桂林电子科技大学2024-2025学年 第2学期

**数据库系统原理综合 实验报告**

**课号： 2422430 任课老师：唐麟**

|  |  |  |  |
| --- | --- | --- | --- |
| 项目名称 | **物流派送系统** | | |
| 院 系 | **计算机与信息安全学院** | 专业 | **软件工程** |
| 实验日期 | **2025年5月17日** | | |

团队任务分工表

|  |  |  |  |
| --- | --- | --- | --- |
| 题目 | 物流派送系统 | | |
| 负责人 | 杨一婕 | | |
| 合作  完成 | 设计数据库具体表项内容，将数据库上传阿里云端，完成前后端并实现对接，优化数据库逻辑结构，设置触发器、视图函数 | | |
| 序号 | 学号 | 姓名 | 个人负责模块 |
| 1 | 2300320304 | 杨一婕 | 登录注册，存储用户信息，用户权限登录模块，增加任务查询任务状态 |
| 2 | 2300320401 | 陆丽叶 | 管理员权限登录模块，完成配送任务，增加删除派送员、仓库 |

1. 需求分析

1.1 业务需求：

模拟物流派送系统需实现用户基于仓储与派送能力选择仓库下单，布置派送任务，并实时追踪订单状态。用户端需展示仓库的存储容量、当前派送负荷，支持筛选最优仓库下单；下单后生成派送任务，用户可查看订单的派送进度及派送员信息。管理员需管理仓库信息（增删改查仓库数量、容量及派送方式参数），统计并管理不同派送方式（如陆运、空运）的余额；维护派送员信息（添加、删除、修改联系方式及派送范围），并向派送员推送订单详情（收件人地址、联系方式及特殊要求）。系统需确保订单流程透明化，提升资源利用率与派送时效，减少人工干预，满足用户、派送员与管理端的高效协同需求。

1.2 用户需求：

1.2.1 客户需求  
 用户需通过系统便捷查看各仓库的仓储容量、派送能力，并基于此选择最优仓库下单；下单后系统自动生成派送任务，用户可随时查看订单状态（“未发货”“配送中”“已签收”）。界面需简洁直观，适配网页与移动端，确保操作流畅。1.2.2 管理员需求

管理员需动态管理仓库信息（增删查改仓库数量），实时监控各仓库负载状态及各派送方式的剩余容量；需维护派送员档案（录入/删除人员），系统自动将订单派发至对应派送员并推送客户信息（地址、电话、备注）。后台需提供数据，进行权限分级实现安全管理。

1.3 系统需求

系统需采用分层架构，前端为支持用户与管理员的交互界面网站，后端集成仓储管理、订单分配与前端分离。数据库需存储仓库数据（名称、容量、派送方式、仓库容量）、订单信息（ID、状态、创建时间、派送员）、派送员档案（ID、姓名、联系方式）。实现动态计算各运输方式派送能力并更新前端展示；管理仓库和派送员信息。安全性需保障用户隐私数据加密、权限分级控制（用户仅查看自身订单，管理员全权限）。

1. 所用开发工具和技术

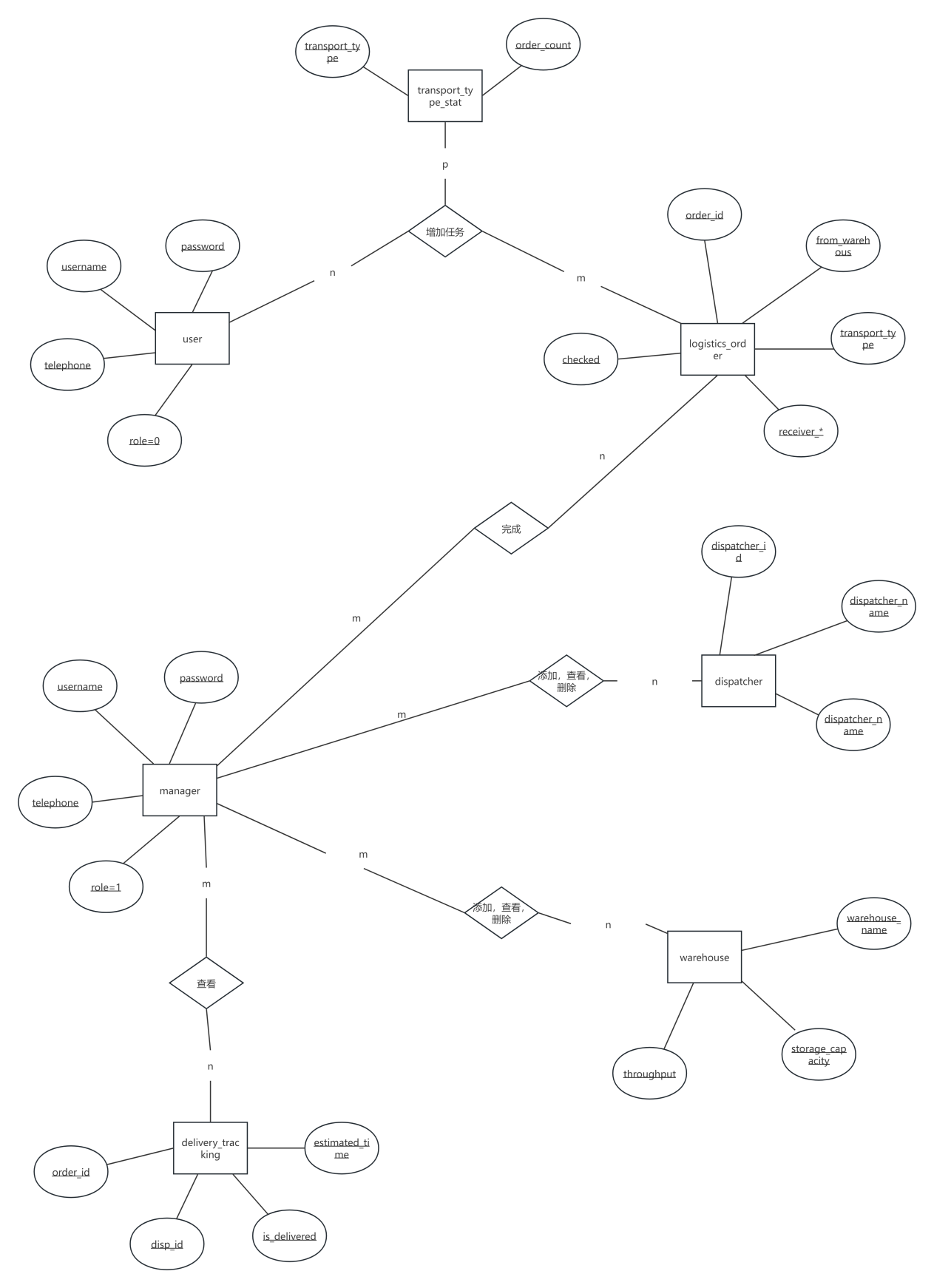
物流派送系统使用了阿里云服务器，将数据库上传云服务器实现数据库远程连接，并且通过宝塔面板实现服务器与数据库的可视化管理，管理安全端口，降低运维门槛，支持一键部署、监控及日志分析，提升效率。前端使用了Vue框架创建响应式界面，后端使用了Python的Flask框架，实现前后端分离，使得代码结构更清晰，易于维护和扩展。

数据库方面使用了MySQL＋Redis，管理派送任务、仓库信息、用户信息、配送员信息等数据。MySQL为关系型数据库，提供复杂查询能力，适合存储结构化数据（如用户信息、仓库配置、派送任务关系），确保数据一致性与完整性；Redis作为内存数据库，用于提高接口相应速度，快速验证token有效性，减小数据库响应压力。两者结合兼顾存储与缓存需求，优化系统整体数据处理量与用户体验。

1. 数据库设计

3.1 概念结构设计

3.1.1 E-R图



3.1.2 关系设计

1:N:一个配送员处理多个物流跟踪记录（dispatcher<->delivery\_tracking）

1:N:一个仓库关联多个物流订单（warehouse<->logistics\_order）

1:1:用户基础信息对应该用户详细信息（user<->user\_msg）

3.2 逻辑结构设计

3.2.1 数据库表项

表3.1：warehouse表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名** | **类型** | **长度** | **不是null** | **是否为主键** | **注释** |
| warehouse\_name | varchar | 50 | yes | √ | 仓库名称 |
| storage\_capacity | int |  | yes |  | 仓库容量 |
| throughput | int |  | yes |  | 月吞吐量 |

表3.2：logistics\_order表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名** | **类型** | **长度** | **不是null** | **是否为主键** | **注释** |
| order\_id | int |  | yes | √ | 自增订单编号 |
| from\_warehouse | varchar | 50 | yes |  | 发货仓库 |
| size | int |  | yes |  | 订单大小 |
| fright | int |  | yes |  | 运费 |
| transport\_type | varchar | 50 | yes |  | 运输方式 |
| receiver\_phone | varchar | 50 | yes |  | 收件人电话 |
| receiver\_name | varchar | 50 | yes |  | 收件人姓名 |
| receiver\_address | varchar | 50 | yes |  | 收件人地址 |
| checked | int |  |  |  | 订单状态 |
| create\_time | varchar | 50 |  |  | 创建时间 |

表3.3：dispatcher表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名** | **类型** | **长度** | **不是null** | **是否为主键** | **注释** |
| dispatcher\_id | varchar | 50 | yes | √ | 配送员编号 |
| dispatcher\_name | varchar | 50 | yes |  | 配送员姓名 |
| dispatcher\_phone | varchar | 50 | yes |  | 配送员联系方式 |

表3.4：delivery\_tracking表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名** | **类型** | **长度** | **不是null** | **是否为主键** | **注释** |
| order\_id | int |  | yes | √ | 订单编号 |
| receiver\_phone | varchar | 50 | yes |  | 收件人电话 |
| disp\_id | varchar | 50 | yes |  | 配送员编号 |
| estimated\_time | varchar | 50 | yes |  | 预计到达时间 |
| is\_delivered | int |  | yes |  | 签收状态 |

表3.5：transport\_type\_stat表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名** | **类型** | **长度** | **不是null** | **是否为主键** | **注释** |
| transport\_type | varchar | 50 | yes | √ | 运输方式 |
| order\_count | int |  | yes |  | 订单数量 |

表3.6：user表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名** | **类型** | **长度** | **不是null** | **是否为主键** | **注释** |
| id | int |  | yes | √ | 用户编号 |
| username | varchar | 20 | yes |  | 用户姓名 |
| password | varchar | 20 | yes |  | 用户密码 |
| telephone | varchar | 20 | yes |  | 用户联系方式 |
| role | int |  | yes |  | 权限 |

表3.7：user\_msg表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **名** | **类型** | **长度** | **不是null** | **是否为主键** | **注释** |
| id | int |  |  |  | 用户编号 |
| real\_name | varchar | 50 | yes |  | 真实姓名 |
| sex | varchar | 50 | yes |  | 性别 |
| age | int |  | yes |  | 年龄 |
| mail | varchar | 50 | yes |  | 邮箱 |
| phone | varchar | 50 | yes |  | 电话 |
| user\_name | varchar | 50 | yes |  | 用户名 |

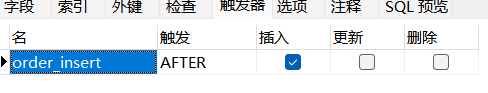
3.2.2 完整性约束

主键约束（如dispatcher\_id、order\_id等）保证记录唯一性；外键约束（如logistics\_order.from\_warehouse引用warehouse.warehouse\_name）确保跨表关联有效；唯一索引（如warehouse\_name）防止重复值；非空约束（如dispatcher\_name、receiver\_phone）强制关键字段必有值；默认值（如checked=0定义初始状态）减少空值风险

3.3 物理结构设计

3.3.1 触发器设置

（1）order\_insert触发器

作用logistics\_order表，当新增一个物流订单时，自动更新transport\_type\_stat表中对应运输方式（如“空运”“陆运”）的订单数量（order\_count+1），使管理员能实时看到仓库运输的情况。  
 

1. update\_warehouse\_capacity触发器

作用logistics\_order表，当新增一个物流订单时，自动更新warehouse表中对应仓库的储存能力（storage\_capacity-size），使用户在创建下一个派送任务时，可以实时依据自身情况选择仓库。



（3）delivery\_tracking\_insert触发器

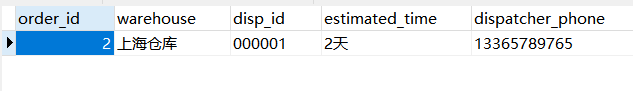
作用于delivery\_tracking表，当管理员将用户布置的派送订单下发时，自动将关联订单状态（checked）标记为1（配送中），使派送系统合理化。



3.3.2 视图设置

(1)completed\_deliveries视图

关联派送订单表（logistics\_order）、物流跟踪表（delivery\_tracking）和配送员表（dispatcher），根据订单基础信息、配送状态及时间、配送员联系方式，展示所有已签收的订单（checked = 2）的完成配送信息，包括订单ID、仓库名称、配送员编号、预计到达时间及配送员电话。



1. active\_deliveries视图

关联订单表（logistics\_order）和物流跟踪表（delivery\_tracking），根据订单基础信息、配送状态及时间，展示所有配送中的订单（checked = 1）的实时配送信息，包括订单ID、仓库名称、配送员编号及预计到达时间。



3.3.3 索引设置

(1)主键索引和普通索引：每个表的主键建立unique索引，有助于快速定位记录。例通过`order\_id`快速找到物流订单。在其余属性上建立了normal索引加速查询条件中的字段，如`from\_warehouse`用于按仓库查询订单，`transport\_type`用于按运输方式筛选订单。



（2）复合索引：在`logistics\_order`表中，经常需要根据`from\_warehouse`和`transport\_type`联合查询，现有的单独索引可能无法有效加速这类查询，所以设置复合索引（from\_warehouse, transport\_type)：同时按`from\_warehouse`和`transport\_type`查询订单，查询更高效。



3.3.4 数据库安全性考虑

通过SQLAlchemy的参数化查询防范SQL注入，所有用户输入均使用命名参数绑定（如:tel），避免直接拼接SQL语句。采用基于角色的访问控制（RBAC），关键接口（如仓库管理）通过验证用户角色（如role=1为管理员）限制操作权限；用户订单操作时通过receiver\_phone与token解析的手机号比对，实现行级数据权限控制，确保用户只能操作自身数据。敏感操作（如删除订单）使用事务保证原子性，异常时自动回滚。输入验证方面，强制校验必填字段（如订单ID），动态更新时通过白名单（allowed\_fields）限制可修改字段，防止非法参数篡改。同时采用Token认证机制，关键接口通过get\_token\_phone解析用户身份，结合数据库查询实时验证权限有效性。删除操作优先清理关联表数据（如delivery\_tracking），维护数据完整性和一致性。

3.3.5 Redis的使用

1. 详细设计与实现

分模块进行设计，小组成员分着写这一部分，然后合在一起。

这一个部分就是详细的将自己的系统详细设计与实现介绍清楚。设计就是画流程图之类，并配上文字说明。实现就是实现效果（截图）配上关键代码、文字说明是怎么做出来这样的结果的。

结构可以灵活些，比如根据流程顺序分块介绍，比如根据用户分类的功能介绍。

但是需要注意：（1）不能只有图和代码，必须有文字说明介绍（2）不能大段大段代码，**只能是关键的代码**。而且如果出现代码，那么必须有文字说明（不是代码上的注释）。（3）如果有用到设计模式的同学，这里可以好好的写写。

1. 测试

主要写功能测试：根据需求，将系统分为多个场景，以场景为主线，结合等价类划分、边界值分析等等设计测试用例，进行测试。测试发现问题，进行修改，修改后进行回归测试。

如有web开发，可以写写兼容性测试和性能测试。

6 AI大模型在项目中的使用

在这里请具体写出，项目小组是如何使用AI大模型辅助设计开发的。

有何具体使用心得体会。

7 存在的问题及改进思路

**注意：论文中的任何部分都不能用AI写！！**