Project 1 Part 2

Kunpeng Liu (kl3070) Zichen Pan (zp2197)

Used UNI: zp2197

1. Queries

(1) select the customer with the most products

SELECT customers.user\_id

FROM products, customers, carts

WHERE customers.use\_id = carts.customer\_id and

carts.cart\_id = products.cart\_id

GROUP BY cutomers.user\_id

ORDER BY count(\*) DESC

LIMIT 1;

(2) select the best seller in term of the number of sold products

SELECT sellers.user\_id

FROM sellers, products

WHERE sellers.user\_id = products.seller\_id and

cart\_id in (select cart\_id from carts)

GROUP BY user\_id

ORDER BY count(\*) DESC

LIMIT 1;

(3) a complex one: select the customer with most expensive products after using coupons

CREATE VIEW discounted\_products AS(

SELECT products.product\_id, products.price \* (1 -(coupons.discount))

FROM products, coupons, coupon\_applied

WHERE coupons.coupon\_id = coupon\_applied.coupon\_id and

coupon\_applied.product\_id = products.product\_id

);

CREATE VIEW final\_price AS(

(SELECT product\_id, price

FROM products

WHERE product\_id not in (SELECT product\_id FROM discounted\_products))

UNION

(SELECT \* FROM discounted\_products)

);

SELECT customers.user\_id

FROM products, customers, carts, final\_price

WHERE customers.user\_id = carts.customer\_id and

carts.cart\_id = products.cart\_id and

products.product\_id = final\_price.product\_id

GROUP BY customers.user\_id

ORDER BY sum(final\_price.price) DESC

LIMIT 1;

2. Changes to schema design

(1) Drop column total\_price in table Carts, which can be calculated and selected from other base data in database.

(2) Drop column discounted\_price in table Products for the same reason.

(3) Create a trigger for table Coupon\_applied to ensure the owners of a row of coupon and product are the same, which enforces that a coupon delivered by a seller can only be applied to his products.

CREATE FUNCTION checkcoupon() RETURNS trigger

AS $$

BEGIN

IF (NEW.product\_id in (

SELECT P.product\_id

FROM Coupons C, Products P

WHERE New.coupon\_id = C.coupon\_id and

C.seller\_id = P.seller\_id)) THEN

RETURN NEW;

ELSE

RETURN null;

END IF;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER t\_checkcoupon BEFORE INSERT ON coupon\_applied

FOR EACH ROW

EXECUTE PROCEDURE checkcoupon();

(4) Delete table Customers\_Carts. The relationship between Customers and Carts are complicated. Carts are regarded as a weak entity to Customers and the relationship is one-to-one. Instead of creating one more table Customers\_Carts to enforce the one-to-one relationship, which adds redundancy, we add UNIQUE for column customer\_id in table Carts which ensures that one customer can only have one cart. However, we still cannot ensure full participation of customers. We made a tradeoff to reduce redundancy.