同濟大學
TONGJI UNIVERSITY

# 数据库系统原理课程设计

# 智慧城市安全管理系统

学　　号 **2053459**

姓　　名 毛朝炜

专　　业 计算机科学与技术

授课老师 关佶红

日期：2022/06/23

# 目录

# 摘要

随着互联网的深入发展和广泛应用，国家对城市管理信息化的要求逐步提高，积极推进城市管理数字化、精细化、智慧化，旨在提高城市管理效能。传统城市管理模式缺乏部门间的统一协调机制，城市问题暴露不及时、漏报、重复上报等问题突出，且缺乏对执法人员的监察机制，相关部门对问题的推诿导致难以有效满足公众需求。面对上述现实问题，本文基于数据库系统原理的相关理论方法设计了一个能协调有关部门管理且面向公众需求的智慧城管系统，并基于目前企业常用的 Vue.js 框架和 Django 后端框架基本实现了该系统。设计思路为：针对城市管理的主要需求进行分析，并基于关键需求对某城市管理系统进行概念设计、逻辑设计和物理设计。



**关键词**：城市管理，智慧城市，数据库设计与开发


# 一、引言

有效、稳定的城市安全管理机制是城市健康发展的重要基础。然而，城市在发展的过程中难以避免的存在许多负面问题，比如：车辆非法停放、恶意交通事故、聚众打斗、非法经营、民生安全问题等。城市问题的有效解决作为城市管理的一项重要任务，直接影响着城市风貌和民众生活质量。

目前许多城市针对城市问题的管理存在诸多问题，表现在：发现问题不及时；责权不清，市民不知道向哪一个部门反馈问题，部门各自职责划分不清楚；巡查监控存在盲区，繁华路段重点监察，而相对偏僻的地区监察少或者无监察，一旦发生事故则缺乏事实证据；部门间工作重复冗余以及协同困难等。城市管理中存在的上述各种问题，主要原因是缺少了对现代化城市发生的问题进行统一处理规划和现代化管理手段的有力支撑。

针对城市问题的主要矛盾，本次“智慧城市安全管理系统”设计的主要目的是实现城市管理的高效化、数字化、精细化，建立一个能协调有关部门监管且面向公众需求的管理系统，旨在解决城市管理中存在的以下问题：1、城市问题的权责划分和解决。2、建立有效的巡查监督机制。3、建立统一的管理指挥平台，帮助多部门联合办案。4、建立面向民众的开放的举报系统。

# 二、需求与可行性分析

## 2.1 功能需求

### 2.1.1 系统功能概述

通过查阅相关文献资料，确定了城市管理系统的核心功能包括：违规事件处理结果的上报和查询；普通用户举报周围发生的违规事件；城管系统的团队管理；城管责任区划分和安排；巡查人员状态监控。

具体需求分析如下：

**A、案件上报处理功能**

**• 城市问题的巡查、上报、登记、受理**。问题上报可以由监督员和普通居民发动，监督员在负责的责任区进行巡查，发现问题则通过手机设备拍照记录、对问题详细描述，居民也可通过系统进行违规事件举报，把相关信息上传到信息中心进行立案处理。完成上报和登记后，由受理员对问题进行初步筛选，对于明显与事实不相符的问题置为不予受理，对于存疑的问题进行派遣核实。

**• 存疑案件现场核实**。对于民众上报的存疑案件，监督员根据管理人员发派的核查任务，前往指定现场进行核实，如果现场与描述不相符，则反馈“问题不属实”，反之“问题属实”，同样进行拍照记录现场及文字描述。

**• 立案调查**。值班人员对于监督员上报的问题，以及监督员核实通过的居民上报问题进行进一步核实，对情况属实者进行立案调查。

**• 案件派遣、处置、反馈**。派遣员对立案的案件进行分析，确定相关案件的负责部门，并把案件派遣给相关部门；专业部门接收到案件后，进行专业分析和审查诊断，并将结果反馈到系统；最后，派遣员对来自专业部门的处置反馈案件进行分析，对于处理结果不符合要求的驳回重新派遣，符合要求的结果反馈给追中心进行核查。

**• 案件结果核查**。受理员接收到处理反馈的案件后，交给监督员进行现场核查；监督员根据分配的任务前往现场，把现场情况同专业部门案件处理后的案件描述及图片进行对照，如果结果与描述一致则反馈情况属实，反之则反馈不属实。

**• 结案审核**。值班人员接收到现场审核的反馈案件，对情况属实的进行结案操作，反之，则转交给指挥部门重新派遣、审核。

**B、组织和管理功能**

**• 责任区划分及安排。**把当前团队所在的城市管理区域作为一个“责任区”，把责任区划分成若干“子责任区”，并绑定到巡查人员的巡查区域上，每个单元

网络至少有一个巡查人员负责。

- **巡查人员工作排班。**管理部门能够实现对巡查人员工作时间的具体安排。
- **城管系统团队管理。**够对参与系统运行的所有部门进行管理和维护。
- **用户管理。**对参与系统运行的人员进行权限管理、角色管理、密码管理等。

### 2.1.2 业务流程分析

基于 2.1.1 的业务需求分析，可以将系统用户的角色概括为三类：管理员、巡查员和普通用户。管理员主要负责管理巡查员处理违规事件，巡查员负责对违规行为进行监察和及时处理，普通用户能通过系统向管理机构（管理员）上报发生的违规行为。智慧城管的整体业务流程图如图 2.1 所示。

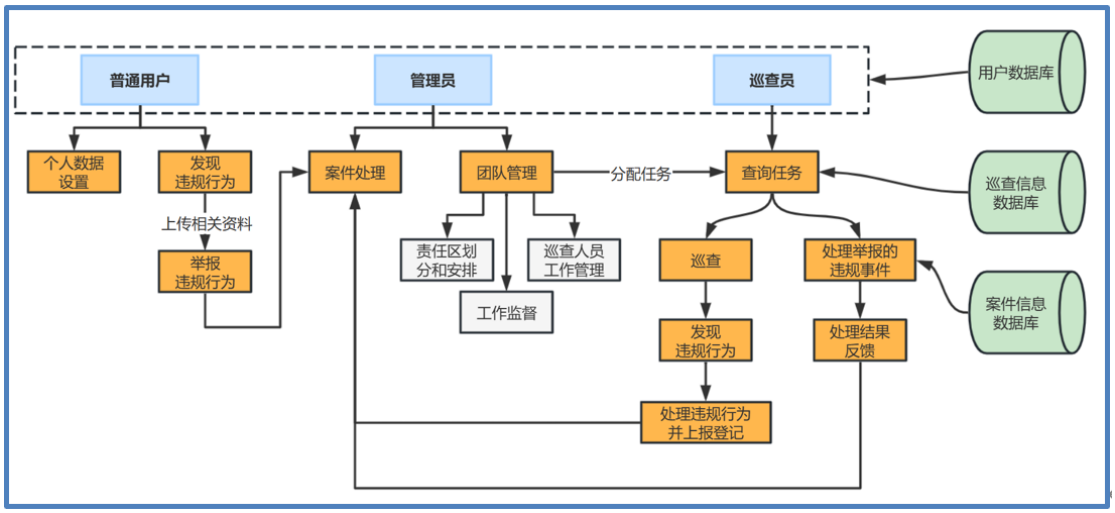



图 2.1 智慧城市安全管理业务流程图

### 2.1.3 功能设计

根据 2.1.2 中各角色的业务流程将系统划分为以下三个功能模块。

**（1）管理员模块**

• 团队管理：能建立自己的巡查团队，并对巡查范围和巡查人员的任务进行安排管理；能邀请用户加入自己的巡查团队。

• 责任区划分和安排：能对负责管理的区域细分成单元网络，并安排给巡查队员进行巡查。

• 巡查人员工作管理：能安排巡查员去责任区的特定单元网络巡查；能安排巡查员去处理居民上报的违规事件；能安排巡查员的工作时间等基础信息。

•巡查人员工作监督：能查看各巡查员的工作状态信息，及其当前所处位置。

• 案件处理：能对居民上报的案件进行立案审查；能对巡查员处理的案件结

果进行审核评估。

**（2）巡查员模块**

• 巡查监督：根据管理员安排的巡查任务进行日常巡查。

• 案件及时处理：对居民上报的且通过管理员立案的案件进行实地处理，并对处理结果进行反馈上报。

**（3）普通用户模块**

• 能便捷高效地上报生活中遇到的违规事件。

• 能管理个人信息。

## 2.2 非功能需求

**（1）系统性能需求**

系统实施的速度应该要高效。通过对用户信息化现状展开深入、细致、全面的调查，从用户的角度出发设计方案。要尽量保证系统能在规定的时间内完成任务。

**（2）系统可视化需求**

为方便居民使用该系统，应当确保系统具备简单易用的人机交互界面，通过设计用户友好型交互方式能在很大程度上提高用户的使用感受。

**（3）系统可使用性需求**

系统设计和实施过程中，应当充分考虑用户的特点以及需求，使系统应用操作方便简洁、学习成本低。

**（4）安全性和稳定性需求**

系统在开发时应当充分考虑整个系统运行的机制和安全策略，保证其具备自我恢复的能力，能保证系统维持稳定、安全的运行。尤其要重视安全性的保障，防止信息泄露，如安装防火墙防止被入侵而系统损坏。

## 2.3 数据需求

### 2.3.1 静态数据

静态数据是指在系统执行期间很长一段时间内不会发生变化的数据。智慧城管系统中的静态数据主要包括：人员基本身份信息，用户角色信息，巡查信息如巡查路线，管理信息如工作时间的安排、管理区域的划分。

### 2.3.2 动态数据

动态数据是在系统应用过程中可能随时间变化而改变的数据。智慧城管系统中的动态数据主要包括：案件信息如案件内容、案件事发地点，居民上报信息等。

### 2.3.3 数据字典

下面列出了数据库设计过程需要用到的主要数据流信息。

• 案件信息

| 名称 | 数据类型 | 默认值 | 约束 |
| --- | --- | --- | --- |
| 案件 ID | INT(20) | | 主键非空，自增长 |
| 案件名称 | VARCHAR(20) | | 非空 |
| 处理团队 ID | INT(10) | | 外键，非空 |
| 案件类型编号 | VARCHAR(5) | | 非空 |
| 事发时间 | DATE | | 非空 |
| 案件状态 | INT(1) | 0 | 非空，取值 0~3 |
| 案件内容描述 | TEXT | | 非空 |
| 事发地址描述 | TEXT | | 非空 |
| 处理结果 | INT(1) | 0 | 非空，取值 0~4 |

• 用户信息

| 名称 | 数据类型 | 默认值 | 约束 |
| --- | --- | --- | --- |
| 用户 ID | INT(10) | | 主键、非空、自增长 |
| 用户角色 | INT(1) | 0 | 非空，取值 0~2 |
| 用户名 | VARCHAR(10) | | 非空 |
| 对于的人员 ID | INT(10) | | 外键 |
| 登陆状态 | INT(1) | 0 | 非空，取值 0~1 |
| 密码 | INT(10) | | 非空 |

• 人员信息

| 名称 | 数据类型 | 默认值 | 约束 |
| --- | --- | --- | --- |
| 身份证号码 | VAERCHAR(20) | | 主键、非空 |
| 真实姓名 | VAECHAR(10) | | 非空 |
| 性别 | VARCHAR(5) | | 取值 male、female |
| 生日日期 | DATE | | |
| 年龄 | INT(5) | | |
| 手机号码 | INT(11) | | |
| 电子邮箱 | VARCHAR(20) | | |

• 团队信息

| 名称 | 数据类型 | 默认值 | 约束 |
| --- | --- | --- | --- |
| 团队 ID | INT(10) | | 主键，非空，自增长 |
| 责任区编号 | VARCHAR(20) | | 外键，非空 |
| 团队名称 | VARCHAR(20) | | 非空 |
| 直接上级团队 ID | INT(10) | | 外键 |
| 团队负责人 ID | INT(10) | | 外键非空 |
| 地址 | TEXT | | |
| 邮箱 | VARCHAR(20) | | |
| 联系电话 | INT(11) | | |

• 上报材料信息

| 名称 | 数据类型 | 默认值 | 约束 |
| --- | --- | --- | --- |
| 材料 ID | INT(10) | | 主键非空自增长 |
| 上传时间 | DATE | | 非空 |
| 案件 ID | INT(20) | | 外键非空 |
| 上传用户 ID | INT(10) | | 外键非空 |
| 文件类型 | VARCHAR(10) | | 非空 |
| 支撑材料文件名 | VARCHAR(20) | | 非空 |
| 支撑材料存储路径 | VARCHAR(50) | | 非空 |

综上完成了智慧城市安全管理系统的需求分析，深入了解了系统的背景知识，明确了系统的核心功能和主要业务流程，并建立了数据字典。

## 2.4 可行性分析

• **技术可行性：**本次智慧城市安全管理系统模块清晰、逻辑简练，编程工作量适中。学生具备一定的编程基础，具备前端开发的能力。通过进一步学习数据库后端开发知识，预期能实现该系统，技术上可行。

• **经济可行性：**该系统服务于政府和民众，通过将流水式的管理过程信息化，能提高决策效率，同时可以有效减少人力沟通成本，且所需数据是政府易调用的，数据成本不高，通过配备稳定的服务器就能维持系统运行，经济上可行。

• **运营可行性：**城市管理信息化能在极大程度上提高管理人员的工作效率、改善民众问题解决效率，具有运营维护价值。系统交互模块是根据用户角色有所区分的，易于使用。系统的实现轻量易于维护，运行上可行。

# 三、概念设计

## 3.1 实体 ER 图设计

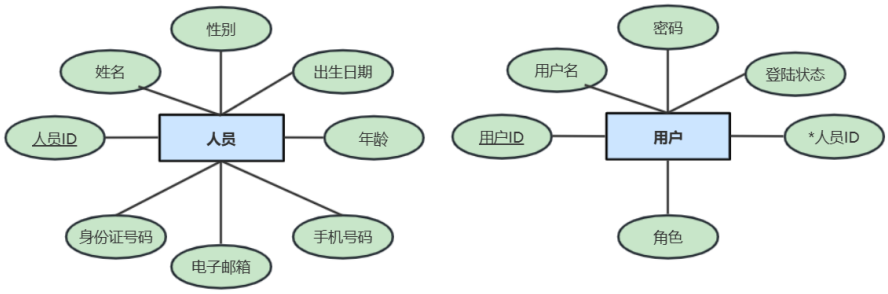
下面对各个实体的内涵和数据项进行说明。

### 3.1.1 人员实体和用户实体

人员实体是指管理区域内具体的每一个人，用户实体是指在该城管系统中具备某种角色的人。人员只描述人的基本特征和信息，用户主要描述人员在系统中扮演的角色。之所以区分开是因为：人员不一定是用户，可能只是系统收录了该人的信息，但他并没有加入系统；人员也可能有多个用户，即它们是 1 对多的关系。

• 人员实体数据项：人员 ID、姓名、性别、出生日期、年龄、身份证号码、电子邮箱、手机号码。

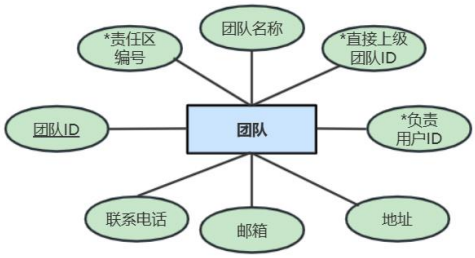
• 用户实体数据项：用户 ID、角色、用户名、密码、登陆状态、人员 ID(实名认证)。其中角色有管理员、巡查员和普通用户三种。




### 3.1.2 团队实体

团队由用户组成，包括各类管理人员和巡查人员，是城市安全管理的基本组织单位，每一个团队都唯一地负责一个责任区内的所有案件。

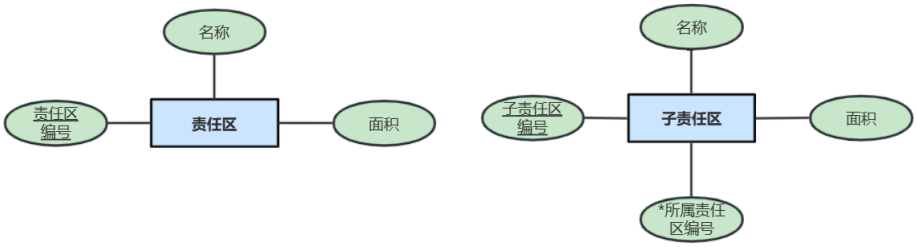
• 团队实体数据项：团队 ID、团队名称、责任区编号、直接上级团队 ID、负责用户 ID、联系电话、邮箱、地址。



### 3.1.3 责任区实体和子责任区实体

责任区是当前团队管辖的区域，子责任区是管理员按照城市行政需求对责任区进行划分得到的面积更小的管理区域，比如某团队“A 市 a 区城管队伍”负责责任区为 A 市 a 区，则其子责任区可按照社区和街道划分。规定子责任区是团队管理的基本单位，即每一个子责任区中必须部署至少一个巡查人员。

- 责任区实体数据项：责任区编号、名称、面积。
- 子责任区实体数据项：子责任区编号、名称、面积、所属责任区编号。

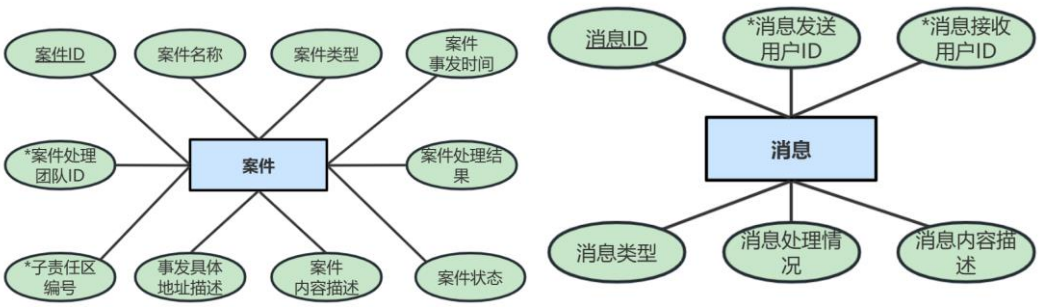



其中，（子）责任区编号是管理员根据行政情况预先划分、设置的，属于管理员的职能之一。

### 3.1.4 案件实体和消息实体

案件实体是指巡查员巡逻时发现的违规事件，或用户上报的并经管理员审核立案的违规事件；消息实体是指普通用户举报违规事件时向管理员发出的信息。管理员处理普通用户上报的消息，若通过审核则立案为案件，之后再派遣巡查员进行调查。

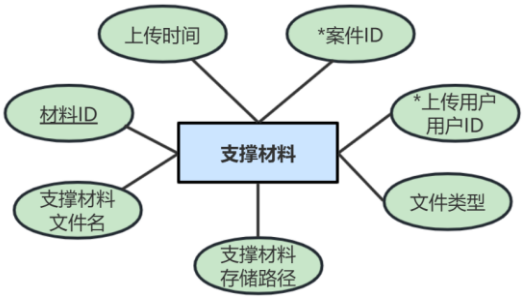
- 案件实体数据项：案件 ID、案件名称、案件类型、案件事发时间、子责任区编号、事发具体地址描述、案件内容描述、案件状态、案件处理结果、案件处理团队 ID。
- 消息实体数据项：消息 ID、消息发送用户 ID、消息接收用户 ID、消息内容描述、消息类型、消息处理情况。



### 3.1.5 支撑材料实体

案件的审查只靠案件实体中提供的案件描述是远远不够的，支撑材料实体记录了案件相关佐证材料如照片、视频、音频、笔录等内容，支撑材料由多部门搜集提供，在本系统中，普通用户和巡查员均能向管理员上报支撑材料。

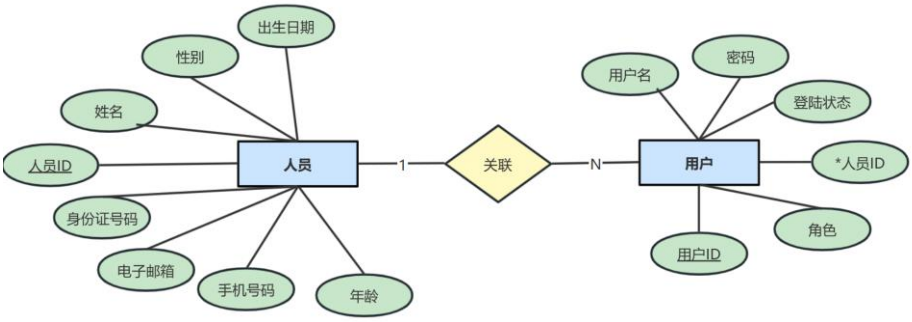
• 支撑材料数据项：材料 ID、案件 ID、上传用户的 ID、上传时间、支撑材料文件名、文件类型、支撑材料存储路径。




## 3.2 实体联系及局部 ER 图设计

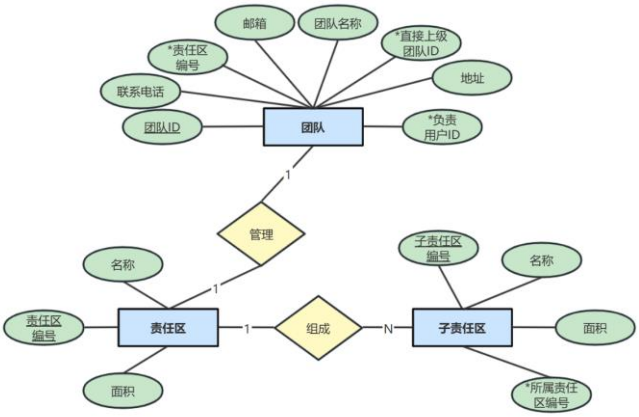
### 3.2.1 人员和用户的联系

人员和用户之间具有一对多的关系，即 1 个人员可能存在多个用户账号，也可能不存在任何用户账号（未加入系统）。ER 联系图如下所示。



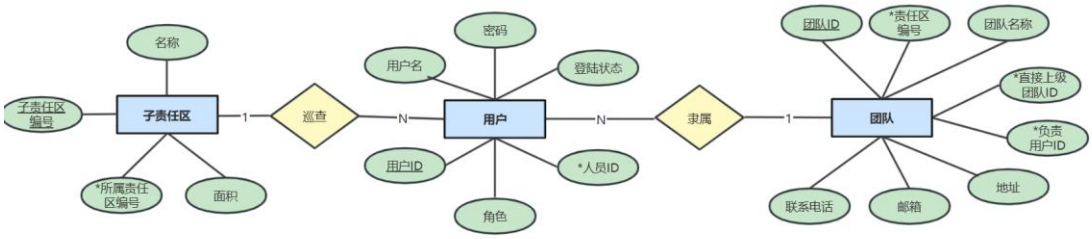

### 3.2.2 团队、责任区和子责任区的联系

一个责任区由若干子责任区组成，每一个责任区由一个管理团队唯一负责管理，下面是其 ER 联系图。



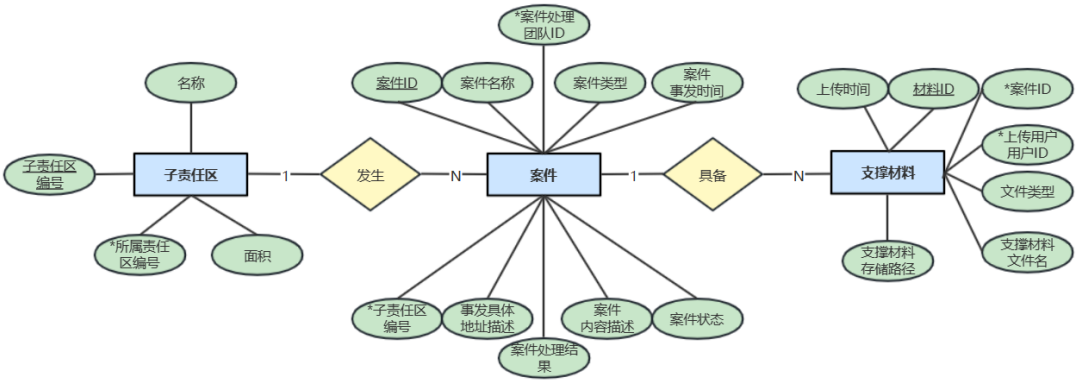

### 3.2.3 子责任区、用户和团队的联系

一个团队中含有若干不同角色的用户，用户和团队具有多对一的关系。每一个子责任区都至少安排了一个巡查员，所以子责任区和用户之间具有一对多的关系。



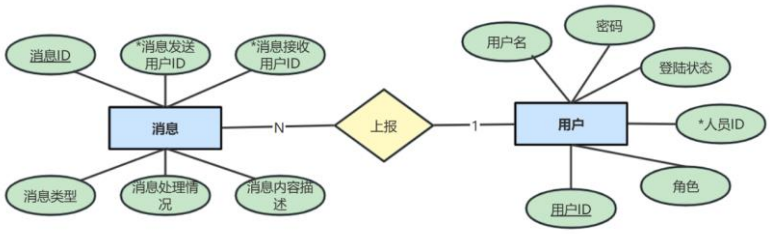

### 3.2.4 案件、材料信息、子责任区的联系

子责任区中可能发生多个案件，每个案件又可具备多份支撑材料。所以子责任区和案件、案件和支撑材料之间均为一对多的关系。




### 3.3.5 用户和消息的联系

每一个用户都可以上报多条消息，所以用户和消息是一对多的关系。




## 3.3 全局 ER 图设计

基于上述分析，总结出智慧城市安全管理系统的全局 ER 图。包含的实体包括：用户、人员、团队、消息、案件、支撑材料、责任区、子责任区等 8 个实体。各实体间存在 1：1 或 1：n 的关系，具体如图 3.1 所示。

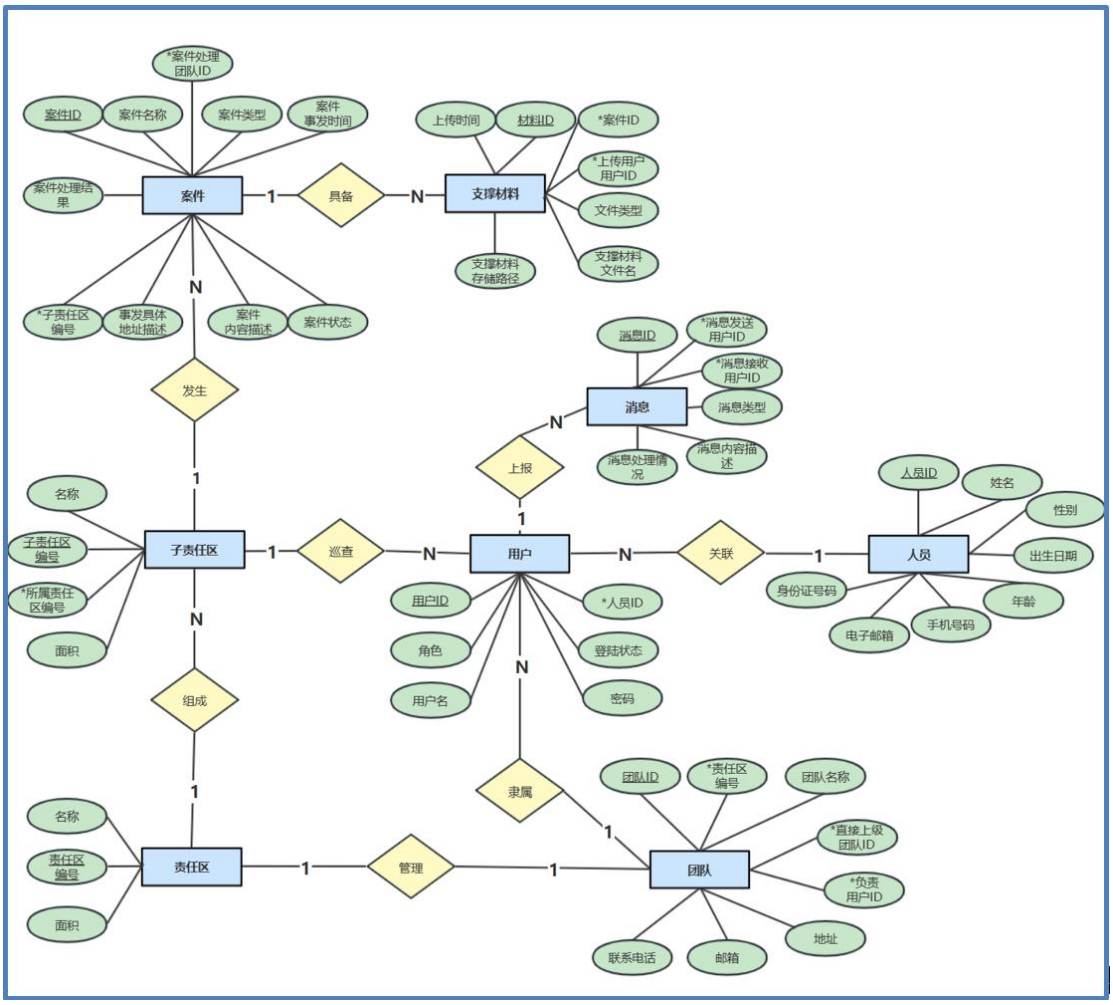



图 3.1 全局 ER 图

# 四、逻辑设计

逻辑结构设计的任务是将概念模型转换成特定 DBMS 所支持的数据模型的过程。下面给将主要从关系模型的设计和表结构的设计展示逻辑设计过程。

## 4.1 关系模型设计

在概念设计时已经保证了各关系满足 1NF，即关系中各属性的域都是原子的。接下来需要根据概念设计的 E-R 图，转化得到关系模型，并对关系模型进行审视，删除冗余属性、合并冗余关系、并保证满足 3NF。下面是转换并处理后的最终关系模型。

1、案件信息表：

Case(CaseID,CaseName,TeamID,CaseType,CaseTime,CaseStatement,CaseContent,CaseLocation,CaseResult)

2、用户信息表：

User(UserID,Role,UserName,PersonID,LoginState,Password)

3、人员信息表：

Person(PersonID,PersonName,Sex,Birthday,Age,PhoneNumber,Email)

4、团队信息表：

Team(TeamID,DutyGridID,TeamName,SuperiorTeamID,headUserID,TeamPos,TeamEmail,TeamPhoneNumber)

5、用户所属表：

UserBelonging(UserID,TeamID)

6、责任区表：

DutyGrid(DutyGridID,DutyGridName,Area)

7、子责任区表：

SubGrid(SubGridID,SubGridName,DutyGridID,SubArea)

8、支撑材料表：

EvidenceFile(FileID,UpLoadTime,CaseID,UserID,FileType,FileName,FilePath)

9、消息上报表：

Message(MessageID,SrcUserID,DstUserID,MessageType,MessageContent,isReaded)

10、用户巡查表

Patrol(UserID,SubGridID)

分析各个关系，发现其中不存在大量冗余信息、插入异常和删除异常的问题。且各关系中每一个非主属性都不传递依赖于关系的任何码，即上述关系模型是满足 3NF 的。

## 4.2 表结构设计

### 4.2.1 案件信息表 Case

| 属性 | 说明 | 数据类型 | 默认值 | 约束 |
|---|---|---|---|---|
| **CaseID** | 案件 ID | INT(20) | | 主键非空，自增长 |
| **CaseName** | 案件名称 | VARCHAR(20) | | 非空 |
| **TeamID** | 处理团队 ID | INT(10) | | 外键，非空 |
| **CaseType** | 案件类型编号 | VARCHAR(5) | | 非空 |
| **CaseTime** | 事发时间 | DATE | | 非空 |
| **CaseStatement** | 案件状态 | INT(1) | 0 | 非空，取值 0~3 |
| **CaseContent** | 案件内容描述 | TEXT | | 非空 |
| **CaseLocation** | 事发地址描述 | TEXT | | 非空 |
| **CaseResult** | 处理结果 | INT(1) | 0 | 非空，取值 0~4 |

案件状态 CaseStatement 有 4 种状态：“0”表示未立案，“1”表示已经立案，“2”表示处理中，“3”表示终止。处理结果 CaseResult 有 5 种状态：“0”表示未完成、“1”代表结案、“2”代表立案作废、“3”代表不予立案、“4”代表不予受理。事发地址描述方式按照特定格式约束：社区-街道-具体地址（经纬度信息）。

### 4.2.2 用户信息表 User

| 属性 | 说明 | 数据类型 | 默认值 | 约束 |
|---|---|---|---|---|
| **UserID** | 用户 ID | INT(10) | | 主键、非空、自增长 |
| **Role** | 用户角色 | INT(1) | 0 | 非空，取值 0~2 |
| **UserName** | 用户名 | VARCHAR(10) | | 非空 |
| **PersonID** | 对于的人员 ID | INT(10) | | 外键 |
| **LoginState** | 登陆状态 | INT(1) | 0 | 非空，取值 0~1 |
| **Password** | 密码 | INT(10) | | 非空 |

Role 取值 0~2，依次代表管理员、巡查员和普通用户。LoginState 取值 0~1，依次代表不在线、在线。PeopleID 是用户关联的人员的 ID，如果非空则说明用户完成了实名认证。

### 4.2.3 人员信息表 Person

| 属性 | 说明 | 数据类型 | 默认值 | 约束 |
|---|---|---|---|---|
| **PersonID** | 身份证号码 | VAERCHAR(20) | | 主键、非空 |
| **PersonName** | 真实姓名 | VAECHAR(10) | | 非空 |
| **Sex** | 性别 | VARCHAR(6) | | 取值”male”、”female” |

| | | | | |
|---|---|---|---|---|
| **Birthday** | 生日日期 | DATE | | |
| **Age** | 年龄 | INT(5) | | |
| **PhoneNumber** | 手机号码 | INT(11) | | |
| **Email** | 电子邮箱 | VARCHAR(20) | | |

因为身份证号码就是人员的唯一标识，故主码设置为身份证号码。

### 4.2.4 团队信息表 Team

| 属性 | 说明 | 数据类型 | 默认值 | 约束 |
|---|---|---|---|---|
| **TeamID** | 团队 ID | INT(10) | | 主键，非空，自增长 |
| **DutyGridID** | 责任区编号 | VARCHAR(20) | | 外键，非空 |
| **TeamName** | 团队名称 | VARCHAR(20) | | 非空 |
| **SuperiorTeamID** | 直接上级团队 ID | INT(10) | | 外键 |
| **headUserID** | 团队负责人 ID | INT(10) | | 外键非空 |
| **TeamPos** | 地址 | TEXT | | |
| **TeamEmail** | 邮箱 | VARCHAR(20) | | |
| **TeamPhoneNumber** | 联系电话 | INT(11) | | |

### 4.2.5 用户所属表 UserBelonging

| 属性 | 说明 | 数据类型 | 默认值 | 约束 |
|---|---|---|---|---|
| **UserID** | 用户 ID | INT(10) | | 主键，外键，非空 |
| **TeamID** | 团队 ID | INT(10) | | 主键，外键，非空 |

该表描述了用户和团队的所属关系，主码由用户 ID 和团队 ID 组合而成。之所以单独设置用户所属表，是考虑到用户因其角色不同而不一定都隶属于某一团队，如果将团队 ID 作为外键加入用户信息表中，则普通用户不需要该属性。因此，设置用户所使表更为合适。

### 4.2.6 责任区表 DutyGrid

| 属性 | 说明 | 数据类型 | 默认值 | 约束 |
|---|---|---|---|---|
| **DutyGridID** | 责任区编号 | VARCHAR(20) | | 主键非空 |
| **DutyGridName** | 责任区名称 | VARCHAR(20) | | 非空 |
| **Area** | 面积 | FLOAT | | |

### 4.2.7 子责任区表 SubGrid

| 属性 | 说明 | 数据类型 | 默认值 | 约束 |
|---|---|---|---|---|
| **SubGridID** | 子责任区编号 | VARCHAR(20) | | 主键非空 |
| **SubGridName** | 子责任区名称 | VARCHAR(20) | | 非空 |
| **DutyGridID** | 所属责任区编号 | VARCHAR(20) | | 外键非空 |
| **SubArea** | 面积 | FLOAT | | |

### 4.2.8 支撑材料表 EvidenceFile

| 属性 | 说明 | 数据类型 | 默认值 | 约束 |
|---|---|---|---|---|
| **FileID** | 材料 ID | INT(10) | | 主键非空自增长 |
| **UpLoadTime** | 上传时间 | DATE | | 非空 |
| **CaseID** | 案件 ID | INT(20) | | 外键非空 |
| **UserID** | 上传用户 ID | INT(10) | | 外键非空 |
| **FileType** | 文件类型 | VARCHAR(10) | | 非空 |
| **FileName** | 支撑材料文件名 | VARCHAR(20) | | 非空 |
| **FilePath** | 支撑材料存储路径 | VARCHAR(50) | | 非空 |

### 4.2.9 消息上报表 Message

| 属性 | 说明 | 数据类型 | 默认值 | 约束 |
|---|---|---|---|---|
| **MessageID** | 消息 ID | INT(10) | | 主键非空自增长 |
| **SrcUserID** | 消息发送用户 ID | INT(10) | | 外键非空 |
| **DstUserID** | 消息接收用户 ID | INT(10) | | 外键非空 |
| **MessageType** | 消息类型 | VARCHAR(10) | | 非空 |
| **MessageContent** | 消息内容描述 | TEXT | | 非空 |
| **isReaded** | 消息处理情况 | INT(1) | 0 | 取值 0、1 |

### 4.2.10、用户巡查表 Patrol

| 属性 | 说明 | 数据类型 | 默认值 | 约束 |
|---|---|---|---|---|
| **UserID** | 巡查员 ID | INT(10) | | 主键、外键、非空 |
| **SubGridID** | 负责巡查的子责任区域 ID | VARCHAR(20) | | 主键、外键、非空 |

该表的设置初衷同 4.2.5 的情况，即不是所有用户都是巡查员，所以单独设置用户巡查表关联巡查员和负责的区域。

# 五、物理设计

物理设计是根据所选择的关系数据库的特点对逻辑模型进行存储结构设计的过程。物理设计中除了要确定数据的存储方式，还要建立起数据库的索引结构，虽然索引结构不是数据库的必要部分，但灵活的索引结构能显著提高数据库查询的效率，所以也是物理设计中重要的一环。该部分将重点介绍物理设计中数据库索引的建立。

## 5.1 索引设计方法概述

### 5.1.1 整理查询条件

设计索引的目的主要是为了加快查询，所以，设计索引的第一步是整理需要用到的查询条件，也就是我们会在 where 子句、join 连接条件中使用的字段。一般来说会整理程序中除了 insert 语句之外的所有 SQL 语句，按不同的表分别整理出每张表上的查询条件。也可以根据对业务的理解添加一些暂时还没有使用到的查询条件。对索引的设计一般会逐表进行，所以按数据表收集查询条件可以方便后面步骤的执行。

### 5.1.2 分析字段的可选择性

字段的可选择性指的就是字段的值的区分度，例如一张表中保存了用户的手机号、性别、姓名、年龄这几个字段，且一个手机号只能注册一个用户。在这种情况下，像手机号这种唯一的字段就是可选择性最高的一种情况；而年龄虽然有几十种可能，但是区分度就没有手机号那么大了；性别这样的字段则只有几种可能，所以可选择性最差。所以可选择性从高到低排列就是：手机号 > 年龄 > 性别。通常，我们会把可选择性高的字段放到前面，可选择性低的字段放在后面，如果可选择性非常低，一般不会把这样的字段放到索引里。根据字段的可选择性对索引字段进行确定是十分重要的，好的索引字段起到事半功倍的效果。

### 5.1.3 合并查询条件

虽然索引可以加快查询的效率，但是索引越多就会导致插入和更新数据的成本变高，因为索引是分开存储的，所有数据的插入和更新操作都要对相关的索引进行修改。所以设计索引时还需要控制索引的数量，不能盲目地增加索引。

一般我们会根据最左匹配原则来合并查询条件，尽可能让不同的查询条件使用同一个索引。例如有两个查询条件 where a = 1 and b = 1 和 where b = 1，那么我们就可以创建一个索引 idx_eg(b, a)来同时服务两个查询条件。

同时，因为范围条件会终止使用索引中后续的字段，所以对于使用范围条件查询的字段也会尽可能放在索引的后面。

### 5.1.4 全覆盖索引技术的引入

最后，需要考虑是否需要使用全覆盖索引，因为全覆盖索引没有回表的开销，效率会更高。所以一般我们会在回表成本特别高的情况下考虑是否使用全覆盖索引，例如根据索引字段筛选后的结果需要返回其他字段或者使用其他字段做进一步筛选的情况。

下面就根据上述思路指引，结合课程中介绍的索引技术，实现对索引的初步设计。

## 5.2 数据库索引设计

### 5.2.1 用户信息和人员信息索引表设计

用户信息的 UserID 作为主键会高频地出现再查询语句和连接语句中，考虑在主键 UserID 上建立唯一性索引。同理，在人员信息的主键 PersonID 上也建立唯一性索引。

### 5.2.2 团队信息索引表设计

因为经常要对属于同一团队的人员进行统一操作，即 TeamID 会常出现在 group by 语句后，而 group by 实质会先排序后分组，所以初步设计使用松散索引扫描实现 group by，对 TeamID 字段建立聚集索引。这样做能明显提高处于同一团队中各用户信息的检索效率。

### 5.2.3 责任区和子责任区索引表设计

考虑到巡查员会高频的调用有关子责任区查询的视图，且责任区和子责任区信息不会经常变化，所以对其建立是有意义的。考虑对其主键 DutyGridID 和 SubGridID 分别建立唯一性索引。且子责任区主键 SubGridID 和巡查员 ID 的组合值是唯一，故建立组合索引。

### 5.2.4 其它索引表设计

案件信息表、消息表等，其中含有许多动态数据，表内容会随时间变化发生较大变化，如果在其上建立索引，则对索引的维护代价可能较大，故不建立索引。另外，对于数据量较小但高频使用的用户所属表和用户巡查表，可以采用 B+树的方法存储，从而提高查询效率。

# 六、系统实现

## 6.1 技术栈及环境说明

本次数据库管理平台采用前后端分离的方法实现，前端采用 Vue.js 框架实现了系统的 Web 界面，后端基于 Django 和 Sqlite 实现了数据库系统的搭建和管理。主要的技术列举并说明如下。

**Vue.js 2**

Vue 是一套用于构建用户界面的渐进式框架。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与现代化的工具链以及各种支持类库结合使用时，Vue 也完全能够为复杂的单页应用提供驱动

**Vuex**

Vuex 是 Vue.js 的状态管理模式和库，用于集中管理应用程序的所有组件的状态。通过 Vuex，我们可以实现全局状态的管理和共享，方便不同组件之间的通信和数据同步

**Vue-Router**

Vue Router 是 Vue.js 的官方路由器，用于实现单页面应用程序中的路由功能。利用 Vue Router 可以轻松地定义路由规则，并实现页面之间的无刷新切换和导航，提供更流畅的用户体验。

**Axios**

Axios 是一个基于 Promise 的 HTTP 客户端，用于在前端与后端之间进行数据交互。它支持浏览器和 Node.js 环境，提供了简洁的 API 接口，使得前端与后端的数据通信变得更加便捷和灵活。

**Element-UI**

Element UI 是一套基于 Vue.js 的桌面端 UI 组件库，提供了丰富的组件和样式，以及易用的交互方式。它具有美观、简洁的设计风格，并且具备良好的扩展性和可定制性。我选择了 Element UI 来构建数据库管理平台的用户界面，

以实现简洁明了的 UI 风格。

### Django

Django 是一个强大且高效的 Python Web 框架，用于快速开发可扩展的 Web 应用程序。它采用了 MTV（模型-模板-视图）的设计模式，提供了丰富的功能和组件，使得后端开发变得更加简单和高效。我选择 Django 作为后端框架，以便构建可靠的数据库管理平台。

### Django REST framework

Django REST framework 是基于 Django 的一个功能强大且灵活的工具包，用于构建 RESTful 风格的 Web 服务。它提供了一套简单且一致的 API 设计和开发工具，使得后端的 API 开发变得更加快速和可靠。通过使用 Django REST framework，我们可以轻松地构建和暴露出数据库管理。

### Less

Leaner Style Sheets（简称为 Less）是一种动态样式表语言，它扩展了 CSS（层叠样式表）的功能并引入了更多的特性。Less 使得前端开发人员能够更加方便地编写可维护和可重用的样式代码。它提供了变量、嵌套规则、混合（Mixin）等功能，使样式的编写和管理更加灵活和高效。使用 Less 可以在开发过程中轻松地管理和扩展数据库管理平台的样式。

下面是学生运行项目的环境说明。

| 系统环境说明 | |
|---|---|
| **操作系统** | Windows 10 |
| **语言** | 前端采用 html、css、javascript，后端采用 python3.10 |
| **前端框架** | Vue.js 2.0 |
| **后端框架** | Django 4.2.1 |
| **开发工具** | 前端 VScode 后端 Pycharm |
| **测试工具** | Google 浏览器 |

部分开发工具的版本信息如下

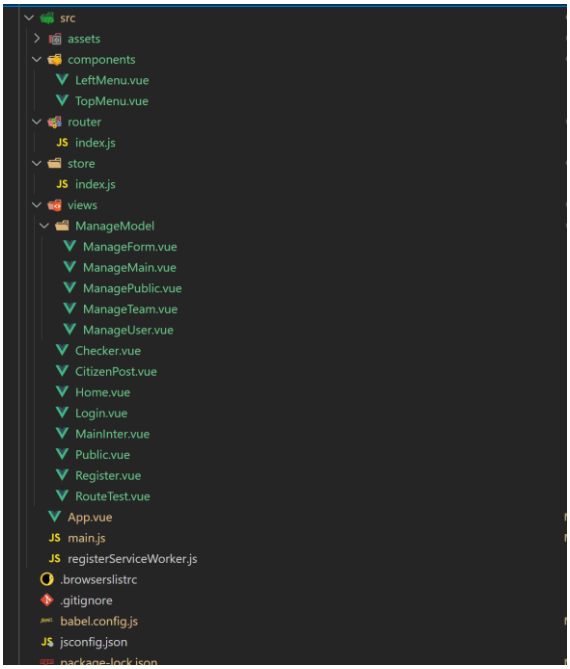
```
// vue.js 中的包管理文件
{
  "name": "city-security",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build"
  },
  "dependencies": {
    "axios": "^1.4.0",
    "core-js": "^3.8.3",
    "element-ui": "^2.15.13",
    "font-awesome": "^4.7.0",
    "less": "^4.1.3",
    "less-loader": "6.0",
    "register-service-worker": "^1.7.2",
    "vue": "^2.6.14",
    "vue-router": "^3.6.5"
  },
  "devDependencies": {
    "@vue/cli-plugin-babel": "~5.0.0",
    "@vue/cli-plugin-pwa": "~5.0.0",
    "@vue/cli-service": "~5.0.0",
    "babel-plugin-component": "^1.1.1",
    "vue-template-compiler": "^2.6.14"
  }
}
```

## 6.2 前后端实现概述

### 6.2.1 前端架构

案件信息通过用户“来信”的方式上报，反馈信息通过管理员（代表最终决策）“回信”的方式实现。

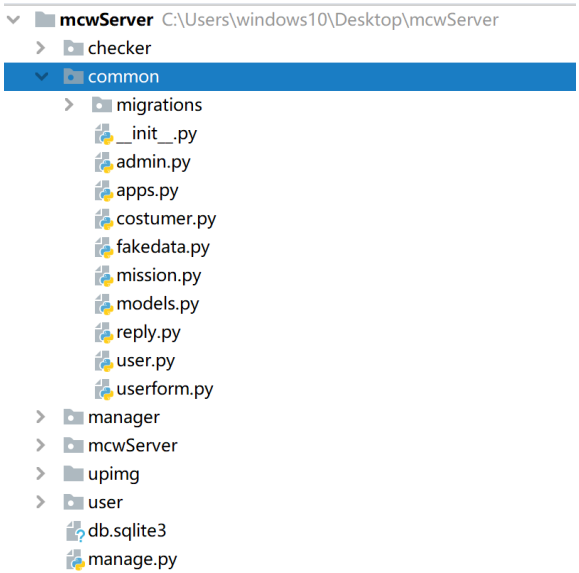
前端的界面设计思路：顶部是主导航栏，采用子路由的方式实现导航栏下的主要内容，从而减少页面渲染时的重复渲染开销。前端的数据请求采用 axios 执行同步请求（防止时序错），公共数据采用 vuex 管理。前端的项目架构如下：




assets 中存放相关图片，components 中存放共用 vue 子组件，router 中存放路由配置，store 中存放 vuex 公共管理的数据，views 中存放所有组成界面的 vue 组件。

### 6.2.2 后端架构

后端的 django 项目架构如下所示。其中 common 文件夹下定义了共用的表结构，mcwServer 中的 urls.py 文件存放后端的主路由，根据操作表对象的不同，会转向对应的子 urls 表。




### 6.2.3 前后端联调

前后端分离项目的实现重在前后端的有效分工和协同，前端负责渲染界面，

并接受用户的交互请求，通过 js 定义的逻辑向后端发起数据请求。后端负责管理数据库的所有表单和数据，是数据库系统的重要组成部分吗，它接受前端发起的网络请求，http 代理服务将接通后端项目，后端按照预设的逻辑处理 http 请求，并将数据以 Json 数据包格式返回给网页。

具体流程如图 6.1 所示，概括如下。

- ✧ **定义 API 接口**：在后端开发阶段，使用 Django 和 Django REST framework 来定义和实现数据库管理平台的 API 接口。通过创建 URL 路由和视图函数，可以确定前端通过何种方式与后端进行通信，并明确 API 的请求格式和响应数据的结构。
- ✧ **搭建后端服务**：使用 Django 提供的开发工具和服务器，学生搭建了后端服务，使得数据库管理平台的 API 可以在本地或远程服务器上运行。后端服务处理前端发送的请求，执行相应的逻辑操作，并返回处理结果给前端。
- ✧ **开发前端界面**：在前端开发阶段，我使用 Vue.js 和 Element UI 来构建用户界面，实现数据库管理平台的各种交互和展示功能。并利用 Vue Router 来定义前端的路由规则，以便在不同页面之间进行无刷新的切换和导航。
- ✧ **发起异步请求**：前端使用 Axios 来发起异步请求与后端 API 进行通信。通过 Axios 的 API 接口，可以发送 GET、POST、PUT、DELETE 等类型的请求，并传递相应的数据。这样，前端可以与后端进行数据交互，获取所需的数据或提交更新。
- ✧ **处理响应结果**：在前端接收到后端的响应后，我使用 Promise 和同步的方式（async 和 await）来处理返回的数据。根据 API 的设计，可以对不同的响应结果进行相应的处理，例如更新页面内容、显示错误信息或进行下一步的操作。

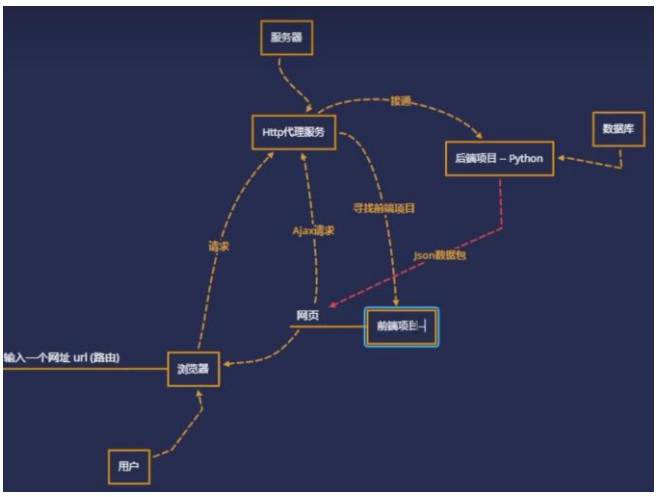



图 6.1 前后端分离的信息交互结构

## 6.3 权限管理

基于需求分析的说明可知（见图 2.1），本系统需要面向 3 类不同角色的用户使用，分别为：普通用户、巡查员、管理员。他们的系统操作界面必然有不同之处，需要通过登录时进行权限判断以进行管理。

本系统基于 token 机制实现权限管理， Token 机制是一种常见的权限管理方式，它通过在客户端和服务器之间传递令牌（token）来验证和授权用户访问权限。其原理和作用机制简述如下：

- ✧ **用户认证**：用户在登录时提供用户名和密码，服务器验证用户的身份凭证是否正确。如果验证通过，服务器会生成一个令牌并返回给客户端。
- ✧ **token 生成**：服务器生成一个具有唯一标识的令牌，并将用户的身份信息、权限等相关数据进行加密和签名。生成的令牌通常包括用户 ID、角色、有效期等信息。
- ✧ **token 传递**：客户端在后续的请求中将令牌通过 HTTP 的授权头或其他方式传递给服务器。常见的方式是在请求头中添加"Authorization"字段，并将令牌放在"Bearer"后面，形如："Bearer <token>"。
- ✧ **token 验证**：服务器接收到请求后，会解析和验证令牌的有效性和完整性。这包括检查令牌的签名、有效期、用户角色等信息。如果令牌验证通过，服务器会根据用户的权限执行相应的操作，否则返回相应的错误信息。
- ✧ **权限控制**：服务器根据令牌中包含的用户角色或其他权限信息，对请求的操作进行权限验证。这可以包括验证用户是否具有访问特定资源的权限、执行特定操作的权限等。如果权限验证不通过，服务器会返回相应的权限错误信息。
- ✧ **token 更新**：为了增强安全性并防止 token 被滥用，token 通常具有一定的有效期。当 token 过期时，客户端可以通过特定的方式（例如刷新 token）获取新的 token，以延长用户的会话时间。

接下来将依次介绍各用户的界面和功能。

## 6.4 登录注册

登录是进入系统的必要动作，如果没有登录而直接访问系统内部 url，会基于 token 机制重定向到登录界面，这样也保证了系统的安全性。

登录和注册界面设计上简洁，如图 6.2 和 6.3 所示。登录注册的功能也十分

简单，注册将用户的注册信息写入 auth_user 表中（django 自带，用于管理账户），该表中含有诸多属性，学生主要使用了 user_name、is_superuser、password。

一般用户的受制于其权限，登录后不能看到系统的“全貌”，而仅有一名用户在 auth_user 表中的 is_superuser 属性设置为 true，表示能同时看到所有功能模块，这样做是出于前期系统实现和后期调试方便考虑的。


图 6.2 登录界面

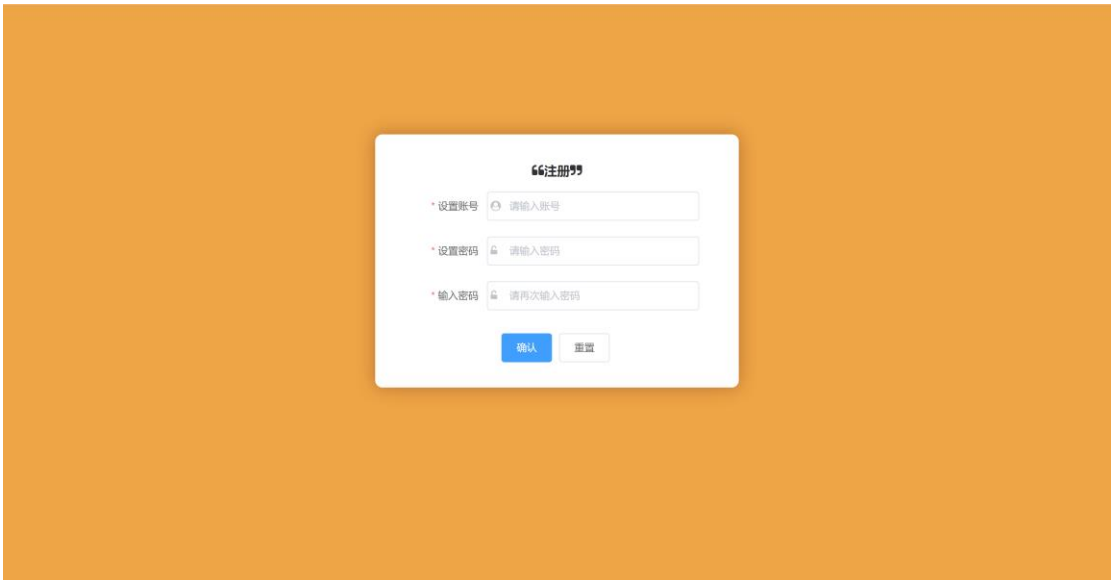



图 6.3 注册界面

## 6.5 普通用户界面

普通用户能看到三个模块，展示在顶部导航栏中，分别是“个人中心”、“信息公开”、“我要写信”。下面依次介绍各模块。

- **个人中心**

如图 6.4 所示，左侧是个人信息，右侧是个人信箱（即举报和求助信息）。左侧的个人信息包括姓名、电话、邮箱等。点击“修改个人信息”会弹出修改框，修改框中默认填写着当前值，如图 6.5 所示。

右侧的个人信息按照时间顺序陈列了用户上报过的所有信件。每一条信件信息展示了标题、是否公开、内容、执行进度、发布时间及相关交互按键。

“查看反馈”的作用是当巡查员完成任务且管理员提交反馈后，支持用户查看处理结构的按钮，如图 6.6 所示。“删除”按钮则直接删除这条上报信息（上报信息只在上报者处删除，管理者不应该具备删除用户上报信息的权限）。

顶部能通过搜索栏根据标题进行关键词检索。后端通过 filter 过滤器返回对应 json 数据包。


图 6.4 个人中心界面

图 6.5 修改个人信息


图 6.6 查看反馈信息

## 信息公开

信息公开的界面如图 6.7 所示。左侧是“回信选登”，展示了管理员选登的处理来信和回信。点击某一行，能展开之，从而看到具体内容，如图 6.8 所示。

右侧上方统计了 4 项基本来信和回信数据。

右侧下方是信访要问栏目，提供相关信息的跳转链接。


图 6.7 信息公开界面


图 6.8 回信选登的具体内容

## 我要写信

该模块提供用户上报的信箱，如图 6.9 所示。包括必填项：访信目的、标题、内容、详细地址、是否公开选项，以及可选项：图片资料。

最后还有提交和重置两个按钮。

表单提交时会自动进行正则检测，如果没有满足要求会标红显示，如图 6.10 所示。此外，是否公开选项是指是否公开发布者姓名。

提交前也会有确认窗口提示。


图 6.9 我要写信界面

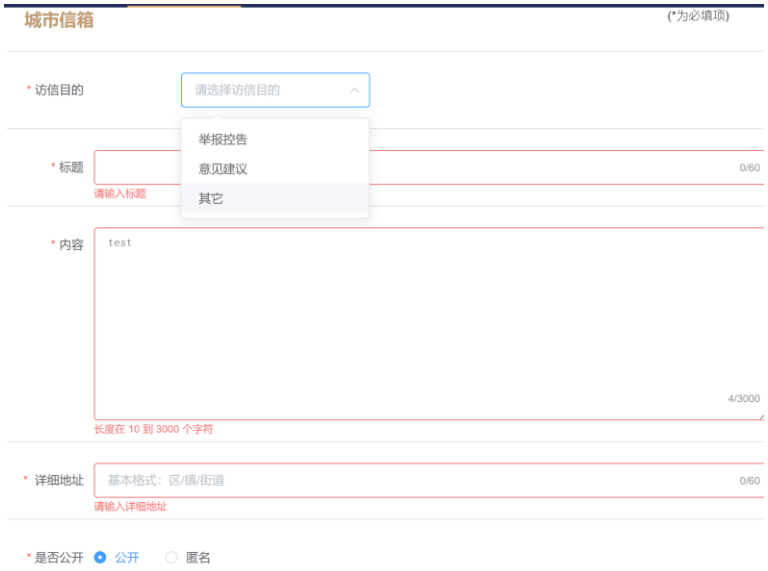

城市信箱
(*为必填项)
* 访信目的
请选择访信目的
举报控告
意见建议
其它
* 标题
0/60
请输入标题
* 内容
test
4/3000
长度在 10 到 3000 个字符
* 详细地址
基本格式：区/镇/街道
0/60
请输入详细地址
* 是否公开
公开
匿名

图 6.10 正则检测提示

## 6.6 管理员界面

管理员界面包括“个人中心”、“信息公开”和“管理系统”三个模块，其中前两者与普通用户并无差异。管理系统模块时管理员管理后台的界面，下面介绍管理系统模块的内容。

管理系统模块有一个左侧子导航栏，包括“用户信息管理”、“事务管理”、“信息公开管理”三部分。

- **用户信息管理**

如图 6.11 所示，用户信息管理的右侧包括所有用户的所有基本信息，采用分页展示（每页十条），底部的分页器可以选择跳页，顶部的新增按钮可以新增用户数据，顶部右侧的搜索栏可以根据姓名检索用户信息。

对于该用户表格，每一行展示一条用户数据，包括姓名、权限、性别、年龄、电话、邮箱、生日、地址等用户信息，最右一列是操作按钮，包括编辑和删除两种功能。

下面说明各交互按钮的使用。点击“新增”，会弹出新增的弹窗，如图 6.12 所示。点击“编辑”图标，会弹出编辑的弹窗，其布局与新增弹窗一致，但表单中已经存放好相应行的数据，可直接修改，如图 6.13 所示。

最后，点击“删除”会弹窗消息提示框，以防止误删，如图 6.14 所示。


图 6.11 用户信息管理


图 6.12 用户信息新增


图 6.13 用户信息修改


图 6.14 用户信息删除

### 信息公开管理

信息公开管理模块支持管理员管理信息公开模块发布的来信选登的内容，界面如图 6.15 所示。该模块展示的数据是已经执行完成的案件（信息）且支持公开的。点击选登，信件信息则能加入信息公开模块，反之点击取消则能摘下该来信信息。


图 6.15 信息公开管理

## 事务管理

事务管理负责管理所有上报信息，如图 6.16 所示，其中工作进度和操作包括以下步骤。

- ✧ 来信未处理——>等待管理员分配任务
- ✧ 已分配任务——>等待巡查员接受任务
- ✧ 巡查处理中——>巡查员实地执行
- ✧ 处理完成——>管理员提交最终回信
- ✧ 完成

上述步骤不仅在事务管理中有所体现，在用户个人中心的我的信箱中也能通过进度条追踪案件进度。

图 6.16 事务管理界面

点击“分配任务”，如图 6.17 所示。因为巡查员 id 是唯一的，所以采用基于 id 的方式分配，任务说明是对任务的补充，能在被分配任务的巡查员的工作界面中看到。


图 6.17 分配任务

点击“提交反馈”，为用户提供最终回信，能在用户个人界面看到对应的回信，如图 6.18 所示。


图 6.18 提交反馈

## 6.7 巡查员界面

巡查员的工作模块如图 6.19 所示，操作按钮包括提交反馈和接受任务。接受任务后，巡查员进入实地执行，完成后通过提交反馈按钮总结工作情况。点击每一行都可以展开查看具体信息。


图 6.19 我的工作界面

至此，完成了对各个主要界面的布局及功能介绍。

# 七、系统使用说明

✧ **项目部署：前后端分别部署。**

前端，在 vue 项目工作目录下打开 cmd 窗口（也可通过集成环境打开），输入“npm run serve”，即会默认运行在 localhost 的 8080 端口。显示如下即打开成功。

```
App running at:
- Local:   http://localhost:8080/
- Network: http://192.168.10.101:8080/
```

后端，进入到 manage.py 所在目录，在 python 环境中启动 django 项目的指令为“python manage.py runserver 8088”。若显示如下则说明启动成功。（8088

端口是学生设计接口时固定的，若已被占用请先释放 8088 端口）

```
PS C:\Users\windows10\Desktop\mcwServer> python manage.py runserver 8088
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
June 23, 2023 - 10:09:04
Django version 4.2.1, using settings 'mcwServer.settings'
Starting development server at http://127.0.0.1:8088/
Quit the server with CTRL-BREAK.
```

✧ **登入**

进入系统时可以用以下四个账号（密码均为 8888）

- user，普通用户权限
- worker，巡查员权限
- admin，管理员权限
- superuser，超级管理员，能看到所有界面，能同时执行 user、worker、admin 三个账号。

✧ **登出**

直接点击顶部导航栏的登出按钮即可。

# 八、系统测试与分析

## 8.1 前后端联调测试

先后端联调是前后端分离项目中重要的一环，学生采用 Google 的网络检测功能监察前端请求和后端 json 数据包的返回情况，如图 8.1 所示。

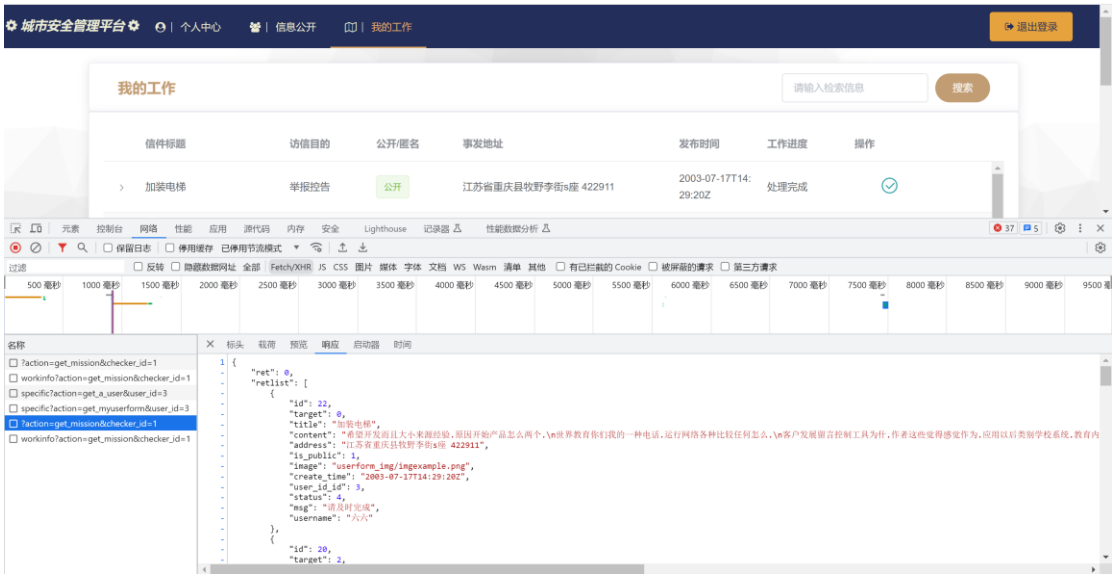



图 8.1 网络测试

## 8.2 数据库中表的增删查改

对数据库数据是否准确操控也是测试的重要环节。学生采用 SQLiteStudio 数据库系统检查前端请求是否被正确执行，如图 8.2 所示。

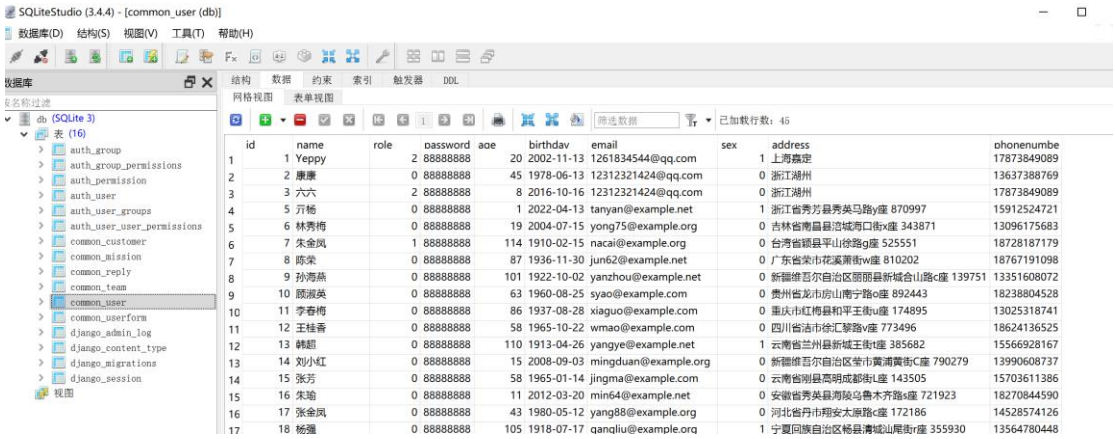



图 8.2 后端数据库管理系统

## 8.3 关键逻辑测试

主要的逻辑体现在一个案件从上报到完成的全过程，其执行情况是否在各个界面都是保持同步的，即数据的准确性、唯一性和实时性是否得到了保证。其逻辑过程如下：

- 来信未处理时等待管理员分配任务
- 已分配任务后等待巡查员接受任务
- 巡查处理中巡查员实地执行
- 处理完成后管理员提交最终回信

通过超级用户权限执行上述流程，发现准确无误，基本说明逻辑正确。

# 九、总结与展望

## 9.1 数据库系统设计

本次大作业研究了“城市计算”中智慧城市管理的问题。通过本次数据库系统设计大作业，我在实践过程中对数据库设计的各流程的作用、重点、难点有了进一步的理解和掌握，不仅提高了自身的问题分析和解决能力，同时也巩固了所学的数据库课程知识，收获颇丰!

数据库设计主要是构造出优化的数据库逻辑模式和物理结构，并在此基础上实现数据库及其引用系统，目标是要做到根据用户需求有针对性地建立一个能有效存储和管理数据的数据库系统。要实现该目标，需要经历需求分析、概念结构设计、逻辑结构设计、物理结构设计及系统实现等重要过程。在上述各阶段探索和设计工作中，我都有不同的感悟和收获。

首先对于需求分析，通过文献资料搜集和相关政策查询，我理清了城市管理的具体流程，较好地把握了传统城市管理的常见问题和多方主体的核心需求。在此基础上，我确定了本次智慧城市管理系统的服务主体是民众、管理层和监察层，以及核心功能为：违规事件处理结果的上报和查询；普通用户举报周围发生的违规事件；城管系统的团队管理；城管责任区划分和安排；巡查人员状态监控。同时分析了数据流、建立了数据字典，便于后续设计。

接下来，通过对用户需求进行综合、归纳和抽象，构建了一个独立于具体数据库管理系统的概念模型。确定了实体及其属性，以及实体间的联系，并绘制了全局 E-R 图。随后，将概念结构转换成为了 mysql 数据库管理系统支持的数据模型，并进一步对模型进行优化，保证了各关系符合第三范式，并尽可能的删除了冗余属性，合并了冗余关系。最后，对物理结构设计的索引设计进行了分析和初步设计，确定了基本表的索引结构组织。

## 9.2 数据库系统实现

在本次课程设计的实现过程中，学生顺利实现了一个功能基本完整的数据库管理平台，基本满足了上学期的需求设计，收获颇丰！

首先，我的知识面得到了很大程度的完善，这是我第一次独立完成一个数据库管理系统的前后端，基本是从 0 开始接触各种技术，在眼花缭乱的知识海洋里寻觅，过程确实坎坷，中间也熬了不少夜。但是，付出必有回报，如果不是课程的激励作用，我很难在短时间内掌握全栈工程的来龙去脉。我理解了前后端分离

项目的架构，http 协议的具体应用，对后端表结构的设计有了更深入和切实的理解，也加深了查询等语句的使用。尤其是对索引结构的设计，让我意识到数据库的设计规划对系统性能起着何其关键的作用。

虽然，本次项目只能算是一个规模不大的历练，我在数据库系统设计和实现上大概还只是初入门槛，但迈开腿的这一步，也是最艰难的一步已经落下了，之后我会一步一个脚印，继续努力，在学习中耕耘与收获~

# 十、参考资料

下面是学生在系统实现中主要使用的参考和学习资料。

[1] Vue.js 框架指导文档：https://v2.cn.vuejs.org/v2/guide/

[2] Django 开发指导：https://www.byhy.net/tut/webdev/django/09/

[3] Element-ui 指导文档：https://element.eleme.cn/

[4] 前后端基础知识学习：https://developer.mozilla.org/zh-CN/docs/Web/