附件 1

《数据库系统原理》大作业

系统设计报告

题目名称：　寝室出物管理系统

学号及姓名：_23371341_黎欣妍_____

_____23371251_吴思齐_____

2025 年　12　月　30　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：系统功能设计与建表功能数据库操作实现	子任务 2：系统数据库设计与后端相关功能数据库操作实现	
黎欣妍	参与系统功能设计	参与数据库设计 实现查询、插入等数据库操作方法接口 实现后端逻辑 补充前端	50%
吴思齐	参与系统功能设计 数据库建表	参与数据库设计 实现前端 补充后端逻辑	50%

一．需求分析

1．需求描述

我们观察到宿舍楼群里经常有出物/收物的需求，虽然可以在宿舍楼群里直接沟通，但往往混杂于其他消息中，且通常需要加微信私聊（导致微信朋友列表混乱），较为不便。所以我们设计了宿舍出物管理系统，方便有收购/出售需求的同学发帖出物或寻找自己想购入的物品。

2. 系统功能设计

（1）用户相关功能：

• 用户注册、登录：

含密码错误、账号被封禁等报错，使用“管理员注册码”注册管理员

• 查看个人信息、修改个人信息、修改密码

• 查看用户列表（管理员权限）

• 封禁用户（管理员权限）

• 解封用户（管理员权限）

（2）业务功能：

帖子相关功能：

• 查看帖子列表：

可在首页、出物大厅栏、我的出物栏查看所有/部分帖子，且可根据上架/下架状态区分可视范围

• 查看帖子详情：

含发布者、标签、价格等信息

• 发布帖子：

输入帖子标题（物品名）、内容、价格、数量，可上传图片、选择 Tag 标签，进行发布

• 修改帖子：

发布者可修改帖子信息

• 删除帖子（下架）：

帖子状态变为“下架”，帖子可视范围发生变化（他人不可见，只发布者可见）

• 修改帖子可见范围（寝室/楼层/楼栋/全楼）

收藏相关功能：

• 添加收藏

• 取消收藏

• 查看收藏列表

搜索相关功能：

• 综合搜索：包括按关键词+标签+范围搜索相关帖子

订单相关功能：

• 买家创建订单（下单）：

创建一个订单，设置订单初始状态变为“待交接”

• 查看买家订单列表

• 查看卖家订单列表

• 查看订单详情：

含物品名、价格、买家、卖家等

• 卖家确认交接

• 买家确认订单完成

• 取消订单

投诉相关功能：

• 用户提交投诉：

对某一订单发起投诉，输入投诉原因等内容并提交

• 用户查看自己的投诉记录

• 查看投诉详情：

包含投诉订单号、投诉原因等

• 查看全部投诉（管理员权限）

• 处理投诉（管理员权限）：

写入投诉结果，提交投诉状态（处理中/已处理/驳回）

消息和公告相关功能：

• 查看消息：

用户查看订单状态变化的消息，可标记为已读

• 查看公告：在系统公告栏查看管理员发布的公告

• 发布公告、删除公告（管理员权限）

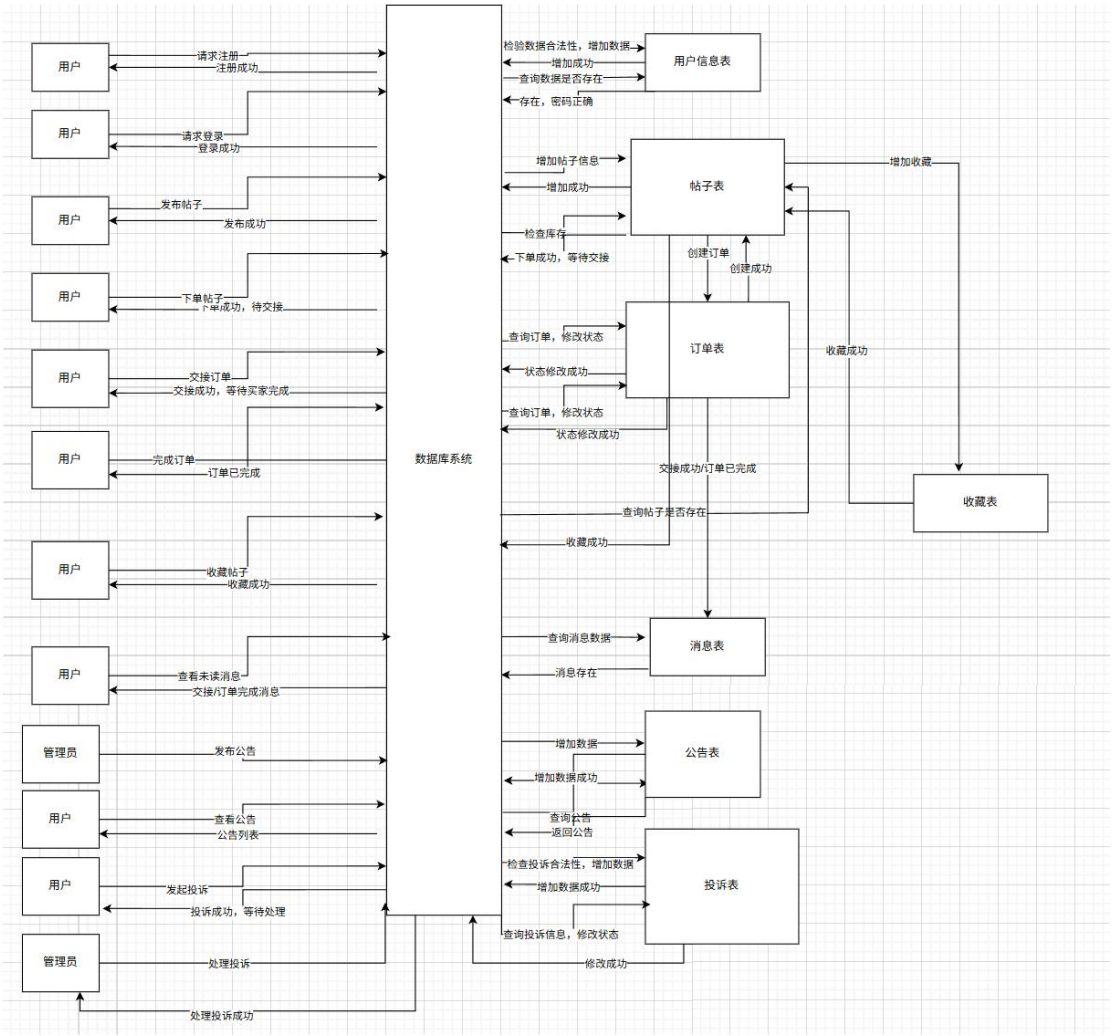
数据统计相关功能：

• 用户个人统计：包含出物量、订单完成率、被投诉率等数据信息

• 系统总览统计（管理员权限）：

管理员可查看平台月度订单统计等数据

3．数据流图



4．数据元素表

room

字段名	数据类型	约束/默认	说明
room_id	INT	PK, AUTO_INCREMENT	寝室主键
building	VARCHAR(50)	NOT NULL	楼栋
floor	INT	NOT NULL	楼层
room_no	VARCHAR(20)	NOT NULL	寝室号
created_at	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	创建时间
updated_at	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP	更新时间

user

字段名	数据类型	约束/默认	说明
user_id	INT	PK, AUTO_INCREMENT	用户主键
username	VARCHAR(150)	NOT NULL	用户名
password	VARCHAR(128)	NOT NULL	密码（哈希/存储值）
email	VARCHAR(150)	NULL	邮箱
wechat	VARCHAR(30)	UNIQUE, NULL	微信号
student_id	VARCHAR(20)	UNIQUE, NULL	学号
role	TINYINT	NOT NULL, DEFAULT 0, CHECK(role IN (0,1))	角色字段（项目内约束）
user_role	TINYINT	NOT NULL, DEFAULT 1, CHECK(user_role IN (1,2,3))	用户身份（如普通/卖家/管理员等口径）
status	ENUM('正常','封禁')	NOT NULL, DEFAULT '正常'	账号状态
room_id	INT	NOT NULL, FK → room(room_id)	所属寝室
is_staff	BOOLEAN	DEFAULT FALSE	Django 兼容字段
is_superuser	BOOLEAN	DEFAULT FALSE	Django 兼容字段
created_at	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	创建时间
updated_at	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP	更新时间

tag

字段名	数据类型	约束/默认	说明
tag_id	INT	PK, AUTO_INCREMENT	标签主键
tag_name	VARCHAR(50)	NOT NULL, UNIQUE	标签名
ref_count	INT	NOT NULL, DEFAULT 0	引用次数/热度
created_at	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	创建时间
updated_at	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP	更新时间

posting

字段名	数据类型	约束/默认	说明
posting_id	INT	PK, AUTO_INCREMENT	帖子主键
title	VARCHAR(100)	NOT NULL	标题
content	TEXT	NULL	内容描述
price	DECIMAL(10,2) UNSIGNED	NOT NULL	价格
quantity	INT UNSIGNED	NOT NULL, DEFAULT 0	库存/数量
brand	VARCHAR(50)	NULL	品牌
image_url	VARCHAR(255)	NULL, COMMENT '封面图片URL(演示/展示用)'	封面图路径/URL
condition	ENUM('全新','几乎全新','轻微使用痕迹','空')	NULL	成色/新旧程度
tag_id	INT	NULL, FK → tag(tag_id)	标签
status	ENUM('上架','下架','已约满')	NOT NULL, DEFAULT '上架'	上下架与售罄状态
scope	ENUM('寝室','楼层','楼栋','全楼')	NOT NULL, DEFAULT '全楼'	可见范围
owner_id	INT	NOT NULL, FK → user(user_id)	发布者
created_at	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	创建时间
updated_at	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP	更新时间

favorite

字段名	数据类型	约束/默认	说明
f_id	INT	PK, AUTO_INCREMENT	收藏记录主键
user_id	INT	NOT NULL, FK → user(user_id)	收藏者
posting_id	INT	NOT NULL, FK → posting(posting_id)	被收藏帖子
created_at	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	收藏时间

order

字段名	数据类型	约束/默认	说明
order_id	INT	PK, AUTO_INCREMENT	订单主键
posting_id	INT	NOT NULL, FK → posting(posting_id)	对应帖子
buyer_id	INT	NOT NULL, FK → user(user_id)	买家
seller_id	INT	NOT NULL, FK → user(user_id)	卖家
num	INT	NOT NULL, CHECK(num > 0)	购买数量
status	ENUM('待交接','已交接','完成','取消')	NOT NULL, DEFAULT '待交接'	订单状态
created_at	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	创建时间
confirmed_at	DATETIME	NULL	确认交接时间
cancel_reason	VARCHAR(200)	NULL	取消原因
updated_at	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP	更新时间

notice

字段名	数据类型	约束/默认	说明
notice_id	INT	PK, AUTO_INCREMENT	通知主键
type	ENUM('系统','交接提醒','公告')	NOT NULL	通知类型
content	TEXT	NOT NULL	内容
receiver_id	INT	NOT NULL, FK → user(user_id)	接收者
related_order_id	INT	NULL, FK → order(order_id)	关联订单（可空）
status	ENUM('未读','已读')	NOT NULL, DEFAULT '未读'	阅读状态
created_at	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	创建时间
updated_at	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP	更新时间

complaint

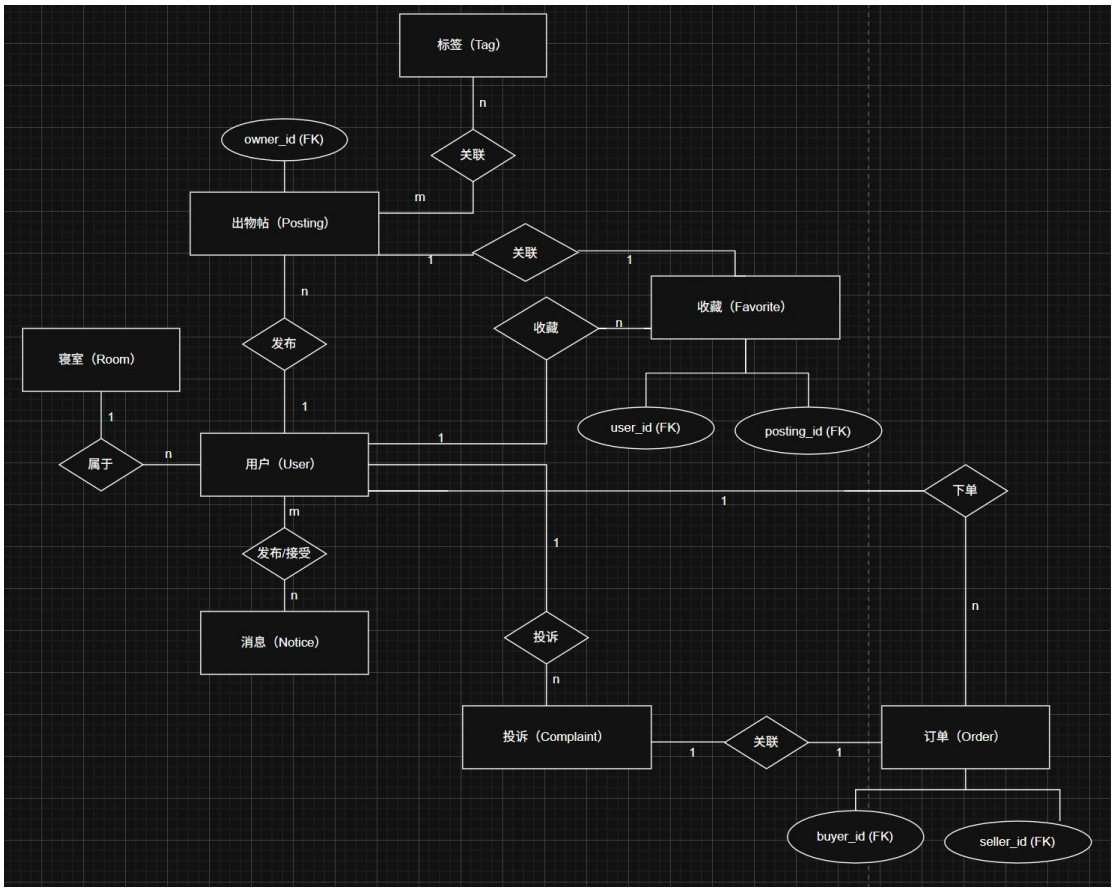
字段名	数据类型	约束/默认	说明
complaint_id	INT	PK, AUTO_INCREMENT	投诉主键
order_id	INT	NOT NULL, FK → order(order_id)	关联订单
complainant_id	INT	NOT NULL, FK → user(user_id)	投诉人
accused_id	INT	NOT NULL, FK → user(user_id)	被投诉人
content	TEXT	NOT NULL	投诉内容
status	ENUM('待处理','已处理','驳回','处理中')	NOT NULL, DEFAULT '待处理'	处理状态
result	VARCHAR(200)	NULL	处理结论
created_at	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	创建时间
handled_at	DATETIME	NULL	处理时间
updated_at	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP	更新时间

image

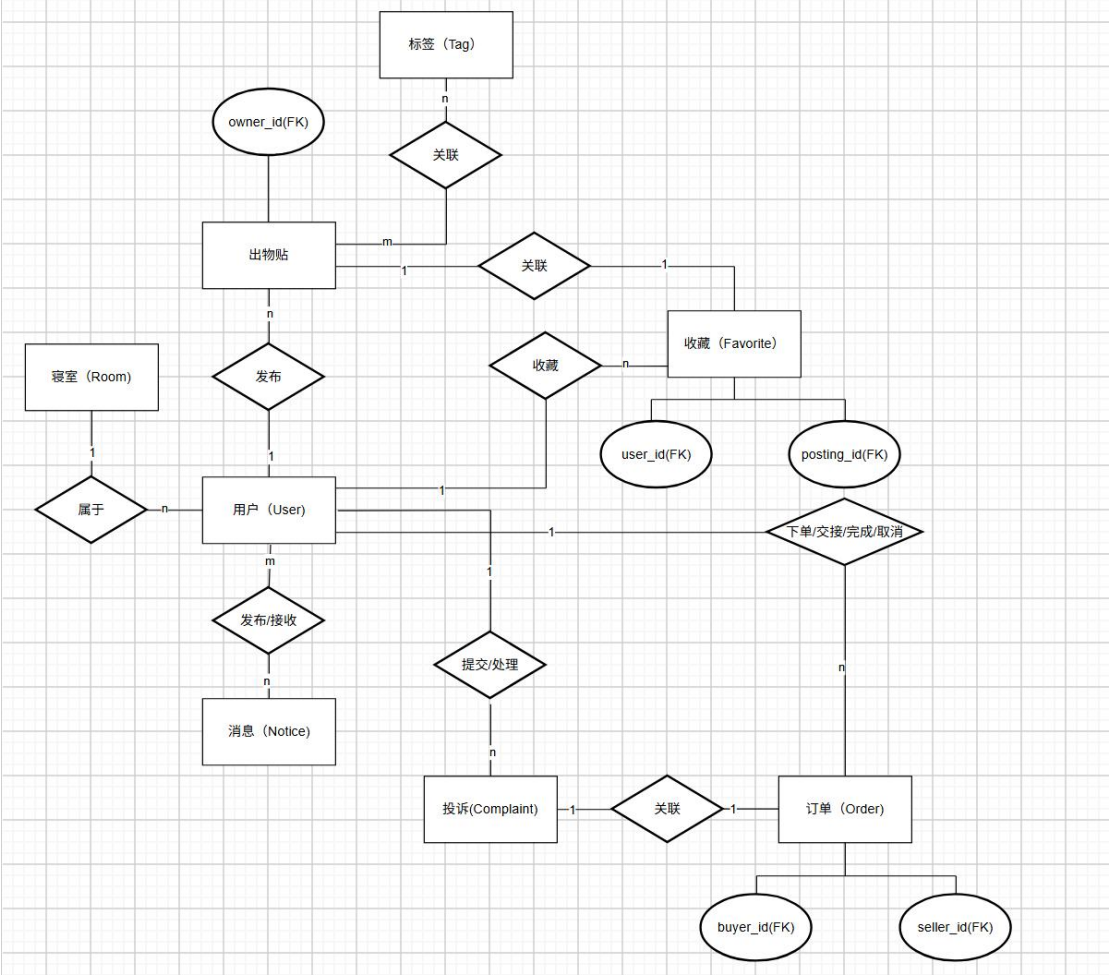
字段名	数据类型	约束/默认	说明
image_id	INT	PK, AUTO_INCREMENT	图片主键
posting_id	INT	NULL, FK → posting(posting_id)	关联帖子（可空）
uploader_id	INT	NOT NULL, FK → user(user_id)	上传者
path	VARCHAR(255)	NOT NULL	图片路径
category	VARCHAR(50)	NULL	图片类别（如封面/详情）
created_at	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	上传时间

二．数据库概念模式设计

1．系统初步E-R 图


标签（Tag）
n
owner_id (FK)
关联
m
出物帖（Posting）
1
关联
1
n
收藏（Favorite）
发布
收藏
n
寝室（Room）
1
user_id (FK)
posting_id (FK)
1
属于
n
用户（User）
1
下单
1
m
1
发布/接受
n
n
消息（Notice）
投诉
n
投诉（Complaint）
1
关联
1
订单（Order）
buyer_id (FK)
seller_id (FK)

2．系统基本 E-R 图



三、数据库逻辑模式设计与优化

1. 数据库关系模式定义

注：由 E-R 图得到的关系模式。

Room（寝室）： Room(room_id, building, floor, room_no, created_at, updated_at)

User(用户)：User(user_id, username, password, email, wechat, student_id,

role, user_role, status, room_id,is_staff, is_superuser,created_at, updated_at)

Tag（标签）： Tag(tag_id, tag_name, ref_count, created_at, updated_at)

Posting（出物帖）： Posting(posting_id, title, content, price, quantity, brand,image_url, condition, tag_id, status, scope,owner_id, created_at, updated_at)

Favorite（收藏）： Favorite(f_id, user_id, posting_id, created_at)

Order（订单）：Order(order_id, posting_id, buyer_id, seller_id, num, status, created_at, confirmed_at, cancel_reason, updated_at)

Notice（通知）：Notice(notice_id, type, content, receiver_id, related_order_id, status,created_at, updated_at)

Complaint（投诉）：Complaint(complaint_id, order_id, complainant_id, accused_id, content, status, result, created_at, handled_at, updated_at)

Image（图片，弱实体）：Image(image_id, posting_id, uploader_id,path, category, created_at)

2. 关系模式范式等级的判定与规范化

注：要规范到 3NF。

Room:
主键 room_id；其余属性（building, floor, room_no）完全函数依赖于主键，无部分依赖与传递依赖，至 BCNF。

User:
主键 user_id，student_id、wechat 可作为候选码；其余非主属性完全函数依赖于主键，不存在非主属性间的传递依赖，至 BCNF。

Tag:
主键 tag_id，tag_name 具有唯一性约束；非主属性完全依赖于主键，无传递依赖，至 BCNF。

Posting:
主键 posting_id，owner_id、tag_id 为外键；标题、价格、状态等属性均完全依赖主键，不存在由外键引起的传递依赖，满足 BCNF。

Favorite:
候选码为 (user_id, posting_id)，采用代理主键 f_id；不存在非主属性对非码属性的依赖，满足 BCNF。

Order:
主键 order_id；buyer_id、seller_id、posting_id 为外键。
其中 seller_id 理论上来源于 Posting.owner_id，属于**可控冗余**，通过业务规则约束（如触发器或应用层校验）保证一致性；非主属性对主键完全依赖，满足 3NF / BCNF。

Notice：
主键 notice_id，receiver_id 为外键；与订单的关联通过 related_order_id 实现，无非主属性传递依赖，满足 BCNF。

Complaint：
主键 complaint_id；order_id、complainant_id、accused_id 为外键；投诉内容与状态均完全依赖主键，不存在传递依赖，满足 BCNF。

Image：
主键 image_id；posting_id、uploader_id 为外键；图片路径等属性完全依赖主键，满足 BCNF。

3. 数据库关系模式优化

在初始设计中，用户可能直接包含寝室楼栋、楼层、房间号等信息，存在如下潜在函数依赖关系：

user_id → room_id → (building, floor, room_no)

这将导致非主属性对非主属性的**传递函数依赖**，违反第三范式。

优化措施：将寝室信息单独抽象为 Room 关系模式

User 表中仅保留 room_id 作为外键进行关联

优化结果：

消除了 User 表中的传递依赖

用户与寝室信息解耦，避免更新异常

User 与 Room 均满足 BCNF

四、数据库物理设计

注：说明所选择的存取方法，给出索引定义。

本系统基于 TaurusDB for MySQL 数据库管理系统实现，在完成数据库逻辑结构设计的基础上，对数据库的物理存储结构和访问路径进行了合理设计，以提高系统在实际运行过程中的查询效率和整体性能。

4.1 存取方法的选择

本系统选用 InnoDB 存储引擎 作为所有关系表的底层存储引擎，其主要存取方法如下：

1. 基于 B+ 树的索引存取方式

InnoDB 引擎采用 B+ 树结构来组织索引，通过索引可以快速定位数据记录，显著降低查询的磁盘 I/O 开销。

2. 聚簇索引与辅助索引相结合

每张表的主键自动形成聚簇索引，数据记录按照主键顺序物理存储；

在非主键字段上建立辅助索引，用于加速常见的条件查询和连接操作。

3. 索引扫描与顺序扫描结合使用

对于高频访问、条件过滤明显的查询，优先通过索引扫描完成；

对于数据量较小或无过滤条件的查询，采用顺序扫描。

该存取方法在保证数据一致性和事务安全的同时，能够满足系统对查询性能的需求。

4.2 索引设计原则

在索引设计过程中，遵循以下基本原则：

1. 所有关系表均定义主键索引；

2. 外键字段建立辅助索引，以提高多表连接效率；

3. 对高频查询条件建立单列索引或联合索引；

4. 对具有唯一性约束的字段建立唯一索引，保证数据一致性；

5. 联合索引的设计遵循最左前缀原则，兼顾过滤与排序需求。

4.3 索引定义说明

（1）主键索引

系统中所有关系表均定义了主键，数据库自动为主键建立聚簇索引，例如：

Room(room_id)

User(user_id)

Posting(posting_id)

Order(order_id)

Complaint(complaint_id)

Notice(notice_id)

Image(image_id)

作用：

支持按主键的高效精确查询；

保证记录唯一性；

提高实体详情查询的访问效率。

（2）唯一索引

在需要保证数据唯一性的字段上建立唯一索引，例如：

User(wechat)

User(student_id)

Tag(tag_name)

Favorite(user_id, posting_id)

作用：

防止重复数据插入；

支持基于唯一字段的快速查询；

保证系统数据一致性。

（3）外键辅助索引

为提高多表连接操作的性能，在外键字段上建立辅助索引，包括：

User(room_id)

Posting(owner_id)

Posting(tag_id)

Favorite(user_id)

Favorite(posting_id)

Order(posting_id)

Order(buyer_id)

Order(seller_id)

Notice(receiver_id)

Notice(related_order_id)

Complaint(order_id)

Complaint(complainant_id, accused_id)

Image(posting_id)

Image(uploader_id)

作用：

提高 JOIN 操作效率；

避免外键约束带来的全表扫描；

支持用户、订单、帖子等实体之间的关联查询。

（4）联合索引

针对系统中常见的多条件组合查询，设计了联合索引，例如：

Room(building, floor, room_no) —— 支持寝室的精确定位查询；

Posting(status, scope)—— 支持按商品状态与可见范围筛选出物帖；

Complaint(complainant_id, accused_id) —— 支持投诉关系相关查询。

作用：

减少多条件查询的检索范围；

提高复合条件过滤效率；

降低排序和回表操作的开销。