Online Shopping Website Database Design Report

Group 10
Wu Yuhang
Li Yuanyang
Li Zhili
Wang Teng
Lu Tianyu
Zhou Heng
Ouyang Wenzhuo

1 Table of Contents

2 Demand Analysis

Our e-commerce platform database design has the following requirements:

There are two main bodies:

- **Users**: We need to record the information of each user who registers with the e-commerce platform, including their user ID, username, user contact number, user address, and merchants they follow.
- **Shops**: There are many shops on the e-commerce platform, and each shop has its corresponding shop ID, shop name, shop contact number, type of shops (e.g. electrical store, furniture store, etc.) and all of its products.
  - **Product**: Each shop has a number of products, and each product has a unique identifier of the e-commerce platform. At the same time, we also need to store the name of the product, the product type, the product image, as well as the product price and product reviews.
    - **Product reviews**: This e-commerce platform has several user reviews for each specific product. For each review, we need to record the review user ID, review time, review score (1 to 5 stars), and review text.

Logging user's actions:

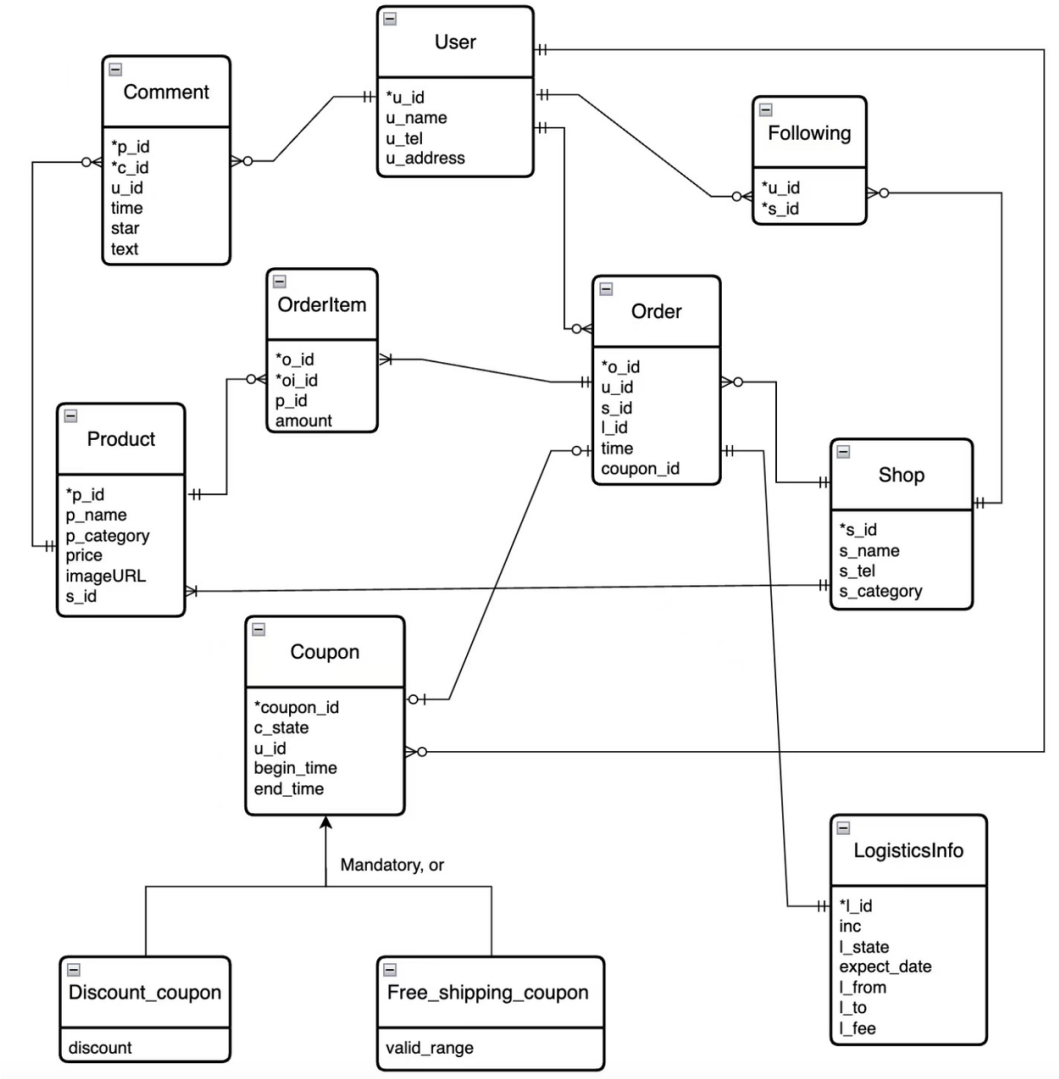
- **Order**: We specify that every time a user makes a purchase, an order will be generated, and each order can only have products from the same shop. In other words, the user can only purchase at one shop in each round. For each order, we need to record its time, order item, shipping information, and coupon usage.
  - **Logistics message**: The logistics message should include logistics tracking number, logistics company, logistics status, expected delivery time, shipping address and receiving address, and logistics cost.
  - **Coupons**: We specify that there are two types of coupons on the e-commerce platform, one is discount coupon, the other is free shipping coupon. Each coupon has a unique identifier of the platform, coupon status (whether used or not), the user to which the coupon belongs, the effective time and the expiration time of the coupon. We stipulate that only one coupon can be used per purchase, and each coupon can only be used once.
    - **Discount coupon**: The discount coupon has a discount attribute on top of this, which records the percentage discount of the item ($0\% \sim 100\%$).

- **Free shipping coupon**: On this basis, the free shipping coupon needs to note the valid range, that is, free shipping can be provided within this delivery range.

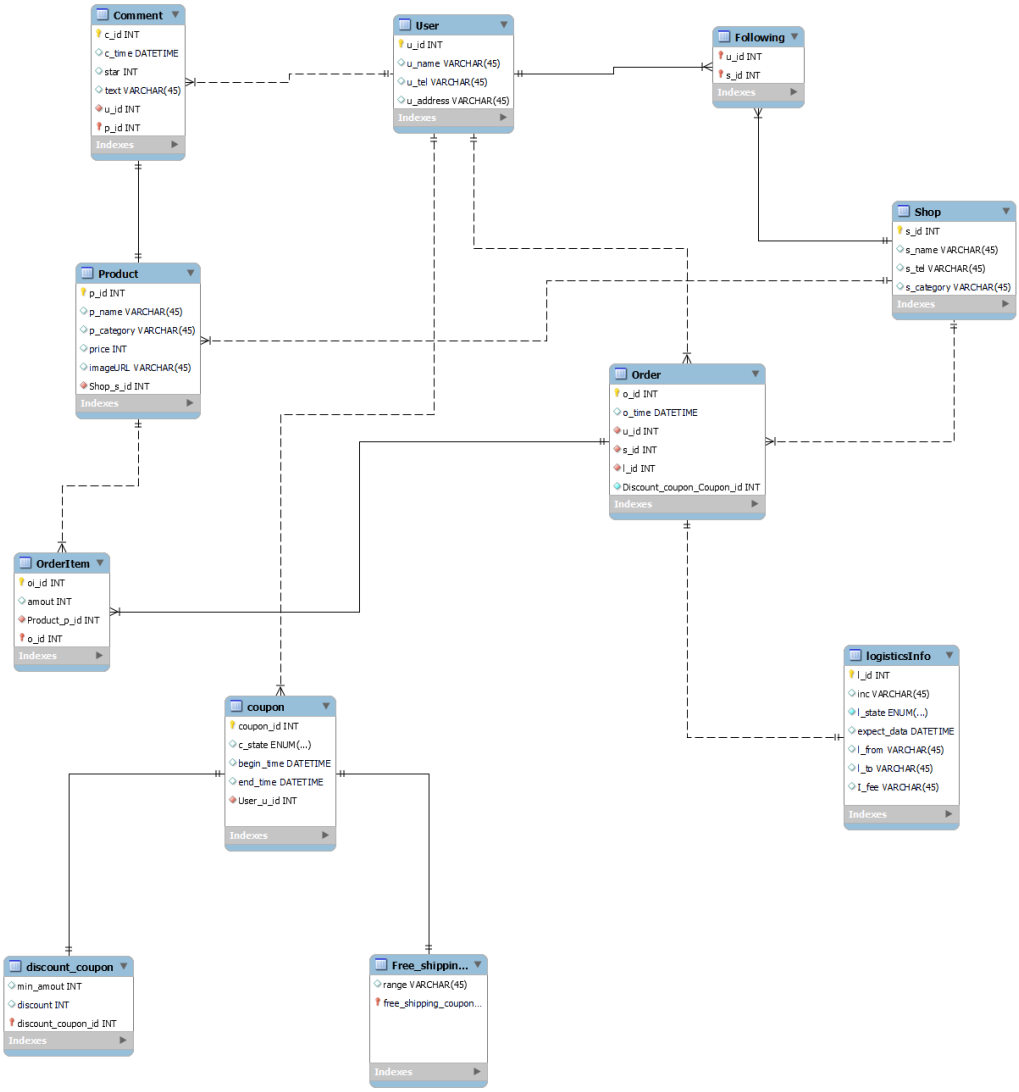
3 EER Diagram of the Conceptual Model

Below is the EER diagram of our designed conceptual model, illustrating the relationships and attributes among entities:

3.1 EER Diagram



Entity relationship diagram drawn on the workbench:



3.2 Entity Definitions

Note:

- **User**: The primary key is the unique identifier u_id, with attributes for username, user phone number, and user address.
- **Shop**: The primary key is the unique identifier s_id, with attributes for shop name, shop phone number, and shop category.
- **Product**: The primary key is the unique identifier p_id, with attributes for product name, product category, product price, and product image URL. It also has a foreign key s_id to associate it with the corresponding shop.
- **Following**: This entity represents the relationship between users and shops they follow. Since a user can follow multiple shops, and a shop can be followed by multiple users, an additional entity Following is needed to describe this many-to-many relationship. The primary key is (u_id, s_id), with no other non-key attributes.
- **Comment**: This entity is used to record user comments on products. The primary key is (p_id, c_id), which represents the associated product and comment number. The attributes include u_id, time, star rating, and comment text.

- **Order**: The primary key is the unique identifier o_id, with foreign keys u_id and s_id. It also has attributes for order time, total transaction amount, and a foreign key coupon_id, which can be NULL to indicate the absence of a coupon for the order.
- **OrderItem**: This entity stores information about each product item in an order. The primary key is (o_id, oi_id), representing the associated order and order item number. It also has a foreign key p_id and an attribute for product quantity (amount).
- **LogisticsInfo**: This entity stores the logistics information for each order. The primary key is the logistics tracking number l_id, with attributes for order ID, logistics company, shipment status, expected delivery time, shipping address, and delivery address.
- **Coupon**: We use specialization techniques to design the coupon's parent class. The primary key is coupon_id, and it has an attribute for the state. It also has attributes for the associated user ID u_id, start time begin_time, and end time end_time. There are two subclasses:
  - **Discount_coupon**: This subclass has attributes for minimum transaction amount min_amount and discount percentage discount.
  - **Free_shipping_coupon**: This subclass has an attribute for the eligible shipping range.

This EER diagram design supports our requirements by clearly defining the relationships and attributes among entities, enabling the association and data flow between customers, products, orders, order items, payments, shopping carts, comments, coupons, and logistics.

4 Relational Database Schema

4.1 Relational Schema

User($\underline{u\_id}, u\_name, u\_tel, u\_address$)

Primary Key: u_id

Product($\underline{p\_id}, p\_name, p\_category, price, imageURL, s\_id$)

Primary Key: p_id

Foreign Key: s_id **references Shop**(s_id)

Shop($\underline{s\_id}, s\_name, s\_tel, s\_category$) **Primary Key**: s_id

Comment($\underline{c\_id}, \underline{p\_id}, u\_id, c\_time, star, text$)

Primary Key: (c_id, p_id)

Foreign Key: u_id **references User**(u_id), p_id **references Product**(p_id)

Following(*u_id*, *s_id*)

Primary Key: (u_id, s_id)

Foreign Key: u_id **references User**(u_id), s_id **references Shop**(s_id)

Order(*o_id*, *u_id*, *s_id*, *l_id*, *coupon_id*, *o_time*)

Primary Key: o_id

Foreign Key: u_id **references User**(u_id), s_id **references Shop**(s_id), l_id **references LogisticsInfo**(l_id), coupon_id **references Coupon**(coupon_id)

OrderItem(*oi_id*, *o_id*, *product_p_id*, *amount*)

Primary Key: (oi_id, o_id)

Foreign Key: product_p_id **references Product**(p_id), o_id **references Order**(o_id)

LogisticsInfo(*l_id*, *inc*, *l_state*, *expect_date*, *l_from*, *l_to*, *l_fee*)

Primary Key: l_id

Coupon(*coupon_id*, *u_id*, *c_state*, *begin_time*, *end_time*)

Primary Key: coupon_id

Foreign Key: u_id **references User**(u_id)

Discount_coupon(*discount_coupon_id*, *min_amount*, *discount*)

Primary Key: discount_coupon_id

Foreign Key: discount_coupon_id **references Coupon**(coupon_id)

Free_shipping_coupon(*free_shipping_coupon_id*, *range*)

Primary Key: free_shipping_coupon_id

Foreign Key: free_shipping_coupon_id **references Coupon**(coupon_id)

4.2 Validation using Normalization Techniques

Non-primary-key attributes are all fully functionally dependent on the primary key in each relation. So, all relations are in 2NF. There is no transitive dependency in each relation. All relations are in 3NF. Every determinate is a candidate key in each relation. Then all relations are in BCNF. There is no multi-valued dependency in each relation. Therefore, all relations are in 4NF.

5 Sample Test Data

In order to test the database querying operations, we have added the following sample test data to the database:

The test samples are data inserted by the program, combining datasets and randomly generated scripts.

Program detail see:Wu0219/sample_test_data_insert: a python script to insert sample test data to dataset_coursework of Group 10 (github.com)

There are totally 450k+ data inserted to the database and here are some samples for each table:

[comment]170000000,2020-09-14 20:46:51,5,A bit shorter than expected,110006252,130009901

[coupon]180000000,unused,110007299,1997-06-29 14:43:00,2025-04-06 00:00:00

[discount coupon]29,180000000

[free shipping coupon]190000000,only Beijing

[following]110000004,120000000

[logistics info]160000000,J.B. Hunt,in transit,2020-02-21,"34084 Thomas Estate Gravesmouth, PA 51572","2558 Zoe Isle Lake Mariamouth, SC 99031",10

[order]150000000,2022-03-03 17:02:54,110009950,120000103,160004740,180000000

[order item]140000000,130008819,150009480,6

[product]130000000,Garlic Oil - Vegetarian Capsule 500 mg,Beauty & Hygiene,918,https://vQGVtxDYDAhX/xEf.png,120000151

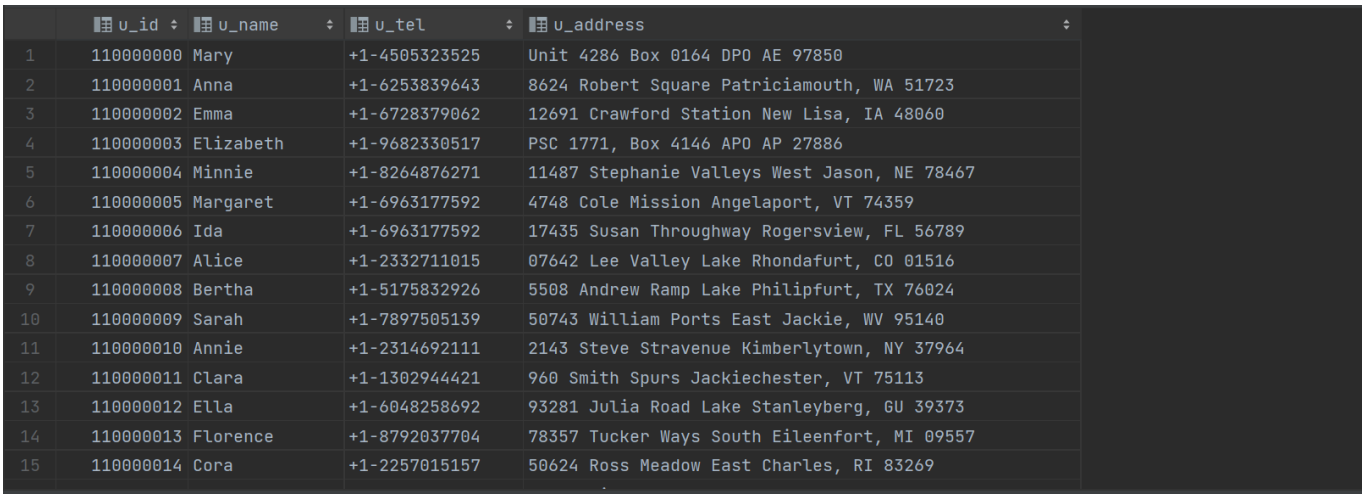
[shop]120000000,Amazon,+1-8061713222,"Foodgrains, Oil & Masala"

[user]110000000,Mary,+1-4505323525,Unit 4286 Box 0164 DPO AE 97850

6 SQL Queries and Query Results

In this section, we will demonstrate some common SQL query examples and provide the corresponding query results.

```sql
# Query the information about all users
select * from User;
```



	u_id	u_name	u_tel	u_address
1	110000000	Mary	+1-4505323525	Unit 4286 Box 0164 DPO AE 97850
2	110000001	Anna	+1-6253839643	8624 Robert Square Patriciamouth, WA 51723
3	110000002	Emma	+1-6728379062	12691 Crawford Station New Lisa, IA 48060
4	110000003	Elizabeth	+1-9682330517	PSC 1771, Box 4146 APO AP 27886
5	110000004	Minnie	+1-8264876271	11487 Stephanie Valleys West Jason, NE 78467
6	110000005	Margaret	+1-6963177592	4748 Cole Mission Angelaport, VT 74359
7	110000006	Ida	+1-6963177592	17435 Susan Throughway Rogersview, FL 56789
8	110000007	Alice	+1-2332711015	07642 Lee Valley Lake Rhondafurt, CO 01516
9	110000008	Bertha	+1-5175832926	5508 Andrew Ramp Lake Philipfurt, TX 76024
10	110000009	Sarah	+1-7897505139	50743 William Ports East Jackie, WV 95140
11	110000010	Annie	+1-2314692111	2143 Steve Stravenue Kimberlytown, NY 37964
12	110000011	Clara	+1-1302944421	960 Smith Spurs Jackiechester, VT 75113
13	110000012	Ella	+1-6048258692	93281 Julia Road Lake Stanleyberg, GU 39373
14	110000013	Florence	+1-8792037704	78357 Tucker Ways South Eileenfort, MI 09557
15	110000014	Cora	+1-2257015157	50624 Ross Meadow East Charles, RI 83269

```sql
# Query the number of followers of all merchants and sort by the number of followers
select
    s.s_id as s_id,
    s.s_name as s_name,
    count(f.u_id) as following_number
from
    Shop as s
left join
    Following as f
on
    s.s_id = f.s_id
group by
    s_id
order by
    following_number desc;
```

	s_id	s_name	following_number
1	120000236	ConocoPhillips	540
2	120000240	Tyson	526
3	120000285	Enterprise	521
4	120000396	Best Buy	519
5	120000023	Allianz Group	513
6	120000183	Metlife	511
7	120000127	Adidas	510
8	120000132	KPMG	509
9	120000085	Ford	508
10	120000211	LIC	507
11	120000046	Volkswagen	504
12	120000328	McKesson	504
13	120000348	HCLTech	504
14	120000017	Starbucks	503
15	120000313	Kelloggs	503
16	120000314	Fresenius	503
17	120000320	Taco Bell	503
18	120000004	Walmart	502
19	120000114	Postal Savings Bank	501
20	120000180	Zara	501
21	120000292	Movistar	501
22	120000022	Disney	500
23	120000118	BP	500

```sql
# Query the cost of all orders
select
    sum(oi.amount * p.price) * if(o.coupon_id like '18%', (select discount from
Discount_coupon as c1 where c1.discount_coupon_id = o.coupon_id) / 100, 1) + if(o.coupon_id
like '19%', 0, (select l_fee from Logisticsinfo as l where l.l_id = o.l_id)) as total_price
# Different prefixes represent different coupons
from
    `Order` as o
left join
    OrderItem as oi on oi.o_id = o.o_id
left join
    Product as p on oi.Product_p_id = p.p_id
group by
    o.o_id;
```

	o_id	total_price
1	150000000	565536.0000
2	150000001	1080307.0000
3	150000002	928675.0000
4	150000003	390426.0000
5	150000004	677327.0000
6	150000005	759507.0000
7	150000006	747826.0000
8	150000007	942441.0000
9	150000008	540811.0000
10	150000009	503988.0000
11	150000010	735898.0000
12	150000011	718697.0000
13	150000012	1057917.0000
14	150000013	456834.0000
15	150000014	880159.0000
16	150000015	764934.0000
17	150000016	512447.0000
18	150000017	763430.0000
19	150000018	929933.0000
20	150000019	287660.0000
21	150000020	693491.0000
22	150000021	486958.0000
23	150000022	687084.0000
24	150000023	1231177.0000
25	150000024	811510.0000
26	150000025	370776.0000
27	150000026	623134.0000
28	150000027	772414.0000

```sql
# Query the average rating of all items for each shop
select
    s.s_id as s_id,
    s.s_name as s_name,
    avg(c.star) as average_star
from
    Shop as s
left join
    Product as p on s.s_id = p.Shop_s_id
inner join
    comment as c on p.p_id = c.p_id
group by
    s_id
order by
    average_star desc;
```

	s_id	s_name	average_star
1	120000093	Optum	5.0000
2	120000006	ICBC	4.5000
3	120000219	NBC	4.1250
4	120000116	NVIDIA	4.0000
5	120000235	Telstra	4.0000
6	120000319	Subway	4.0000
7	120000349	Pall Mall	4.0000
8	120000003	Microsoft	4.0000
9	120000309	Reliance	4.0000
10	120000272	Japan Post Holdings	3.8571

```sql
# Query the number of valid coupons for the all users
select
    u.u_id as u_id,
    u.u_name as u_name,
    sum(if(c.c_state = 'unused' and (now() between c.begin_time and c.end_time), 1, 0)) as
valid_coupon_number
from
    User as u
left join
    Coupon as c
on
    u.u_id = c.u_id
group by
    u.u_id
order

 by
    valid_coupon_number desc;
```

	u_id	u_name	valid_coupon_number
1	110000011	Clara	4
2	110001615	Oswald	4
3	110000000	Mary	3
4	110001304	Clem	3
5	110007487	Bennie	3
6	110009254	Margarete	3
7	110000106	Ollie	3
8	110001710	Elton	3
9	110001743	Tyler	3
10	110002119	Luella	3
11	110003273	Evan	3