FINAL REPORT

BCD Mart

Hyungtaek Kwon

Youngwon Choi

Table of Contents

1. Database Design

1.1 Goals

This project has the purpose to successfully create a web application with JSP and MySQL. The main domain consists of shoe market where users can buy shoes, upload shoes, give certain feedback about the shoes, post articles and receive coupons. It was aimed to simulate real-life trading shoe marketplaces with multiple functions.

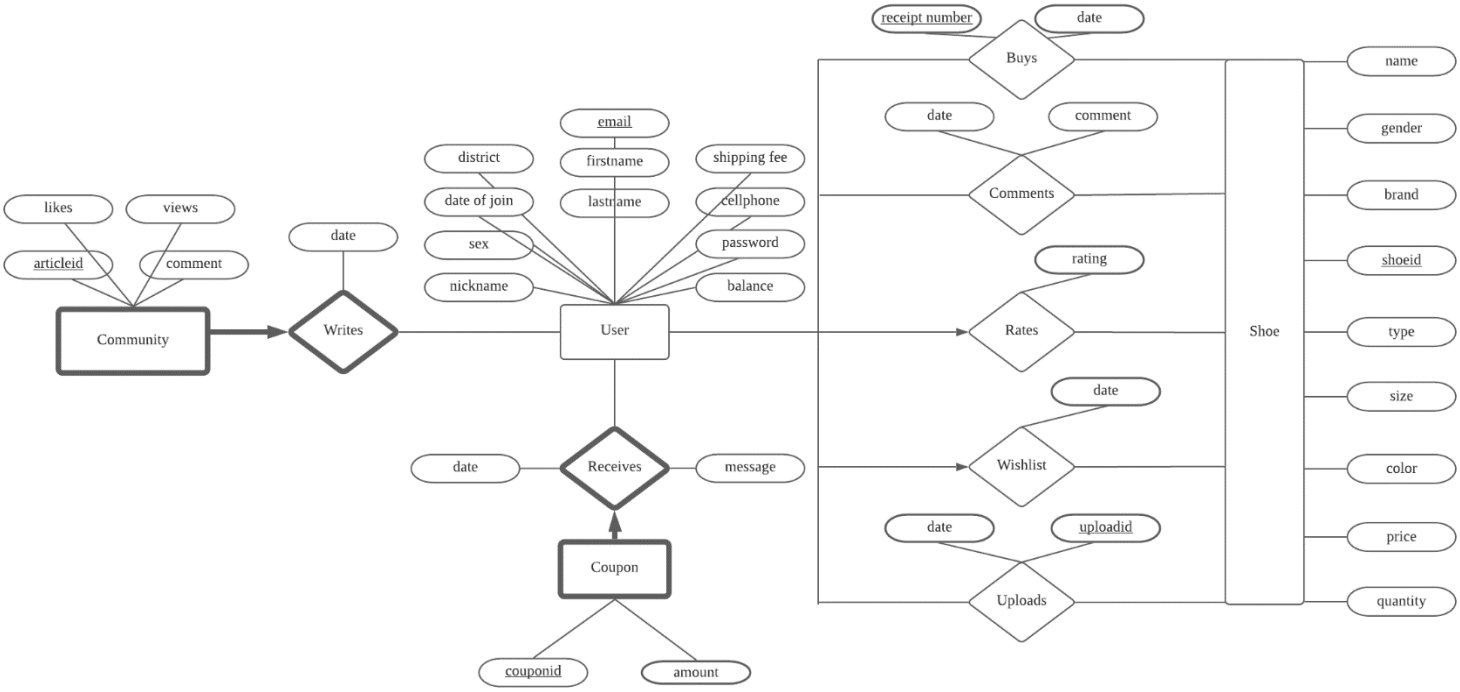
1.2 Requirements

This project was developed in a Windows 10 environment using XAMPP and MariaDB as the database client. The server that hosted the website was Apache Tomcat and the programming language was Java. Additionally, JSP, Javascript, HTML, and CSS were used to develop the website and JDBC was used to connect front and back end. MySQL was used as the query language.

1.3 FEATURES

1) Login/Logout
2) Register Account
3) Modify Account Information
4) Buy shoe
5) Rate, Comment, Wish shoe
6) Sell Shoe
7) Post Articles
8) Like Articles
9) Modify, Delete Articles
10) Receive Coupons

1.4 ER DIAGRAM



1.5 DB TABLES

- Sizes (sizes, gender)
- Brand (brand, type)
- Brand_to_Name (brand, name, price)
- District (district, shipping)
- Users (email, nickname, first_name, last_name, password, district, cell_num, balance, gender, joindate)
- Shoe (upload_id, brand, shoe_name, color, quantity, sizes, address)
- Uploads (upload_id, email, dates)
- Community (article_id, likes, views, title, content, email)
- Coupon (coupon_id, email, amount)
- Writes (article_id, email, dates)
- Receives (email, coupon_id, dates, message)
- Buys (receipt_num, dates, email, upload_id)
- Comments (email, upload_id, dates, comment)

- Wishes (email, upload_id, dates)
- Rates (email, upload_id, rating)

1.6 NORMALIZATION

Two tables in the database were not in BCNF: "Shoe" and "User".

1.6.1 Shoe relation.

Given shoe as:

Shoe(upload_id, gender, brand, shoe_name, type, size, color, price, quantity)

Represented as:

Shoe{UGBNTSCPQ} in 2NF

Where:

U = upload_id ; G= gender ; B= brand ; N = shoe_name T= type ; S=size ; C=color ; P=price ; Q=Quantity

Functional Dependencies:

1. U→GBTSCPQNE
2. BN→P; Breaks BCNF because BN is not a candidate key or P is a subset of BN
3. S→G ; Breaks BCNF because S is not candidate key or G a subset of S
4. B→T ; Breaks BCNF because B is not a candidate key or T a subset of S

Decomposition:

D1 = BNPT ; Breaks BCNF because left hand side of BN→P is not superkey or PT a subset of BN.

D2 = UGBNSCQ : Breaks BCNF because left hand side of S→G is not superkey or G subset of S.

D1.1 = BT : is in BCNF

D1.2 = BNP : is in BCNF

D2.1 = SG : is in BCNF

D2.2 = UBSCQNE : is in BCNF

Lossless join:

D1.1 ∩ D1.2 = B

S5 ∩ S6 = S

Final decomposition:

Brand(BT) and Brand_to_Name(BNP)

Sizes(SG) and Shoe (UBSCQNE)

The final relations are lossless and dependency preserving since all FD's hold over the decomposed relations. BNP relation helps reduce price redundancies and SG relation reduces shoe size redundancies.

1.6.2 User relation.

Given User as:

*User(*email, nickname, first_name, last_name, password, district, cell_num, balance, gender, shipping, joindate)

Represented as:

User{ENFLPDCBGSJ}

Where:

E = email ; N = nickname ; F = firstname ; L = lastname ; P = password ;
D= district ; C= cellphone number ; B= Balance ; G= Gender ;
S=Shipping fee ; J=joindate

Functional Dependencies:

1. E→NFLPDCBGS
2. D→S (Transitive FD)

Decomposition:

D1 = DS

D2 = ENFLPDCBGJ

Lossless join:

D1 ∩ D2 = D

Final Decomposition:

District (DS) and User (ENFLPDCBGJ)

The final relations are lossless and dependency preserving since all FD's hold over the decomposed relations.

2. Implementation (use cases)

1) Login/Logout

For logging in, users insert their email and password to get accessed as the user of the web page. The process is continued by determining whether the information is valid.

If all the information match, the session would set this email as 'userID' and redirect you to main page. If the information does not match, it would alert the user to reinsert email and password.

2) Register

The user can register in the database by inputting their email and password. Emails are unique and there cannot be duplicates.

3) Modify User Information

Once registered and logged in into the database, users can choose to change additional information about themselves such as nickname, district, gender, first name, last name, phone number, and balance (for demonstration purposes)

4) Buy Shoe

Users access the buy tab on the website and can scroll down a list of shoes available. Once a user decides on a shoe, the user selects the shoe and can purchase the shoe. The purchase is then recorded, and the price of the shoe and shipping fee is discounted from the user's balance. If the user does not have enough balance or if the shoe is out of stock, the user cannot purchase the shoe.

5) Rate, Comment, Wish Shoe

A user can interact with individual shoe listings on the buy tab. A user can rate the shoe on scale from 0 to 5, comment on the shoe, or add the shoe to the wish list.

6) Sell shoe

Users can sell their shoes by selecting the sell tab on the website. After clicking, users can enter data about the shoes that includes name of the shoe, brand, color, a picture, quantity, sizes, price of shoe and type of shoe. Once the user is satisfied with inputted data, the user can upload the shoe for purchase. Once a shoe has been purchase, the amount is automatically added to the users balance. But the name and the brand and the price should match to be uploaded.

7) Post Articles

A user can write articles on the website community.

8) Like Articles

A user can "like" other user's articles on the community.

9) Modify, Delete Articles

A user can modify or delete the articles they have created on the community.

10) Receive Coupons

A user can choose to receive a random generated coupon that gives coupon values ranging from 1,000 – 10,000. The user then can choose to accept the coupon, which will be automatically added to the user's balance.

3 Further Steps

1. Goals Met

1) Users can login and logout easily if the database holds their records
2) Building database to hold information of users, shoes, community and coupons
3) Users can buy and sell shoes
4) Users can leave feedback about the shoes so other users can interact each other
5) Users can leave articles on the community section and like the article
6) Users are able to receive coupons that can aid their balance
7) 3 Triggers were made to aid the complex process of connecting other tables
    a. ArticleTrig: simultaneously inserts values into Community and Writes relations upon INSERT on Community.
    b. CouponTrig : after INSERT on coupon, automatically adds coupon amount to user balance and registers coupon in Receives tuple.
    c. BuyTrig: BEFORE INSERT on Buys relation, Checks whether user has enough balance to buy a shoe and whether a shoe is in stock or not. If both conditions are satisfied, the trigger deducts the shoe price from the user's balance and reduces the stock of the shoe.

2. Goal's Not Met

1) BuyTrig: We could not prevent the insertion of "buys" tuple if User does not have money to buy the shoes or if the shoes are not in stock. MySQL currently does not have a function to prevent or skip insertions in Triggers.
2) Website user interface and design was minimal unlike other shoe market's online.

3. Further Improvements

Due to time constraints, we were not able to add the previous features but with more time, these features could be developed or other methods could be used. Currently, MySQL does not support the preventing or skipping insertions through triggers and thus other methods could be used to achieve the same goal.

4. Future Potential Features

Bank Account Feature: Provide section where users can add their bank and card number and purchase shoes using a card rather than manually setting the balance.

Shopping Cart: Feature where users can put multiple shoes inside a shopping card and purchase all at once instead of one by one.

Percentage Coupons: Coupons with percentage discounts applied at checkout rather than fixed coupons that add to a user's balance.