数据库系统设计报告

题目名称：航游集市——北航学子的交易&社交平台

小组成员：

- 张瀚文 22373321
- 魏鹏程 22373151
- 石伊聪 22371207

组内同学承担任务说明

学生姓名	子任务1 系统功能设计与数据库设计	子任务2 系统服务器端开发	子任务3 系统客户端开发	工作量占比
张瀚文	想法讨论 需求分析	接口文档敲定与测试	前端程序架构设计 前端架构实现 网页测试 前后端代码测试	33.3%
魏鹏程	想法讨论 需求分析	接口文档敲定与测试	前端布局美化 前端架构实现 网页测试 前后端代码测试	33.3%
石伊聪	想法讨论 需求分析 构造数据表	数据库连接 数据库存储定义与实现 后端架构设计与实现	前后端代码测试	33.3%

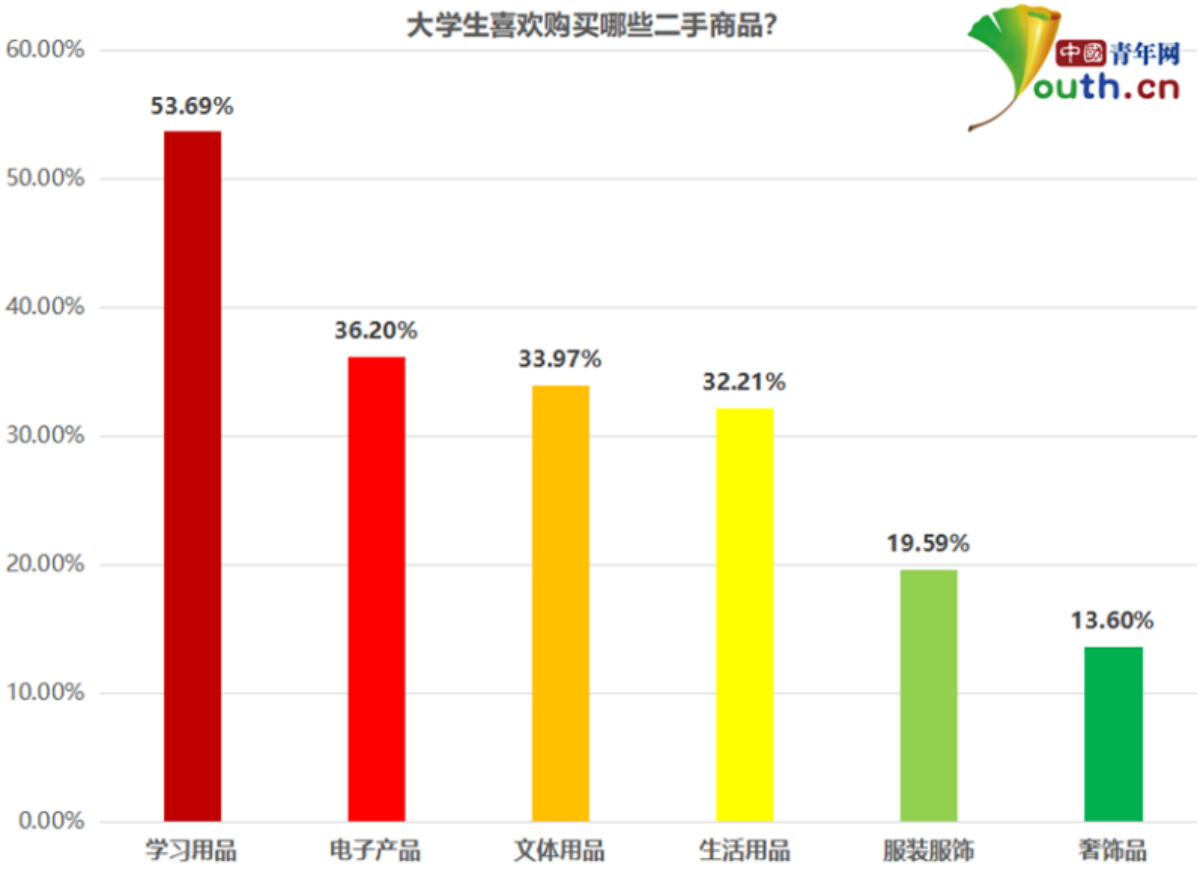
一、系统需求分析

1.1 需求描述

1.1.1 背景调研

随着线上交易的日渐普及，二手交易逐渐成为互联网用户尤其是大学生的主要交易方式之一。二手商品交易是一种秉持有节约和绿色理念的交易模式，能促进资源的合理流通和分配，减少闲置资源，使资源得到充分的利用。

大学生作为二手交易市场的主要群体，其交易市场也逐渐迸发出巨大的潜力，根据中国青年网记者在2023 年的调查显示，**72.04%** 的大学生有闲置物品，**80.84%** 愿意进行闲置物品交易，**52.44%** 的大学生有过购买二手商品的经历。二手商品所具有的流通性广、性价比高等特点深受大学生的青睐。同时大学生购买的二手商品以学习用品、电子产品等为主体，更侧重于日常学习生活需求，具体统计详情如下图:



然而目前的校园二手交易的主要方式还是借助于社交媒体和群聊的方式进行。这样的方式存在着难以按需寻找、难以沟通交流、被骗概率较高等问题，并不是一个成熟、完善、长久的交易方式。同时市面上活跃度较高的二手交易平台又难以具备校园特色，需要进行卖家发货、买家收货的具体流程；同时也难以匹配到校园附近的同学，并不能很好地契合 "校园交易" 的特色需求。过于普遍广泛的交易平台却也让商品质量保障成为大学生进行二手交易的主要担忧。

拥有广大的市场空间和前景，同时也存在急需解决的问题和隐患。基于此本组希望打造一款具有校园特色的、交易社交为一体的集市平台，让安全、便捷、高质量能够成为 "校园二手交易" 的代名词，满足同学们的日常生活需求，增强同学们的幸福感。

1.1.2 需求总结

经过上述的需求调研和小组成员的详细拟定，我们可以针对不同身份的使用者总结出如下需求：

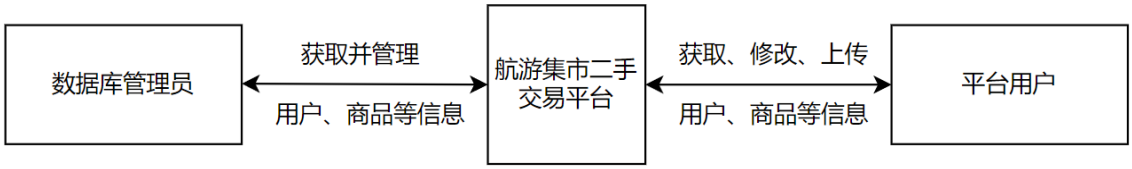
需要说明的是，每一个用户都可以是买家和卖家，这里只是从**不同身份角度**对功能加以区分。

- 用户
    1. 用户注册和学号认证
    2. 修改个性信息(头像、昵称等)
- 买家
    1. 接收商品推荐
    2. 根据商品 tag 或商品名称进行商品查找
    3. 对商品进行评论、收藏
    4. 和卖家进行沟通交流
    5. 商品购买
    6. 记录个人收货地址

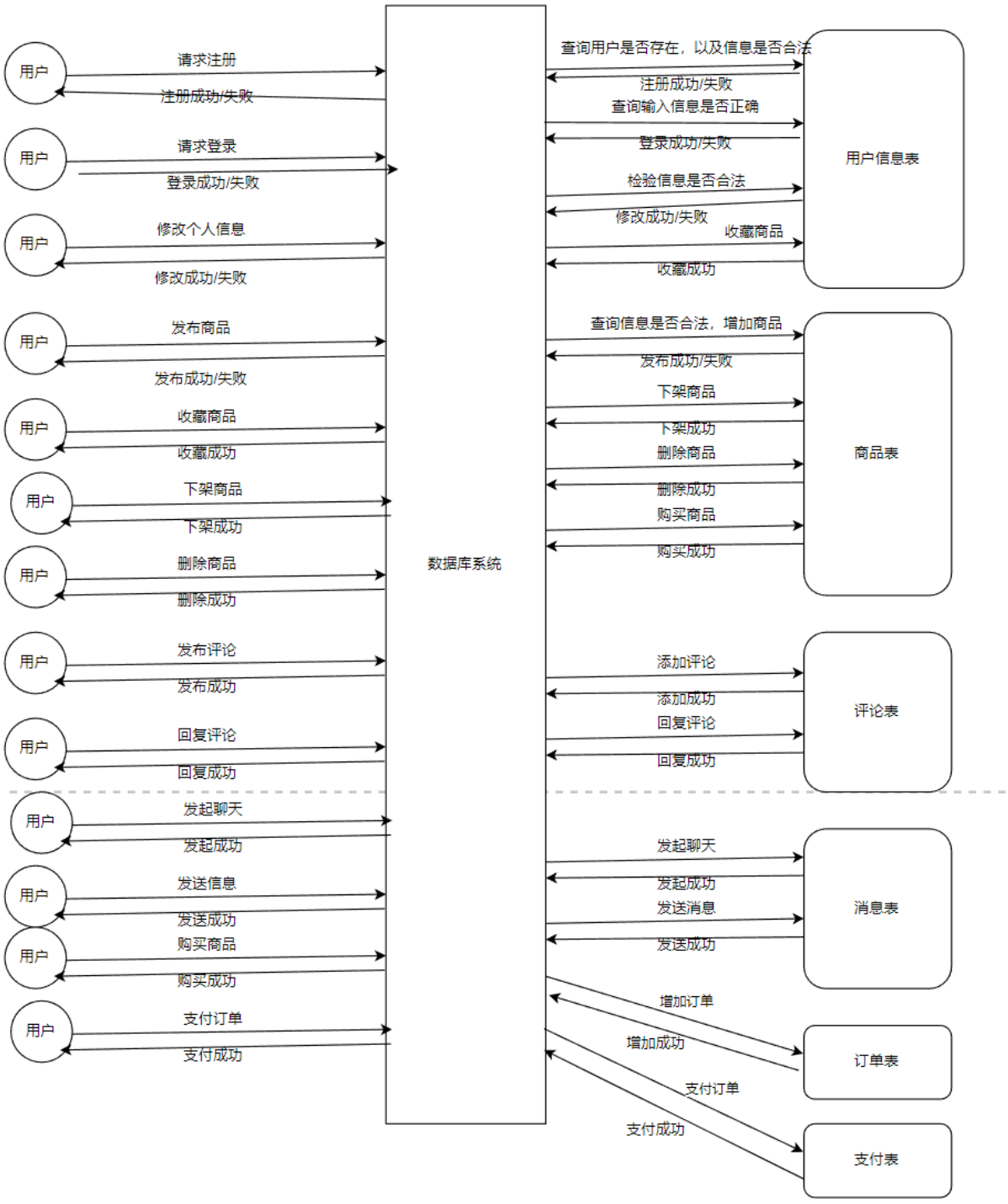
- 卖家
    1. 创建商品并完善商品信息
    2. 发布商品
    3. 商品状态查询
    4. 下架商品、删除商品
    5. 个人消息管理
- 管理员
    1. 用户注册管理及用户信息管理
    2. 商品发布管理及商品信息管理
    3. 商品交易信息统计

1.2 数据流图

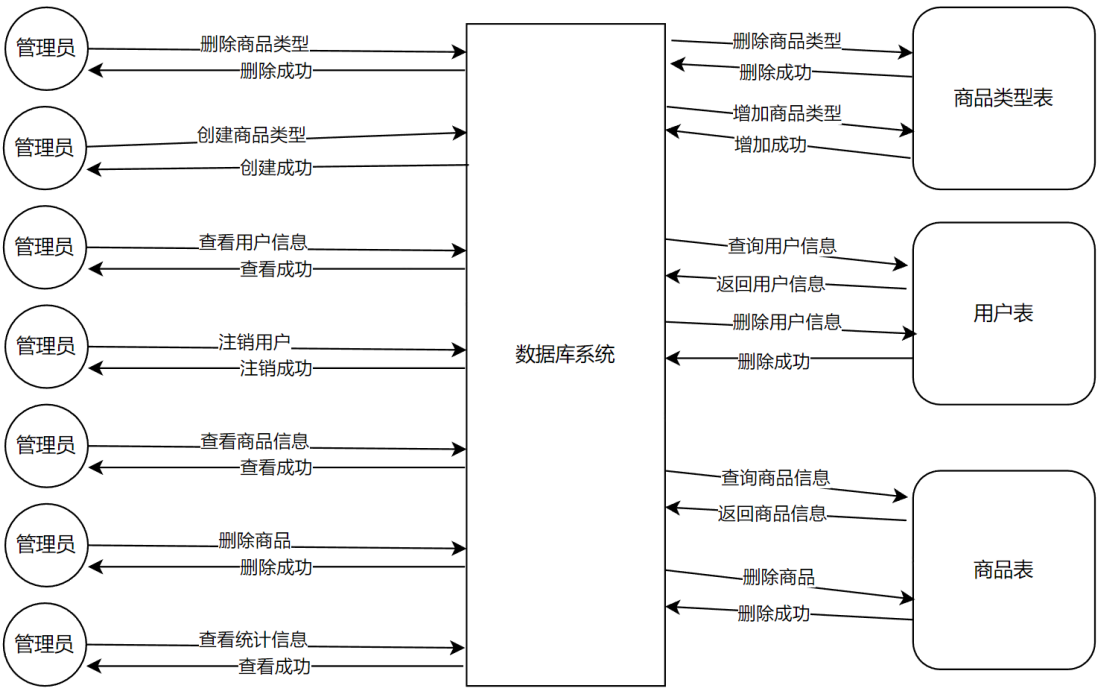
1.2.1 顶层简略数据流图



1.2.2 用户功能数据流图



1.2.3 管理员功能数据流图



1.3 数据元素表

1.3.1 用户表

字段名称	数据类型	可否为空	说明
id	VARCHAR(36)	否	主键
avatar	VARCHAR(255)	否	用户头像地址
intro	VARCHAR(255)	否	用户简介
nick_name	VARCHAR(100)	否	用户昵称
username	VARCHAR(100)	否	用户名
email	VARCHAR(100)	否	电子邮箱
student_id	VARCHAR(20)	否	学生ID
password	VARCHAR(100)	否	用户密码
status	INTEGER	是	用户状态
update_time	BIGINT	是	更新时间戳
create_time	BIGINT	是	创建时间戳
address	VARCHAR(10)	是	地址
check_nick_name	VARCHAR(100)	是	昵称审核内容
check_intro	VARCHAR(255)	是	简介审核内容

字段名称	数据类型	可否为空	说明
check_avatar	VARCHAR(255)	是	头像审核内容
check_status	INTEGER	是	审核状态

1.3.2 商品表

字段名称	数据类型	可否为空	说明
user_id	VARCHAR(36)	是	外键，user(id)
title	VARCHAR(100)	是	商品标题
id	VARCHAR(36)	否	主键
intro	TEXT	是	商品介绍
image	TEXT	是	商品图片
price	BIGINT	是	商品价格
original_price	BIGINT	是	商品原价
type_code	VARCHAR(10)	是	外键，product_type(type_code)
type_name	VARCHAR(20)	是	商品类型名称
post_type	INTEGER	是	发布类型
like_count	INTEGER	是	点赞数量
adcode	VARCHAR(10)	是	行政区划代码
province	VARCHAR(10)	是	省份
city	VARCHAR(10)	是	城市
district	VARCHAR(10)	是	区县
status	INTEGER	是	商品状态
create_time	BIGINT	是	创建时间戳
update_time	BIGINT	是	更新时间戳

1.3.3 消息表

字段名称	数据类型	可否为空	说明
id	VARCHAR(36)	否	主键
product_id	VARCHAR(36)	是	外键，product_info(id)
product_image	VARCHAR(255)	是	商品图片
from_user_id	VARCHAR(36)	否	外键，user(id)

字段名称	数据类型	可否为空	说明
from_user_avatar	VARCHAR(255)	是	发送方头像
from_user_nick	VARCHAR(100)	是	发送方昵称
to_user_id	VARCHAR(36)	否	外键，user(id)
to_user_nick	VARCHAR(100)	是	接收方昵称
to_user_avatar	VARCHAR(255)	是	接收方头像
create_time	BIGINT	是	创建时间戳
update_time	BIGINT	是	更新时间戳

1.3.4 评论表

字段名称	数据类型	可否为空	说明
id	VARCHAR(36)	否	主键
product_id	VARCHAR(36)	是	外键，product_info(id)
parent_id	VARCHAR(36)	是	外键，comment(id)
pub_user_id	VARCHAR(36)	是	外键，user(id)
pub_nickname	VARCHAR(100)	是	评论者昵称
parent_user_id	VARCHAR(36)	是	被回复者用户ID
parent_user_nickname	VARCHAR(255)	是	被回复者昵称
content	TEXT	是	评论内容
create_time	BIGINT	是	创建时间戳

1.3.5 订单表

id	VARCHAR(36)	否	主键
order_number	VARCHAR(32)	是	订单号
user_id	VARCHAR(36)	是	外键，user(id)
pay_price	BIGINT	是	支付金额
pay_type_id	BIGINT	是	外键，payment_type(id)
pay_type_name	VARCHAR(200)	是	支付类型名称
order_status	INTEGER	是	订单状态
payment_pay_id	VARCHAR(36)	是	支付流水号
payment_status	INTEGER	是	支付状态

id	VARCHAR(36)	否	主键
payment_type	VARCHAR(20)	是	支付方式
process_status	INTEGER	是	处理状态
time_create	TIMESTAMPTZ	是	创建时间
time_update	TIMESTAMPTZ	是	更新时间，需大于等于创建时间
time_finish	TIMESTAMPTZ	是	完成时间，需大于等于更新时间

1.3.6 角色表

字段名称	数据类型	可否为空	说明
id	VARCHAR(36)	否	主键
role_code	VARCHAR(20)	是	角色编码,唯一
role_name	VARCHAR(20)	是	角色名称
create_time	BIGINT	是	创建时间戳
update_time	BIGINT	是	更新时间戳

1.3.7 管理员表

字段名称	数据类型	可否为空	说明
id	VARCHAR(36)	否	主键
username	VARCHAR(100)	是	用户名
password	VARCHAR(100)	是	密码
name	VARCHAR(20)	是	姓名
role_id	VARCHAR(36)	是	外键，system_role(id)
role_code	VARCHAR(20)	是	角色编码
role_name	VARCHAR(20)	是	角色名称
create_time	BIGINT	是	创建时间戳
update_time	BIGINT	是	更新时间戳

1.3.8 商品类型表

字段名称	数据类型	可否为空	说明
id	VARCHAR(36)	否	主键
type_code	VARCHAR(10)	是	类型编码，唯一约束
type_name	VARCHAR(20)	是	类型名称

字段名称	数据类型	可否为空	说明
create_time	BIGINT	是	创建时间戳
update_time	BIGINT	是	更新时间戳

1.3.9 消息列表表

字段名称	数据类型	可否为空	说明
id	VARCHAR(36)	否	主键
chat_list_id	VARCHAR(36)	否	外键，chat_list(id)
from_user_id	VARCHAR(36)	否	外键，user(id)
to_user_id	VARCHAR(36)	否	外键，user(id)
from_user_nick	VARCHAR(100)	是	发送方昵称
to_user_nick	VARCHAR(100)	是	接收方昵称
content	TEXT	是	消息内容
send_time	BIGINT	是	发送时间戳
is_read	INTEGER	是	是否已读

1.3.10 支付表

字段名称	数据类型	可否为空	说明
id	VARCHAR(36)	否	主键
user_id	VARCHAR(36)	是	外键，user(id)
order_id	VARCHAR(36)	是	外键，payment_order(id)
payment_price	BIGINT	是	支付金额
payment_type	VARCHAR(30)	是	支付方式
payment_result_data	TEXT	是	支付结果数据
payment_time_start	VARCHAR(50)	是	支付开始时间
payment_time_expire	VARCHAR(50)	是	支付过期时间
payment_status	INTEGER	是	支付状态，默认值0
process_status	INTEGER	是	处理状态，默认值0
time_create	TIMESTAMPTZ	是	创建时间
time_update	TIMESTAMPTZ	是	更新时间，需大于等于创建时间
time_finish	TIMESTAMPTZ	是	完成时间，需大于等于更新时间

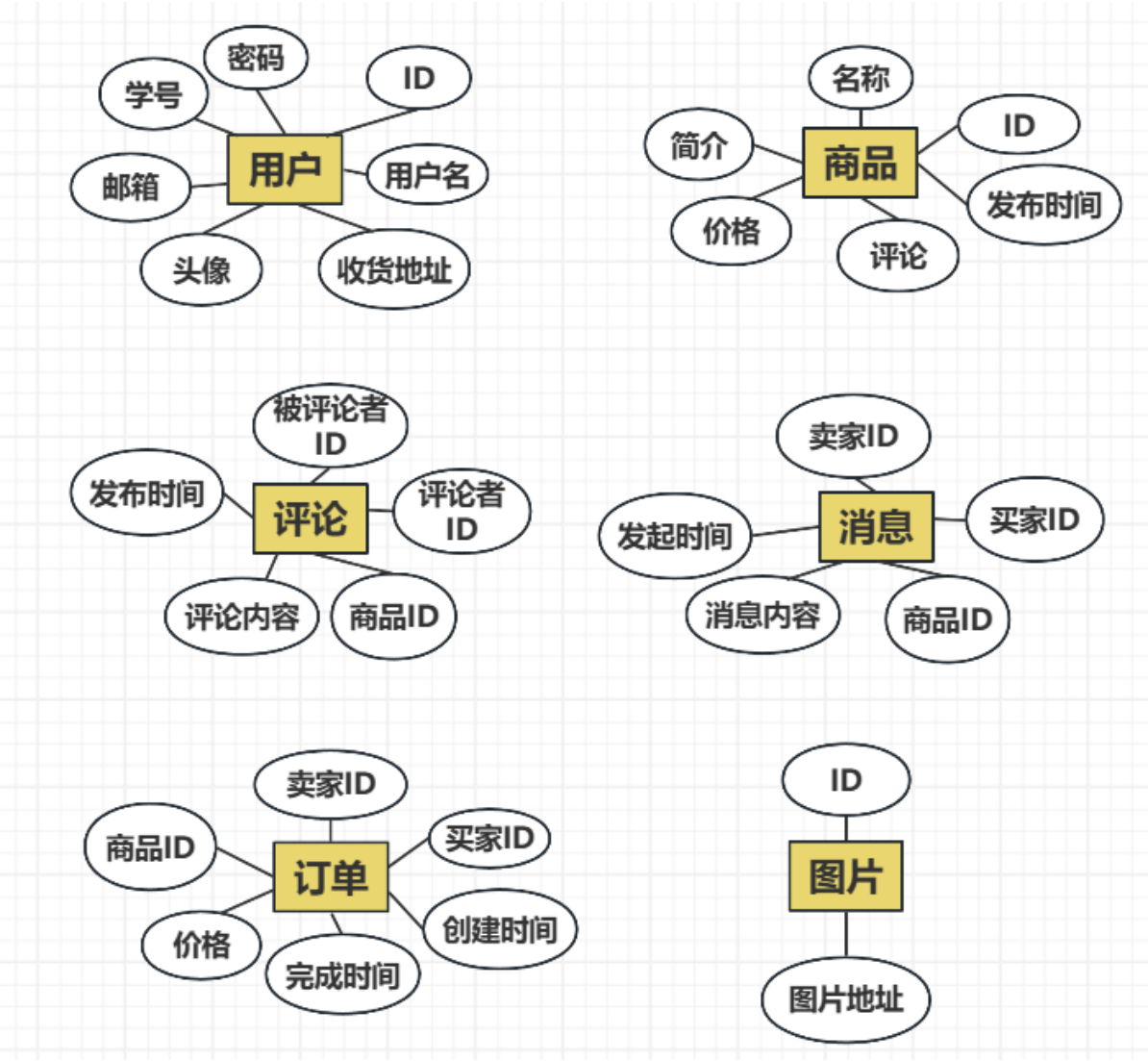
1.3.11 发布信息表

字段名称	数据类型	可否为空	说明
id	VARCHAR(36)	否	主键
user_id	VARCHAR(36)	是	外键，user(id)
product_id	VARCHAR(36)	是	外键，product_info(id)
create_time	BIGINT	是	创建时间戳
update_time	BIGINT	是	更新时间戳

二、数据库系统的概念模式

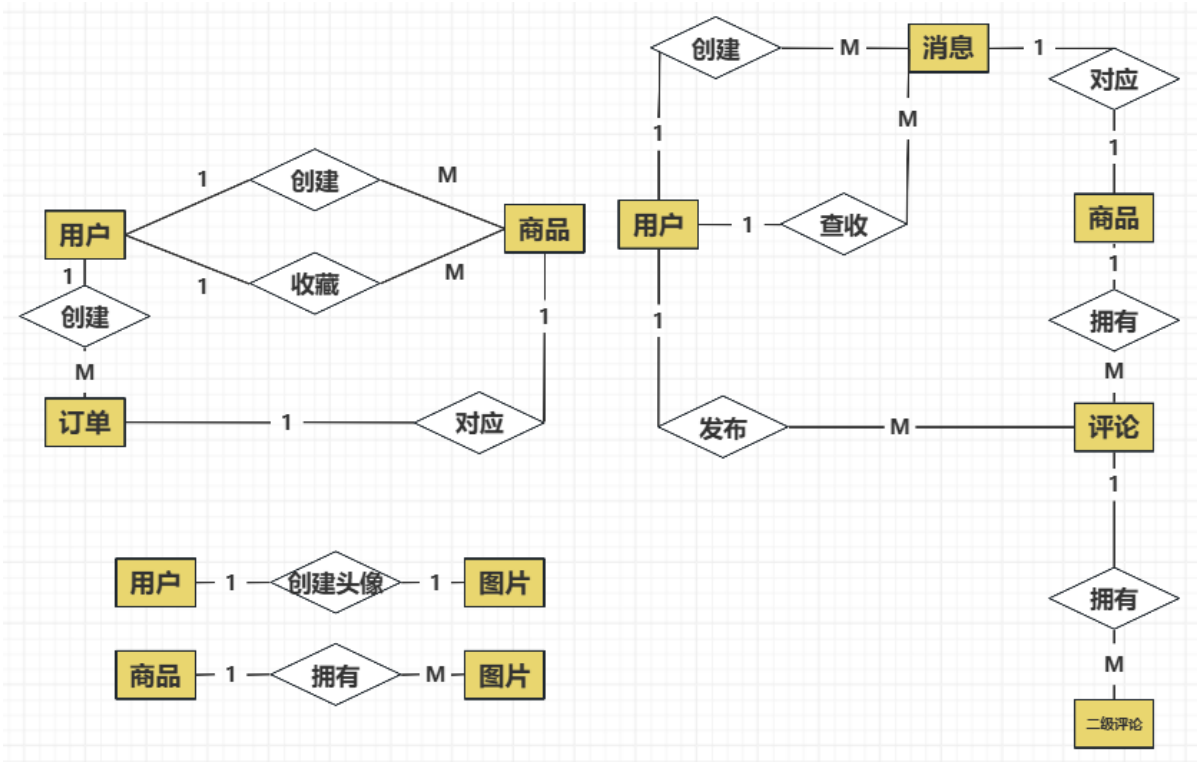
2.1 系统初步ER图

2.1.1 实体ER图



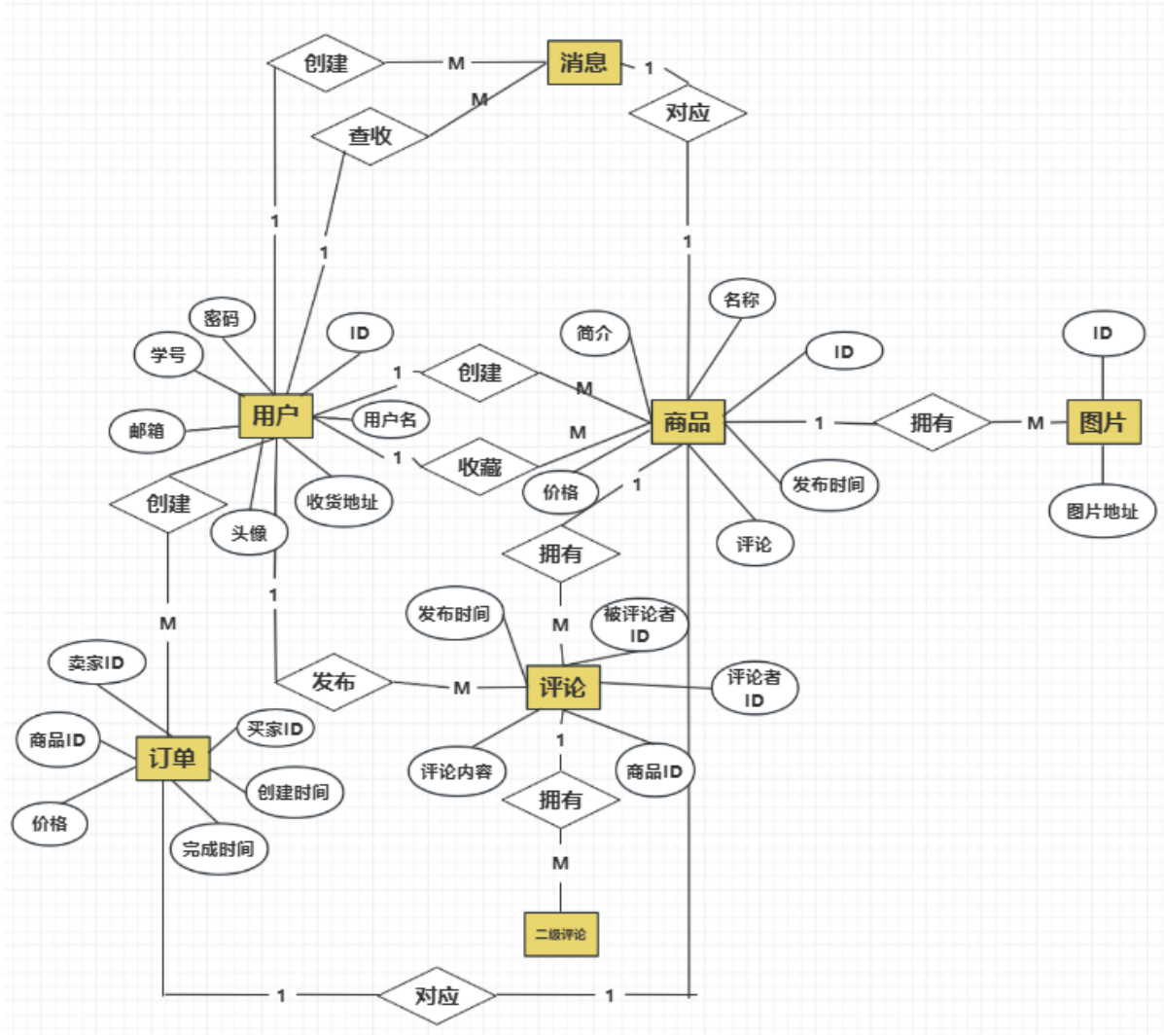
2.1.2 关系ER图

下图分别为商品购买模块、用户交流模块和图片模块对应的 ER 图：



2.2 系统基本ER图

分析上述实体 ER 图，结合关系 ER 图，合并并消除冗余之后可以得到系统基本 ER 图如下：



三、数据库系统的逻辑模式

3.1 数据库关系模式

- 用户

  user(**id**, avatar, intro, nick_name, username, email, student_id, password, status, update_time, create_time, address, check_nick_name, check_intro, check_avatar, check_status)

- 商品信息

  product_info(**id**, user_id, title, intro, image, price, original_price, type_code, type_name, post_type, like_count, adcode, province, city, district, status, create_time, update_time)

- 商品种类

  product_type(**id**, type_code, type_name, create_time, update_time)

- 消息列表

  chat_list(**id**, product_id, product_image, from_user_id, from_user_avatar, from_user_nick, to_user_id, to_user_nick, to_user_avatar, create_time, update_time)

- 聊天消息

  chat_message(**id**, chat_list_id, from_user_id, to_user_id, from_user_nick, to_user_nick, content, send_time, is_read)

- 评论

comment(**id**, product_id, parent_id, pub_user_id, pub_nickname, parent_user_id, parent_user_nickname, content, create_time)

- 支付订单

  payment_order(**id**, order_number, user_id, pay_price, pay_type_id, pay_type_name, order_status, payment_pay_id, payment_status, payment_type, process_status, time_create, time_update, time_finish)

- 支付

  payment_pay(**id**, user_id, order_id, payment_price, payment_type, payment_result_data, payment_time_start, payment_time_expire, payment_status, process_status, time_create, time_update, time_finish)

- 支付种类

  payment_type(**id**, type_name, wx_pay, ali_pay)

- 商品收藏

  product_collect(**id**, user_id, product_id, create_time, update_time)

- 商品订单

  product_order(**id**, order_number, product_user_id, product_id, user_id, product_title, product_img, product_price, product_type, product_type_name, product_sell_price, product_num, product_post, product_post_status, product_money, product_info, buy_money_all, buy_money, buy_info, post_mode, post_self_code, post_username, post_phone, post_address, ship_username, ship_phone, ship_address, ship_num, ship_company, ship_time, pay_status, pay_order_id, deal_status, eva_score, eva_content, create_time, update_time)

- 系统角色

  system_role(**id**, role_code, role_name, create_time, update_time)

- 系统用户

  system_users(**id**, username, password, name, role_id, role_code, role_name, create_time, update_time)

- 用户地址

  user_address(**id**, user_id, name, phone, address, create_time, update_time)

3.2 关系模式范式等级的判定与规范化

- 用户表：符合3NF

  user(id, avatar, intro, nick_name, username, email, student_id, password, status, update_time, create_time, address, check_nick_name, check_intro, check_avatar, check_status)

  - 所有属性都是原子的
  - 所有属性都完全依赖于主键id
  - 没有非主键属性传递依赖于主键

- 商品种类表：符合3NF

  product_type(id, type_code, type_name, create_time, update_time)

  - 所有属性都是原子的
  - 所有属性完全依赖于主键id

    - 没有非主键属性传递依赖于主键
- 商品信息表：符合3NF

  product_info(id, user_id, title, intro, image, price, original_price, type_code, type_name, post_type, like_count, adcode, province, city, district, status, create_time, update_time)

    - 所有属性都是原子的
    - 所有属性完全依赖于主键id
    - 没有非主键属性传递依赖于主键
- 聊天信息表：符合3NF

  chat_message(id, chat_list_id, from_user_id, to_user_id, from_user_nick, to_user_nick, content, send_time, is_read)

    - 所有属性都是原子的
    - 所有属性完全依赖于主键id
    - 没有非主键属性传递依赖于主键
- 聊天列表表：符合3NF

  chat_list(id, product_id, product_image, from_user_id, from_user_avatar, from_user_nick, to_user_id, to_user_nick, to_user_avatar, create_time, update_time)

    - 所有属性都是原子的
    - 所有属性完全依赖于主键id
    - 没有非主键属性传递依赖于主键
- 评论表：符合3NF

  comment(id, product_id, parent_id, pub_user_id, pub_nickname, parent_user_id, parent_user_nickname, content, create_time)

    - 所有属性都是原子的
    - 所有属性完全依赖于主键id
    - 没有非主键属性传递依赖于主键
- 支付订单表：符合3NF

  payment_order(id, order_number, user_id, pay_price, pay_type_id, pay_type_name, order_status, payment_pay_id, payment_status, payment_type, process_status, time_create, time_update, time_finish)

    - 所有属性都是原子的
    - 所有属性完全依赖于主键id
    - 没有非主键属性传递依赖于主键
- 支付表：符合3NF

  payment_pay(id, user_id, order_id, payment_price, payment_type, payment_result_data, payment_time_start, payment_time_expire, payment_status, process_status, time_create, time_update, time_finish)

    - 所有属性都是原子的
    - 所有属性完全依赖于主键id
    - 没有非主键属性传递依赖于主键
- 支付种类表：符合3NF

payment_type(id, type_name, wx_pay, ali_pay)

- 所有属性都是原子的
- 所有属性完全依赖于主键id
- 没有非主键属性传递依赖于主键

- 商品收藏表：符合3NF

  product_collect(id, user_id, product_id, create_time, update_time)

  - 所有属性都是原子的
  - 所有属性完全依赖于主键id
  - 没有非主键属性传递依赖于主键

- 商品订单表：符合3NF

  product_order(id, order_number, product_user_id, product_id, user_id, product_title, product_img, product_price, product_type, product_type_name, product_sell_price, product_num, product_post, product_post_status, product_money, product_info, buy_money_all, buy_money, buy_info, post_mode, post_self_code, post_username, post_phone, post_address, ship_username, ship_phone, ship_address, ship_num, ship_company, ship_time, pay_status, pay_order_id, deal_status, eva_score, eva_content, create_time, update_time)

  - 所有属性都是原子的
  - 所有属性完全依赖于主键id
  - 没有非主键属性传递依赖于主键

- 系统角色表：符合3NF

  system_role(id, role_code, role_name, create_time, update_time)

  - 所有属性都是原子的
  - 所有属性完全依赖于主键id
  - 没有非主键属性传递依赖于主键

- 系统用户表：符合3NF

  system_users(id, username, password, name, role_id, role_code, role_name, create_time, update_time)

  - 所有属性都是原子的
  - 所有属性完全依赖于主键id
  - 没有非主键属性传递依赖于主键

- 用户地址表：符合3NF

  user_address(id, user_id, name, phone, address, create_time, update_time)

  - 所有属性都是原子的
  - 所有属性完全依赖于主键id
  - 没有非主键属性传递依赖于主键

3.3 数据库设计优化

3.3.1 建立索引

数据库默认建立有主键和外键的索引。除此之外，对于高频查询的候选码，我们**建立了额外的索引**来提高查询速度。

```
CREATE INDEX idx_user_username ON "user" (username);
Run | New Tab
CREATE INDEX idx_user_email ON "user" (email);
Run | New Tab
CREATE INDEX idx_product_user_id ON product_info (user_id);
Run | New Tab
CREATE INDEX idx_chat_list_time ON chat_message (chat_list_id, send_time);
Run | New Tab
CREATE INDEX idx_payment_order_number ON payment_order (order_number);
Run | New Tab
CREATE INDEX idx_product_order_number ON product_order (order_number);
Run | New Tab
CREATE INDEX idx_collect_user ON product_collect (user_id, create_time DESC);
Run | New Tab
CREATE INDEX idx_collect_product ON product_collect (product_id);
Run | New Tab
CREATE INDEX idx_buyer_orders ON product_order (user_id, deal_status);
Run | New Tab
CREATE INDEX idx_seller_orders ON product_order (product_user_id, deal_status);
Run | New Tab
CREATE INDEX idx_order_pay_status ON product_order (pay_status);
```

3.3.2 建立数据约束

我们根据数据表的语义设置了合理的外键、UNIQUE、NOT NULL等约束，**保证数据的一致性和完整性**，在应用程序层面提供额外的数据验证保护。

```
username VARCHAR(100) NOT NULL,
email VARCHAR(100) NOT NULL,
student_id VARCHAR(20) NOT NULL,
password VARCHAR(100) NOT NULL,
```

```
time_create TIMESTAMPTZ,
time_update TIMESTAMPTZ CHECK (time_update ⩾ time_create),
time_finish TIMESTAMPTZ CHECK (time_finish ⩾ time_update),
```

3.3.3 利用触发器

我们设置了一些**触发器**来**维护数据完整性和业务逻辑**。在方便**表间数据同步更新**的同时减轻开发者的心智负担。

```
CREATE OR REPLACE FUNCTION update_user_timestamp()
RETURNS TRIGGER AS $$
BEGIN
    NEW.update_time = EXTRACT(EPOCH FROM NOW()) * 1000;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

▷ Run | New Tab
CREATE TRIGGER trigger_user_timestamp BEFORE
UPDATE ON "user" FOR EACH ROW
EXECUTE FUNCTION update_user_timestamp ();
```