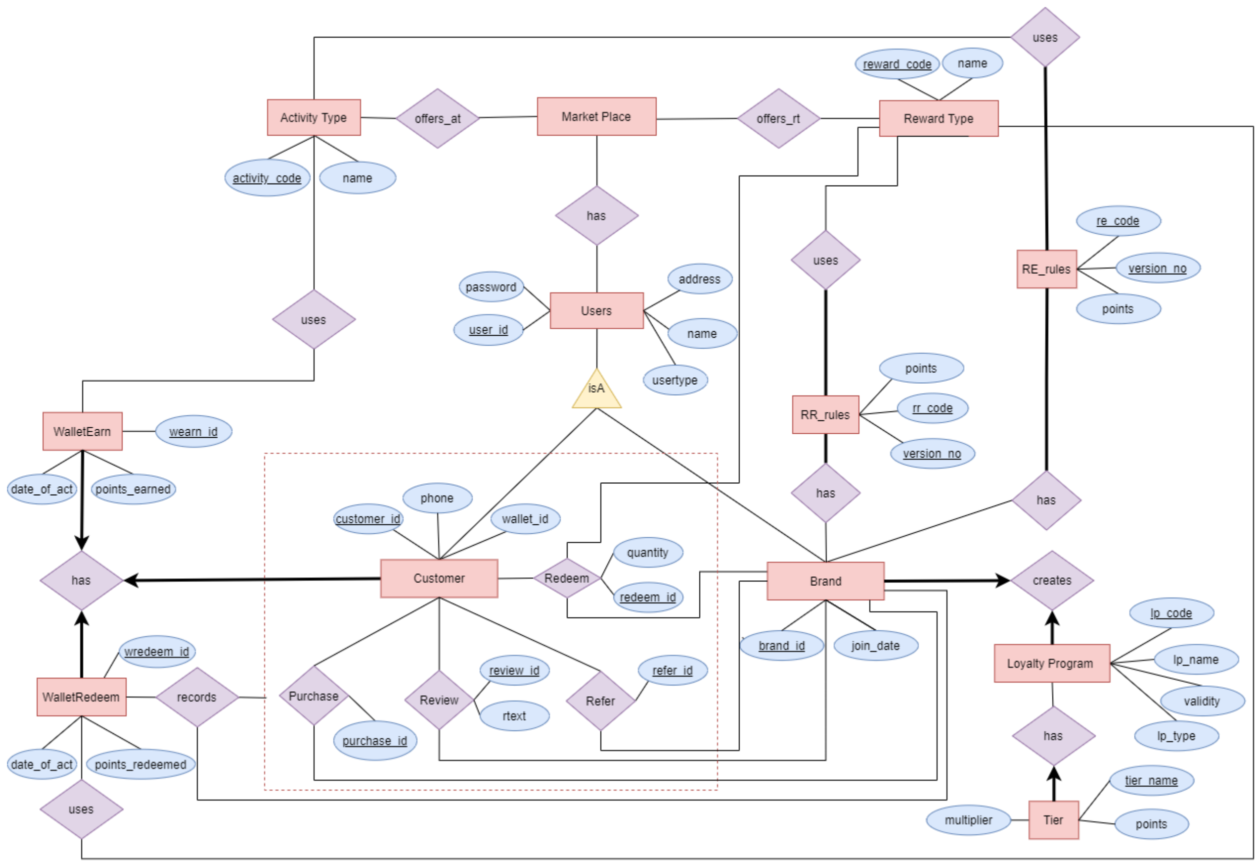
Final Report

Customer Loyalty Marketplace Application

CSC 540 - Database Application Programming

Submitted by Team 10

1. Final ER model



2. Two SQL files:

a) Triggers, tables, constraints, procedures

The ***DDL.sql*** file is attached.

b) Queries for populating the tables with the sample data

The ***DML.sql*** file is attached.

3. Short Description

3.1 Check Constraints and Stored Procedures

3.1.1 **Check constraints:** We have used a few of them, which are described below:

(i) In the USERS table, we use a check constraint to verify the user type (USERTYPE). By using this, we ensure that the users (customer, brand or admin) are logged in properly under the user types they are supposed to be, and there is no conflict as data is inserted into the database.

(ii) In the LOYALTYPROGRAM table, we use two check constraints. The first one is for validating the creation of a loyalty program (ISVALID) upon adding activities, rewards and associated rules by the corresponding brand. The second one is for maintaining the loyalty program type (LPTYPE), which helps us keep the regular LP differentiate from the tiered LP.

(iii) In the GIFTCARD table, we use a check constraint to check the status of a gift card. It is a binary variable (GCUSED), which helps us verify its status before inserting it into the database.

3.1.2 **Procedures:** We have used a list of procedures and their uses can be described as follows.

i. validate_loyalty_program

In this procedure, we check if a loyalty program is tiered without multiple tiers or has no reward earning/redeeming rule.

ii. add_re_rule

We used this function to add a reward earning rule in the **re_rules** table. In this function we check if there is already an existing rule with the same activity type. If it is the case, we return without adding the rule in the table.

iii. update_re_rule

In this stored procedure, we check if there is already an existing reward earning rule, if not, we return without updating anything.

iv. add_rr_rule

We used this function to add a reward redeem rule in the **rr_rules**
table. In this function we check if there is already an existing rule with the same reward type. If it is the case, we return without adding the rule in the table.

v. update_rr_rule

In this store procedure, we check if there is already an existing reward redeeming rule, if not, we return without updating anything.

vi. customer_redeem

In this procedure, we check if there are enough earned points in the customer wallet while redeeming points for a gift.

vii. get_next_id

We use this function to generate different unique codes which are used in different tables as a primary key.

Moreover, there are primary key constraints in all tables, and foreign key constraints in most of the tables.

3.2 Functional dependencies: In our design, we have the following relations.

Users(UserID, Password, UserType);
Functional Dependencies: {UserID → Password, UserType}
Primary Key: <u>UserID</u>

Brand(Bid, BName, Address, JoinDate)
Functional Dependencies: {Bid → BName, Address, JoinDate}
Primary Key: <u>Bid</u>

ActivityType(ActivityCode, ActivityName)
Functional Dependencies: {ActivityCode → ActivityName}
Primary Key: <u>ActivityCode</u>

RewardType(RewardCode, RewardName)
Functional Dependencies: {RewardCode → RewardName}
Primary Key: <u>RewardCode</u>

ReRules(BrandId, ActivityCode, Points, VersionNo)
Functional Dependencies: {BrandId, ActivityCode, VersionNo → Points}
Primary Key: <u>BrandId</u>, <u>ActivityCode</u>, <u>VersionNo</u>

RrRules(BrandId, RewardCode, Points, VersionNo)
Functional Dependencies: {BrandId, RewardCode, VersionNo → Points}
Primary Key: <u>BrandId</u>, <u>RewardCode</u>, <u>VersionNo</u>

LoyaltyProgram(LpCode, LpName, LpType, IsValid, BrandId)
Functional Dependencies: {LpCode → Points, LpName, LpType, IsValid, BrandId,
BrandId → LpName, LpType, IsValid, LpCode}
Keys: {LpCode, BrandId }
Primary Key: <u>LpCode</u>

Tier(LpCode, TierName, Points, Multipliers)
Functional Dependencies: {LpCode, TierName → Points,Points, Multipliers}
Primary Key: <u>LpCode</u>, <u>TierName</u>

BrandActivityType(BrandId, ActivityCode)
Functional Dependencies: {}
Primary Key: <u>BrandId</u>, <u>ActivityCode</u>

BrandRewardType(BrandId, RewardCode, TotalQuantity, CurQuantity)
Functional Dependencies: {BrandId, RewardCode → TotalQuantity, CurQuantity}
Primary Key: <u>BrandId</u>, <u>RewardCode</u>

Customer(CustomerId, FName, LName, Address, PhoneNumber, WalletID)
Functional Dependencies: {CustomerId → FName, LName, Address, PhoneNumber, WalletID
WalletID → FName, LName, Address, PhoneNumber, CustomerId}
Keys: {CustomerId, WalletID }
Primary Key: <u>CustomerId</u>

EnrolLP(CustomerId, LpCode, PointsEarned, EnrolDate)
Functional Dependencies: {CustomerId, LpCode → PointsEarned, EnrolDate}
Primary Key: <u>CustomerId</u>, <u>LpCode</u>

WalletRE(Ser, CustomerID, Bid, ActivityCode, PointsEarned, DateOfActivity)
Functional Dependencies: {Ser → CustomerID, Bid, ActivityCode, PointsEarned, DateOfActivity}
Primary Key: <u>Ser</u>

WalletRR(Ser, CustomerID, Bid, RewardCode, PointsRedeemed, DateOfActivity)
Functional Dependencies:{Ser→CustomerID, Bid, RewardCode, PointsRedeemed, DateOfActivity}
Primary Key: <u>Ser</u>

Review(ReviewID, CustomerID, Bid, ReviewText)
Functional Dependencies: {ReviewID → CustomerID, Bid, ReviewText}
Keys: {ReviewID, (CustomerID, Bid)}
Primary Key: <u>ReviewID</u>

Purchase(PurchaseID, CustomerId, Bid, GiftCardUsed)
Functional Dependencies: {PurchaseID → Points, CustomerId, Bid, GiftCardUsed}
Primary Key: <u>PurchaseID</u>

Redeem(Redeem, RewardCode, CustomerId, Bid, Quantity)
Functional Dependencies: {RedeemID → RewardCode, CustomerId, Bid, Quantity}
Primary Key: <u>RedeemID</u>

ReferFriend(RfId, CustomerId, Bid)
Functional Dependencies: {RfId → CustomerId, Bid}
Primary Key: <u>RfId</u>

4. **Executable file (e.g., executable JAR file) and source Java Code.**
   - Attached.

5. **README.txt file that contains the names of the team members and explains with any additional instructions on how to compile and execute your code.**
   - Attached.