

河海大学物联网工程学院

数据库系统原理 课程设计报告

项目实践部分

学年学期	<u>2021—2022（1）</u>
组 号	<u>21</u>
题 目	<u>流浪宠物领养管理系统</u>
专 业	<u>计算机科学与技术 19 级</u>
授课班号	<u>c0601046</u>
组长	<u>1961010516 秦骁</u>
组成员	<u>1961310203 顾书宁</u>
	<u>1961310319 王子潇</u>
授课教师	<u>陈慧萍</u>

任务分配表

任务	姓名	备注
项目实践报告	秦骁	由王子潇完成 Java 技术部分的文档
E-R 图设计	秦骁	
前端界面	王子潇	顾书宁设计，王子潇使用 Javafx 技术编写
数据库设计	顾书宁	PowerDesigner 转换，并结合实际应用稍修改
前后端连接	王子潇	JDBC 实现前端与后端数据库连接
存储过程、触发器设计	顾书宁	秦骁辅助完成部分工作

目录

一、课题背景及意义	4
二、需求分析.....	4
1. 业务需求	5
2. 功能需求	5
3. 数据需求	6
三、概要设计.....	9
1. 系统组成	9
2. 系统结构设计	10
3. 数据库设计	11
(1) 概念结构设计	11
(2) 逻辑结构设计	12
(3) 关联图	14
四、详细设计与实现	15
五、结束语.....	36
参考文献.....	36

一、课题背景及意义

1. 课题背景

饲养宠物是社会的一大潮流，但是宠物遗弃、不良商贩无节制繁育宠物，产生了一系列伦理问题以及社会环境问题。首先，宠物购买推动商贩不断繁育宠物造成母体早亡、幼体宠物存活率低；其次，买家一时兴起的购买宠物可能导致宠物被遗弃，危害生态环境；再次，喜欢宠物的人可能面临高昂的费用，而没法购买、拥有自己喜欢的宠物。

2. 课题目标

为此，项目组成员提出领养代替购买的理念，收取低额手续费将流浪宠物交给真正喜欢宠物的人饲养，同时一定程度缓解上述问题。目标系统为三方提供服务：领养者、志愿者、管理员。其中，志愿者救助流浪宠物，在系统平台上传；领养者申请领养心仪的流浪宠物，待志愿者批准后，交付保证金，交接流浪宠物开始试养，并且每一阶段上传宠物的近况信息；管理员对处于试养期的流浪宠物考察，对违反领养合同的领养者进行惩罚，没收保证金并收回流浪宠物；试养期结束后，退还未违约领养者交付的保证金。

系统要求功能完善可靠，为用户提供友好的界面，提供一定的并发度控制，保证信息安全性以及响应时间在可接受范围内。

3. 课题意义

- A. 该项目一定程度上缓解了社会问题，减轻流浪宠物的社会负担；
- B. 为喜欢宠物的人提供了低价饲养宠物的途径，免去宠物店购买昂贵费用；
- C. 将原有系统的线下模式改为线上模式，数据存储在服务器，信息难以丢失，对于用户而言，方便操作流程，减少合同纠纷。

二、需求分析

1. 业务需求

系统的业务处理流程：领养者首先浏览待领养宠物，对心仪的宠物进行领养申请，然后支付保证金；志愿者进行流浪宠物信息管理，审核领养者申请表，对试养宠物进行管理，并申请惩罚领养者；平台管理员进行惩罚判决，并处理保证金退回操作。

以下是业务流程图（见图 1）：

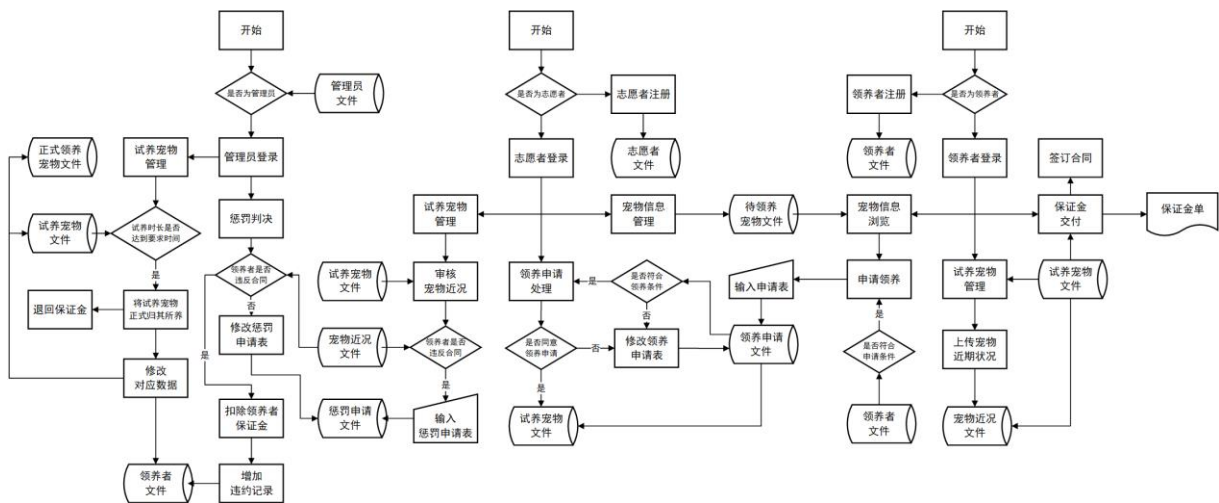


图 1

2. 功能需求

系统功能模型描述具体的功能要求，此处用功能列表（见表 1）描述：

表 1 系统功能列表

编号	功能名称	功能说明
01	流浪宠物信息管理	志愿者增加、修改、删除、查询流浪宠物信息
02	流浪宠物领养申请	领养者对心仪的流浪宠物提交领养申请
03	领养申请审批	志愿者审核领养者条件，决定是否批准领养
04	保证金支付	领养者支付已被批准领养宠物的试样保证金
05	试养期违约惩罚申请	领养者申请没收违约领养者宠物及保证金
06	违约惩罚审批	管理员依据宠物近况，决定是否惩罚

3. 数据需求

3.1 顶层图

顶层图采用的 IPO 模型，主要描述了该系统的主要的输入数据，输出数据，相关实体。（见图 2）

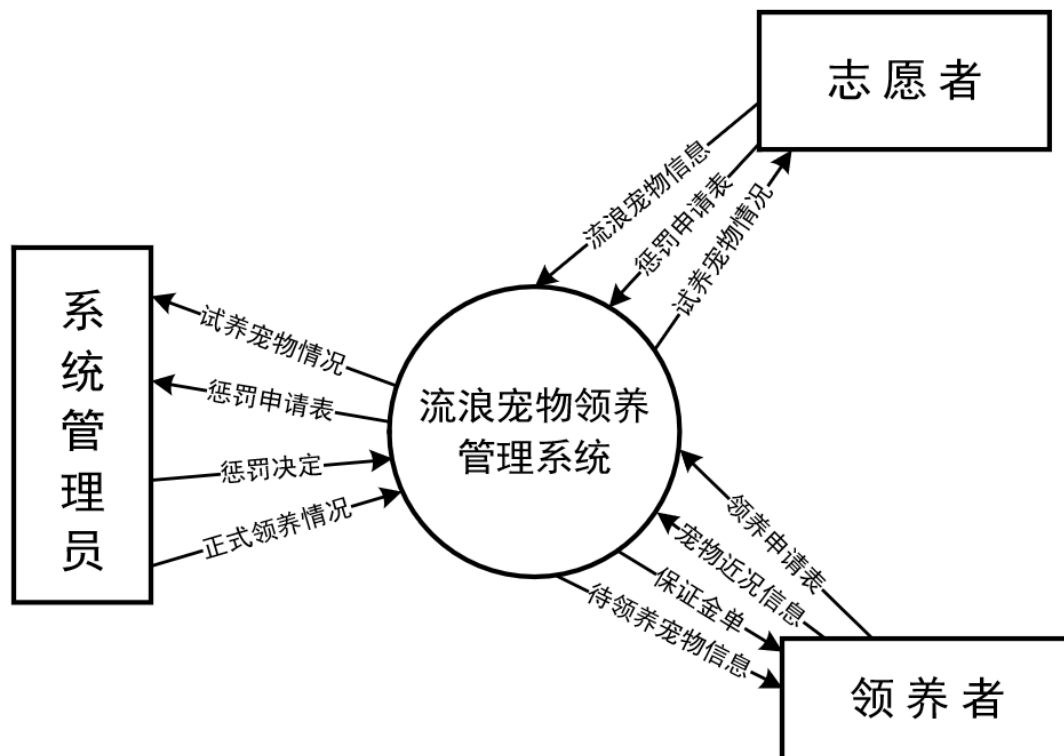
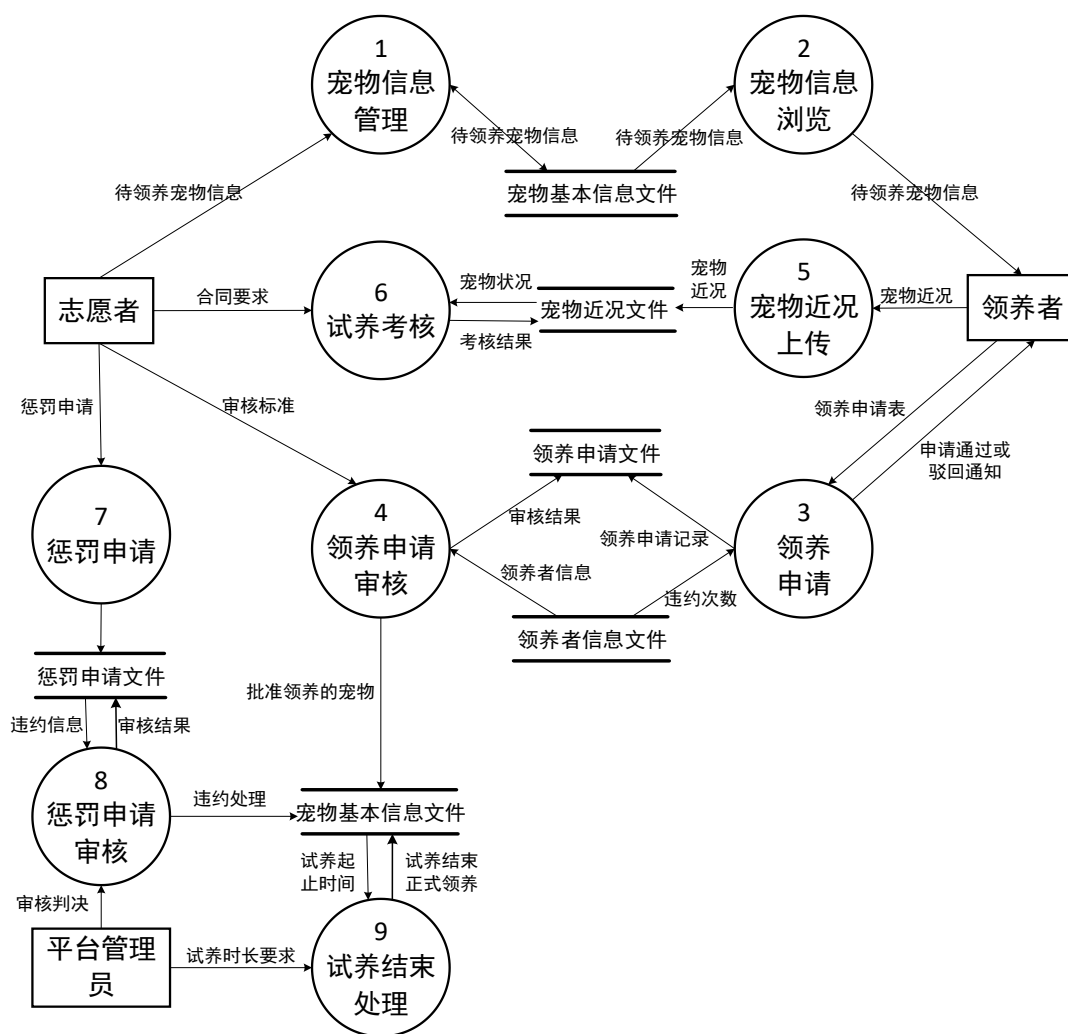


图 2

3.2 0 层图

0 层图（见图 3）对于系统进行了模块的划分，分为了九个部分：待领养宠物信息管理、待领养宠物信息浏览、领养宠物申请与审核、违约惩罚申请、违约惩罚审核、宠物近况上传。其中在该层图中也显示了相关的存储数据。



3.3 1 层图

1 层数据流图(见图 4)中对 0 层图中划分的九个模块分别进行详细描述。

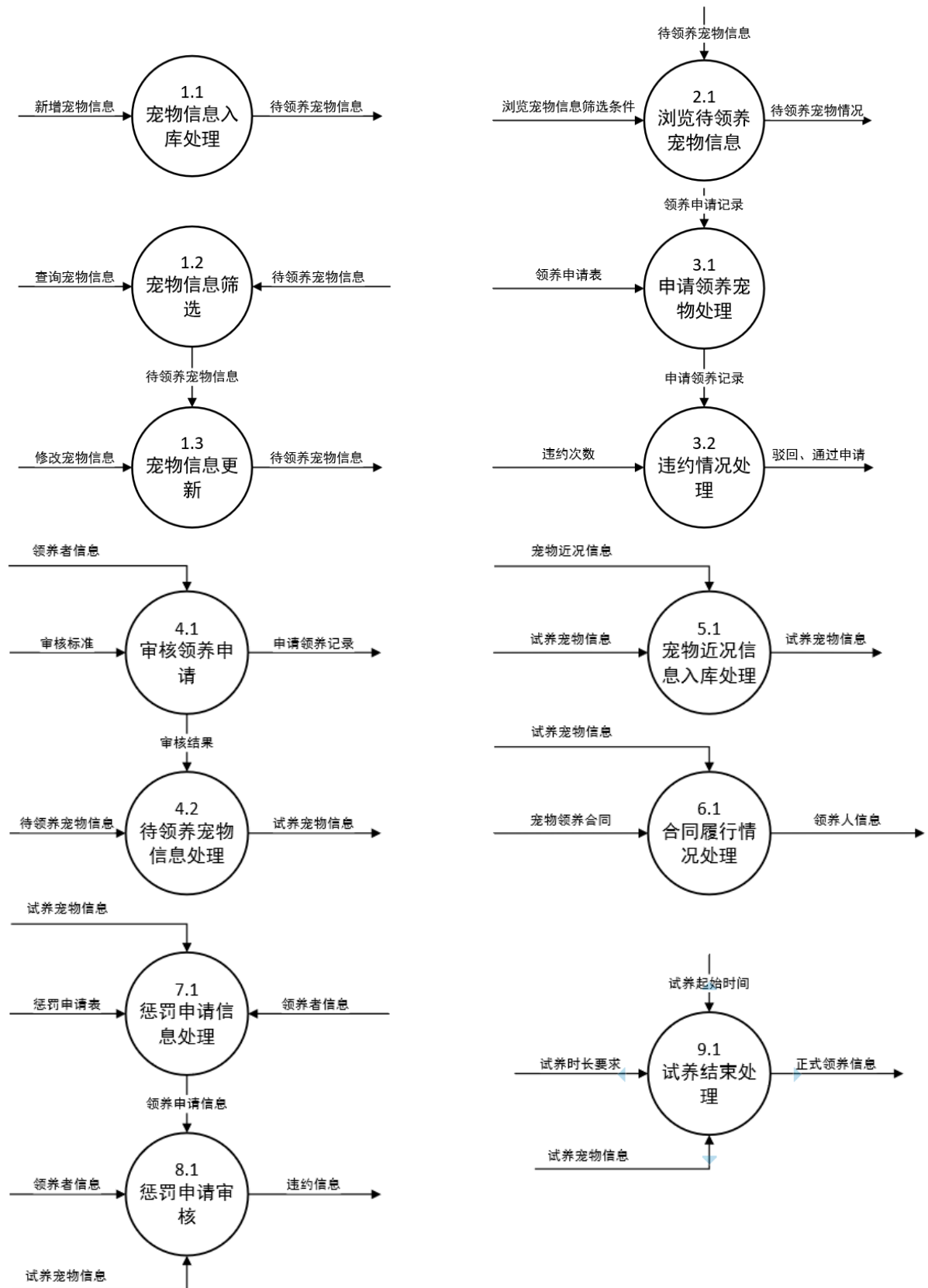


图 4

三、概要设计

1. 系统组成

系统采用 C/S 架构，目前仅支持 Windows 操作系统。
运行环境限制：

普通客户机	
CPU	Intel Core i5 2.3GHz 以上
内存	8G 以上

服务器	
CPU	Intel Core i5 2.0GHz 以上
内存	8G 以上
硬盘	80GB 以上

软件环境	
操作系统	windows10
数据库	Microsoft SQL Server 2012

2. 系统结构设计

软件的总体逻辑结构，采用面向功能的设计方法，分为注册登录模块、待领养宠物管理模块、流浪宠物领养申请模块、试养期宠物管理模块。

其中，登陆注册业务主要为志愿者和领养者的注册登录以及管理员的登录。待领养宠物管理业务主要为志愿者的新增、修改、查询待领养宠物信息管理。流浪宠物领养申请业务主要包括领养者申请领养、志愿者批准领养、保证金支付。试养期宠物管理业务主要分为领养者的定期对试养期宠物的近况更新、志愿者对试养期内有违约行为的领养者惩罚申请和管理者的惩罚申请表审批和保证金退回或没收。

功能模块结构图（见图 5）如下：

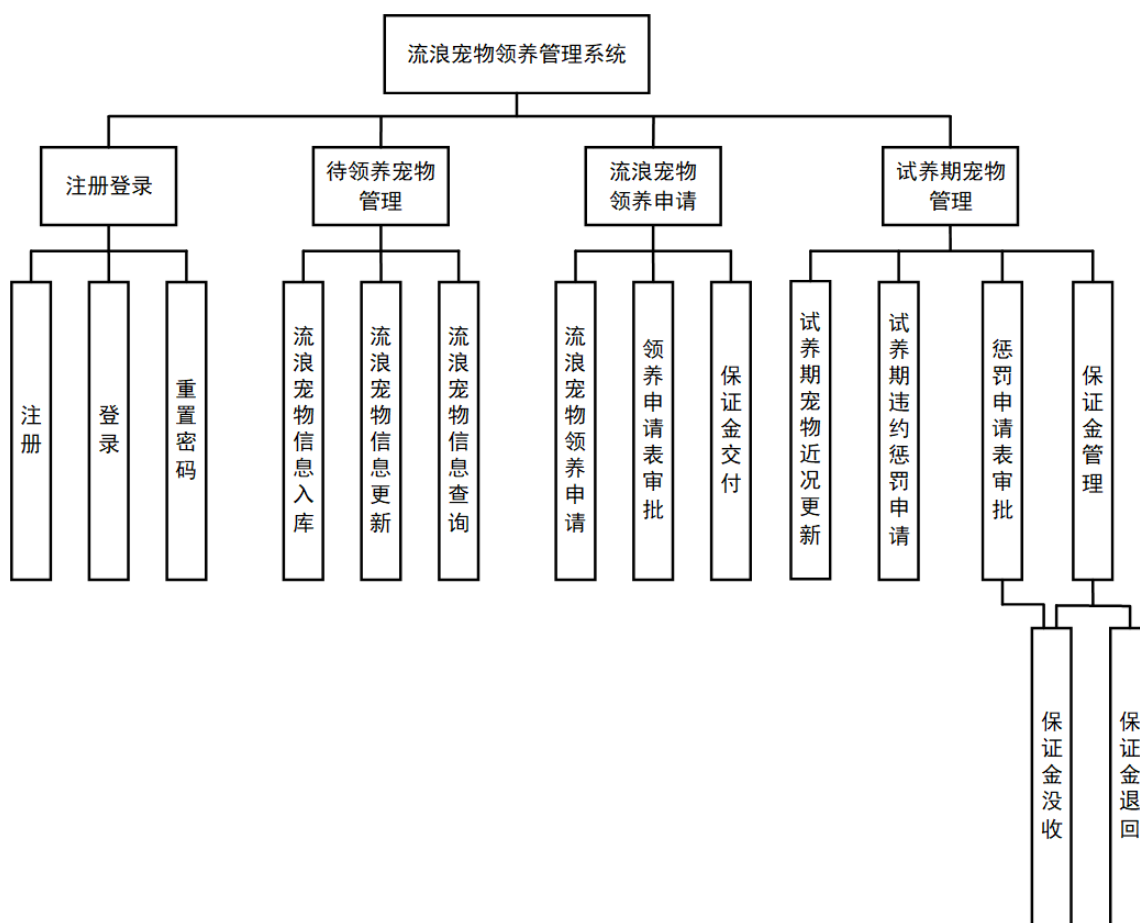


图 5

3. 数据库设计

(1) 概念结构设计

采用 E-R 图（见图 6）描述系统的该概念模型，其中实体包括：领养者、志愿者、流浪宠物、宠物类别、领养单、宠物近况、管理员、违约情况；联系包括：宠物类属于流浪宠物“1 对多”、志愿者负责流浪宠物“1 对多”、流浪宠物被领养单申请“1 对多”、志愿者审核领养单“1 对多”、领养者提交领养单“1 对多”、领养单对应宠物近况“1 对多”、宠物近况-违约情况-管理员“多对多”。

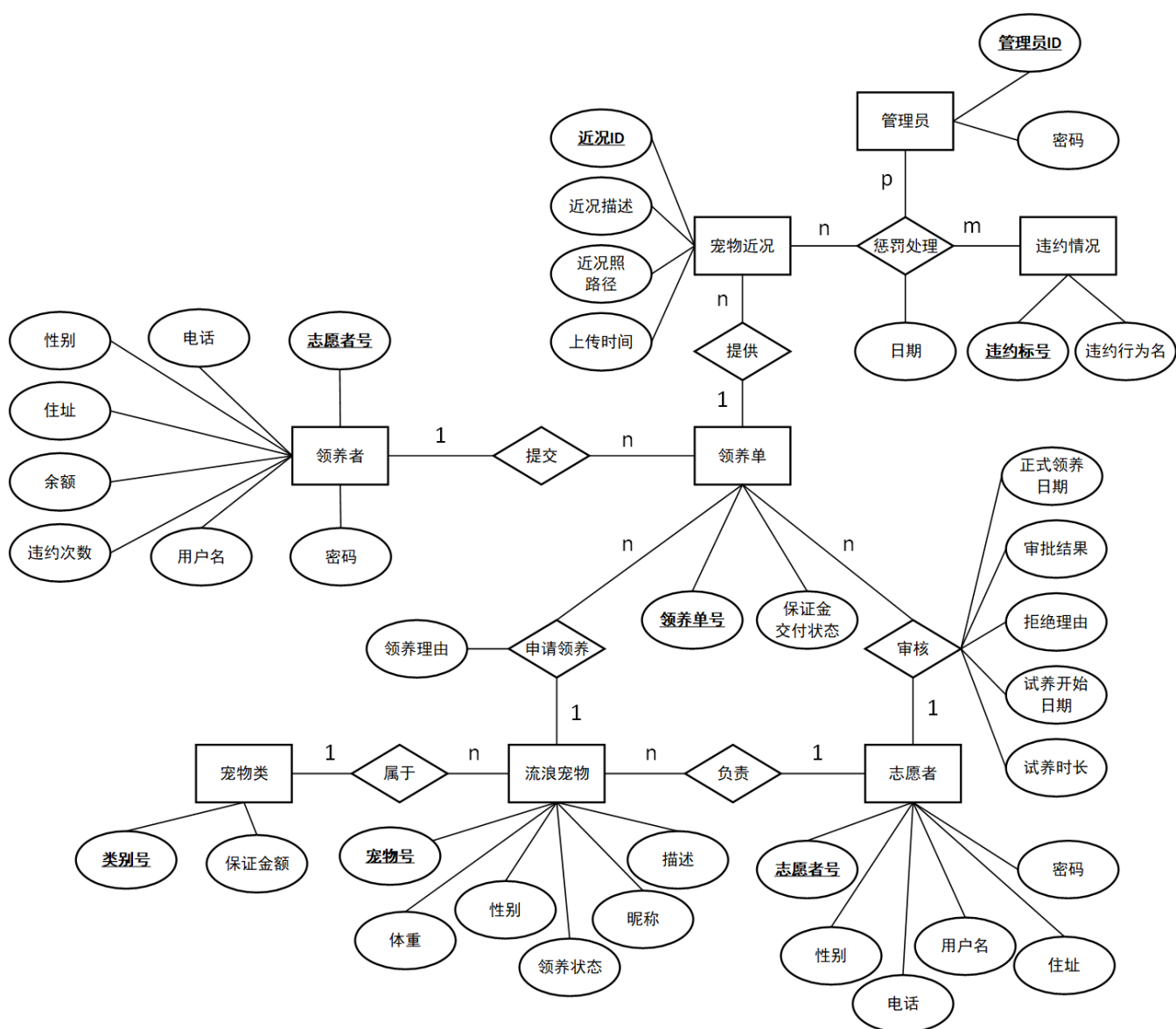


图 6

(2) 逻辑结构设计

根据系统的概念模型设计结果，即 E-R 图，转换成系统的关系模型。此处列出每张数据库表的详细清单：

表 2 名：volunteer 志愿者表

序号	列名	类型	描述	索引/关键字	备注
01	Vno	Char(8)	志愿者编号	Primary key	NOT NULL
02	Vsex	Char(2)	志愿者性别		NOT NULL
03	Tel	Char(11)	志愿者电话		NOT NULL UNIQUE
04	Vloc	Varchar(12)	志愿者所在地		NOT NULL
05	Vname	Varchar(10)	志愿者姓名		NOT NULL
06	Vpsw	Varchar(16)	志愿者密码		NOT NULL

表 3 名：adopter 领养者表

序号	列名	类型	描述	索引/关键字	备注
01	Ano	Char(8)	领养者编号	Primary key	NOT NULL
02	Asex	Char(2)	领养者性别		NOT NULL
03	Tel	Char(11)	领养者电话		NOT NULL UNIQUE
04	Aloc	Varchar(12)	领养者所在地		NOT NULL
05	Aname	Varchar(10)	领养者姓名		NOT NULL
06	Apsw	Varchar(16)	领养者密码		NOT NULL
07	Contractcnt	Smallint	违约次数统计		NOT NULL，默认 0
08	Money	Int	账户余额		NOT NULL，非负

表 4 名：pettype 宠物类别表

序号	列名	类型	描述	索引/关键字	备注
01	Ptype	Varchar(20)	宠物类别号	Primary key	NOT NULL
02	Bond	Float(2)	应付保证金		NOT NULL，非负

表 5 名：pet 宠物表

序号	列名	类型	描述	索引/关键字	备注
01	Pno	Char(8)	宠物编号	Primary key	NOT NULL
02	Pname	Varchar(10)	宠物姓名		NOT NULL
03	Psex	Char(2)	宠物性别		NOT NULL
04	Vno	Char(8)	志愿者编号	Foreign key	NOT NULL
05	Ptype	Varchar(20)	宠物类别号	Foreign key	NOT NULL
06	Pweight	Int	宠物体重		NOT NULL，非负
07	Dscrib	Varchar(200)	情况描述		NOT NULL
08	Isadpated	Char(6)	领养状态		NOT NULL In (‘试养期’，‘待领养’，‘已收养’)

表 6 名: admins 管理员表

序号	列名	类型	描述	索引/关键字	备注
01	Adminno	Char(8)	管理员编号	Primary key	NOT NULL
02	Pswd	Varchar(16)	管理员密码		NOT NULL

表 7 名: adpapply 领养表

序号	列名	类型	描述	索引/关键字	备注
01	Adp_applyno	Char(8)	领养单号	Primary key	NOT NULL
02	Ano	Char(8)	领养者编号	Foreign key	NOT NULL
03	Pno	Char(8)	宠物编号	Foreign key	NOT NULL
04	Apply_reason	varchar(200)	申请领养理由		NOT NULL
05	Isapproval	Char(4)	领养申请状态		NOT NULL In (‘未批’, ‘通过’, ‘拒绝’)
06	Refuse_reason	Varchar(100)	拒绝理由		
07	Trail_time	Date	试养开始日期		
08	Traillenght	Int	试养时长		
09	Formal_time	Date	正式领养日期		
10	Bondstate	Char(4)	保证金状态		NOT NULL In (‘未付’, ‘已付’, ‘归还’)

表 8 名: breach 惩罚类别表

序号	列名	类型	描述	索引/关键字	备注
01	Bno	Char(1)	违约编号	Primary key	NOT NULL
02	Behaviour	Varchar(8)	违约行为		NOT NULL UNIQUE

表 9 名: recentstand 近况表

序号	列名	类型	描述	索引/关键字	备注
01	Rno	Char(8)	近况编号	Primary key	NOT NULL
02	Adp_applyno	Char(8)	领养单号	Foreign key	NOT NULL
03	rdescribe	Varchar(200)	近况描述		NOT NULL
04	Rfile	varchar(100)	近况照片路径		NOT NULL
05	Uptime	Date	上传日期		NOT NULL

表 10 名: punish 惩罚表

序号	列名	类型	描述	索引/关键字	备注
01	Adminno	Char(8)	管理员编号	Primary key	NOT NULL
02	Rno	Char(8)	近况编号	Primary key	NOT NULL
03	Bno	Char(1)	违约编号	Primary key	NOT NULL
04	Pdate	Date	违约日期		NOT NULL

(3) 关联图

在 DBMS 上建立了所需要的表和表之间的关系后，此处展示 SQL Server Management Studio 自动生成表的关联图（见图 7）。

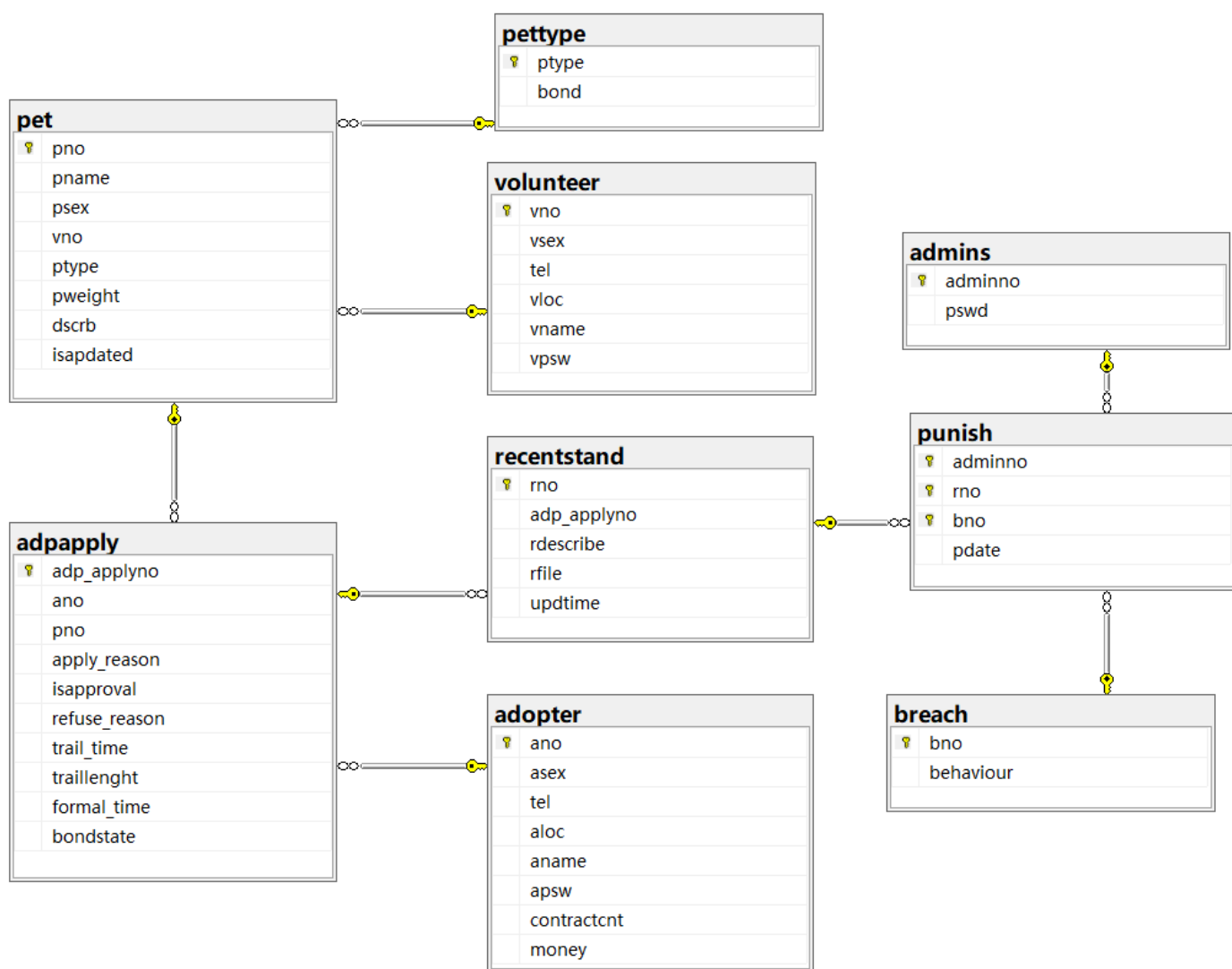


图 7

四、详细设计与实现

①登录注册

(1) 模块编号: RegisterAndLogin

(2) 模块名称:

登录注册模块

(3) 模块功能:

1.登录:用户输入账号密码进行登录,如果未注册则会提示账号未注册,如果账号密码不匹配则,提示密码错误,账号密码均正确,才可进入系统。

2.注册:用户未拥有本系统的账号密码,点击立即注册,填写个人基本信息进行注册,注册成功后即可登录使用本系统。

(4) 模块背景描述:

本模块主要用对用户身份的鉴别,要求用户填写的信息必须是完整有效的才能1注册成为系统用户,用户输入正确的用户标识密码后才能使用系统。

(5) 设计思路及技术要点:

性能要求:

准确性:用户填写的信息必须是有效完整的。

及时性:系统相应操作响应时间不超过2秒。

输入项目:

登录模块输入项如表4.1所示:

名称	标识	数据类型	有效范围	输入方式
用户名	username	Char	4-16字节英文或者数字字符	手动输入/系统输入
密码	password	Char	6-12位英文和数字组成	手动输入

表 4.1 登录模块输入项

输出项目:

登录验证的反馈信息:

- (1) 验证成功:进入系统
- (2) 账号不存在:账号未注册
- (3) 密码不匹配:密码错误
- (4) 输入为空:账号密码为必填项

注册账号的反馈信息:

- (1) 输入为空:用户名、密码、确认密码、手机号不能为空
- (2) 两次密码输入不一致:密码和确认密码需一致
- (3) 密码格式不正确:密码应为6-12位英文和数字组成
- (4) 电话号码不足11位
- (5) 手机号格式不正确
- (6) 该手机号已注册
- (7) 该用户名已注册

界面设计:

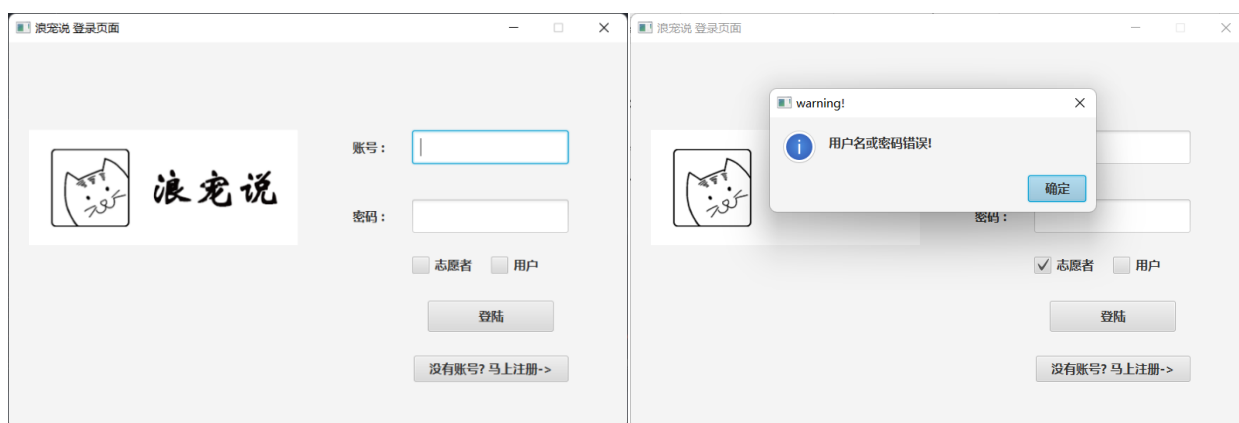


图 4.2 登录界面及信息输入错误后

(6) 模块算法设计:

用户输入账号密码进行登录，系统对用户输入的账号密码进行验证，如果验证失败则反馈相应的验证消息，并且返回登陆界面；如果验证成功，则将用户信息存入 session 中，页面跳转至主界面。

登录模块的算法流程图如图 4.3 所示:

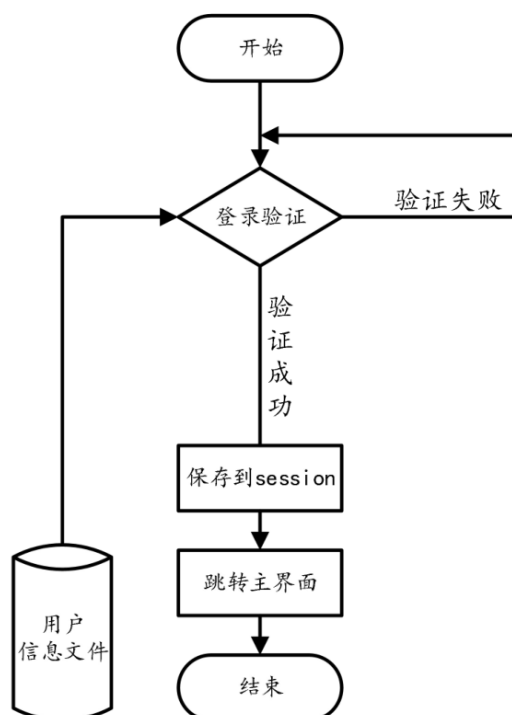


图 4.3 登录模块算法流程图

如果用户未注册，可以点击立即注册按钮，填写个人信息进行注册，填写个人信息时，如果账号已注册，系统会提醒用户该账号已注册；如果手机号已注册，系统会系统用户该手机号已注册；如果密码或者手机号格式不正确，系统会反馈给用户相应的反馈信息，如果所有信息均正确，则注册成功，用户信息写入用户信息文件。

注册模块的算法流程图如图 4.4 所示：

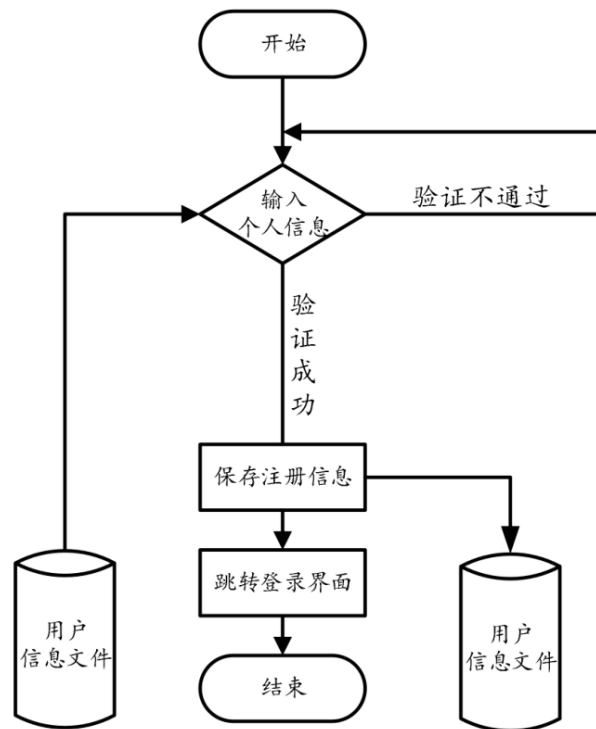


图 4.4 注册模块算法流程图

(7) 关键代码及描述：

```
1. /** Adopter 登录 * @param userId 用户账号 * @param password 用户密码
2. * @return 与数据库中数据对比，若账号密码正确则返回 Adopter 信息，错误则 null**/
3. public static AdopterBean findAdopterForLogIn(String userId, String password
   ) {
4.     String sql = "select * from adopter where ano = ? and apsw = ?";
5.     try (Connection conn = SQLUtil.getConnection()) {
6.         PreparedStatement st = conn.prepareStatement(sql);
7.         st.setString(1,userId);
8.         st.setString(2,password);
9.         try (ResultSet rs = st.executeQuery()) {
10.            if (rs.next()) {
11.                return AdopterUtil.ResultToUser(rs);
12.            }
13.        } catch (Exception inEx) {
14.            DialogBox.error(inEx);
15.            return null;
16.        }
17.    } catch (SQLException e) {
18.        DialogBox.error(e);
19.    }
20.    return null;
21. }
```

```

1. // 从数据库获取用户信息并判断是否在线
2. if (volunteer.isSelected()) {
3.     CacheDataBean.setIdentity("volunteer");
4.     VolunteerBean nowVolunteer = SQLUtil.findVolunteerForLogIn(userId.getText(
        ), password.getText());
5.     if (nowVolunteer == null) {
6.         DialogBox.warning("用户名或密码错误!");
7.         return;
8.     } else {
9.         CacheDataBean.setNowVolunteer(nowVolunteer);
10.    }
11. } else if (adopter.isSelected()) {
12.     CacheDataBean.setIdentity("adopter");
13.     AdopterBean nowUser = SQLUtil.findAdopterForLogIn(userId.getText(), passw
        ord.getText());
14.     if (nowUser == null) {
15.         DialogBox.warning("用户名或密码错误!");
16.         return;
17.     } else {
18.         CacheDataBean.setNowAdopter(nowUser);
19.     }
20. }

```

②待领养宠物管理

(1) 模块编号: **PetManage**

(2) 模块名称:

待领养宠物管理模块

(3) 模块功能:

志愿者可以发布新增的流浪宠物信息, 或者对流浪宠物信息进行修改更新, 志愿者和领养者都对平台上的宠物信息进行查询。

(4) 模块背景描述:

模块主要是用于志愿者对于宠物信息的发布, 修改以及管理查询检索, 同时也用于领养者对于想要领养的宠物信息的查询。

(5) 设计思路及技术要点:

此模块调用 **waitAdopt** 视图

模块性能:

准确性: 志愿者上传的流浪宠物信息需要与流浪宠物实际信息一致

时间特性要求: 志愿者上传的数据需要及时存入数据库中。系统相应操作响应时间不超过 2 秒; 用户查询成绩操作从输入数据, 电脑提交数据到查询结果不超过 2 秒; 数据管理部分, 从提交数据录入到结果返回不超过 2 秒。

输入项目:

待领养宠物管理模块主要分为三部分: 流浪宠物信息入库、流浪宠物信

息更新、流浪宠物信息查询。其中，流浪宠物信息入库的输入项目为：宠物的基础信息，例如宠物种类，性别，年龄，情况描述等等，主要为了领养者能了解宠物的基本情况；流浪宠物信息更新的输入项目也基本相同；流浪宠物信息查询的输入项目是可选的，系统支持模糊查询。

流浪宠物信息入库、流浪宠物信息更新输入项如表 4.5 所示：

名称	标识	数据类型	有效范围	输入方式
宠物 ID	Pno	Char	4-8 位英文数字 中文字符	系统自动生成
宠物名称	pname	Char	4-10 字节英文 数字中文字符	手动输入
性别	psex	Char	雄 / 雌	手动输入
录入志愿者 ID	vno	Char	4-8 位英文数字 中文字符	手动输入
种类	ptype	Char	4-16 字节英文 或者中文字符	手动输入
体重	pweight	Int	2-3 位数字	手动输入
描述	dscrb	Char	200 字节中文	手动输入
领养状态	isapdated	Char	待领养/试养期 /已收养	手动输入

表 4.5 流浪宠物信息入库、流浪宠物信息更新模块输入项目

输出项目：

流浪宠物信息入库的反馈信息：宠物信息录入成功/失败

流浪宠物信息更新的反馈信息：数据更新成功/失败

流浪宠物信息查询模块的输出项目如表 4.6 所示：

名称	标识	数据类型	有效范围
宠物 ID	Pno	Char	4-8 位英文数字 中文字符
宠物名称	pname	Char	4-10 字节英文 数字中文字符

性别	psex	Char	雄 / 雌
录入志愿者 ID	vno	Char	4-8 位英文数字 中文字符
种类	ptype	Char	4-16 字节英文 或者中文字符
体重	pweight	Int	2-3 位数字
描述	dscrb	Char	200 字节中文
领养状态	isapdated	Char	待领养/试养期 /已收养

表 4.6 流浪宠物信息查询模块输出项目

界面设计： 新增宠物信息



该界面用于新增宠物信息。左侧有一个“上传图片”按钮，下方显示了一张柴犬的照片。右侧是“详情”表单，包含以下字段：

- 宠物名称：文本输入框，值为“憨憨”
- 宠物类别：下拉菜单，显示“中华田园犬”
- 宠物性别：两个单选按钮，其中“雄”被选中
- 体重：文本输入框，值为“12”
- 状态描述：多行文本输入框，值为“状态良好，已接种疫苗”

底部有两个按钮：“发布”和“清空页面”。

图 4.7 新增宠物信息输入界面

修改宠物信息



该界面用于修改宠物信息。左侧有一个“更换图片”按钮，下方显示了一张金毛犬的照片。右侧是“修改”表单，包含以下字段：

- 宠物名称：文本输入框，值为“55”
- 宠物类别：下拉菜单，显示“中华田园犬”
- 宠物性别：两个单选按钮，其中“雄”被选中
- 体重：文本输入框，值为“12”
- 状态描述：多行文本输入框，值为“不过是用来自扩充数据库的罢了”

底部有两个按钮：“提交”和“返回”。

图 4.8 修改宠物信息输入界面

查询宠物信息

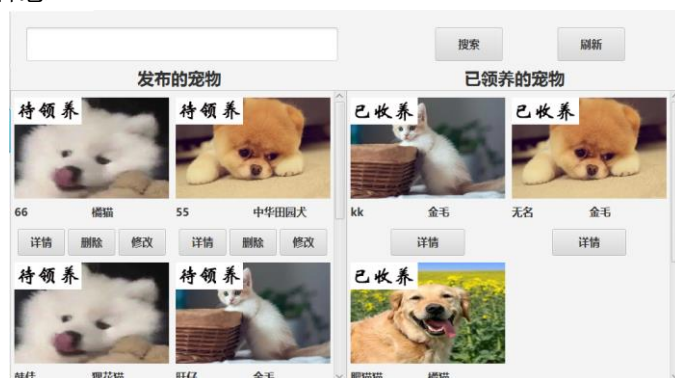


图 4.9 查询宠物信息结果预览界面

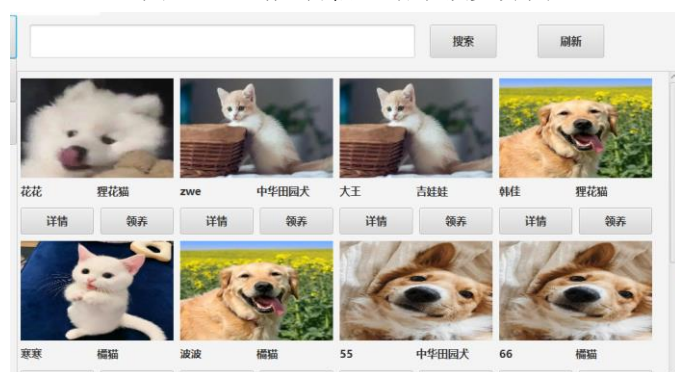


图 4.10 志愿者查询其所负责的宠物信息结果预览界面

(6) 模块算法设计:

流浪宠物信息入库模块算法流程图如图 4.11 所示:

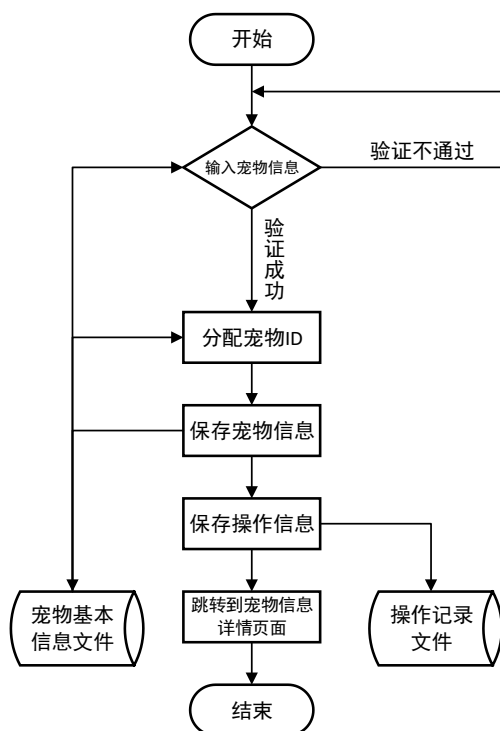


图 11 流浪宠物信息入库模块算法流程图

流浪宠物信息更新模块算法流程图如图 4.12 所示：

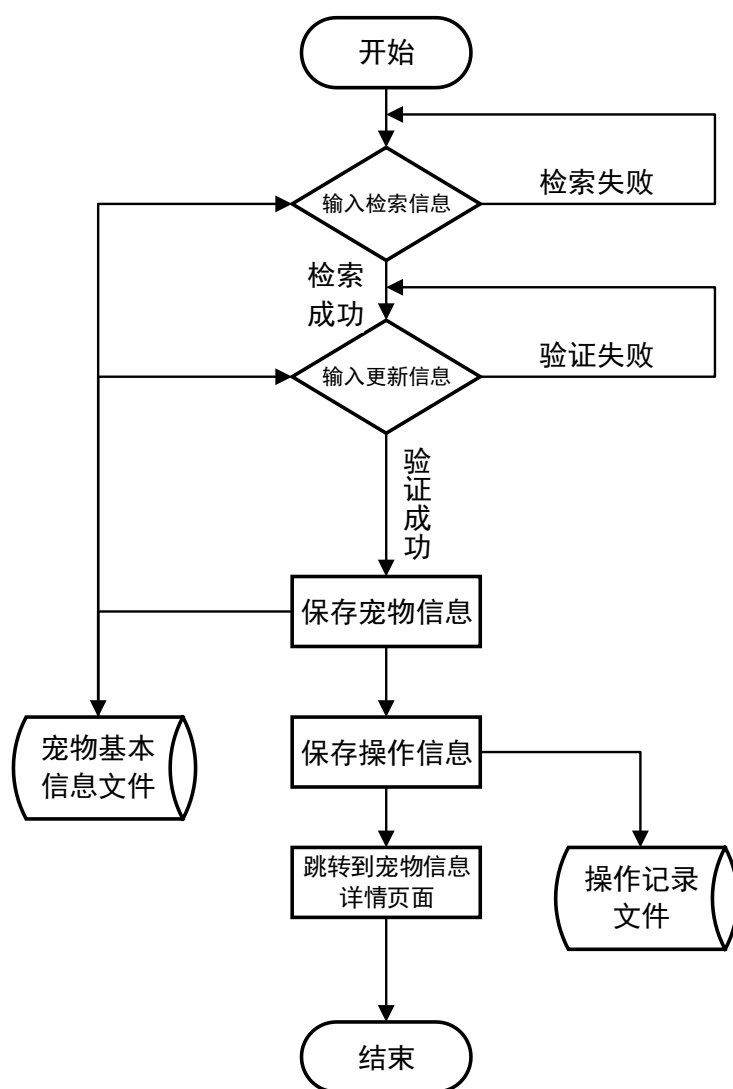


图 4.12 流浪宠物信息更新模块算法流程图
流浪宠物信息查询模块算法流程图如图 4.13 所示：

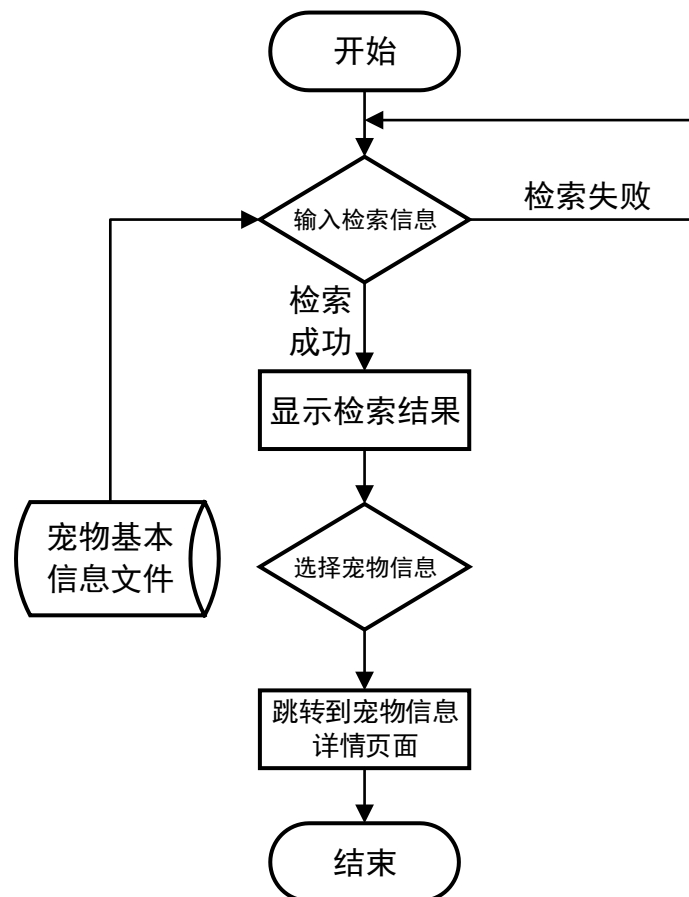


图 4.13 流浪宠物信息查询模块算法流程图

(7) 关键代码及描述:

```

go
create view waitAdopt as
select * from pet where isadopted = '待领养'

```

```

1.  /** 管理员发布宠物信息 使用存储过程 pet_insert
2.  * @param pet 存储宠物信息的类的对象
3.  * */
4.  public static void addPet(PetBean pet) {
5.      String add = "EXEC pet_insert ?, ?, ?, ?, ?, ?, ?, ?";
6.      try (Connection conn = SQLUtil.getConnection()) {
7.          PreparedStatement st = conn.prepareStatement(add);
8.          // 将用户数据输入 SQL 语句
9.          PetUtil.PetAddToSQL(st, pet);
10.         st.executeUpdate();
11.     } catch (SQLException e) {
12.         DialogBox.error(e);
13.     }
14. }

```

```

1.  /**
2.   * 查询检索宠物信息
3.   * @param userKeys 用户输入的检索关键词
4.   * @return 检索到的宠物列表
5.   */
6.  public static List<PetBean> getGoodsByKeyWords(String[] userKeys) {
7.      String sql = "select * from pet";
8.      List<PetBean> goodsList = new LinkedList<>();
9.      try (Connection conn = SQLUtil.getConnection()) {
10.         PreparedStatement st = conn.prepareStatement(sql);
11.         try (ResultSet rs = st.executeQuery()) {
12.             while (rs.next()) {
13.                 PetBean pet = PetUtil.findPetsByKey(rs, userKeys);
14.                 if (pet != null) {
15.                     goodsList.add(pet);
16.                 }
17.             }
18.         } catch (Exception inEx) {
19.             inEx.printStackTrace();
20.         }
21.     } catch (SQLException e) {
22.         e.printStackTrace();
23.     }
24.     return goodsList;
25. }

1.  /**
2.   * 判断 rs 中的数据行是否符合检索要求
3.   * @param rs 数据库检索结果
4.   * @param userKeys 用户输入的检索关键词
5.   * @return 检索到的宠物列表
6.   */
7.  public static PetBean findPetsByKey(ResultSet rs, String[] userKeys) throws
    IOException, SQLException {
8.      PetBean pet = ResultToPet(rs);
9.      String petString = pet.toString();
10.     for (String user : userKeys) {
11.         if (petString.contains(user)) {
12.             return pet;
13.         }
14.     }
15.     return null;
16. }

```


③流浪宠物领养申请

(1) 模块编号: AdpApply

(2) 模块名称:

流浪宠物领养申请模块

(3) 模块功能:

领养者在浏览宠物信息界面可以选择中意的宠物提交领养申请(此处调用领养申请的存储过程,负责该宠物的志愿者收到领养申请后对领养者进行信息审核,与领养者交流后可同意或拒绝领养申请。若领养申请已被同意,领养者可以支付保证金试养流浪宠物。对领养超数的领养者申请,通过触发器禁止申请写入。

(4) 模块背景描述:

模块用于辅助领养者提交对于宠物的领养申请,并等待志愿者的审批以及保证金的提交。

(5) 设计思路及技术要点:

性能要求:

及时性:对于领养者的领养申请要及时传入数据库,防止领养申请得不到及时处理。

输入项目:

领养者申请领养的输入项目如图 4.14 所示:

名称	标识	数据类型	有效范围	输入方式
领养申请号	adp_applyno	char	8 位中文或英文字符	系统自动生成
领养者 ID	ano	char	8 位中文或英文字符	系统识别
宠物 ID	pno	char	8 位中文或英文字符	系统识别
申请原因描述	Apply_reason	Char	200 位字符	人工输入
申请状态	isapproval	Char	拒绝/通过/未批	人工输入
拒绝原因	refuse_reason	Char	100 位字符	人工输入
试养开始时间	trail_time	Date	时间	系统识别

试养期长度	Traillenght	Int	2-3 位数字	人工输入
正式领养时间	formal_time	Date	时间	系统识别
保证金状态	bondstate	Char	没收/归还/已付/未付	人工输入

图 4.14 领养者申请领养的输入项目

志愿者审核领养申请的输入项目如图 4.15 所示：

名称	标识	数据类型	有效范围	输入方式
领养申请号	adp_applyno	char	8 位中文或英文字符	系统识别
申请状态	isapproval	Char	拒绝/通过/未批	人工输入
拒绝原因	refuse_reason	Char	100 位字符	人工输入
试养开始时间	trail_time	Date	时间	系统识别
试养期长度	Traillenght	Int	2-3 位数字	人工输入

图 4.15 志愿者审核领养申请的输入项目

保证金支付的输入项目如图 4.16 所示：

名称	标识	数据类型	有效范围	输入方式
领养申请号	adp_applyno	char	8 位中文或英文字符	系统识别
保证金状态	bondstate	Char	没收/归还/已付/未付	人工输入

图 4.16 保证金支付的输入项目

输出项目：

领养申请提交的反馈信息：领养申请成功/失败

领养申请审批的反馈信息：领养申请审批成功/失败

保证金支付的反馈信息：保证金支付成功/失败

界面设计：

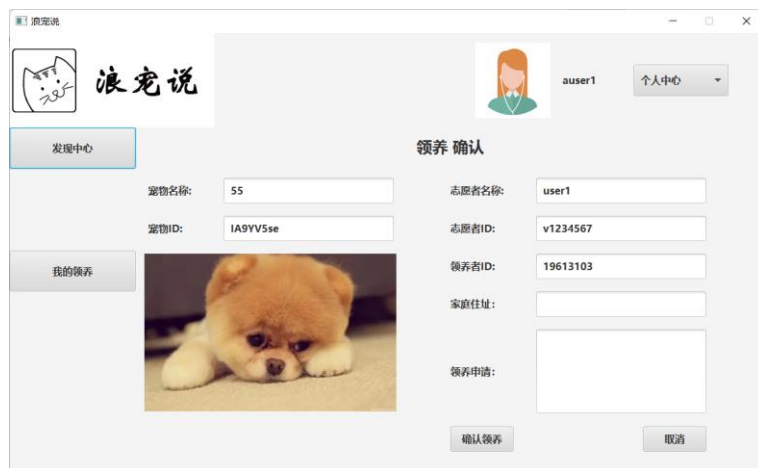


图 4.17 领养申请确认

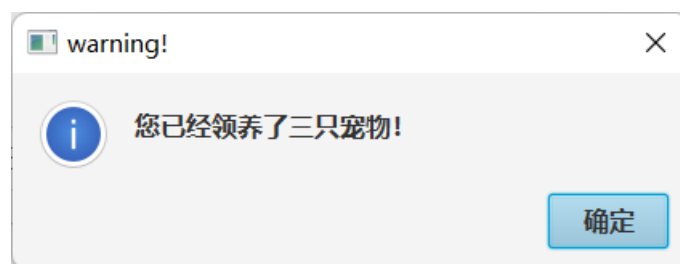


图 4.18 领养数量达到限额警告

(6) 模块算法设计：

领养者提交宠物领养申请的同时，若恰好该宠物被批准被其他人领养，则检索失败，无法领养该宠物；领养申请需要通过约束条件，若输入的申请不满足约束，则验证不通过需要重新输入，若该领养者的违约次数超数，则无权限领养宠物，验证无法通过。若以上条件都通过，则领养申请保存进领养申请文件。

领养者申请领养的算法流程图如图 4.19 所示：

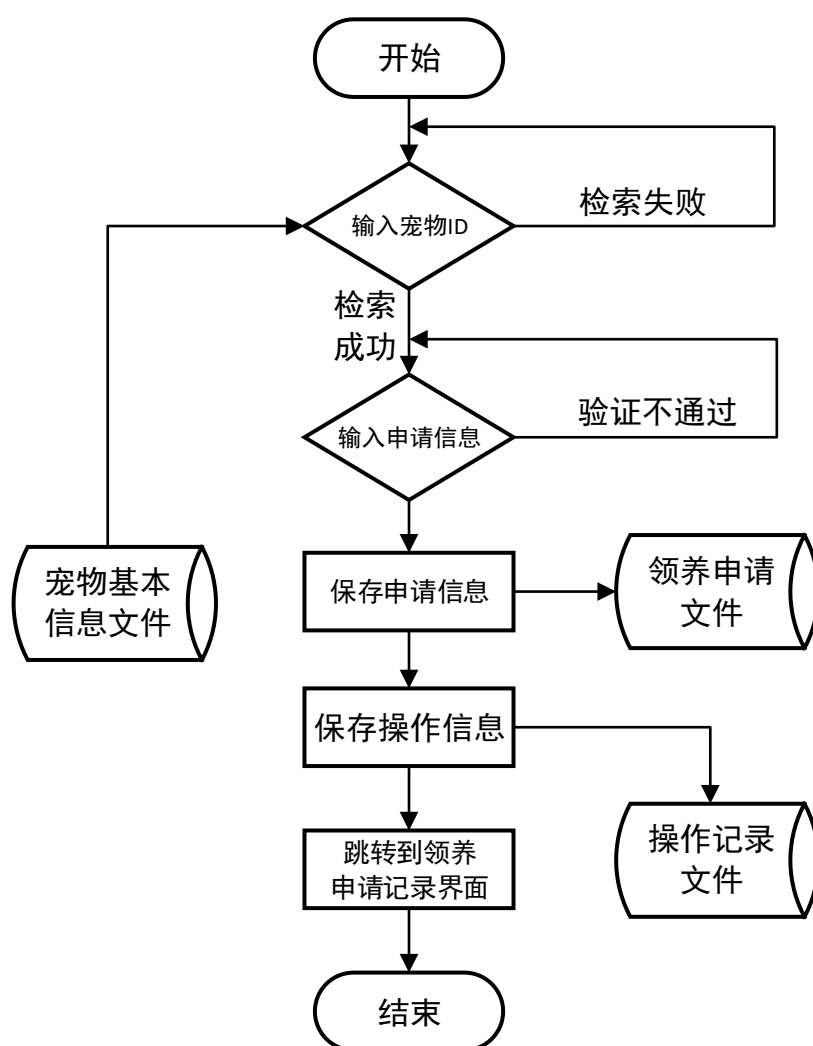


图 4.19 领养者申请领养的算法流程图

领养申请文件中的申请表首先由志愿者预设的条件进行过滤，不满足条件的申请直接被系统拒绝；通过志愿者预设的过滤条件的领养申请，交由志愿者人工审批。

志愿者审核领养申请的算法流程图如图 4.20 所示：

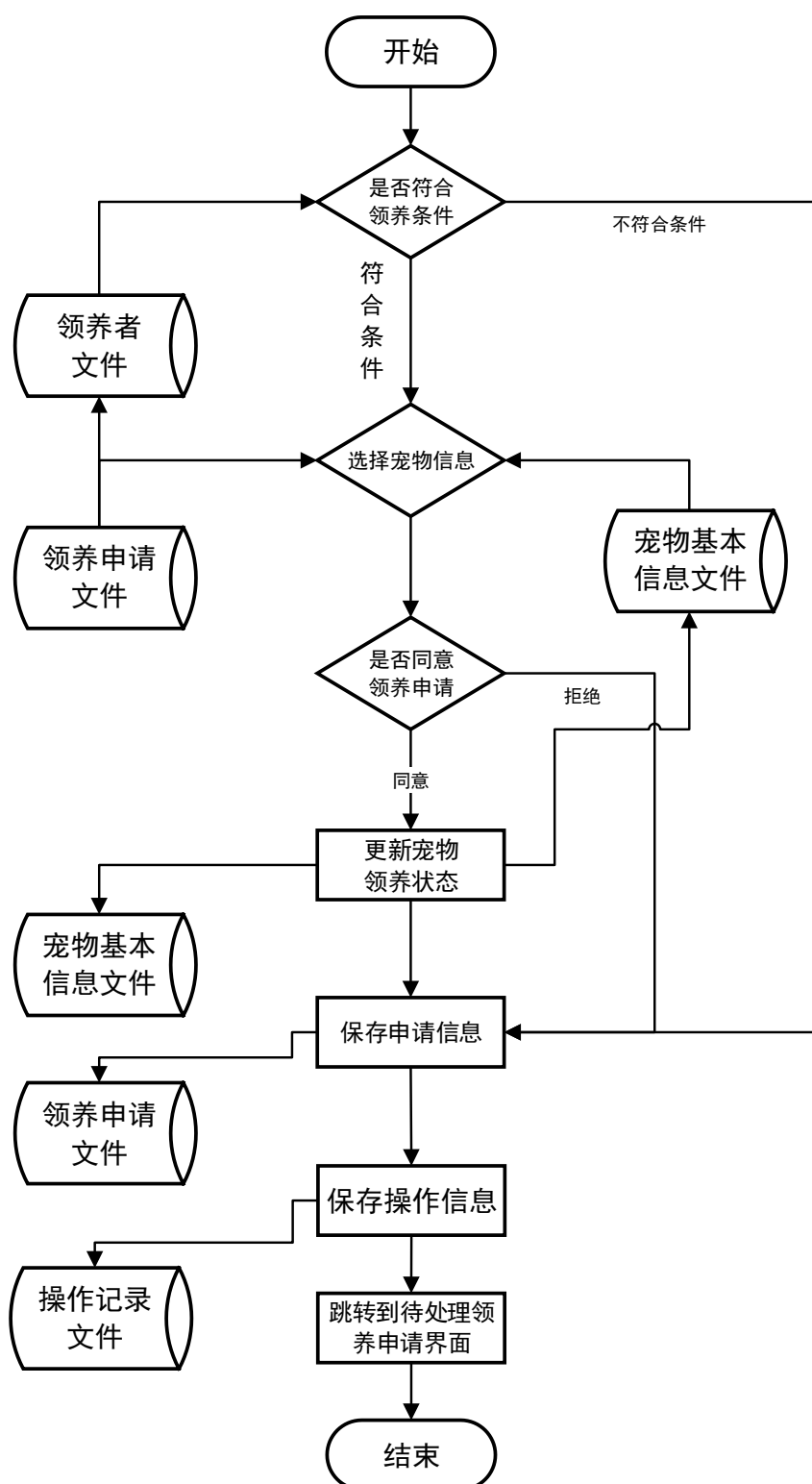


图 4.20 领养申请审核的算法流程图

系统仅允许支付已被批准领养的申请对应的保证金，成功支付保证金后，系统生成分配保证金单 ID 写入保证金文件同时更改宠物基本信息文件的保证金支付情况。

志愿者审核领养申请的算法流程图如图 4.21 所示：

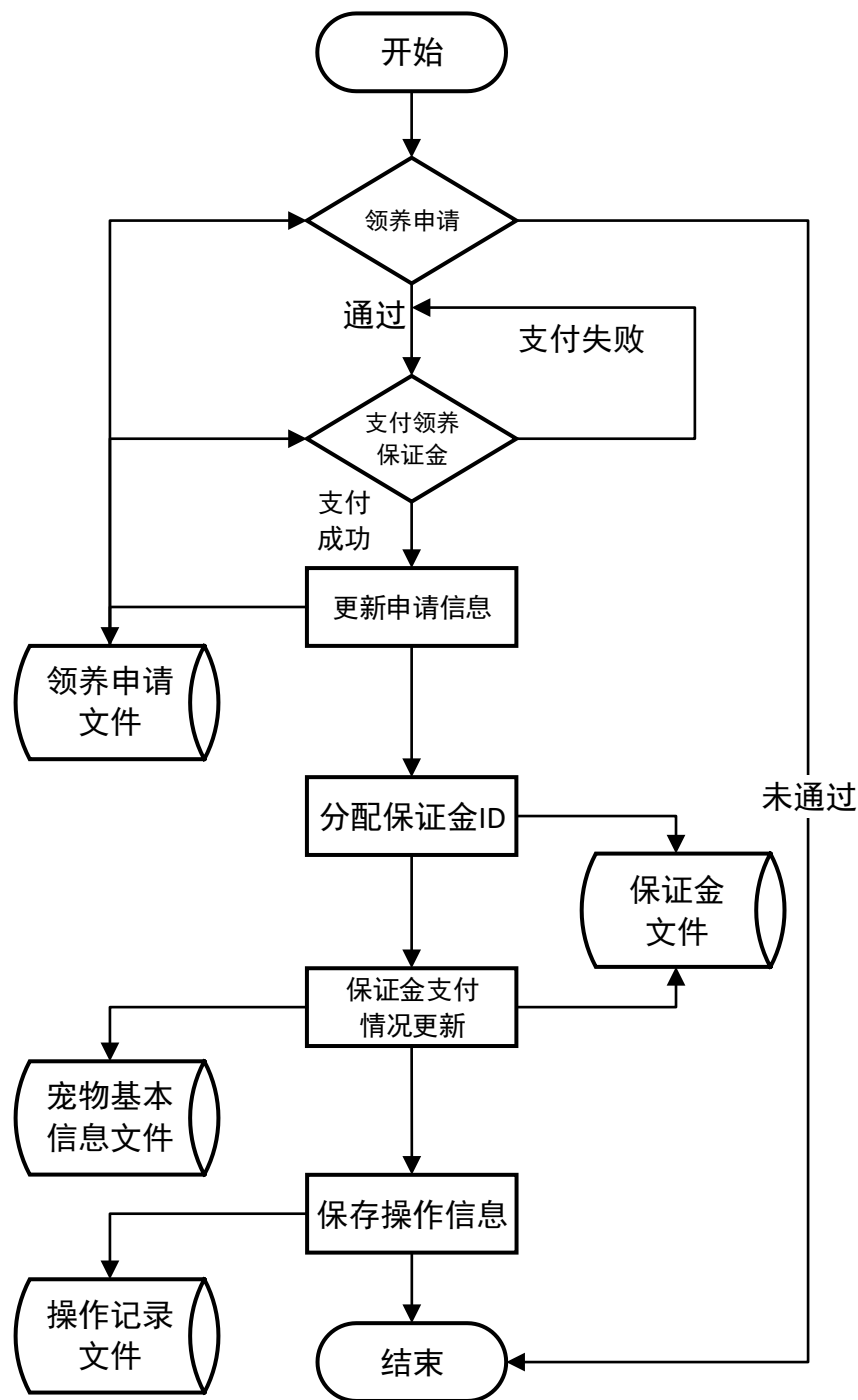


图 4.21 保证金支付的算法流程图

(7) 关键代码及描述:

```
1.  /**
2.   * 领养宠物
3.   * @param adpapply 领养申请信息
4.   * @param pet 宠物信息
5.   * @return 领养是否成功
6.   * */
7.  public static boolean AdoptPet(AdpapplyBean adpapply, PetBean pet) throws SQLException {
8.      String sql = "insert into adpapply(adp_applyno, ano, pno, apply_reason, isapproval, refuse_reason, trail_time, traillength, formal_time, bondstate) values (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
9.      try (Connection conn = SQLUtil.getConnection()) {
10.          PreparedStatement st = conn.prepareStatement(sql);
11.          AdpapplyUtil.AdpapplyAddToSQL(st, adpapply);
12.          if (st.executeUpdate() <= 0 || !SQLUtil.updatePet(pet)) {
13.              pet.setIsapdated("待领养");
14.              SQLUtil.updatePet(pet);
15.              return false;
16.          }
17.      } catch (SQLException e) {
18.          e.printStackTrace();
19.          pet.setIsapdated("待领养");
20.          SQLUtil.updatePet(pet);
21.          return false;
22.      }
23.      return true;
24. }
```

go--存储过程 领养申请 /存储过程

create procedure sendapply

 (@c1 char(8),@c2 char(8),@c3 char(8),@c4 varchar(200)) as

insert into adpapply (adp_applyno,ano,pno,apply_reason,isapproval)

values (@c1,@c2,@c3,@c4,'未批')

go--领养超过3个的不允许申请领养 /触发器

create trigger adopter_apply1 on adpapply for insert as

declare @c1 int

select @c1 = count(*) from adpapply

where ano = (select ano from inserted) AND (trail_time is NOT NULL)

if (@c1>=3)

begin

 ROLLBACK

 PRINT '您已领养了三只宠物'

end

④存储过程

(1) 模块编号: **CreateProc**

(2) 模块名称:

创建存储过程模块

(3) 模块功能:

包括新增志愿者、领养者、宠物信息, 新增领养申请, 同意/拒绝领养审批, 志愿者交保证金的存储过程。

(4) 模块背景描述:

该模块用于数据库创建存储过程, 方便前端调用。

(5) 关键代码:

go--存储过程 插入志愿者

```
create procedure vol_insert
    (@c1 char(8), @c2 char(2), @c3 char(11), @c4 varchar(12), @c5 varchar(10), @c6
    varchar(16)) as
insert into volunteer (vno, vsex, tel, vloc, vname, vpsw)
values (@c1, @c2, @c3, @c4, @c5, @c6)
```

```
EXEC vol_insert v1234567, 男, 15962964901, 江苏省常州市, user1, pswtest1
```

```
EXEC vol_insert v1234568, 男, 15962961111, 江苏省常州市, user2, pswtest2
```

```
EXEC vol_insert v1234569, 男, 15962961221, 江苏省苏州市, user3, pswtest3
```

go--存储过程 插入领养者

```
create procedure adopter_insert
    (@c1 char(8), @c2 char(2), @c3 char(11), @c4 varchar(12), @c5 varchar(10), @c6
    varchar(16), @c7 int) as
insert into adopter (ano, asex, tel, aloc, aname, apsw, contractcnt, money)
values (@c1, @c2, @c3, @c4, @c5, @c6, 0, @c7)
```

```
EXEC adopter_insert a1234567, 女, 18962182821, 江苏省常州市, auser1, pwsss1, 200000
```

```
EXEC adopter_insert a1234568, 女, 18962122821, 江苏省常州市, auser2, pws123, 800
```

```
EXEC adopter_insert a1234569, 女, 18962122811, 江苏省常州市, auser3, pws143, 1000
```

go--存储过程 插入宠物

```
create procedure pet_insert
    (@c1 char(8), @c2 varchar(10), @c3 char(2), @c4 char(8), @c5 varchar(20), @c6 int, @c7
    varchar(200), @c8 char(6)) as
insert into pet (pno, pname, psex, vno, ptype, pweight, dscrbr, isapdated)
values (@c1, @c2, @c3, @c4, @c5, @c6, @c7, @c8)
```

```
EXEC pet_insert p1111111, 旺仔, 雄, v1234567, 金毛, 69, 非常健康且性格活泼, 待领养
```

```
EXEC pet_insert p1111112, 波波, 雄, v1234567, 橘猫, 30, 很粘人, 待领养
```

```
EXEC pet_insert p1111113, 肥猫猫, 雌性, v1234567, 橘猫, 40, 有点肥胖, 待领养
```

```
EXEC pet_insert p1111114, 大王, 雄, v1234568, 吉娃娃, 54, 瘦弱未驱虫, 待领养
```


go--存储过程 领养申请

```
create procedure sendapply
    (@c1 char(8),@c2 char(8),@c3 char(8),@c4 varchar(200)) as
insert into adpapply (adp_applyno,ano,pno,apply_reason,isapproval)
values (@c1,@c2,@c3,@c4,'未批')
```

```
EXEC sendapply s1111111,a1234567,p1111111,特别喜欢金毛
EXEC sendapply s1111117,a1234567,p1111112,特别喜欢橘猫
EXEC sendapply s1111119,a1234568,p1111112,我也特别喜欢橘猫
EXEC sendapply s1121137,a1234567,p1111114,小动物好可爱
```

go--存储过程 领养审批 同意

```
create procedure agreeadp
    (@c1 char(8),@c2 int) as
if ((select count(*) from adpapply where adp_applyno=@c1 AND bondstate='通过')=0)
begin
    update adpapply set isapproval='通过'
    ',trail_time=GETDATE(),traillenght=@c2,bondstate='未付'
    where adp_applyno = @c1
end
```

```
EXEC agreeadp s1111117,1
EXEC agreeadp s1111119,1
```

go--存储过程 领养审批 拒绝

```
create procedure refuseadp (@c1 char(8),@c2 varchar(100)) as
update adpapply set isapproval='拒绝',refuse_reason=@c2
where adp_applyno = @c1
```

```
EXEC refuseadp s1111127,我觉得你不是真正的喜欢
```

go--存储过程 交保证金

```
create procedure paybond (@c1 char(8)) as
declare @c2 int
if ((select bondstate from adpapply where adp_applyno=@c1) = '未付')--未被批准领养的或
已交过保证金的不允许交保证金
begin
    update adopter set money=money-(select bond from pet,pettype,adpapply where
pettype.ptype=pet.ptype AND adpapply.pno = pet.pno AND adpapply.adp_applyno = @c1 )
    update adpapply set bondstate = '已付' where adp_applyno=@c1
    update pet set isapdated = '试养期' where pet.pno = (select pno from adpapply where
adp_applyno=@c1)
end
else PRINT '申请还未通过或已交过保证金，不能交保证金'
```

④触发器

(1) 模块编号: **CreateTrigger**

(2) 模块名称:

创建存储器模块

(3) 模块功能:

对数据库操作进行控制保证安全性,同时防止前端限制不足造成数据库完整性被破坏。

(4) 模块背景描述:

该模块用于数据库创建触发器,和前端限制一同保证数据安全、完整性。

(5) 关键代码:

go--超过3次违约的不允许申请领养

```
create trigger adopter_apply on adpapply
for insert
as
    declare @c1 int
    select @c1 = contractcnt from adopter
    where ano = (select ano from inserted)
    if (@c1>=3)
    begin
        ROLLBACK
        PRINT '您违约次数过多, 不允许申请领养'
    end
EXEC sendapply s1111112, a1234511, p1111111, 我也特别喜欢金毛
```

go--领养超过3个的不允许申请领养

```
create trigger adopter_apply1 on adpapply
for insert
as
    declare @c1 int
    select @c1 = count(*) from adpapply
    where ano = (select ano from inserted) AND (trail_time is NOT NULL)
    if (@c1>=3)
    begin
        ROLLBACK
        PRINT '您领养的宠物过多, 请给其他爱心人士一点机会'
    end
end
```

go--已被领养的宠物不允许申请领养

```
create trigger adopter_apply2 on adpapply
for insert
as
    declare @c1 int
    select @c1 = count(*) from adpapply
```

```

where pno = (select pno from inserted) AND isapproval = '通过'
if (@c1>=1)
begin
    ROLLBACK
    PRINT '您想领养的宠物已被别人领走啦，请看看别的宠物呢'
end
EXEC sendapply s1111113, a1231122, p1111111, 我也也特别喜欢金毛

go--领养者和对应志愿者必须在同一地址
create trigger adopter_apply3 on adpapply
for insert
as
    declare @c1 int
    select @c1 = count (*)
    from adopter, volunteer, pet, adpapply
    where adpapply.pno = pet.pno AND pet.vno = volunteer.vno AND adpapply.pno = (select
pno from inserted) AND volunteer.vloc=adopter.aloc AND adopter.ano = (select ano from
inserted)
    if (@c1 = 0)
    begin
        ROLLBACK
        PRINT '您的所在地和对该宠物负责的志愿者不一致呢，不方便领养，请看看别的宠物呢'
    end

EXEC sendapply s1111114, a1231122, p1111112, 我特别喜欢橘猫

go--批准领养必须在领养者钱大于定金
create trigger volunteer_agree on adpapply
for update
as
    declare @c1 int
    select @c1 = money
    from adopter
    where adopter.ano = (select ano from inserted)
    declare @c2 int
    select @c2 = bond
    from pet, pettype
    where pet.pno = (select pno from inserted) and pet.ptype = pettype.ptype
    if ( @c1 < @c2 )
    begin
        ROLLBACK
        PRINT '领养者账户余额小于该宠物的保证金，不能批准他领养哦'
    end
end

```

五、结束语

在本次数据库系统原理实践项目中，我们深刻体验到软件开发周期的过程充满艰辛和困难，软件的功能需求分析、概念模型设计、数据库设计、编码等每一步都不能有差错，否则就要回头重来。在此过程中，也有许多收获。我们初步掌握了实际项目中概念模型的设计方法，熟练使用 SQL 语句，掌握了前后端连接的方法。通过本次实践，我们认识到数据库在实际开发中的重要作用，概念模型设计是数据库设计合理的基础，我们深刻理解了实体与属性的划分原则、数据模型的优化准则。

目前，系统仍有许多不足之处，

- 1) 系统的违约惩罚模块未完善，不完全满足实际应用。原功能应是志愿者申请惩罚违约的领养者，管理员审批惩罚申请，但考虑到课设时间紧迫，并且此“惩罚申请-审批”模式与已完善的“宠物领养申请-审批”模式类似，在代码实现方法一样，我们将该模块功能改为管理员直接依据宠物近况信息来提交惩罚，省略了相应志愿者申请惩罚的环节。于是将 E-R 图设计为“管理员-宠物近况-违约情况”多对多对多关系，简化了功能并且丰富了 E-R 图中关系的类型。后期改进：将该模块设计符合实际应用，即志愿者申请惩罚，管理员再审批。
- 2) UI 界面不够精致美观。由于时间原因，我们在 UI 界面设计仅满足应用需要的元素，在布局、风格上尚未过多考虑，可能不够美观。
后期改进：调节 UI 界面元素大小、布局，采用更美观的元素图标。
- 3) 缺少并发控制。目前系统仅考虑功能性、安全性、完整性需求，未进行并发度检测。
后期改进：先对进行并发测试，检查并发错误的来源，然后采用封锁技术控制并发。

参考文献

- [1]. 王少锋，面向对象技术 UML 教程 [M]，北京-清华大学出版社，2004 年
- [2]. 王珊，数据库系统概论 [M]，高等教育出版社，2014 年
- [3]. 李波、孙宪丽、关颖，PowerDesigner16 从入门到精通 [M]，清华大学出版社，2016 年
- [4]. 蒋辉，SQL Server2019 数据库应用教程 [M]，重庆大学出版社，2021 年