CSC 540 T14 Project 1 Report

Team Members:

Ashok Kumar Selvam

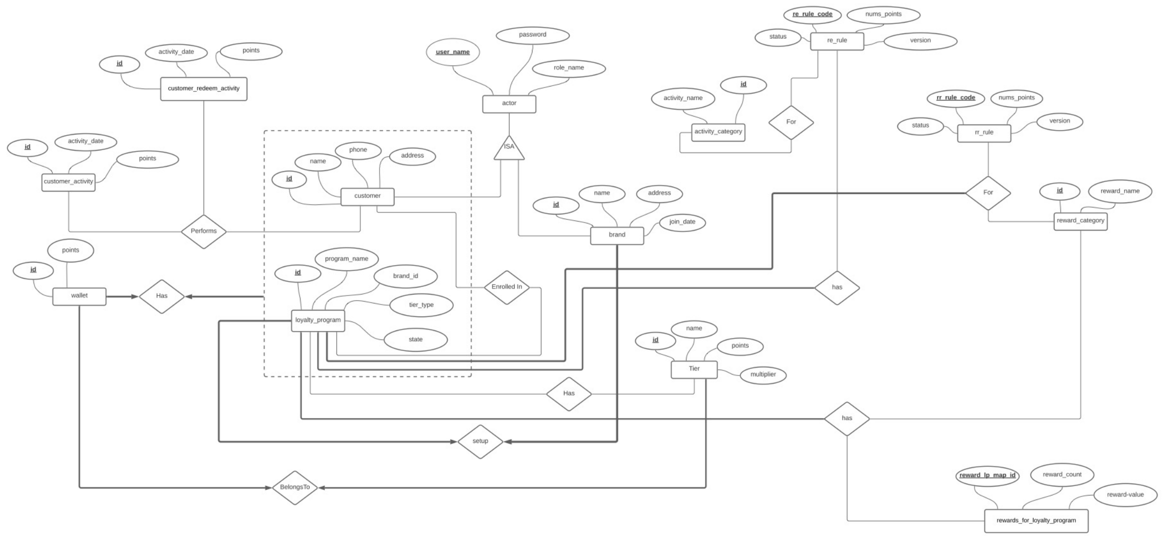
Arvind Srinivas Subramanian

Sumedh Sanjay Salvi

Aakash Satish Poliyath

ER Model:

Customer Loyalty Marketplace Application



SQL Queries:

1. DDL Queries:

   540-P1-team_name/DBMSProject1/DDL.sql

2. DML Queries:

   540-P1-team_name/DBMSProject1/DML.sql

Constraints:

Constraints handled in DB:

Check Constraints:

1. TIER:

   check if points and multiplier $\geq$ 0

2. Wallet:

   Check if points $\geq$ 0

3. Rewards for Loyalty program:

   Check if reward count and reward value $\geq$ 0

4. Customer redeem activity:

   Check if points $\geq$ 0

5. Customer reward activity:

   Check if points $\geq$ 0

Procedures:

1. *`update_customer_tier(customer_id, loyalty_program_id)`*

   This procedure takes customer id and loyalty program as parameters and calculate and update the tier of the customer for a loyalty program based on the points in the wallet.

Triggers:

1. `update_reward_count_and_wallet`

   This trigger is set to execute when the customer performs a redeem activity.

   This trigger performs 3 activities

   a. Decrement reward count

   b. Decrease the points used by the customer to redeem reward from the customer's wallet

    c. Update the tier of the customer based on the updated points (execute update_customer_tier)

2. `calc_points`

   This trigger is set to execute when the customer performs a reward activity.

   This trigger performs 3 activities

    a. Fetch the points set for the performed activity from the rules table and update it in the activities table.

    b. Add the points earned by the customer from the activity in the wallet.

    c. Update the tier of the customer based on the updated points (execute update_customer_tier)

3. `brand_insert_trigger`

   This trigger creates an entry in the actor table(used for credentials) whenever a brand is added. Default password is set to 'abcd1234'

4. `customer_insert_trigger`

   This trigger creates an entry in the actor table(used for credentials) whenever a customer is added. Default password is set to 'abcd1234'

5. `customer_wallet_trigger`

   This trigger creates an entry in the wallet table whenever a customer enrolls in a loyalty program.

Constraints not handled in DB:

1. A loyalty program can only have 3 tiers

   There is no way to add a check constraint to limit number of rows with a particular value in a table. Therefore, this constraint cannot be handled in the DB. We added a check in the application which will enforce this constraint.

Functional Dependencies:

1. Actor(username, password, role_name)

   FD = {username->password,role_name}

   CK = username

   NF = BCNF

2. Activity_category(id, activity_name)

   FD = {id->activity_name}

   CK = id

   NF = BCNF

3. Re_rule(re_rule_code, activity_category_code, num_points, version, status, lp_Code)

   FD = {re_rule_code,version->activity_category_code,nums_points,version,status,lp_code
   activity_category_code, version, lp_Code -> num_points, re_rule_code, status}

   CK = {(re_rule_code, version), (activity_category_code, version, lp_Code)}

   NF = BCNF

4. Rr_rule(rr_rule_code, reward, num_points, version, status, lp_code)

   FD = {rr_rule_code, version->reward,num_points,version,status,lp_code
   reward, version, lp_Code -> num_points, rr_rule_code, status}

   CK = {(rr_rule_code, version), (reward, version, lp_code)}

   NF = BCNF

5. Brand(id,name,address,join_date,user_name)

   FD = {id->name,address,join_date,user_name
   name -> id, address, join_date, user_name}

   CK = id
   NF = BCNF

6. Loyalty_program(id,program_name,brand_id,tier_type,state)

   FD = {id->program_name,brand_id,tier_type,state
   brand_id -> id,program_name,tier_type,state)}

   CK = {id, brand_id}

   NF = BCNF

7. activities_for_loyalty_program(activity_lp_map_id, loyalty_program_code,activity_category_code)

   FD = {activity_lp_map_id->loyalty_program_code,activity_category_code
   loyalty_program_code, activity_category_code->activity_lp_map_id}

   CK = {activity_lp_map_id, (loyalty_program_code, activity_category_code)}

   NF = BCNF

8. Customer(id,name,phone,address,user_name)

   FD = {id->name,phone,address,user_name}

   CK = id

   NF = BCNF

9. Tier(id,name, points, multiplier, lp_program_id)

   FD = {id-> name, points, multiplier, lp_program_id}

   CK = id

   NF = BCNF

10. customer_redeem_activity(id,customer_id,activity_date,redeem_lp_map_id,points)

    FD = {id -> customer_id,activity_date,redeem_lp_map_id,points}

    CK = id

    NF = BCNF

11. customer_activity(id,customer_id,activity_date,activity_lp_map_id, customer_redeem_activity_id,points)

    FD = {id -> customer_id,activity_date,activity_lp_map_id,
    customer_redeem_activity_id,points}

    CK = id

NF = BCNF

12. reward_category(id, reward_value)

    FD = {id -> reward_name}

    CK = id
    NF = BCNF

13. Wallet(id,points,customer_id,loyalty_program_code, tier_id)

    FD = {id -> points,customer_id,loyalty_program_code, tier_id
    customer_id, loyalty_program_code -> id, points, tier_id}

    CK = {id, (customer_id, loyalty_program_code)}

    NF = BCNF

14. customer_lp_enroll(customer_id,loyalty_program_code)

    FD = {}

    CK = (customer_id,loyalty_program_code)

    NF = BCNF

15. rewards_for_loyalty_program(reward_lp_map_id, loyalty_program_code,reward_category_code,reward_count,reward_value)

    FD = {reward_lp_map_id->loyalty_program_code,reward_category_code,reward_count,reward_value
    reward_category_code,loyalty_program_code ->
    reward_lp_map_id,reward_count,reward_value}

    CK = {reward_lp_map_id, (reward_category_code,loyalty_program_code)}

    NF = BCNF

In all the above relations, all the functional dependencies are either driven by the primary key or by a candidate key.
Hence, we can conclude that all the relations are in BCNF.