《数据库系统原理》大作业

系统设计报告

题目名称：具有十三公寓特色的二手交易市场

学号及姓名： 21373061方沐阳

21373119 杨茜木

21182632苏伊

2023年 12 月 2 日

组内同学承担任务说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 学生姓名 | 工作内容 | | | 工作量占比（组内同学总和为1） |
| **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** |
| 方沐阳 | 想法讨论  需求分析 | 后端架构设计与实现  数据库连接  接口定义与具体实现  撰写实现报告 | 前后端代码测试 | 33.3% |
| 杨茜木 | 想法讨论  需求分析 | 测试 | 前端程序架构设计与实现  网页测试  前后端代码测试  撰写实现报告 | 33.3% |
| 苏伊 | 想法讨论  需求分析  构造E-R图、数据表  撰写设计报告 | 测试 | 前端布局美化  前后端代码测试 | 33.3% |

## 一．需求分析

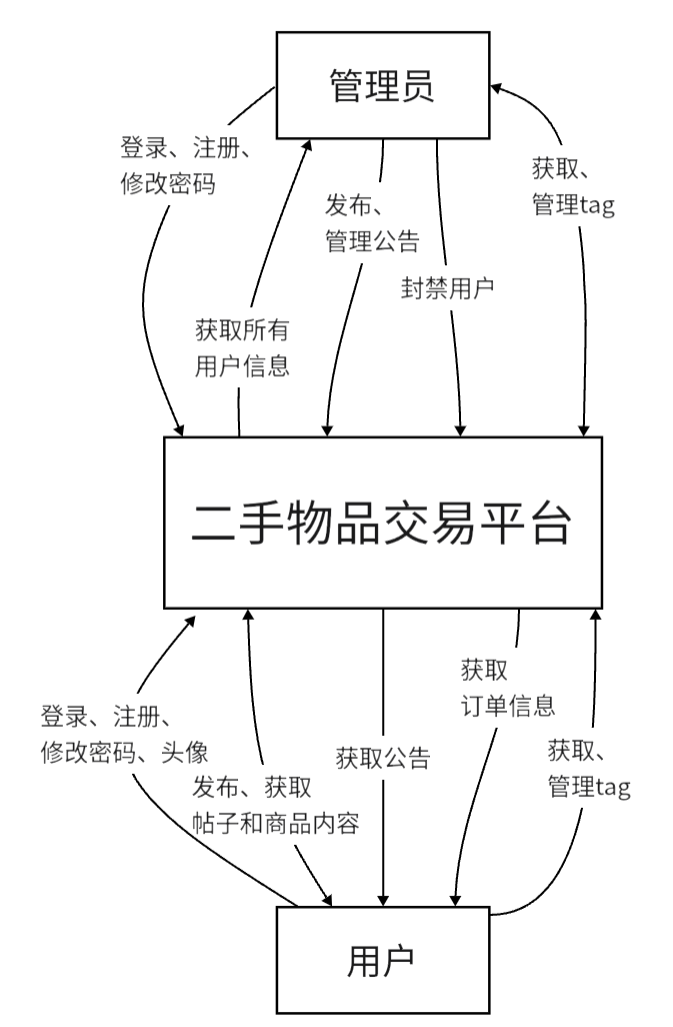
## 1．需求描述

在我们三位组员居住的十三公寓里存在着许多微信二手群，我们发现微信交易并不方便，所以我们提炼出了十三公寓住户主要的二手交易需求，决定写一个二手交易市场的 web 作为此次大作业。

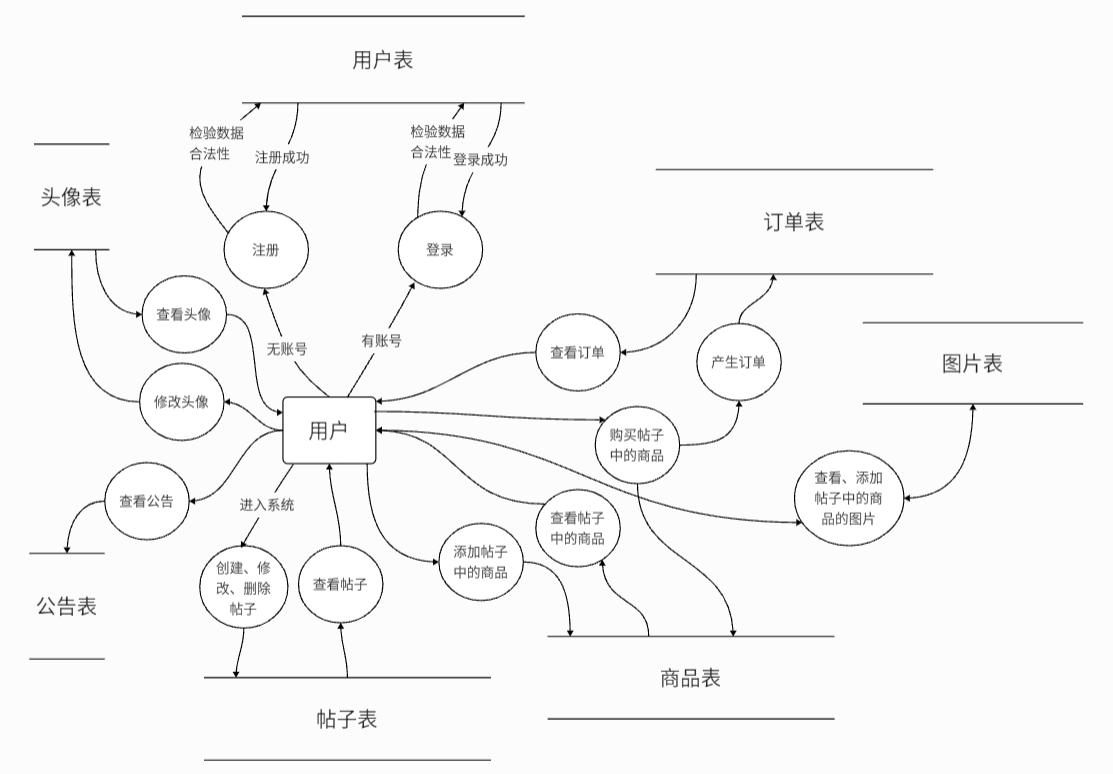
* + 1. **用户、帖子分类**
* 用户
  + 管理员
  + 普通用户
* 帖子
  + 售物贴
  + 信息贴
    1. **普通用户相关功能**
* 注册
* 登录
* 修改密码
* 设置个人信息（头像，个人联系方式等）
* 发帖（发布信息/商品）
* 删除自己发的帖子
* 购买商品
  + 1. **管理员相关功能**
* 注册
* 登录
* 修改密码
* 发公告
* 管理tag
* 封号
  + 1. **售物贴相关功能**
* 添加相应tag
* 删除tag
* 发布者自行修改、删除帖子内容
* 用户购买商品形成订单，订单上内容有买家卖家的个人信息
  + 1. **信息贴相关功能**
* 信息包括：代取外卖、快递；外卖拼单等等
* 发布的帖子上会有个人信息
* 有需求的用户通过联系方式联系发布者
  + 1. **公告相关功能**
* 管理员发布、修改、删除公告
* 用户在首页看到公告

## 2．数据流图

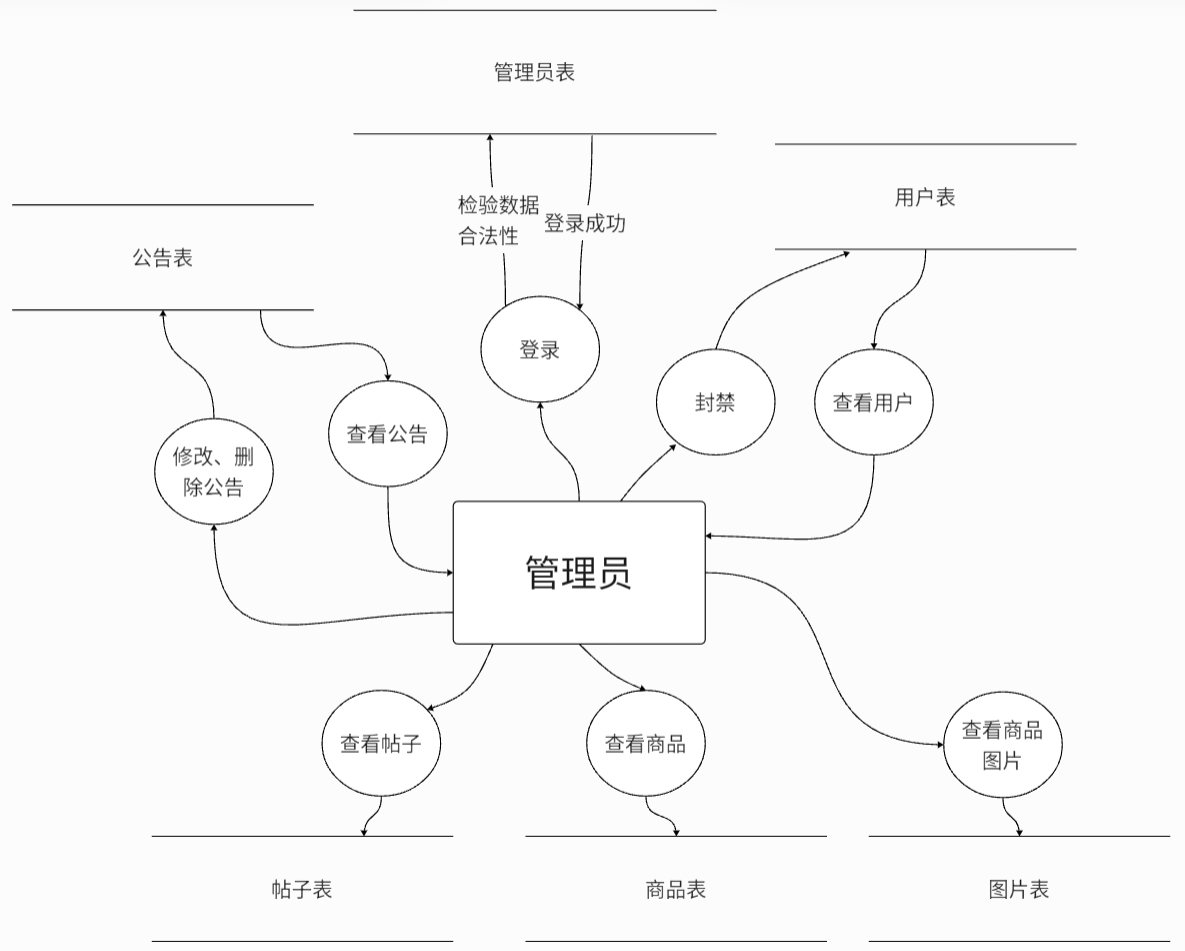
**2.1 顶层数据流图**



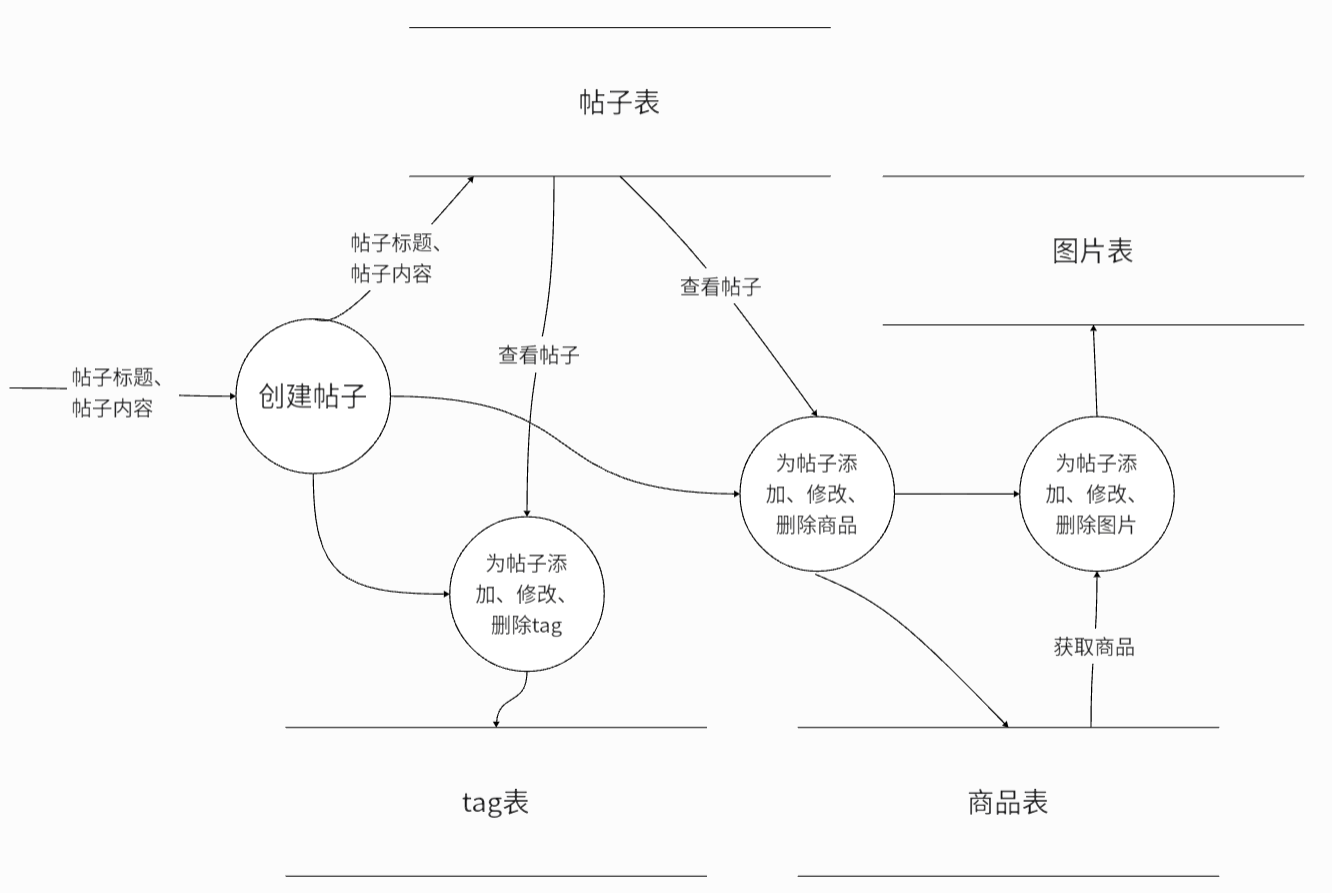
**2.2 用户功能数据流图**

****

**2.3 管理员功能数据流图**



**2.4 帖子相关数据流图**

****

## 3．数据元素表

**1.3.1. 普通用户表**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **数据类型** | **可否为空** | **说明** |
| user\_id | bigint | 否 | 用户id |
| username | varchar(10) | 否 | 用户名（主键） |
| password | varchar(20) | 否 | 密码 |
| phone | bigint | 是 | 联系电话 |
| WeChat | varchar(30) | 否 | 微信号 |
| isBaned | tinyint(1) | 否 | 是否被封禁 |

**1.3.2 管理员表**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **数据类型** | **可否为空** | **说明** |
| administrator\_id | bigint | 否 | 管理员id（主键） |
| username | varchar(10) | 否 | 用户名 |
| password | varchar(20) | 否 | 密码 |

**1.3.3帖子表**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **数据类型** | **可否为空** | **说明** |
| id | bigint | 否 | 帖子id（主键） |
| title | varchar(20) | 否 | 帖子标题 |
| content | text | 否 | 信息说明 |
| time | datatime(6) | 否 | 发帖时间 |
| user\_id | varchar(10) | 否 | 发帖用户（外键） |

**1.3.4. 订单表**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **数据类型** | **可否为空** | **说明** |
| id | bigint | 否 | 订单id（主键） |
| time | datetime(6) | 否 | 产生订单时间 |
| saler\_id | bigint | 否 | 售卖者用户id（外键） |
| buyer\_id | bigint | 否 | 购买者用户id（外键） |

**1.3.5 公告表**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **数据类型** | **可否为空** | **说明** |
| id | bigint | 否 | 公告id（主键） |
| title | varchar(20) | 否 | 公告标题 |
| content | longtext | 否 | 公告内容 |
| time | datetime(6) | 否 | 公告发布时间 |
| admin\_id | varchar(10) | 否 | 管理员id（外键） |

**1.3.6 商品表**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **数据类型** | **可否为空** | **说明** |
| id | bigint | 否 | 商品id（主键） |
| name | varchar(20) | 否 | 商品名称 |
| description | longtext | 否 | 商品描述 |
| price | int | 否 | 售卖价钱 |
| order\_id | bigint | 是 | 所属订单id（外键） |
| post\_id | bigint | 否 | 所属帖子id（外键） |

**1.3.7 tag表**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **数据类型** | **可否为空** | **说明** |
| id | bigint | 否 | tag\_id（主键） |
| name | varchar(10) | 否 | tag名字 |
| num | int | 否 | tag被引用次数 |

**1.3.8 图片表**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **数据类型** | **可否为空** | **说明** |
| id | bigint | 否 | 图片id（主键） |
| commodity\_id | int | 是 | 被用作商品图片的商品id(外键) |
| file | varchar(100) | 否 | 图片名称 |

**1.3.9 帖子-标签表**

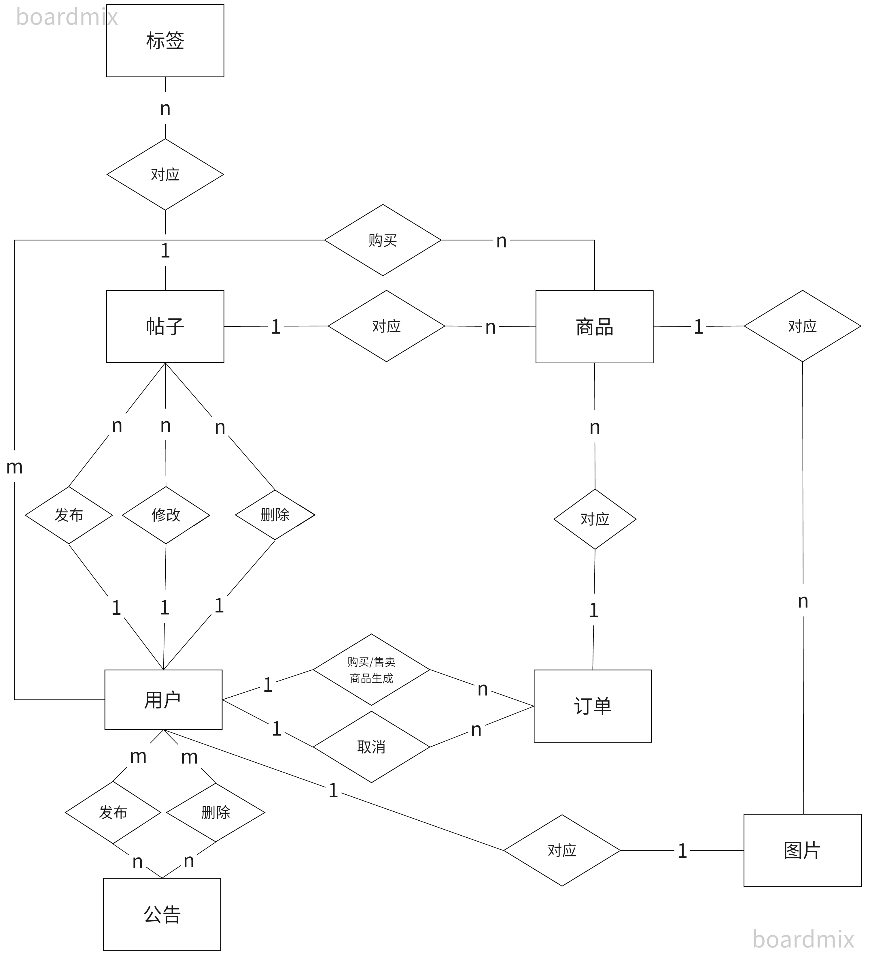
|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **数据类型** | **可否为空** | **说明** |
| id | bigint | 否 | 主键，自增 |
| tag\_id | bigint | 否 | 标签id（外键） |
| post\_id | bigint | 否 | 帖子id（外键） |

**1.3.10 头像表**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **数据类型** | **可否为空** | **说明** |
| id | bigint | 否 | 头像id（主键） |
| file | varchar(100) | 否 | 图片名称 |
| user\_id | varchar(10) | 是 | 普通用户id |
| admin\_id | varchar(10) | 是 | 管理员id |

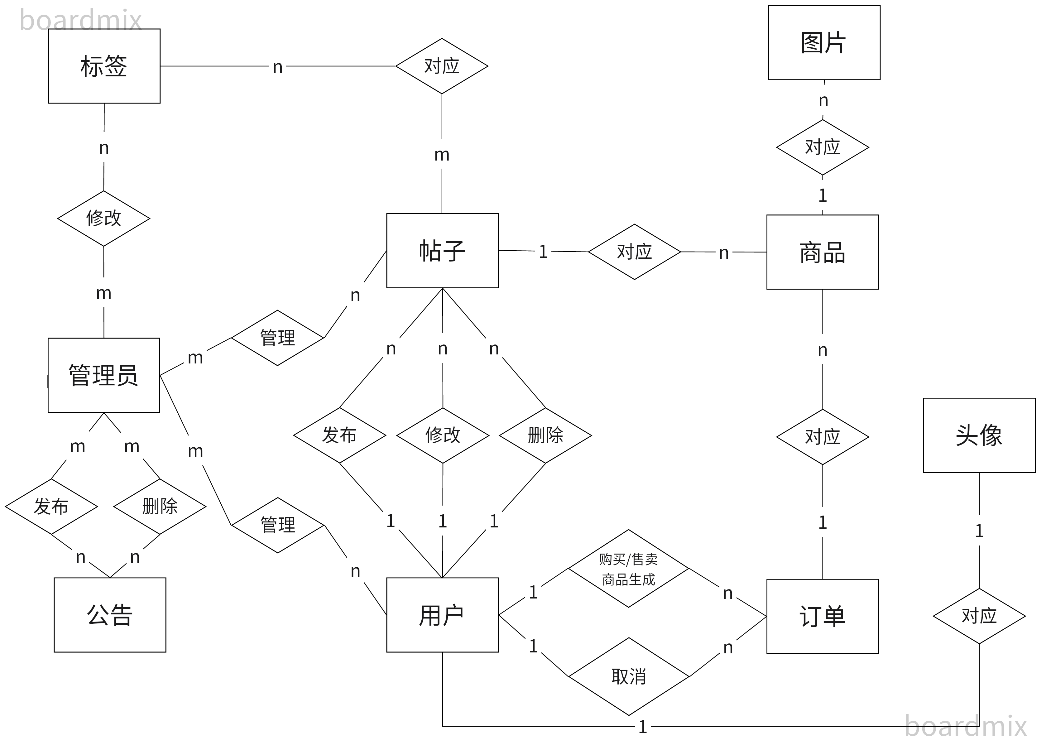
## 二．数据库概念模式设计

## 1．系统初步 E-R图

****

## 2．系统基本E-R图

将用户细分为管理员和普通用户，图片具体分为用作商品的图片和用户的头像。



## 三、数据库逻辑模式设计

## 1．数据库关系模式

**注：由E-R图得到的关系模式**

**3.1.1. 实体**

1. 用户user(user\_id, username, password, phone, WeChat, isBaned);

2. 管理员 administrator(administrator\_id, username, password);

3. 帖子post(id, title, content, time, user\_id);

4. 订单order(id, time, saler\_id, buyer\_id);

5. 公告notice(id, title, content, time, admin\_id);

6. 商品commodity(id, name, description, price, order\_id, post\_id);

7. 标签tag(id, name, num);

8. 图片photo(id, commodity\_id, file);

9. 头像 avatar(id, file, user\_id, admin\_id);

**3.1.2. 联系**

1.帖子-标签 post\_tag (id, tag\_id, post\_id);

（其他联系没有单独设表，均在实体基本表中通过外键联系）

## 2．关系模式范式等级的判定与规范化

**注：要规范到3NF**

**3.2.1.1 实体**

所有表的主键均id，其他属性均依赖于主键；不存在部分函数依赖、传递函数依赖，所有依赖项的左端均为超码，所以属于BCNF。

1. 用户user(user\_id, username, password, phone, WeChat, isBaned);

user\_id作为主键，username和Wechat为候选键，其他属性均依赖于主键，所以是BCNF，所以user∈3NF。

2. 管理员 administrator(administrator\_id, username, password);

F={ administrator\_id→username, administrator\_id→password}

administrator\_id作为主键，其他属性均依赖于主键，所以是BCNF，所以administrator∈3NF。

1. 帖子post(id, title, content, time, user\_id);

id作为主键，无其他候选键，其余属性均依赖于id，所以为BCNF，一定是3NF。

1. 订单order(id, time, saler\_id, buyer\_id);

id作为主键，一个订单只能拥有一个卖家和一个买家，所以其余属性均依赖于主键，所以为BCNF，一定为3NF。

1. 公告notice(id, title, content, time, admin\_id);

id作为主键，一个公告只能由一个管理员发送，其余属性均依赖于主键，所以为BCNF，一定是3NF。

1. 商品commodity(id, name, description, price, order\_id, post\_id);

id作为主键，一个商品只属于一个订单和一个帖子，所以其余属性均依赖于主键，该表∈BCNF，一定为3NF。

1. 标签tag(id, name, num);

id作为主键，其余属性均依赖于id，该关系∈BCNF，一定为3NF。

1. 图片photo(id, commodity\_id, file);

id作为主键，其余属性均依赖于id，该关系∈BCNF，一定为3NF。

9. 头像 avatar(id, file, user\_id, admin\_id);

id作为主键，其余属性均依赖于id，该关系∈BCNF，一定为3NF。

**3.2.1.2 联系**

1. 帖子-标签 post\_tag (tag\_id, post\_id);

该关系为二元关系，所以post\_tag∈BCNF，post\_tag∈3NF。

## 3．数据库设计优化

**3.3.1使用外键**

使用外键可以提高数据完整性、减少冗余以及简化数据管理。在本项目中，绝大多数表都是拥有外键的，一些简单的联系通过外键实现，可以减少多余表的开发。通过在关联表之间建立关系，可以减少数据冗余。相同的数据不需要在多个表中存储，而是可以通过外键引用的方式访问。外键有助于确保数据的一致性。当引用表的数据发生变化时，外键可以协助在相关表之间维护一致性。

**3.3.2 合并相同主码的表**

合并相同主码的表可以减少数据冗余，因为相同的主码不再需要在多个表中存储。合并表也可以简化查询，特别是在需要检索与主码相关的信息时。不再需要进行多个表之间的联接操作，查询语句可能更加简洁。减少表的数量通常可以提高查询性能，因为数据库系统在执行查询时需要处理的表的数量更少。合并表可以促进更好的数据一致性，因为相同的主码在一个表中维护，避免了在多个表中可能出现的一致性问题。

**3.3.3 严格约束加⼊数据库的数据**

严格约束加入数据库的数据，即通过各种约束条件（如主键、外键、唯一性约束、检查约束等）来保证数据的完整性和准确性。

严格约束确保数据库中的数据满足预定义的规则和条件，从而维护了数据的完整性。这包括主键保证唯一性、外键确保引用的完整性、唯一性约束避免重复数据等。通过添加检查约束，可以在插入或更新数据时防止无效或不合法的数据进入数据库。这有助于确保数据库中只包含合法的、符合业务规则的数据。