灾情上报系统（妖六）详细设计说明书

文件状态：正式发布
文件标识：
当前版本：

作者：

姓名：陈治中
学号：2018011474
团队：泗妖六

完成日期：2021/10/21

版本更新信息：

1 引言

1.1 使用人员：

张石一、陈治中、王伟杰、杨鹏飞、刘子昂、胡志国、孙博君、朱康晟

1.2 编写目的：

编写详细设计说明书的目的是为了使开发人员在完成概要设计说明书的基础上，完成概要设计的各项功能规定，为程序员写出实际的程序代码提供依据。它是软件设计阶段所有任务和所有相关人员所需的参考资料。预计的读者为全体系统开发人员以及后续对系统进行拓展和维护的人员。

1.3 背景：

本项目将完成一个多源灾情上报系统。多源社会灾情数据通过接口输入到多源灾情数据管理服务系统平台，进行一体化编码；然后将编码后的数据输入到虚拟化管理系统。数据来源包括业务报送数据、泛在感知数据、舆情感知数据和承载体基础数据等，将来可以扩展更多数据来源。该系统依据数据的时效性进行数据存储，当前时刻为，设置合适的时间窗口 ，对内的数据进行存储，并且随着时间的延续，实现新数据存储旧数据淘汰；最后，当外界发出数据请求时，从管理系统中获取目标数据，并通过接口发送给请求方。需要设计一个通用的架构以适应众多异源异质异构的数据请求（例如多层架构中的每层低耦合，通用化接口 - 例如前后端分离，可以适用需求的变化）。实时处理技术的研究，实现灾情数据统一管理和高效合理利用，构建多源社会灾情数据管理服务系统。
a. 待开发软件系统的名称：灾情信息上报系统
b. 本项目的任务提出者：泗妖六团队
c. 本项目的开发者：泗妖六团队
d. 本项目的用户：系统管理人员，灾情信息上传人员，灾情信息查询人员

1.4 定义与缩写

术语：SQL Server
解释：
系统服务器所使用的数据库管理系统（DBMS）

术语：SQL
解释：
结构化查询语言（Structured Query Language），一种用于访问查询数据库的语言。

术语：事务流
解释：
数据进入模块后可能有多种路径进行处理

术语：主键
解释：
数据库表中的关键域。值互不相同

术语：外键
解释：
数据库表中与其他表主键关联的域

术语：ROLLBACK
解释：
数据库的错误恢复机制

术语：系统
解释：
若未特别指出，均指本系统

术语：VB
解释：
Visual Basic

术语：mud-rock flow
解释：
泥石流，指在山区或者其他沟谷深壑，地形险峻的地区，因为暴雨、雪暴或其他自然灾害引发，为泥土、石头等与大量的水混合后，受重力作用，沿着斜坡或山沟滑动的现象

术语：fire as a disaster
解释：
这里特指由于物体自然燃烧造成的火灾的天灾。

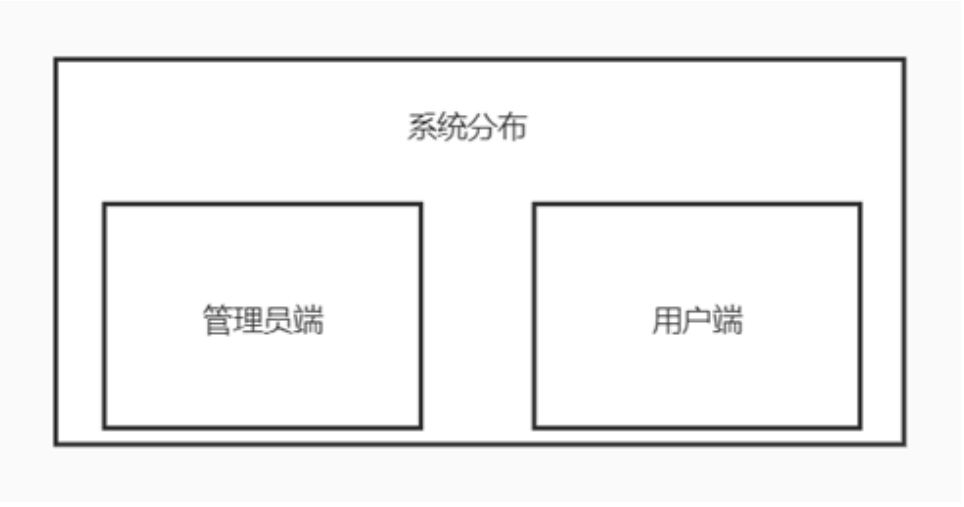
术语：flood
解释：
洪水，一种自然灾害，指河流、湖泊、海洋所含的水水体上涨，超过常规水位的水流现象。洪水常威胁沿河、湖滨、近海地区的安全，甚至造成淹没灾害。

1.5 参考资料

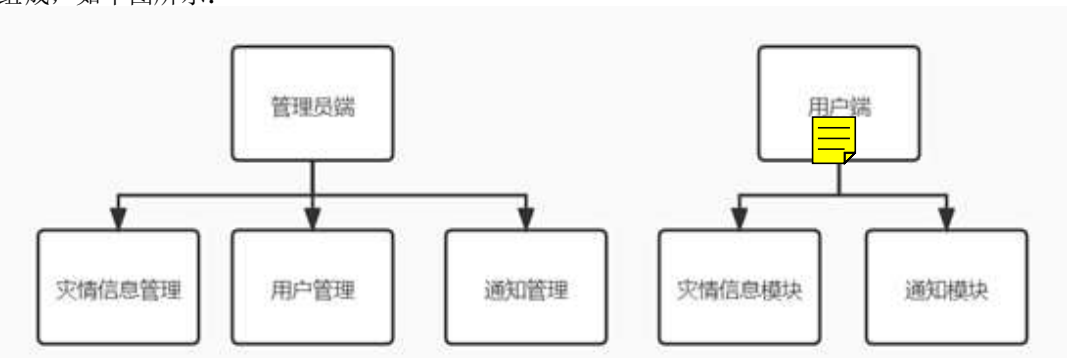
- 

2 程序系统的结构：

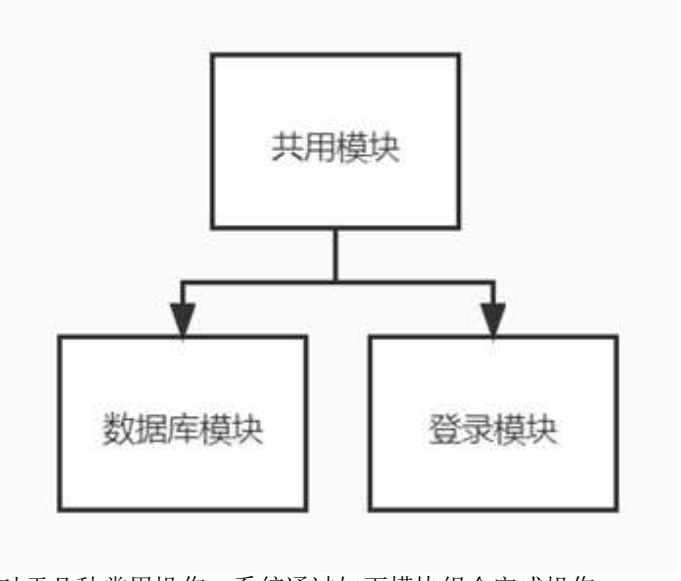
系统主要分为部分，管理员端和用户端，如下图所示：



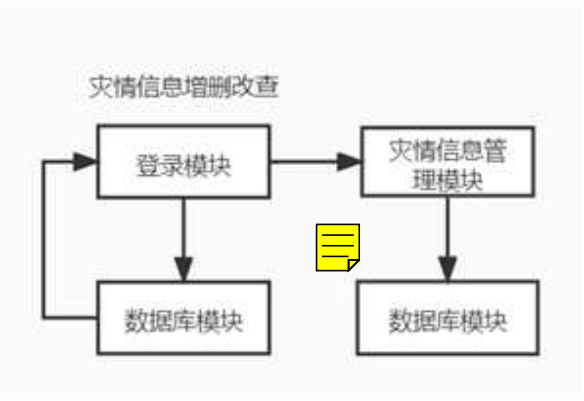
管理员端由灾情信息管理、用户管理、以及通知管理三个大模块组成，用户端由灾情信息模块、通知模块组成，如下图所示：



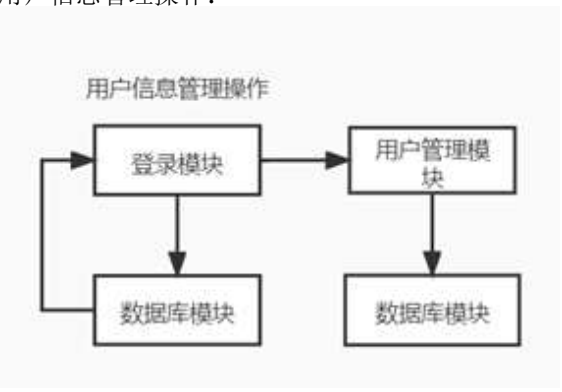
另外，设有数据库模块用于数据的调出与存入，设有登录模块用于管理员和用户的识别与登入：



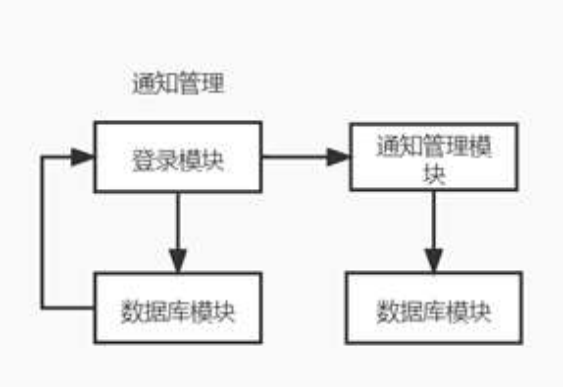
对于几种常用操作，系统通过如下模块组合完成操作：
灾情信息增删改查：



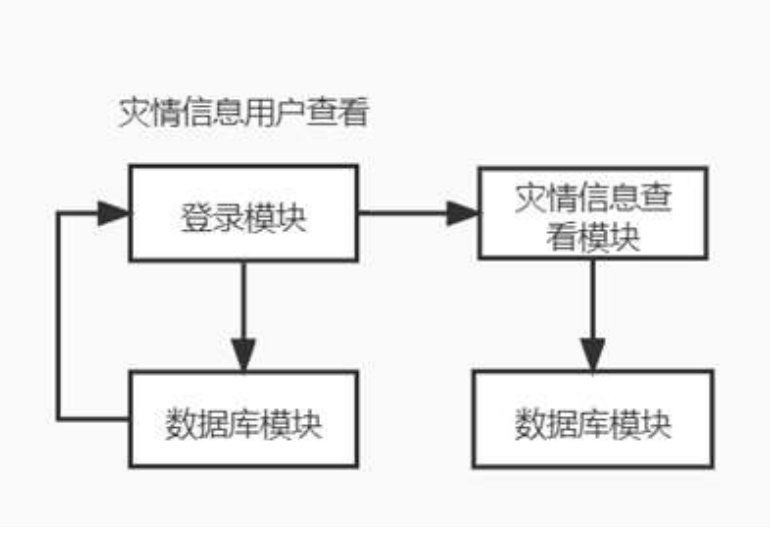
用户信息管理操作：



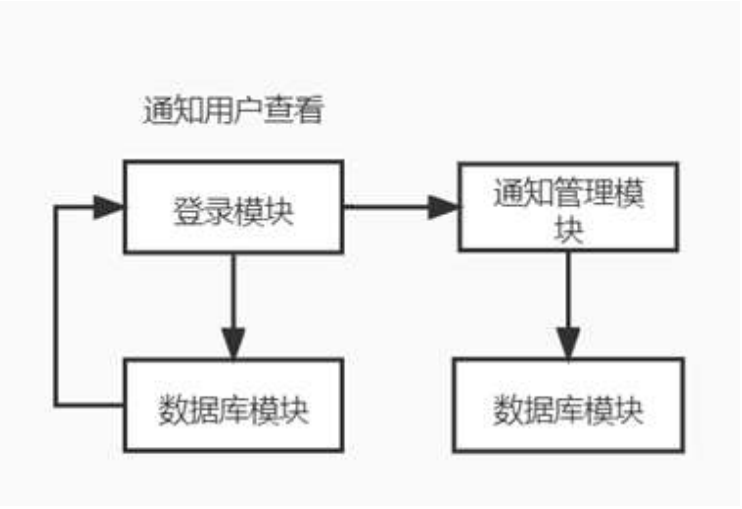
通知管理操作：



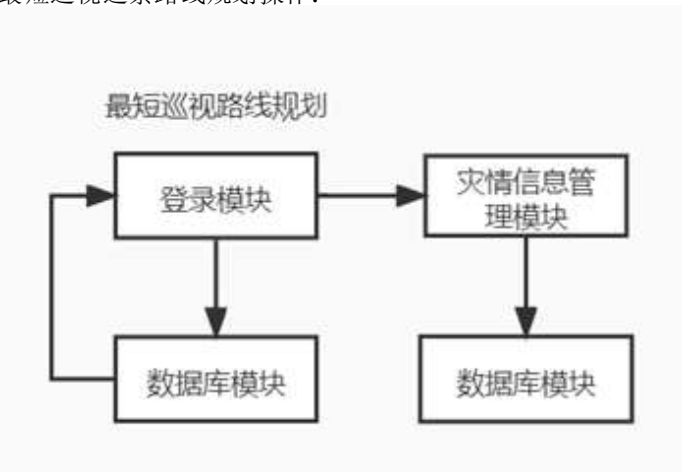
灾情信息用户查看：



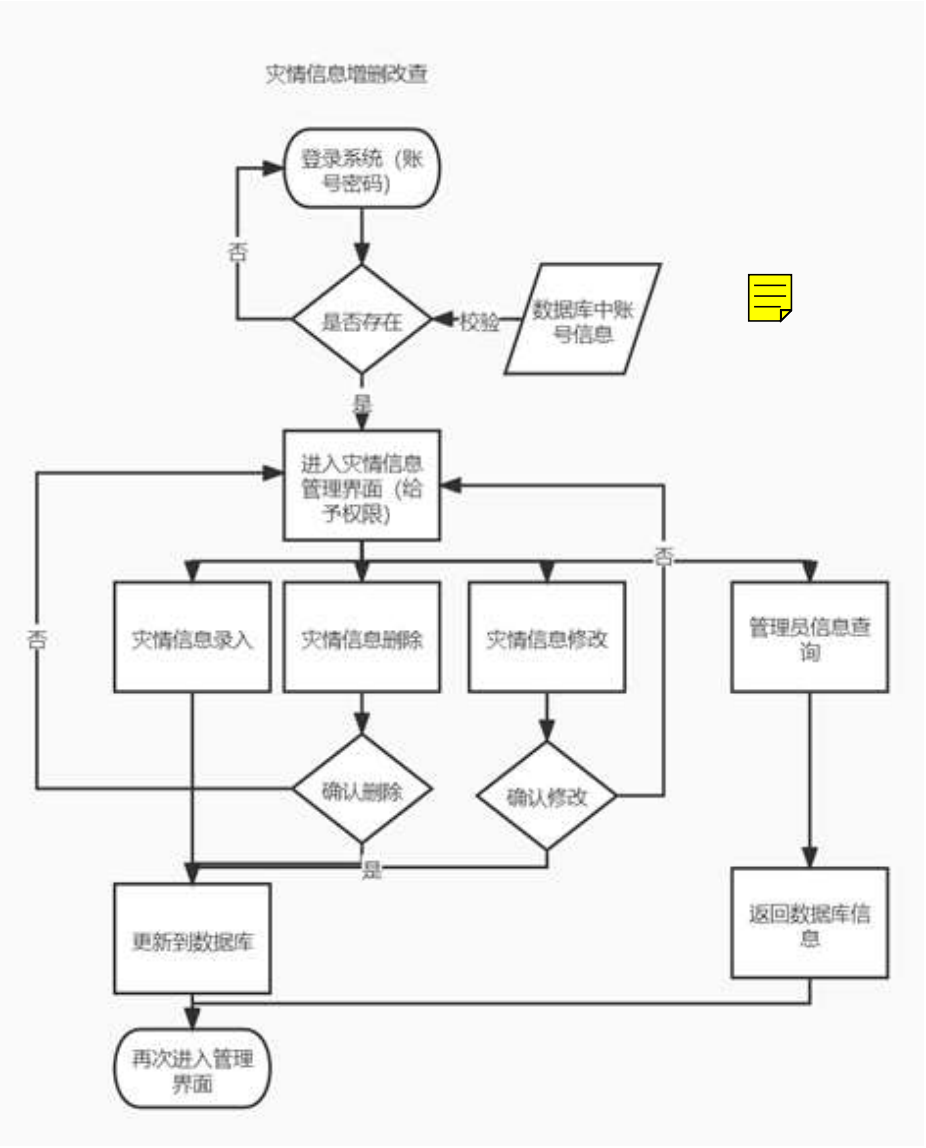
通知用户查看：



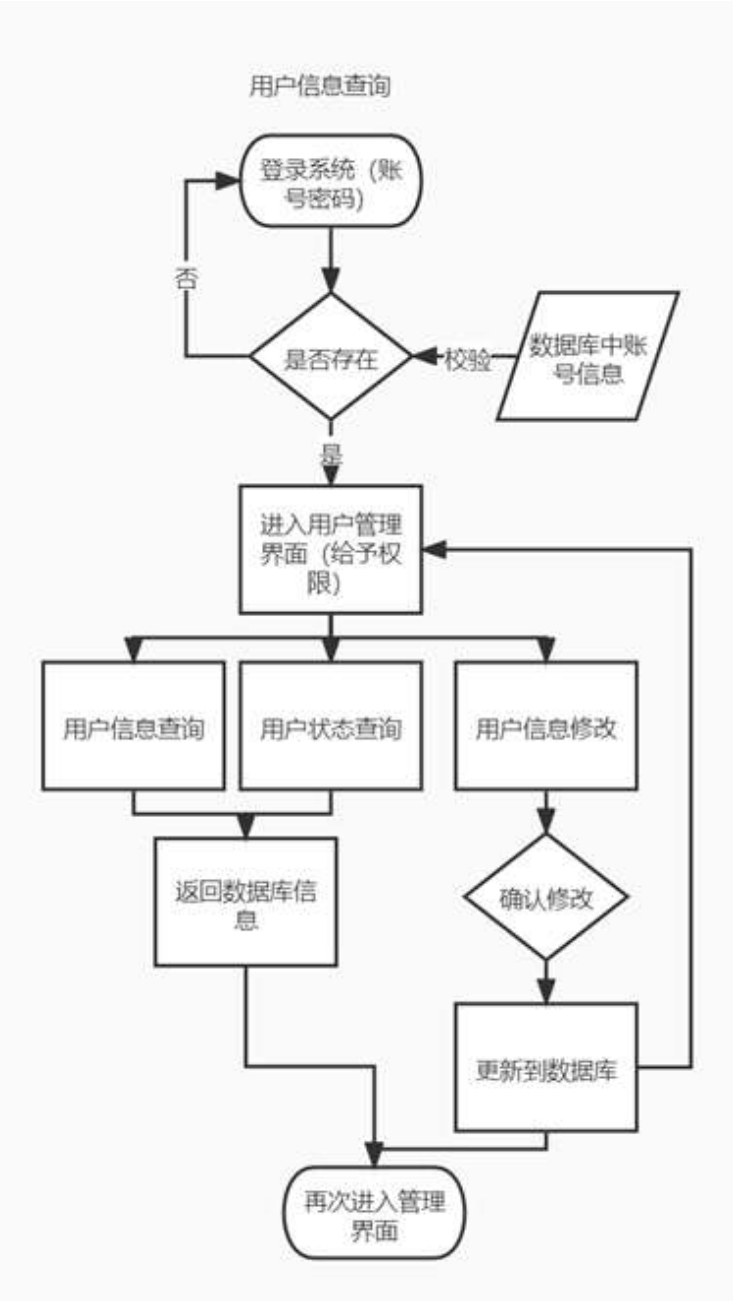
最短巡视巡察路线规划操作：



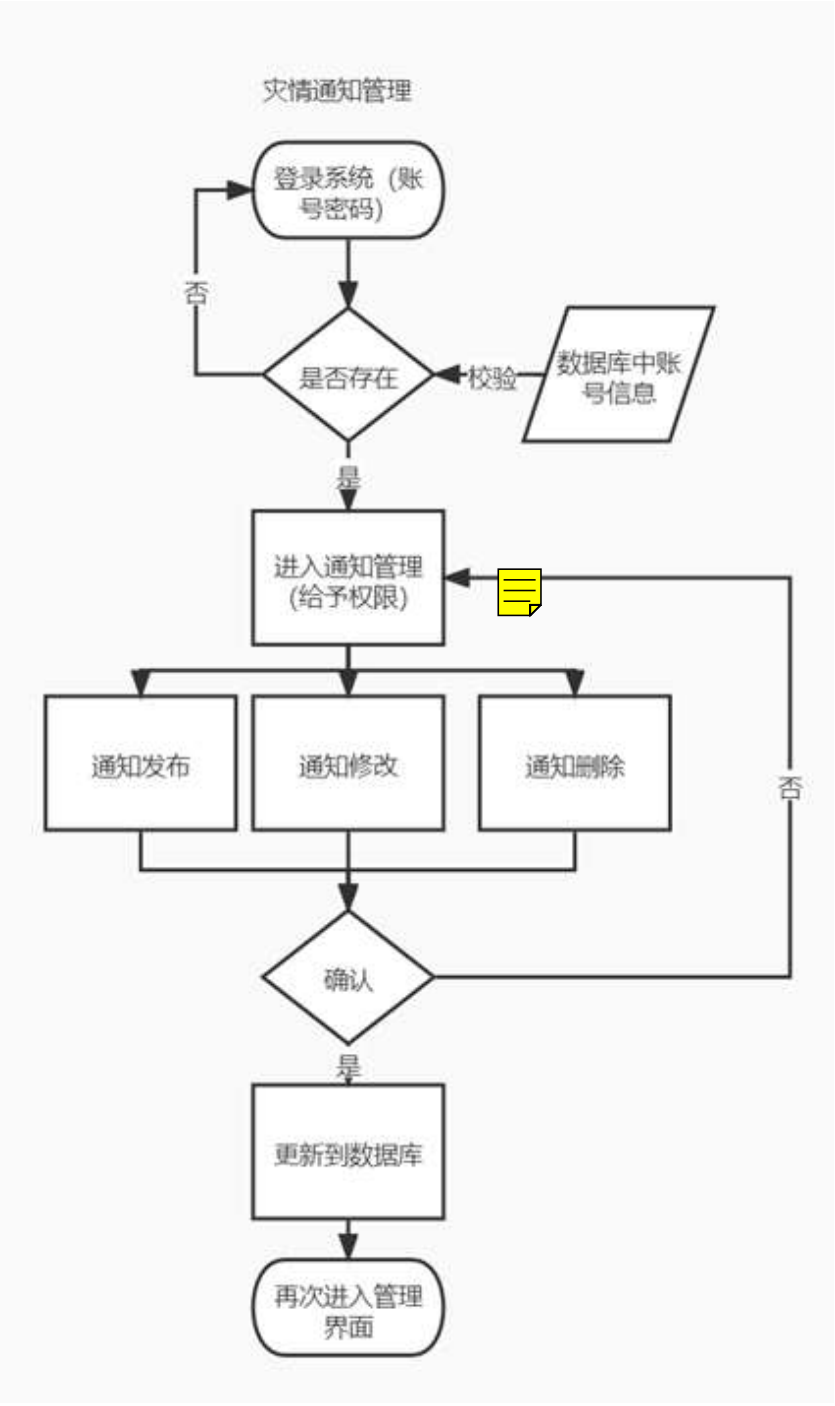
补充：该操作目的是为救护人员、巡视巡察人员规划一条可以将灾区内所有重点地区全部到达并尽量避免多余路线的救助巡查路线。
经过上述描述已经大致描述清楚在几种常用操作中，系统中的模块之前是如何组合完成操作目的，下面将对每种操作进行细致流程地描述：
灾情信息增删改查：


灾情信息增删改查
登录系统（账号密码）
否
是否存在
校验
数据库中账号信息
是
进入灾情信息管理界面（给予权限）
否
否
灾情信息录入
灾情信息删除
灾情信息修改
管理员信息查询
确认删除
确认修改
是
更新到数据库
返回数据库信息
再次进入管理界面

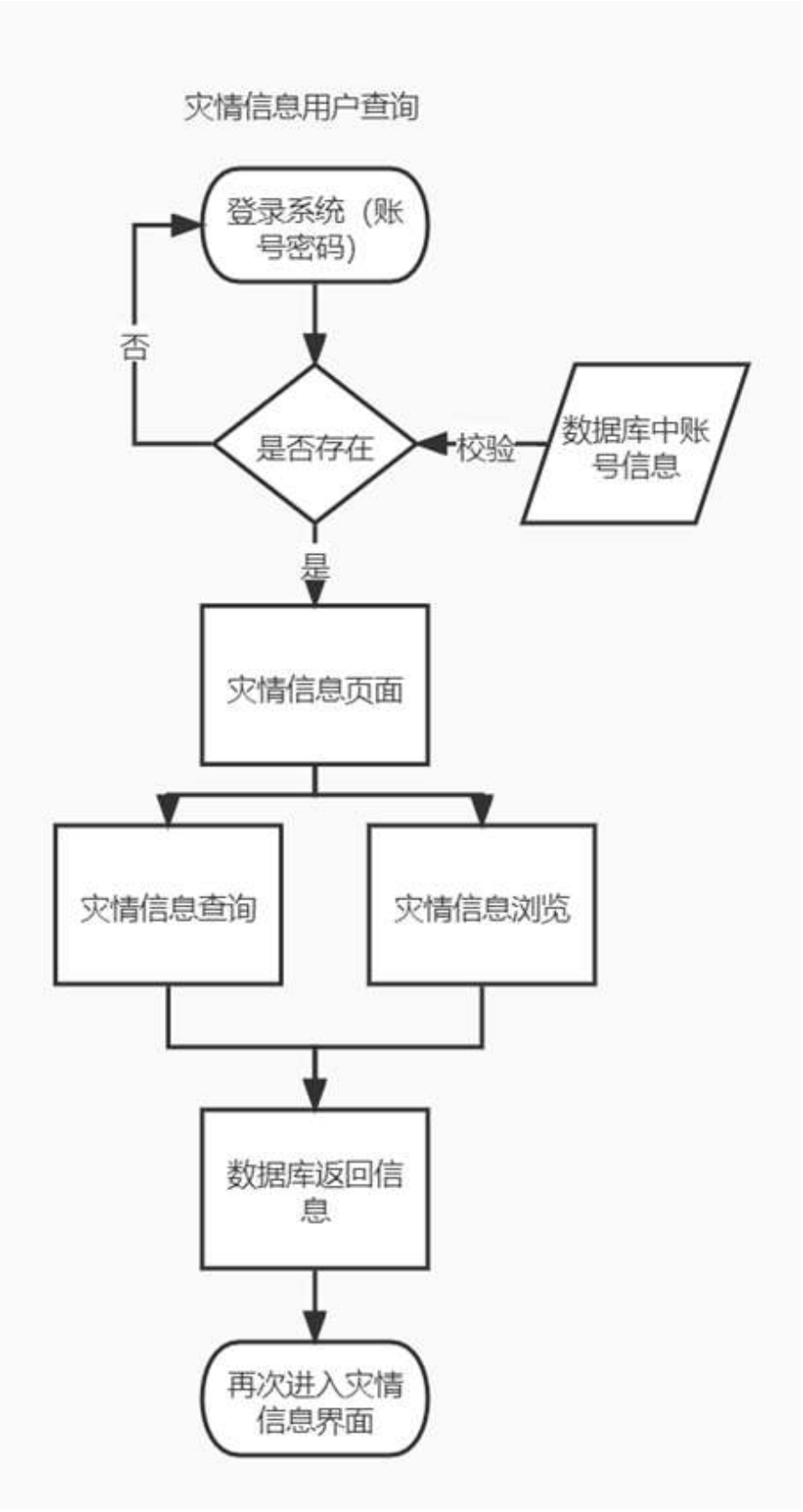
用户信息管理：


用户信息查询
登录系统（账号密码）
否
是否存在
校验
数据库中账号信息
是
进入用户管理界面（给予权限）
用户信息查询
用户状态查询
用户信息修改
返回数据库信息
确认修改
更新到数据库
再次进入管理界面

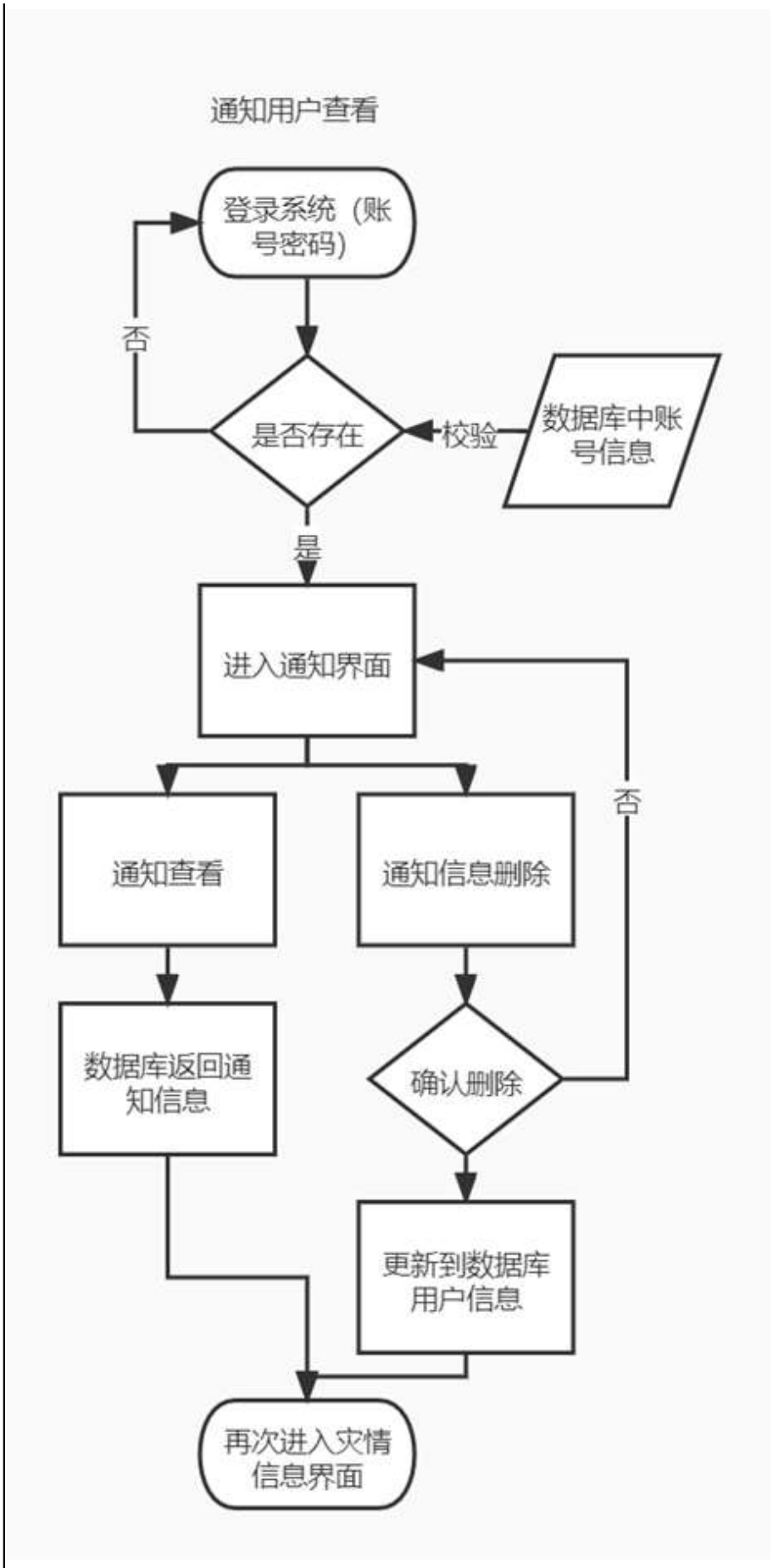
通知管理：


灾情通知管理
登录系统（账号密码）
否
是否存在
校验
数据库中账号信息
是
进入通知管理（给予权限）
通知发布
通知修改
通知删除
否
确认
是
更新到数据库
再次进入管理界面

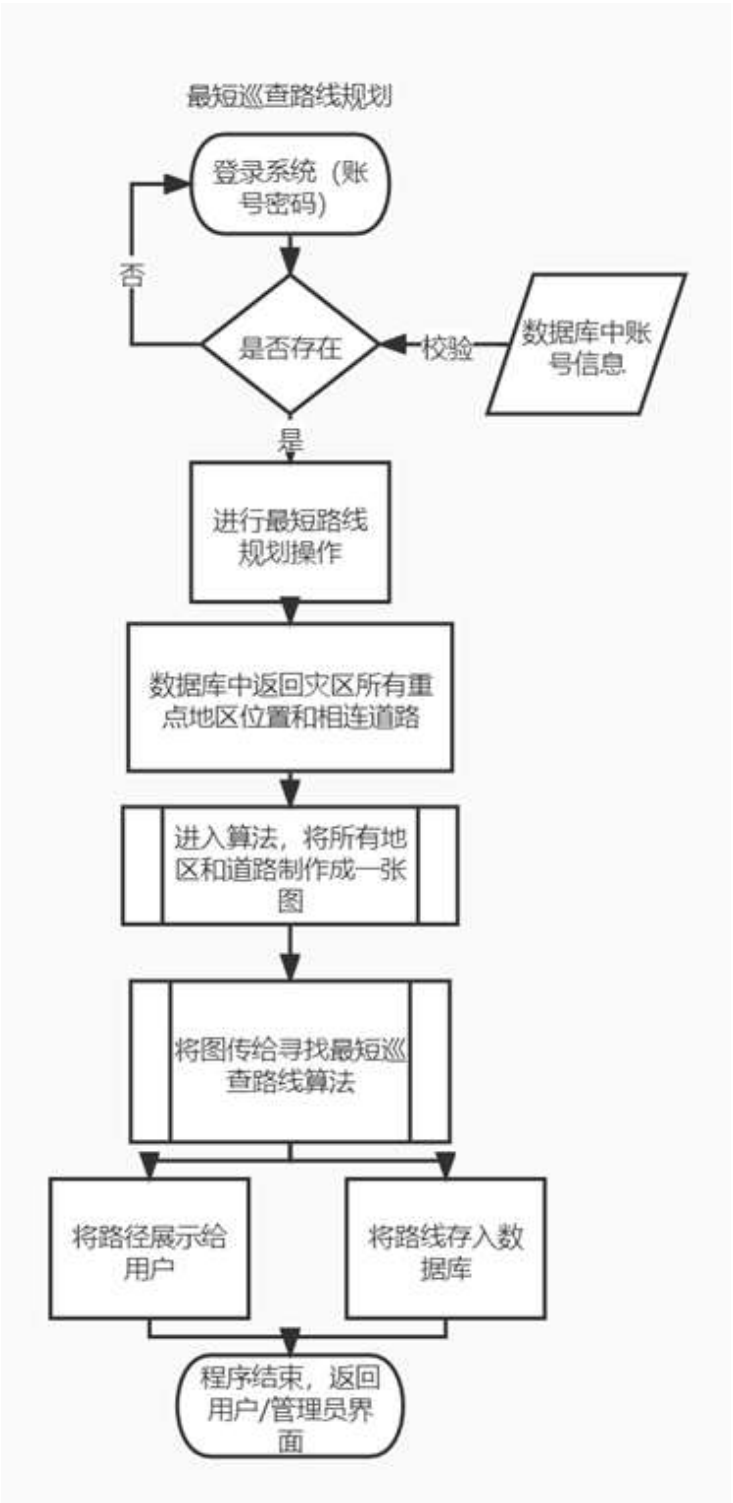
灾情信息用户查看：

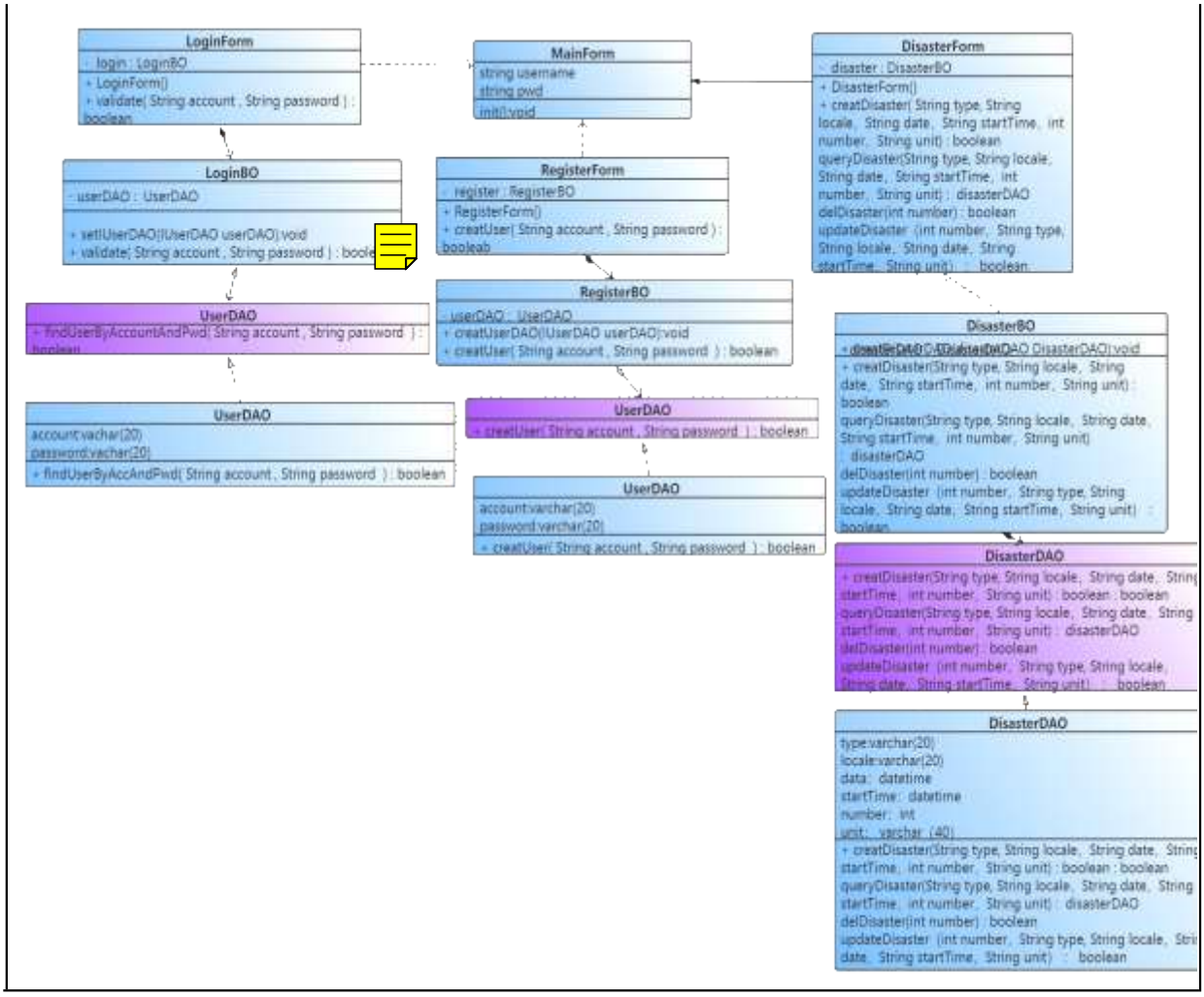

灾情信息用户查询
登录系统（账号密码）
否
是否存在
校验
数据库中账号信息
是
灾情信息页面
灾情信息查询
灾情信息浏览
数据库返回信息
再次进入灾情信息界面

通知用户查看：


通知用户查看
登录系统（账号密码）
否
是否存在
校验
数据库中账号信息
是
进入通知界面
通知查看
通知信息删除
否
数据库返回通知信息
确认删除
更新到数据库用户信息
再次进入灾情信息界面

最短巡查路线规划：


最短巡查路线规划
登录系统（账号密码）
否
是否存在
校验
数据库中账号信息
是
进行最短路线规划操作
数据库中返回灾区所有重点地区位置和相连道路
进入算法，将所有地区和道路制作成一张图
将图传给寻找最短巡查路线算法
将路径展示给用户
将路线存入数据库
程序结束，返回用户/管理员界面



3 单元设计说明

标识符:

（自由表述）

4 数据库设计

4.1 外部设计

（自由表述）

1. 数据库类型:
2. 数据库版本:
3. 数据库运行环境

1. 正式环境

硬件信息:

CPU:_____

内存:_____GB

固态硬盘:无机械

硬盘:_____GB

软件信息:移动客户端服务器部署在linux centos，操作系统为LinuxCentOS_____(版本号），64bit,Web中间件为,Apache/_____(版本号）(Unix)

2. 开发环境

1. windows_____

4.数据库前缀/表前缀
1.jr
5.数据库名称_____
1.数据库标识符：organ
2.数据库表的命名规约:表名首字母大写，表明包含多个单词时，中间单词首字母大写。数据库字段的命名约定:在字段名前加一个表示类型的前缀(除了ID和上传文件字段)，前缀后第一个字母大写，具体可参考下表。

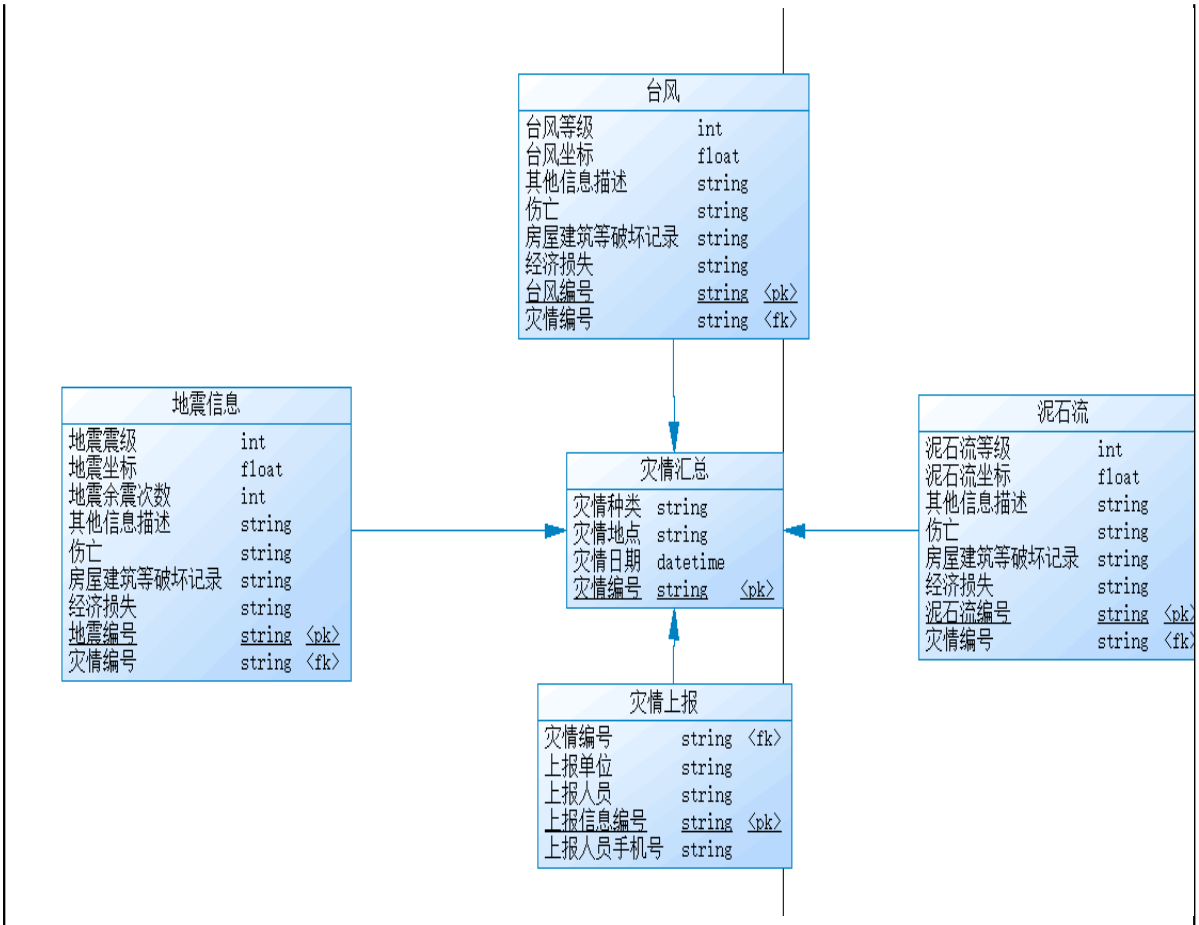
序号	子类型	前缀	示例
1	Boolean	bln	blnFound
2	Byte	byt	bytRasterData
3	Date(Time)	dtm	dtmStart
4	Double	dbl	dblTolerance
5	Error	err	errOrderNum
6	Integer	int	intQuantity
7	Long	lng	lngDistance
8	Object	obj	objCurrent
9	Single	sng	sngAverage
10	String	str	strFirstName
11	Decimal	dec	decCreateTime

该数据库的支撑文件包括:_____
该数据库的数据表将建立与基础数据库,数据库与其他业务系统数据库将使用同一数据库服务器的统一数据库实例。在数据库实例中建立若干数据库，基础数据库使用单独的数据库;所有数据库的数据库容器(container）都放置在实例用户主目录下的DATA子目录下;每个数据库在这个目录下建一个子目录,并在子目录下放置数据库容器和数据库日志,基础数据库容器存放在basedata子目录下;为了保证数据库容器的空间不被其他文件侵占，basedata目录需要单独建立文件系统，并挂载到DATA子目录下。
基础数据库中建立如下三个表空间:
●数据表空间/basedata_tbs:用于存放基础数据表的数据，数据页长度8K，容器名: basedata。
●索引表空间/basedata_tbs:用于存放基础数据表的索引数据,数据页长度8K，容器名: baseindex。
●LOB表空间/basedata_tbs:用于存放基础数据表中 LOB、CLOB、XML字段的数据，数据页长度32K，容器名: baseLOB。
所有数据库容器都用操作系统管理的文件形式存放，每个表空间建一个数据库容器，文件长度采用数据库管理自动增长的方式。
6.数据库权限
1.root用户,只能本地登录
2. migration版本管理用户,可以修改表结构,禁止读写数据
3.api接口账户,大部分表只读,可以远程登录
4.admin后台账户,部分表只读,可以远程登录5.develop开发阶段账户
5.develop开发阶段账户
7.用户名
1.root
2. migration_user版本管理用户,分配给laravel的migration做表修改
3.api_user/admin_user(禁止修改表结构)
4.develop_user(禁止修改表结构,开发阶段用户,正式上线后注销账户)
8.密码要求
1.随机生成16位,其中必须包含A-Z/ a-z /0-9 /!@#$%^&*
2.上线后,数据库密码每个月要更换一次
9.端口
1.3309
10.有效时间
1.开发阶段
11.版本管理工具
1.laravel的migration
12.可视化链接工具要求
1.SQLyog Ultimate - MySQL GUI v12.2.6(64 bit),必须统一,禁用其他工具
13.数据库的修改一律使用migration,禁止在数据库上直接操作,工具只能查看

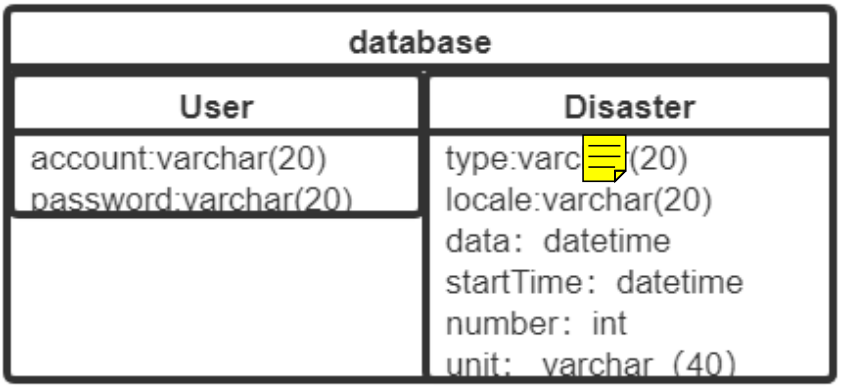
4.2 结构设计

概念设计:

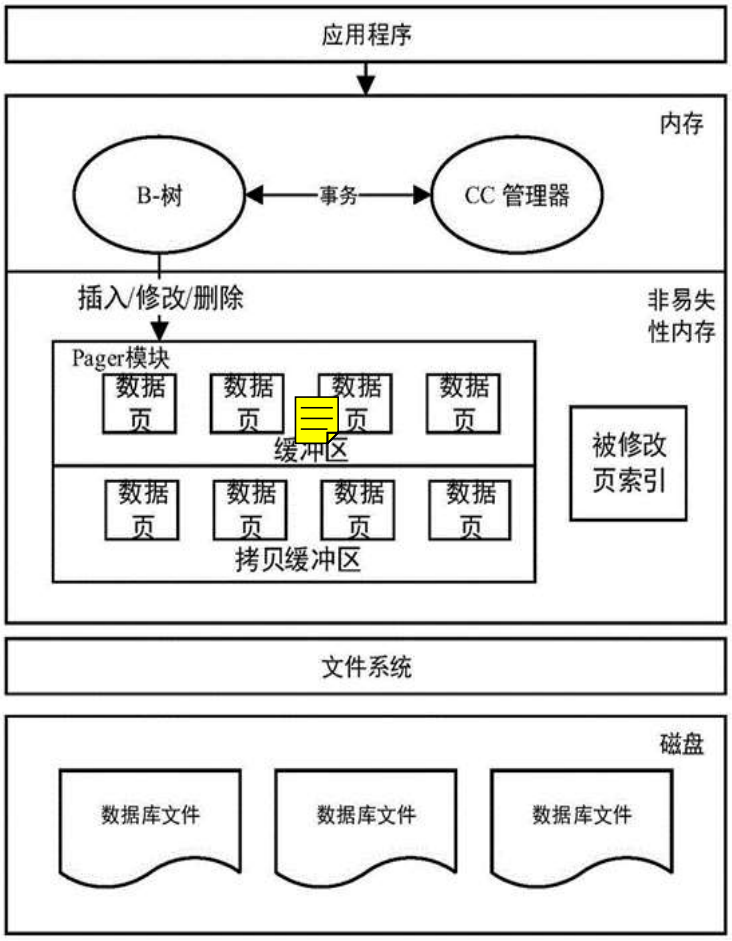
使用PowerDesigner（Physical Digram形式该形式表与表关系体现在integrity里）设计灾情上报管理系统的概念数据模型如下图所示，其中包括4个实体和1个联系。“泥石流”实体、“地震”实体、“台风”实体、“灾情上报”与“灾情汇总”见的关联均是多对一。


台风
台风等级 int
台风坐标 float
其他信息描述 string
伤亡 string
房屋建筑等破坏记录 string
经济损失 string
台风编号 string 〈pk〉
灾情编号 string 〈fk〉
地震信息
地震震级 int
地震坐标 float
地震余震次数 int
其他信息描述 string
伤亡 string
房屋建筑等破坏记录 string
经济损失 string
地震编号 string 〈pk〉
灾情编号 string 〈fk〉
灾情汇总
灾情种类 string
灾情地点 string
灾情日期 datetime
灾情编号 string 〈pk〉
泥石流
泥石流等级 int
泥石流坐标 float
其他信息描述 string
伤亡 string
房屋建筑等破坏记录 string
经济损失 string
泥石流编号 string 〈pk〉
灾情编号 string 〈fk〉
灾情上报
灾情编号 string 〈fk〉
上报单位 string
上报人员 string
上报信息编号 string 〈pk〉
上报人员手机号 string

逻辑结构设计：


database
User
account:varchar(20)
password:varchar(20)
Disaster
type:varc (20)
locale:varchar(20)
data: datetime
startTime: datetime
number: int
unit: varchar (40)

物理设计：



采用基于非易失性内存的拷贝的缓冲区保证事务的原子性和持久化特性的方法，构建了一种新的基于非易失性内存的sqlite缓冲区sqlite-cc（copycache），通过一个分配全局时间戳的事务管理器ccmanager保证事务执行的有序性，以及基于非易失性内存的拷贝的缓冲区保证事务的原子性和持久化特性，并且一定程度上提升并发，通过写后属性的修改页的索引文件，保证非易失性内存与磁盘中数据库文件间的一致性。利用非易失性内存的硬件特性，通过在一个缓冲区中维护两份相同的数据，同时引入cc-manager用于维护事务原子性，并加入修改页面索引updated-pagesindex来保证数据库文件与缓存之间的一致性，使其达到了和sqlite-wal模式相当的并发性能，提升了吞吐量，减少了读写延迟，且有效解决了磁盘中的写放大问题。

采用基于非易失性内存的拷贝sqlite缓冲区方法，通过在一个缓冲区中维护两份相同的数据，同时引入分配全局时间戳的事务管理器cc-manager，并加入修改页面索引updated-pagesindex来保证数据库文件与缓存之间的一致性，使事务执行有序性、原子性和持久化特性得到提升并发，并通过写后属性修改页的索引文件，确保非易失性内存与磁盘中数据库文件之间的一致性。

所述分配全局时间戳的事务管理器cc-manager的实现如下：

a1：一种基于物理时钟的全局时间戳生成功能，在每一个事务开始时，为每个事务分配时间戳，用于协调读写事务之间的先后关系。

a2：管理并检测主缓冲区与拷贝缓冲区之间的一致性状态，通过对两个缓冲区的一致性状态判断，保证事务的acid特性。并且提供恢复点的判断功能。

所述基于非易失性内存的拷贝sqlite缓冲区的实现如下：

b1：每个进程所维护的缓冲区中额外加入一个拷贝的缓冲区，存放于同一个pcache句柄的两个hash表中，一般状态下两者的数据相同，当有写事务进行的时候，主缓冲区承担写入任务，出现不一致状态。结合ccmanager的管理保证是事务的有序进行。

b2：在pcache句柄中记录两个缓冲区的状态，拷贝的副缓冲区在主缓冲区进行修改的时候，保证读事务并发的同时，由于副缓冲区仍然保持着原有数据，则其承担undolog的作用以保障事务的原子性，当出现事务abort的时候或者事务修改未完成宕机时，副缓冲区的值为准确值，可以用作还原主缓冲区中进行的修改。当出现事务修改已完成但是未提交的情况，即主缓冲区已经完成修改，副cache没有同步，同样通过cc-manager的状态检查，判定当前状态，主缓冲区可以承担redolog的作用，将其修改值同步到副缓冲区，并提交事务，以保障事务的持久性。

所述修改页的索引的实现如下：

c1：一个的修改页索引，记录缓冲区中修改的页的编号和偏移量等matedata，标记出与磁盘中数据库文件不同步的页。而后通用延迟同步的方式和被换出缓冲的时刻，将更新同步到磁盘上，以此来保证非易失性内存中的缓冲区与磁盘中数据库文件的一致性。

具体实施方法：基于原始的sqlite数据库架构的面向非易失性内存缓冲区的sqlite系统，将系统分为三个部分，分别是内存中的b树模块和cc-管理器模块、非易失性内存中的pager模块和被修改页索引模块，以及磁盘中的数据库文件存储。cc-管理器模块与b-树模块协同对事务进行管理，b-树模块通过访

问在非易失性内存中的pager模块对缓冲区的数据页进行读写等操作，被修改页索引用于磁盘与非易失性内存同步。
写操作具体流程：通常进行写操作的时候，sqlite通过b-树模块向pager模块发出请求，然后经过缓冲区对数据进行修改。在sqlite-cc中，首先需要经过cc-管理器的sqliteccccheckstate()判定，确保当前两个缓冲区处于同步状态，即可获得写入权限，通过sqliteccgettimestamp()分配给其写时间戳。
修改过程中会在缓冲区中，短暂的形成两个缓冲区不同步的时段，该两个缓冲区由一个pcache句柄统一管理，主缓冲区对应的哈希表处于已修改的状态，已被修改的数据页，也被视为重做日志数据页。拷贝缓冲区对应的哈希表则是保持原有的值不变，所以此处与主缓冲区中被修改处对应的数据页，可以视为回滚日志数据页。即正常的修改流程为经历同步，到不同步，再到同步的流程，并且这个过程通过英特尔的非易失性内存管理库的事务对象型接口保证原子性。由于非易失性内存的持久性，无需担心数据丢失，可以在同步即提交事务sqliteccccommit()，之后记录被修改页索引来确保当缓冲区的页被换出之后的与数据库文件的一致性，不用急切地将修改值刷到磁盘中的数据库文件，减少了磁盘i/o。当发生缓冲区换入换出时，检测换出页是否在修改页索引中标记，如果标记则修改磁盘中的内容，如果未被标记则直接丢弃。
读操作具体流程：在sqlite-cc中，将面临着两种不同的情况，一种情况是当两个缓冲区处于同步状态时，其读取流程跟原始的sqlite相同；另一种情况则是，当前的缓冲区状态不同步，即此时存在正在进行当中的写入操作。事务开始时，会经由cc-manager判断当前状态，并分配时间戳，之后根据读写事务的先后顺序，分为两种情况：当读在写事务之后，它将被阻塞等待，因为它需要通过主缓冲区中新的内容进行读取，读取到修改后的值；反之则通过拷贝的缓冲区进行读取，读取到旧值，其余流程相同。
当写事务开始时，将出现新旧读事务的划分，对之前已经执行完成的读操作的线程没有影响，正在进行的操作如读事务b，读事务d将被作为旧的读事务，通过拷贝缓冲区进行值的读取，然后当旧事务线程全部完成之后，对两个缓冲区进行同步，完成并提交此次操作。之后开始执行的读事务a，读事务c则会在等待执行过程中的写事务提交完成之后通过主缓冲区读取到最新值。
恢复操作具体流程：当出现宕机时，原始的sqlite有自己的方法在重新启动时进行故障检测，当它在“删除日志”模式下运行时，若存在热日志则表示sqlite在重新启动之前处于崩溃状态，其中热日志（即宕机前已经记录，但是由于事务并未提交，仍然存在的日志）可用于恢复数据库一致性。当原始的sqlite以wal日志模式运行时，wal日志里没有提交记录的部分则表示崩溃时正在进行的事务，删除该部分修改即可。sqlite-cc检测故障的方式与前两者均不相同，其事务的原子性和持久性都是经由缓冲区来进行保证的，由于非易失性内存的持久性，任何情况下的故障恢复都能够获取到宕机前缓冲区的状态。如果存在需要恢复的数据，则sqlite-cc的缓冲区中两个缓冲区肯定不是同步的，同时由于在任何时候最多一个写事务都在同一数据库上运行，那么两者必有一个是准确的，所以执行sqliteccrecover()，首先检测副缓冲区的启用状态sqliteccccheckstate()，可以判断所要恢复的状态点，进而做出对应的回滚或者重做操作，使数据库达到一致性状态。另外，如果宕机发生在将缓冲区中换出的页同步到数据库文件时，通过检测被修改页索引中是否还记录有该页和对比换出页与数据库文件里对应页的值，可以判定是否需要再次将缓冲区中的页复制到数据库文件中，以保证数据的一致性。

4.3 运用设计

数据字典设计：

字段名	数据类型	默认值	允许非空	自动递增	备注
earthquakeId	char(30)		No	是	primary key，地震id
province	char(10)		No		省
city	char(10)		No		城市
country	char(10)		No		国家
town	char(10)		No		乡镇
village	char(10)		No		村
category	char(10)		No		类别
location	char(20)		No		位置
longitude	float(5,2)	0	No		经度
latitude	float(5,2)	0	No		纬度
depth	float(5,2)	0	No		深度
magnitude	float(5,2)		No		震级
reportingUnit	char(20)		No		primary key，报告单位
Id		0	No	是	Id
note	char(200)		Yes		记录
grade	char(10)		No		等级
basicallyIntactSquare	char(10)		Yes		完好情况
damagedSquare	char(10)		Yes		损毁情况
destroyedSquare	char(10)		Yes		损毁情况
status	char(10)		Yes		损毁状态
type	char(10)		No		损毁类型

安全保密设计：

一、 数据库安全与保密概述
数据库系统

一般可以理解成两部分一部分是数据库
按一定的方式存取数据另一部分是数据库管理系统(DBMS)
为用户及应用程序提供数据访问
并具有对数据库进行管理、维护等多种功能。
数据库系统安全
包含两层含义

第一层是指系统运行安全
它包括
法律、政策的保护
如用户是否有合法权利
政策是否允许等
物理控制安全
如机房加锁等
硬件运行安全
操作系统安全
如数据文件是否保护等
灾害、故障恢复
死锁的避免和解除
电磁信息泄漏防止。

第二层是指系统信息安全
它包括
用户口令字鉴别
用户存取权限控制
数据存取权限、方式控制
审计跟踪
数据加密。

二、 数据库基本安全架构
1. 用户分类
不同类型的用户授予不同的数据管理权限。一般将权限分为三类数据库登录权限类、资源管理权限类和数据库管理员权限类。
有了数据库登录权限的用户才能进入数据库管理系统
才能使用数据库管理系统所提供的各类工具和实用程序。同时
数据库客体的主人可以授予这类用户以数据查询、建立视图等权限。这类用户只能查阅部分数据库信息
不能改动数据库中的任何数据。
具有资源管理权限的用户
除了拥有上一类的用户权限外
还有创建数据库表、索引等数据库客体的权限
可以在权限允许的范围内修改、查询数据库
还能将自己拥有的权限授予其他用户
可以申请审计。
具有数据库管理员权限的用户将具有数据库管理的一切权限
包括访问任何用户的任何数据
授予(或回收)用户的各种权限
创建各种数据库客体
完成数据库的整库备份、装入重组以及进行全系统的审计等工作。这类用户的工作是谨慎而带全局性的工作
只有极少数用户属于这种类型。
2. 数据分类
同一类权限的用户
对数据库中数据管理和使用的范围又可能是不同的。为此
DBMS提供了将数据分类的功能
即建立视图。管理员把某用户可查询的数据逻辑上归并起来
简称一个或多个视图
并赋予名称
在把该视图的查询权限授予该用户(也可以授予多个用户)。这种数据分类可以进行得很细
其最小粒度是数据库二维表中一个交叉的元素。

3. 审计功能
大型DBMS提供的审计功能是一个十分重要的安全措施
它用来监视各用户对数据库施加的动作。有两种方式的审计
即用户审计和系统审计。用户审计时
DBMS的审计系统记下所有对自己表或视图进行访问的企图(包括成功的和不成功的)及每次操作的用户名、时间、操作代码等信息。这些信息一般都被记录在数据字典(系统表)之中
利用这些信息用户可以进行审计分析。系统审计由系统管理员进行
其审计内容主要是系统一级命令以及数据库客体的使用情况。

三、 数据库加密
一般而言

数据库系统提供的上述基本安全技术能够满足一般的数据库应用
但对于一些重要部门或敏感领域的应用
仅靠上述这些措施是难以完全保证数据的安全性
某些用户尤其是一些内部用户仍可能非法获取用户名、口令字
或利用其他方法越权使用数据库
甚至可以直接打开数据库文件来窃取或篡改信息。因此
有必要对数据库中存储的重要数据进行加密处理
以实现数据存储的安全保护。
1. 数据库密码系统的基本流程
数据加密
就是将明文数据经过一定的交换(一般为变序和代昀)变成密文数据。
数据脱密是加密的逆过程
即将密文数据转变成可见的明文数据。
一个密码系统包含明文集合、密文集合、密钥集合和算法
其中密钥和算法构成了密码系统的基本单元。算法是一些公式、法则或程序
规定明文与密文之间的变换方法
密钥可以看作算法中的参数。
数据库密码系统要求将明文数据加密成密文数据
数据库中存储密文数据
查询时将密文数据取出脱密得到明文信息。