customer contact.
Transaction	Transaction has all information between the customer, store, and products bought.
Product	The product is any piece of clothing.
Top	The top is one type of clothing for Product. It has a pocket, a hood, a zipper, and a size.
Bottom	The bottom is one type of clothing for Product. It has a length size and a waist size.
Shoe	The shoe is one type of clothing for Product. It has a shoe size.

4) Description of relationships on E/R Diagram

Name	Entity 1	Entity 2	Entity 1 -> Entity 2 Role Cardinality	Entity 2 -> Entity 1 Role Cardinality	Description
Carry	Store	Product	many-to-many	many-to-many	Many stores can carry many products
Has	Transaction	Customer	many-to-one	one-to-many	Many transactions correspond to one customer.
Includes	Transaction	Product	one-to-many	many-to-many	One transaction can contain multiple products.
isaTop	Product	Top	one-to-one	one-to-one	One product corresponds to one top.
isaBottom	Product	Bottom	one-to-one	one-to-one	One product corresponds to

					one bottom.
isaShoe	Product	Shoe	one-to-one	one-to-one	One product corresponds to one shoe.

5) Description of Entity attributes

Description of attributes in each entry

Name	Used By	Used By An Identifier	Data Type	Description
storeName	Store	Yes	VARCHAR	Holds the storeName
popularity	Store	No	FLOAT	Holds the popularity, based on a 1-10 rating determined by the amount of transactions.
customerID	Customer	Yes	INT	Holds the customerID.
firstName	Customer	No	VARCHAR	Holds the first name of the customer.
lastName	Customer	No	VARCHAR	Holds the last name of the customer.
address	Customer	No	VARCHAR	Holds the address of the Customer.

email	Customer	No	VARCHAR	Holds the email of the Customer.
cardNumber	Customer	No	DECIMAL(16, 0)	Holds the card number of the Customer.
Total price	Transaction	No	FLOAT	Price of the transaction.
transactionID	Transaction	Yes	INT	transactionID for the specific transaction.
productID	Product	Yes	INT	productID that correspond to a specific Product.
name	Product	No	VARCHAR	Name of the Top.
brandName	Product	No	VARCHAR	The brand name of the top.
color	Product	No	VARCHAR	The color of the product.
price	Product	No	FLOAT	The price of the top.
type	Product	No	VARCHAR	Describes the type of the product i.e. “t-shirt” is a type of top.
size	Top	No	VARCHAR	The letter size of the top (small, medium, large, etc.).
zipper	Top	No	TINYINT(1)	It’s 1 if there is

				a zipper on the top, 0 if there is not.
hood	Top	No	TINYINT(1)	It’s 1 if there is a hood on the top, 0 if there is not.
pocket	Top	No	TINYINT(1)	It’s 1 if there is a pocket on the top, 0 if there is not.
lengthSize	Bottom	No	INT	The length dimension for the size of the pant.
waistSize	Bottom	No	INT	The width dimension for the size of the pant.
shoeSize	Shoe	No	FLOAT	The shoe size of the shoe as a number i.e. 10.

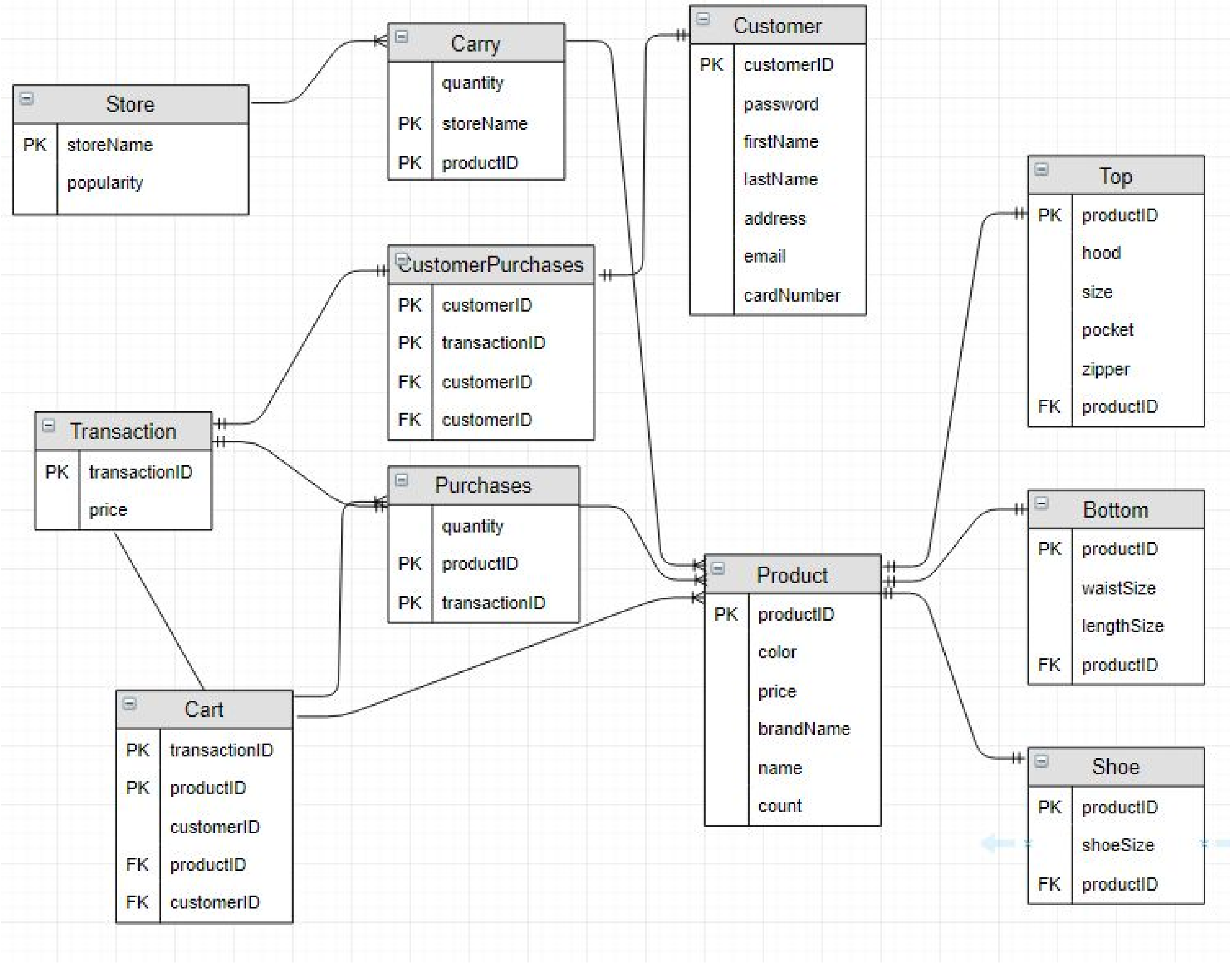
6) Analysis of functional and non-functional requirements

Some assumptions made are each product has a unique ID that tells the brand name, price, color, name and size. Each product will either be a specific top, bottom, and shoe. The user will be able to get the product(s) information and customer information when looking at a transaction.

For functional requirements, the application should allow customer information to be saved with a customerID, name, address, email, and card number. A user can purchase products from multiple stores and looking up the transaction should allow user to track what store the product was purchased from. Products include tops, bottoms, and shoes, and stores carry these products.

For non-functional requirements, there are relationships created between each table to achieve scalability, flexibility, extensibility, efficiency of storage, and efficiency of processing. Scalability is achieved by just adding new stores and products to the database. Flexibility is achieved by the different attributes of each product. The application is extensible because new products can be added easily. It is efficient because just a product ID is needed for the qualities. Processing is easy because each transaction will be identified by a transaction ID.

7) Relational model (translating E/R into table model)



8) Normalization - 3NF

First Normal Form: The tables only contain atomic values, and no columns are repeated.

Second Normal Form: All tables are in FNF, and the non-primary-key attributes are functionally dependent on the primary keys.
Third Normal Form: All tables are in SNF, and there are no transitive functional dependencies.

9) DDL Script that creates a database

```
CREATE DATABASE ClothingDatabase;
USE ClothingDatabase;

CREATE TABLE Store (
        storeName VARCHAR(255),
        popularity FLOAT,
        PRIMARY KEY(storeName)
);

CREATE TABLE Customer (
        customerID INT,
        password VARCHAR(255),
        firstName VARCHAR(255),
        lastName VARCHAR(255),
        address VARCHAR(255),
        email VARCHAR(255),
        cardNumber DECIMAL(16, 0) NOT NULL,
        PRIMARY KEY(customerID)
);

CREATE TABLE Transaction (
        transactionID INT,
        price FLOAT,
        PRIMARY KEY(transactionID),
);

CREATE TABLE Product (
        productID INT,
        color VARCHAR(255),
        price FLOAT,
        brandName VARCHAR(255),
        name VARCHAR(255),
        count INT,
        PRIMARY KEY(productID)
);
```

```
CREATE TABLE Top (
	productID INT,
	hood TINYINT(1),
	size VARCHAR(5),
	pocket TINYINT(1),
	zipper TINYINT(1),
	PRIMARY KEY(productID),
	FOREIGN_KEY (productID) REFERENCES Product(productID)
);

CREATE TABLE Bottom (
	productID INT,
	waistSize INT,
	lengthSize INT,
	PRIMARY KEY(productID),
	FOREIGN_KEY (productID) REFERENCES Product(productID)
);

CREATE TABLE Shoe (
	productID INT,
	shoeSize FLOAT,
	PRIMARY KEY(productID),
	FOREIGN_KEY (productID) REFERENCES Product(productID)
);

CREATE TABLE Carry (
	storeName VARCHAR(255),
	productID INT,
	PRIMARY KEY(storeName, productID)
	FOREIGN_KEY (storeName) REFERENCES Store(storeName),
	FOREIGN_KEY (productID) REFERENCES Product(productID)
);

CREATE TABLE CustomerPurchases (
	customerID INT,
	transactionID INT,
	quantity INT,
	PRIMARY KEY(customerID, transactionID),
	FOREIGN_KEY (customerID) REFERENCES Customer(customerID),
	FOREIGN_KEY (transactionID) REFERENCES Transaction(transactionID)
);
```

```
CREATE TABLE Purchases (
        transactionID INT,
        productID INT,
        quantity INT,
        PRIMARY KEY(transactionID, productID),
        FOREIGN_KEY (transactionID) REFERENCES Transaction(transactionID),
        FOREIGN_KEY (productID) REFERENCES Product(productID)
);

CREATE TABLE Cart (
        transactionID INT,
        customerID INT,
        productID INT,
        PRIMARY KEY(transactionID, productID),
        FOREIGN_KEY (customerID) REFERENCES Customer(customerID),
        FOREIGN_KEY (productID) REFERENCES Product(productID)
);
```

10) Populating tables with sample data

```
        LOAD DATA INFILE 'storeInput.txt'
        INTO TABLE Store COLUMNS TERMINATED BY '\t';

        LOAD DATA INFILE 'customerInput.txt'
        INTO TABLE Customer COLUMNS TERMINATED BY '\t';

        LOAD DATA INFILE 'transactionInput.txt'
        INTO TABLE Transaction COLUMNS TERMINATED BY '\t';

        LOAD DATA INFILE 'productInput.txt'
        INTO TABLE Product COLUMNS TERMINATED BY '\t';

        LOAD DATA INFILE 'topInput.txt'
        INTO TABLE Top COLUMNS TERMINATED BY '\t';

        LOAD DATA INFILE 'bottomInput.txt'
        INTO TABLE Bottom COLUMNS TERMINATED BY '\t';

        LOAD DATA INFILE 'shoeInput.txt'
        INTO TABLE Shoe COLUMNS TERMINATED BY '\t';
```

```
LOAD DATA INFILE 'carryInput.txt'
INTO TABLE Carry COLUMNS TERMINATED BY '\t';
```