《数据库系统原理》大作业 - 系统设计报告

题目名称：校园二手书交易与分享平台

组内同学承担任务说明

学生姓名	工作内容	工作量占比
同学1	子任务1：系统功能设计与用户模块、订单模块数据库操作实现	0.5
同学2	子任务2：系统数据库设计与书籍发布模块数据库操作实现	0.5

一、需求分析

1. 需求描述

1.1 项目背景

随着高校学生人数的增加，每年都会产生大量的闲置书籍。学生在课程结束后，教材往往被搁置，造成资源浪费。同时，新入学的学生又需要购买相同的教材，增加了经济负担。因此，建立一个校园二手书交易与分享平台，可以有效促进书籍资源的循环利用，降低学生购书成本。

1.2 系统目标

本系统旨在为高校学生提供一个便捷、安全的二手书籍交易平台，主要实现以下目标：

1. **用户管理**：支持学生注册、登录，维护个人信息和信誉积分
2. **书籍发布**：用户可以发布想要出售或赠送的二手书籍信息
3. **书籍搜索**：提供关键词搜索、分类筛选、排序等功能
4. **交易管理**：支持下单、确认交易、取消订单等功能
5. **信誉机制**：通过信誉积分系统保障交易安全

1.3 用户需求

用户角色	需求描述
卖家	发布二手书籍信息（包括书名、作者、价格、新旧程度等）；管理自己发布的书籍；确认交易完成
买家	浏览和搜索书籍；查看书籍详情和卖家信息；下单购买；确认收货
系统管理员	管理书籍分类；维护系统数据

2. 系统功能设计

2.1 功能模块划分

```
校园二手书交易平台
├── 用户模块（User Module）
│   ├── 用户注册
│   ├── 用户登录
│   ├── 获取用户信息
│   └── 更新用户信息
├── 书籍与发布模块（Book & Listing Module）
│   ├── 搜索/浏览书籍
│   ├── 获取书籍详情
│   ├── 发布新书籍
│   └── 下架书籍
├── 订单模块（Order Module）
│   ├── 创建订单
│   ├── 获取订单列表
│   ├── 确认交易完成
│   └── 取消订单
└── 分类模块（Category Module）
    └── 获取所有分类
```

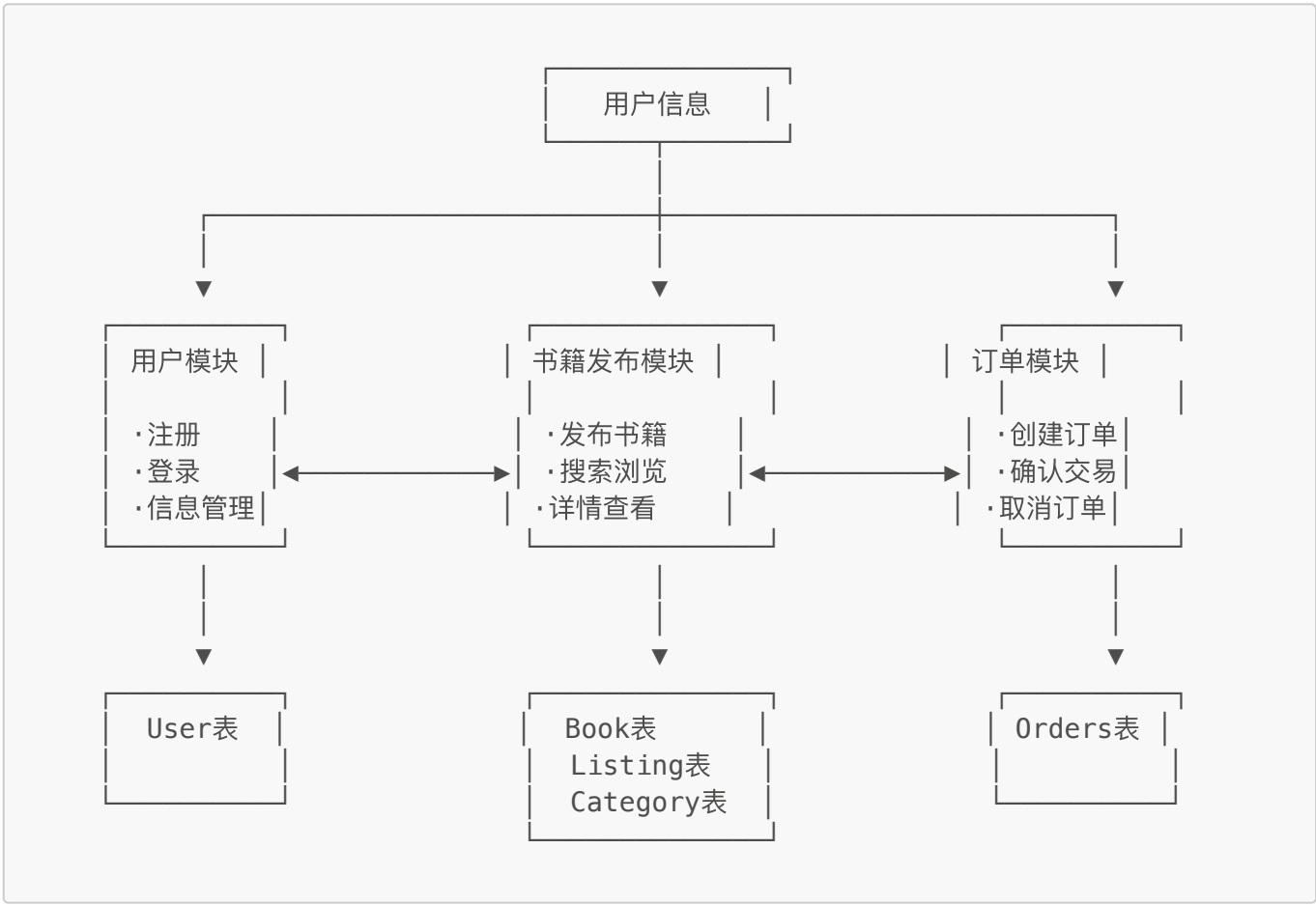
2.2 功能详细说明

模块	功能	说明
用户模块	用户注册	使用学号注册账号，设置昵称、密码和联系方式
用户模块	用户登录	使用学号和密码登录，获取JWT Token
用户模块	获取用户信息	查看当前登录用户的详细信息
用户模块	更新用户信息	修改昵称、联系方式等个人信息
书籍模块	搜索书籍	支持关键词搜索、分类筛选、价格排序、分页
书籍模块	获取详情	查看书籍完整信息、卖家信息
书籍模块	发布书籍	发布出售或赠送的书籍信息
书籍模块	下架书籍	卖家可下架自己发布的书籍
订单模块	创建订单	买家对书籍下单，触发器自动更新书籍状态
订单模块	订单列表	查看作为买家或卖家的所有订单
订单模块	确认交易	调用存储过程完成交易，触发器自动更新信誉分
订单模块	取消订单	取消待确认的订单，触发器自动恢复书籍状态

3. 数据流图

3.1 顶层数据流图（0层DFD）

3.2 一层数据流图



4. 数据元素表

4.1 用户信息表 (User)

数据项	含义	数据类型	长度	取值范围	备注
user_id	用户ID	INT	-	自增	主键
student_id	学号	VARCHAR	20	唯一	登录账号

数据项	含义	数据类型	长度	取值范围	备注
nickname	昵称	VARCHAR	50	非空	显示名称
password	密码	VARCHAR	255	非空	加密存储
register_time	注册时间	DATETIME	-	系统时间	默认当前时间
contact_info	联系方式	VARCHAR	100	可空	QQ/微信/手机
credit_score	信誉积分	INT	-	≥0	默认100分

4.2 书籍分类表 (Category)

数据项	含义	数据类型	长度	取值范围	备注
category_id	分类ID	INT	-	自增	主键
category_name	分类名称	VARCHAR	50	唯一	如：计算机、文学

4.3 书籍信息表 (Book)

数据项	含义	数据类型	长度	取值范围	备注
book_id	书籍ID	INT	-	自增	主键
isbn	ISBN号	VARCHAR	20	唯一	国际标准书号
title	书名	VARCHAR	255	非空	-
author	作者	VARCHAR	100	非空	-
publisher	出版社	VARCHAR	100	可空	-
publication_year	出版年份	VARCHAR	10	可空	-
cover_image_url	封面图片URL	VARCHAR	512	可空	OSS存储地址

4.4 书籍分类关联表 (Book_Category)

数据项	含义	数据类型	长度	取值范围	备注
book_id	书籍ID	INT	-	外键	联合主键
category_id	分类ID	INT	-	外键	联合主键

4.5 发布信息表 (Listing)

数据项	含义	数据类型	长度	取值范围	备注
listing_id	发布ID	INT	-	自增	主键
seller_id	发布者ID	INT	-	外键	关联User
book_id	书籍ID	INT	-	外键	关联Book

数据项	含义	数据类型	长度	取值范围	备注
price	价格	DECIMAL	(10,2)	≥0	-
condition_desc	新旧程度	VARCHAR	20	非空	全新/九成新等
listing_type	发布类型	ENUM	-	出售/赠送	-
status	状态	ENUM	-	在售/已预定/已售出/已下架	默认在售
post_time	发布时间	DATETIME	-	系统时间	默认当前时间
description	详细描述	TEXT	-	可空	-

4.6 订单表 (Orders)

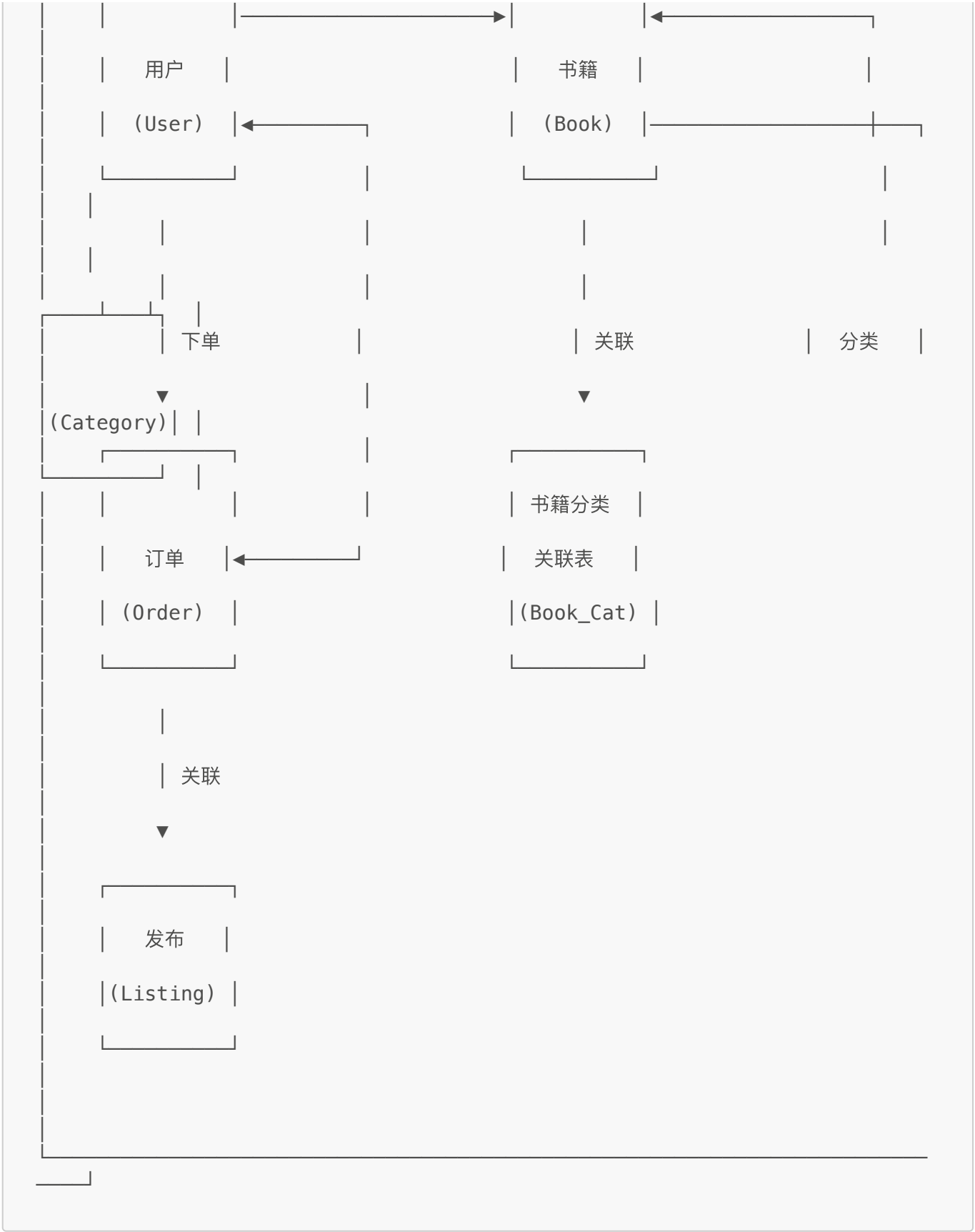
数据项	含义	数据类型	长度	取值范围	备注
order_id	订单ID	INT	-	自增	主键
listing_id	发布ID	INT	-	外键	关联Listing
buyer_id	购买者ID	INT	-	外键	关联User
order_time	下单时间	DATETIME	-	系统时间	默认当前时间
order_status	订单状态	ENUM	-	待确认/已完成/已取消	默认待确认
transaction_price	交易价格	DECIMAL	(10,2)	≥0	下单时价格

4.7 评论表 (Comment)

数据项	含义	数据类型	长度	取值范围	备注
comment_id	评论ID	INT	-	自增	主键
listing_id	发布ID	INT	-	外键	关联Listing
user_id	评论者ID	INT	-	外键	关联User
content	评论内容	TEXT	-	非空	-
comment_time	评论时间	DATETIME	-	系统时间	默认当前时间

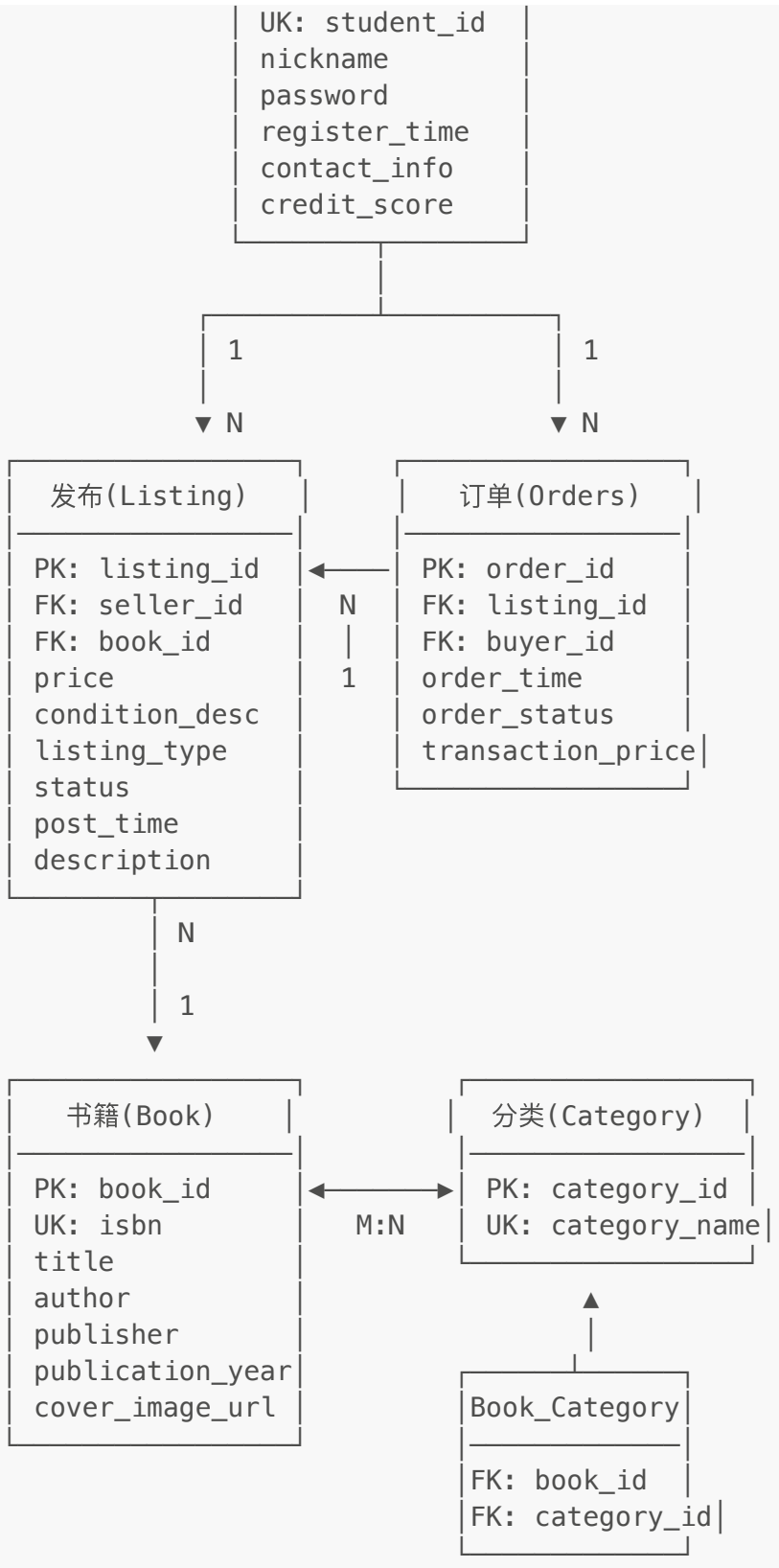
二、数据库概念模式设计

1. 系统初步 E-R 图


用户
(User)
书籍
(Book)
下单
关联
分类
(Category)
书籍分类
订单
(Order)
关联表
(Book_Cat)
关联
发布
(Listing)

2. 系统基本 E-R 图


用户(User)
PK: user_id



关系说明:
- 用户(1) : 发布(N) —— 一个用户可以发布多本书
- 用户(1) : 订单(N) —— 一个用户可以创建多个订单（作为买家）
- 发布(1) : 订单(N) —— 一个发布可以有多个订单（理论上只有一个成功）
- 书籍(1) : 发布(N) —— 一本书可以被多次发布（不同卖家）
- 书籍(M) : 分类(N) —— 多对多关系，通过Book_Category关联

三、数据库逻辑模式设计与优化

1. 数据库关系模式定义

由 E-R 图转换得到以下关系模式：

```
User(user_id, student_id, nickname, password, register_time, contact_info,
credit_score)
    主键: user_id
    候选键: student_id

Category(category_id, category_name)
    主键: category_id
    候选键: category_name

Book(book_id, isbn, title, author, publisher, publication_year,
cover_image_url)
    主键: book_id
    候选键: isbn

Book_Category(book_id, category_id)
    主键: (book_id, category_id)
    外键: book_id → Book(book_id)
    外键: category_id → Category(category_id)

Listing(listing_id, seller_id, book_id, price, condition_desc,
listing_type, status, post_time, description)
    主键: listing_id
    外键: seller_id → User(user_id)
    外键: book_id → Book(book_id)

Orders(order_id, listing_id, buyer_id, order_time, order_status,
transaction_price)
    主键: order_id
    外键: listing_id → Listing(listing_id)
    外键: buyer_id → User(user_id)

Comment(comment_id, listing_id, user_id, content, comment_time)
    主键: comment_id
    外键: listing_id → Listing(listing_id)
    外键: user_id → User(user_id)
```

2. 关系模式范式等级的判定与规范化

2.1 User 关系模式分析

函数依赖集：

- user_id → student_id, nickname, password, register_time, contact_info, credit_score
- student_id → user_id, nickname, password, register_time, contact_info, credit_score

范式判定：

- 1NF：所有属性都是原子的 ✓
- 2NF：非主属性完全函数依赖于主键 ✓
- 3NF：不存在传递依赖 ✓

结论：User 关系模式满足 3NF。

2.2 Book 关系模式分析

函数依赖集：

- book_id → isbn, title, author, publisher, publication_year, cover_image_url
- isbn → book_id, title, author, publisher, publication_year, cover_image_url

范式判定：

- 1NF：所有属性都是原子的 ✓
- 2NF：非主属性完全函数依赖于主键 ✓
- 3NF：不存在传递依赖 ✓

结论：Book 关系模式满足 3NF。

2.3 Listing 关系模式分析

函数依赖集：

- listing_id → seller_id, book_id, price, condition_desc, listing_type, status, post_time, description

范式判定：

- 1NF：所有属性都是原子的 ✓
- 2NF：主键为单属性，自动满足 ✓
- 3NF：所有非主属性直接依赖于主键，无传递依赖 ✓

结论：Listing 关系模式满足 3NF。

2.4 Orders 关系模式分析

函数依赖集：

- order_id → listing_id, buyer_id, order_time, order_status, transaction_price

范式判定：

- 1NF：所有属性都是原子的 ✓
- 2NF：主键为单属性，自动满足 ✓
- 3NF：所有非主属性直接依赖于主键，无传递依赖 ✓

结论：Orders 关系模式满足 3NF。

2.5 Book_Category 关系模式分析

函数依赖集：

- (book_id, category_id) → ∅ （只有主键，无非主属性）

范式判定：

- 满足 BCNF（只有主键）

结论：Book_Category 关系模式满足 BCNF。

3. 数据库关系模式优化

3.1 优化策略

本系统采用以下优化策略：

1. **反规范化设计**：在 Listing 查询时，为避免频繁连接 Book 和 User 表，在查询接口中使用 JOIN 一次性获取所需数据。
2. **冗余字段**：Orders 表中的 transaction_price 是 Listing.price 的冗余副本，目的是保存下单时的价格快照，防止卖家后续修改价格影响历史订单。
3. **状态字段设计**：Listing.status 和 Orders.order_status 使用 ENUM 类型，既节省存储空间，又保证数据完整性。

3.2 查询优化

针对高频查询场景，设计了以下优化：

查询场景	优化方案
按学号登录	为 student_id 建立唯一索引
按 ISBN 查书	为 isbn 建立唯一索引
按分类筛选书籍	为 category_id 建立索引
按卖家查发布	为 seller_id 建立索引
按买家查订单	为 buyer_id 建立索引
按发布查订单	为 listing_id 建立索引

四、数据库物理设计

1. 存取方法选择

本系统选择 **InnoDB** 存储引擎，原因如下：

1. **事务支持**：订单处理需要 ACID 事务保证
2. **外键约束**：表间关系需要外键约束保证参照完整性
3. **行级锁**：支持高并发的读写操作
4. **崩溃恢复**：支持自动崩溃恢复

2. 索引定义

2.1 主键索引（自动创建）

```
-- 所有表的主键自动创建聚簇索引
PRIMARY KEY (`user_id`)      -- User 表
PRIMARY KEY (`category_id`)  -- Category 表
PRIMARY KEY (`book_id`)      -- Book 表
PRIMARY KEY (`listing_id`)   -- Listing 表
PRIMARY KEY (`order_id`)     -- Orders 表
PRIMARY KEY (`comment_id`)   -- Comment 表
PRIMARY KEY (`book_id`, `category_id`)  -- Book_Category 表（联合主键）
```

2.2 唯一索引

```
-- User 表：学号唯一索引
UNIQUE INDEX `uk_student_id` (`student_id` ASC)

-- Category 表：分类名称唯一索引
UNIQUE INDEX `uk_category_name` (`category_name` ASC)

-- Book 表：ISBN 唯一索引
UNIQUE INDEX `uk_isbn` (`isbn` ASC)
```

2.3 外键索引

```
-- Listing 表
INDEX `fk_listing_user_idx` (`seller_id` ASC)
INDEX `fk_listing_book_idx` (`book_id` ASC)

-- Orders 表
INDEX `fk_order_listing_idx` (`listing_id` ASC)
INDEX `fk_order_user_idx` (`buyer_id` ASC)

-- Comment 表
INDEX `fk_comment_listing_idx` (`listing_id` ASC)
INDEX `fk_comment_user_idx` (`user_id` ASC)

-- Book_Category 表
INDEX `fk_book_category_category_idx` (`category_id` ASC)
```

3. 存储结构

表名	预估数据量	字符集	存储引擎

表名	预估数据量	字符集	存储引擎
user	10,000+	utf8mb4	InnoDB
category	20-50	utf8mb4	InnoDB
book	50,000+	utf8mb4	InnoDB
book_category	60,000+	utf8mb4	InnoDB
listing	100,000+	utf8mb4	InnoDB
orders	50,000+	utf8mb4	InnoDB
comment	200,000+	utf8mb4	InnoDB

附录：完整建表 SQL 语句

详见项目文件 `someDetailedFiles/建表语句.txt`