

1 实验环境

实验操作系统：Windows

主要开发工具：后端数据库使用 MySQL8.0 关系数据库管理系统、PowerDesigner。前端界面使用 python 语言实现，环境为 python3.9。前端界面框架为 PyQt5 和 PySide2，使用 pymysql 进行前后端连接，开发工具为 pycharm、Qt Designer。

2 实验过程

2.1 系统功能

功能：登录/注册/退出登录（用户/管理员）、用户端：查看待领养动物、申请领养，查看我的申请；管理员端：添加、修改、删除动物信息，批准/驳回用户申请，查看我已审阅过的申请

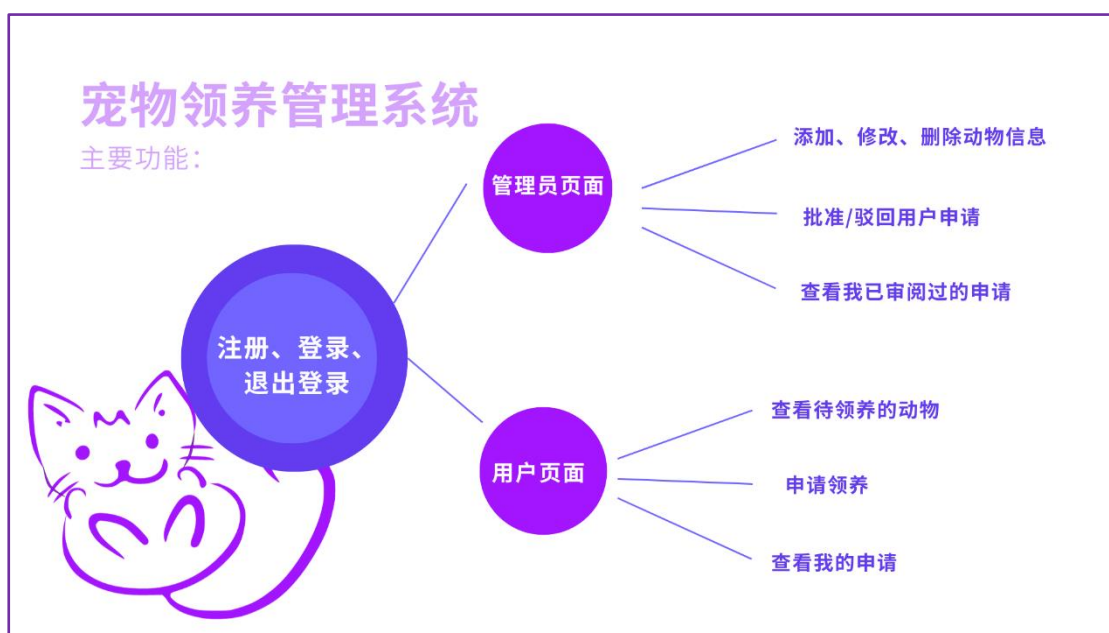


Figure 1.宠物管理系统主要功能

以下对每种功能页面展示：

一、基础的登录/注册/退出登录（用户/管理员）

1. 用户/管理员登录，可以勾选“我是管理员”，登录管理员账户。点击“注册新账号”可以跳转到注册界面。按 esc 键可以退出/打开图形界面。

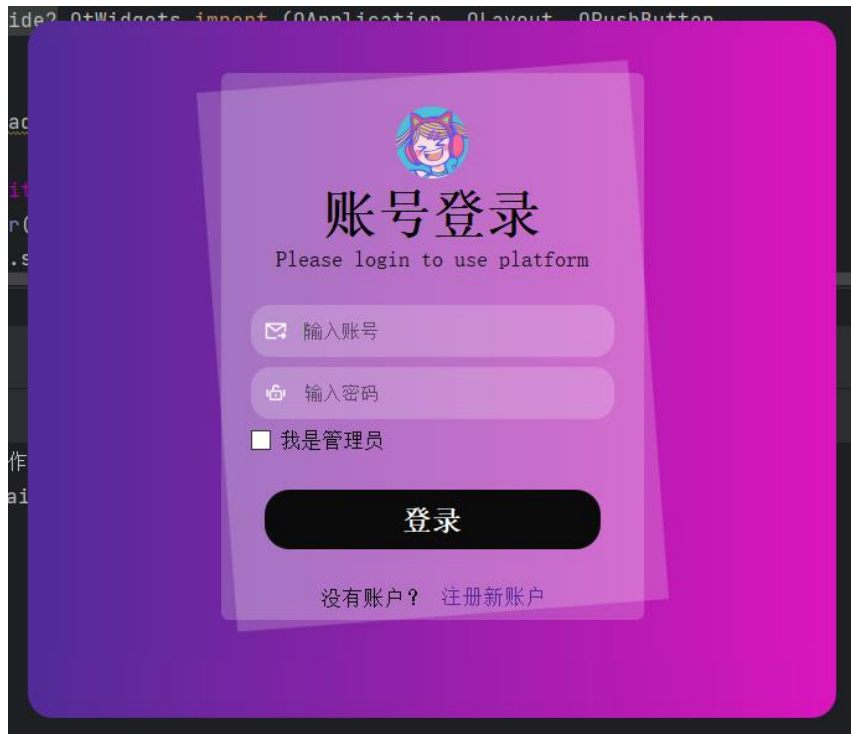


Figure 2.账号登录

2. 用户/管理员注册，可以勾选“我是管理员”，注册管理员账户。点击“登录账号”可以跳转到登录界面。按 **esc** 键可以退出/打开图形界面。



Figure 3.账号注册

3. 退出登录，按 **exit** 退出账户登录，回到登录界面。

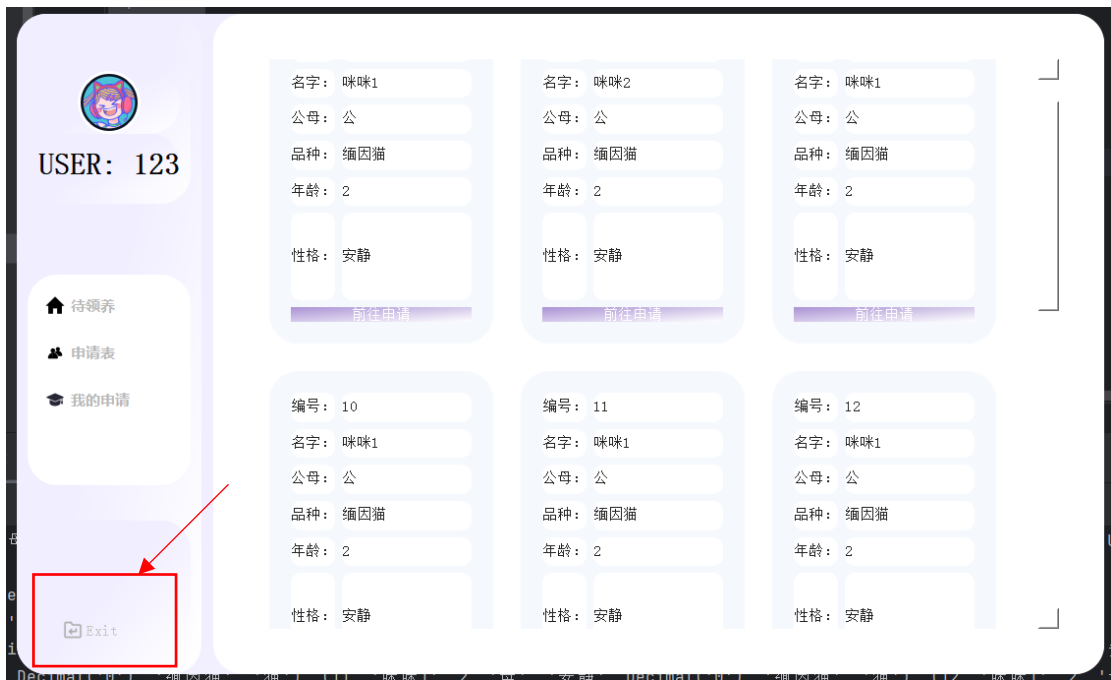


Figure 4.退出登录

二、管理员页面：

(1) 添加、修改、删除动物信息

管理员登录后就进入以下界面，可以进行新增：



Figure 5.添加、查看动物信息

修改删除某条信息：单击修改删除按钮跳转到修改删除页面：

Figure 6.点击可以修改删除信息

Figure 7.修改删除页面

- (2) 处理领养申请，点击“待处理申请”，管理员可以看到未处理的领养申请，选择批准或驳回该申请。



Figure 8.查看待处理申请

(3) 查看我已审阅过的申请，点击“我已处理”查看。

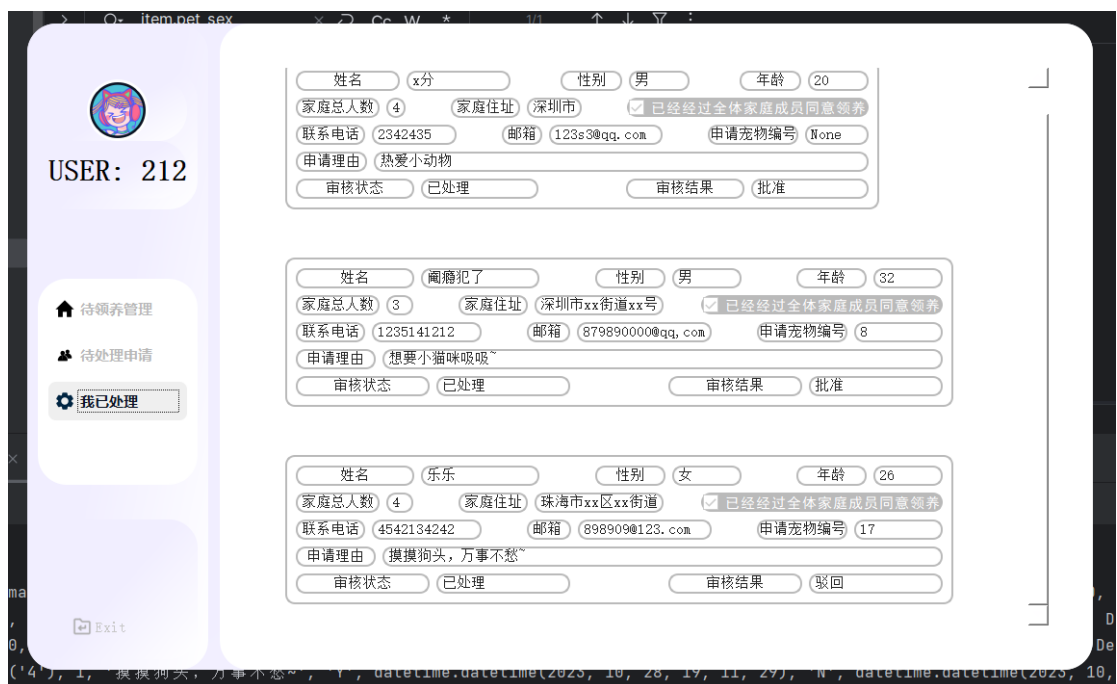


Figure 9.查看已处理申请

三、用户页面：

(1) 查看待领养的动物，可以点击“前往申请”申请该动物

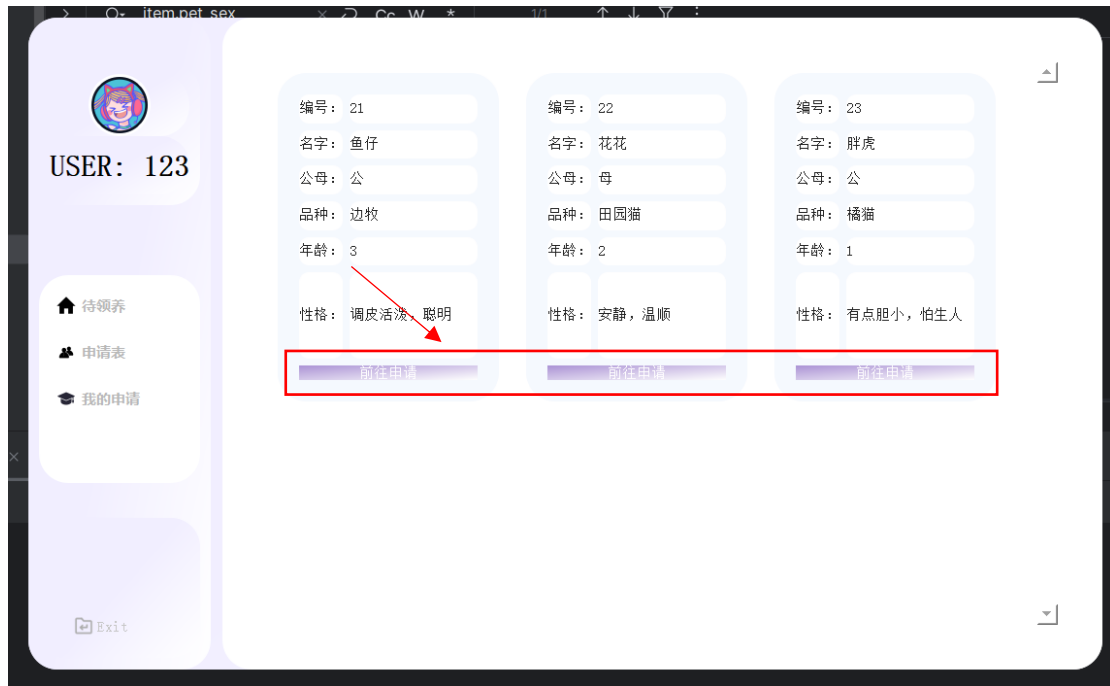


Figure 10.待领养界面

(2) 申请领养，填写领养信息，点击申请即可：

Figure 11.领养申请表填写界面

(3) 查看我的申请，未被处理的申请可以点击撤销申请，已经被处理的申请可以看到最后申请的结果：

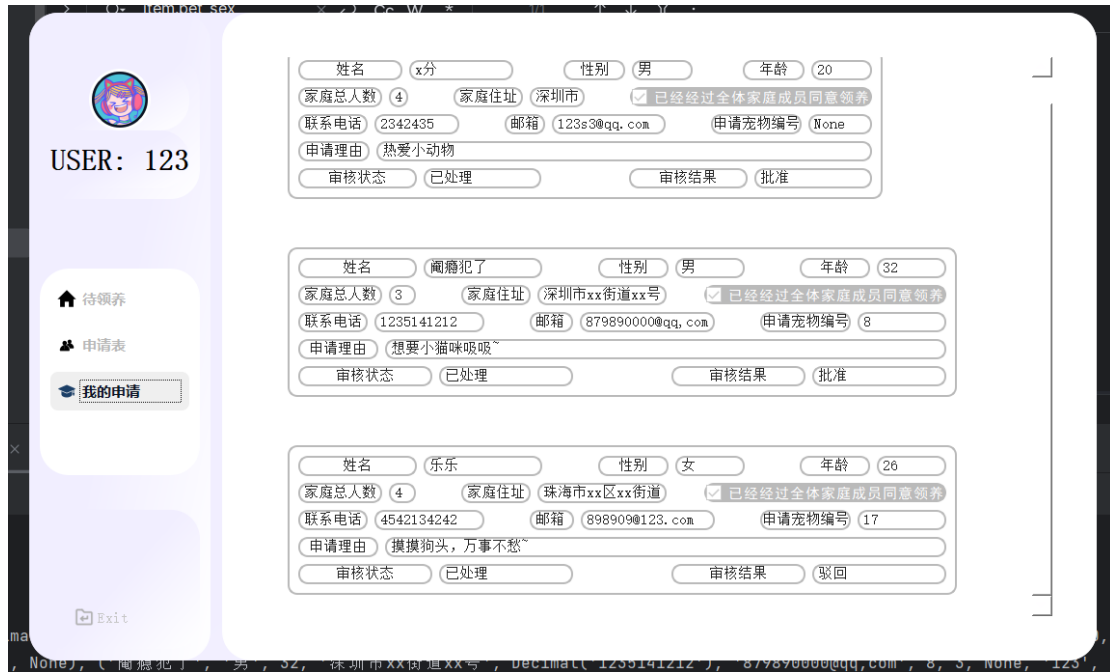


Figure 12.用户看到已经审核过的结果

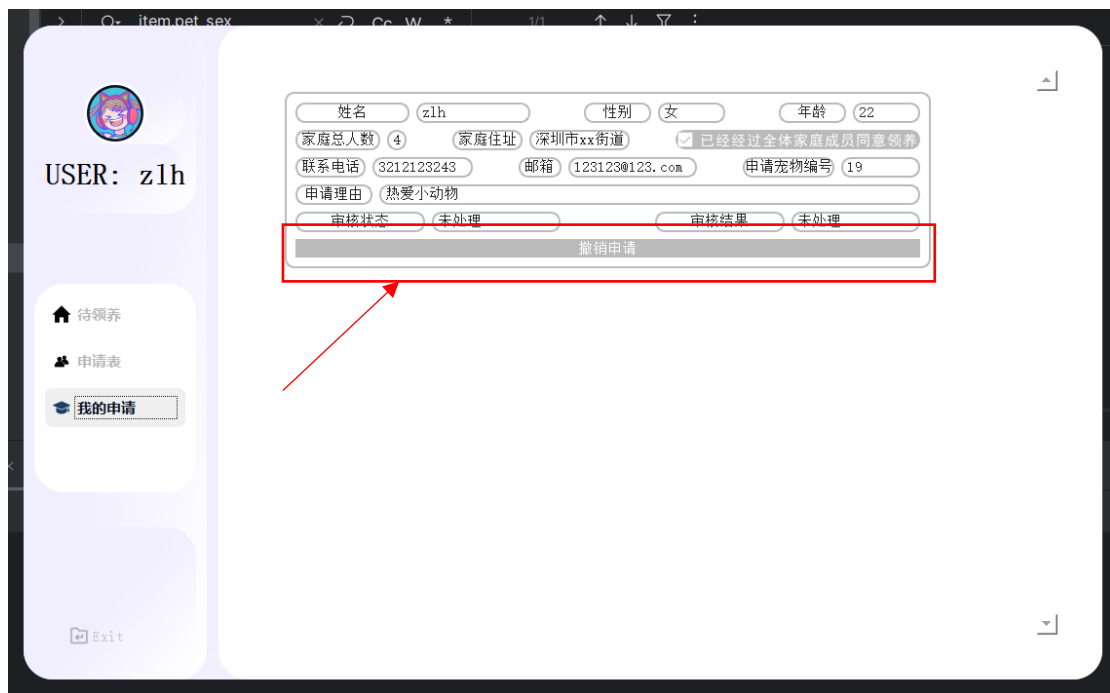
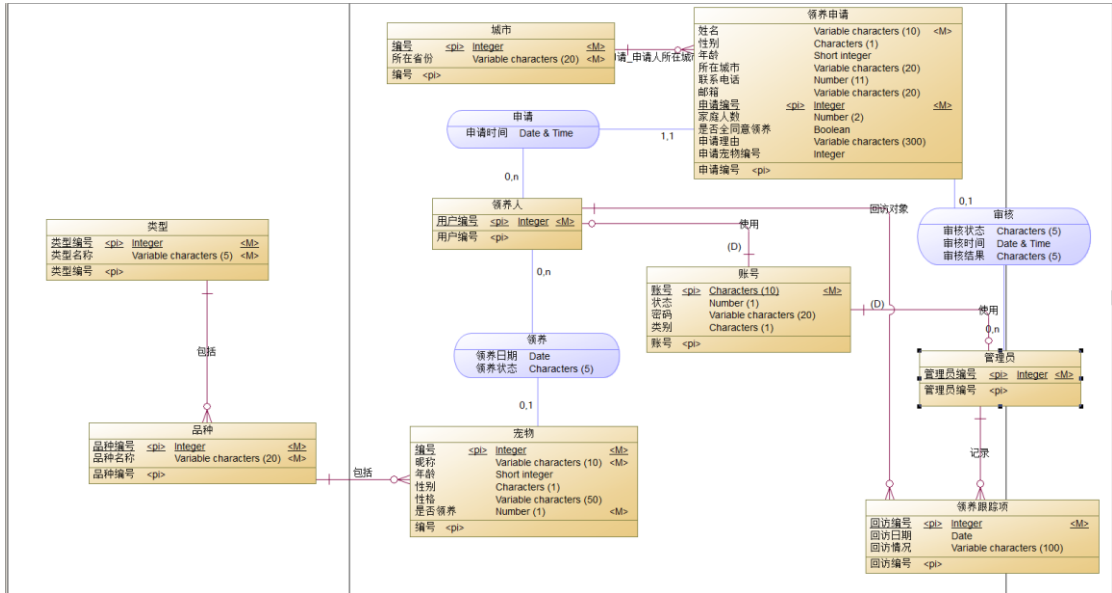


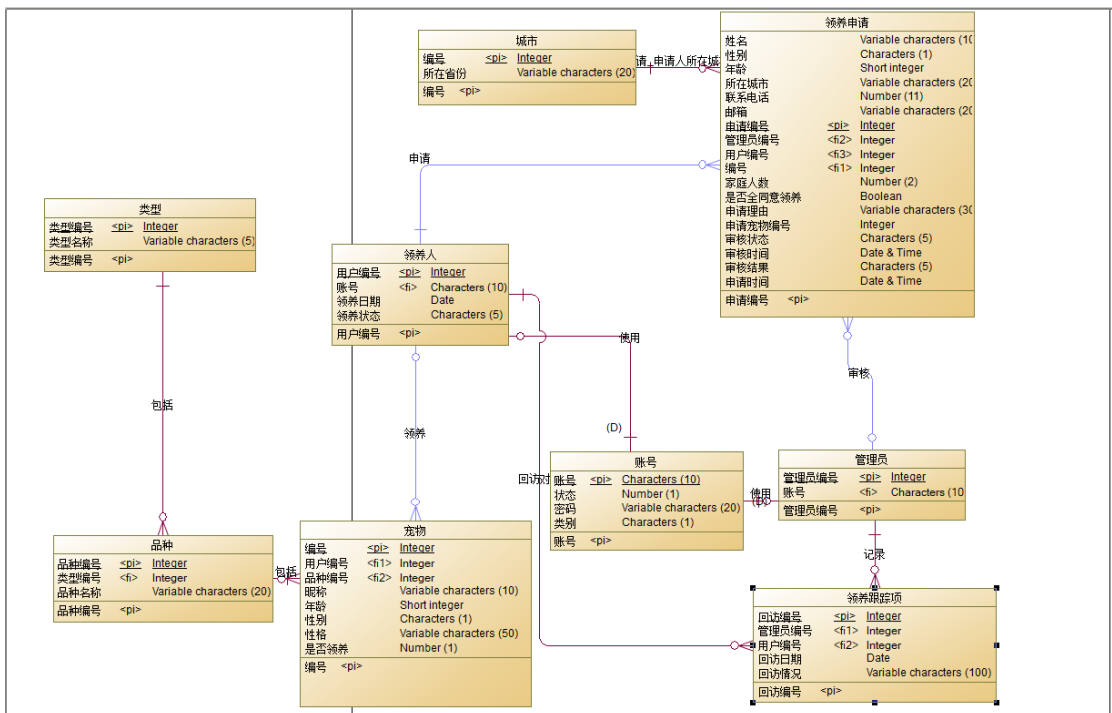
Figure 13.用户可以撤销还未处理的申请

2.2 数据库设计

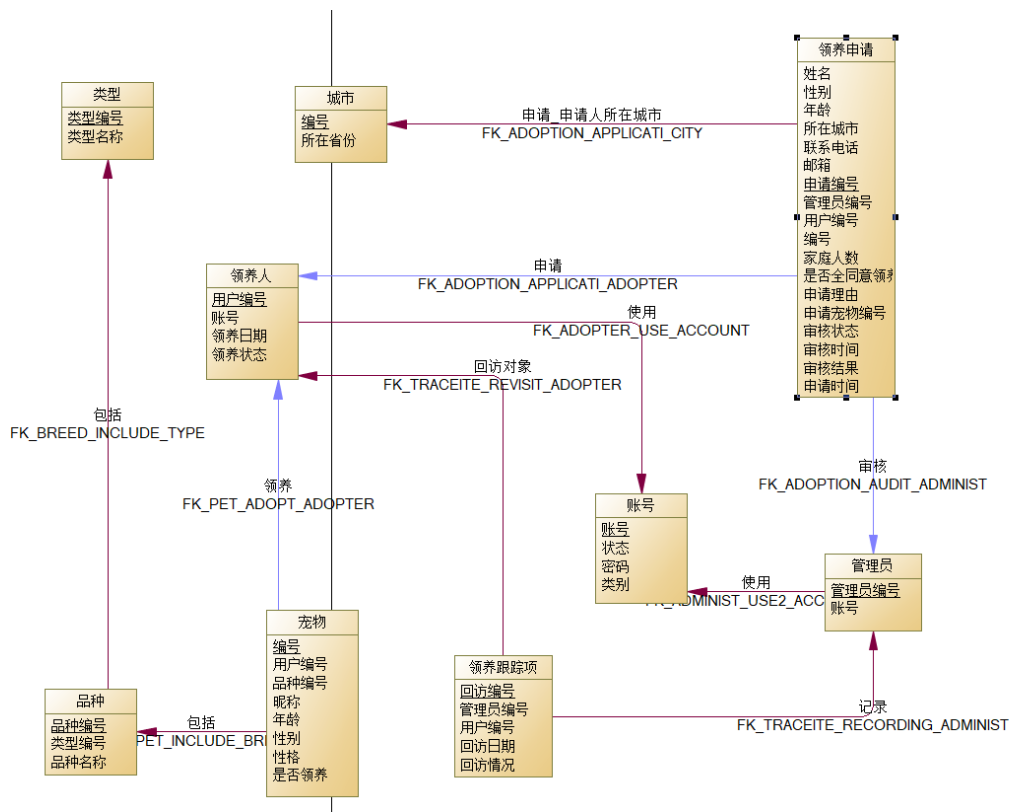
2.1.1 ER 图



2.1.2 LDM 图



2.1.3 PDM 图



2.1.4 数据库表结构

1、 表结构

- 1. pet 表——待领养宠物信息：
主键是宠物编号，设置成 Int 型自增，外键包括用户编号和品种编号，对应领养这个宠物的用户和宠物的品种，主键控制不能为空值。

Table Properties - 领养申请 (AdoptionApplication)

Procedures		Physical Options		MySQL		Definition		Rules		Preview	
General		Columns		Data Protection		Indexes		Keys		Triggers	
Name	Code	Data Type	P	Do	F	M					
姓名	u_name	varchar(10)	<input type="checkbox"/>	<None	<input type="checkbox"/>	<input checked="" type="checkbox"/>					
性别	u_sex	char(1)	<input type="checkbox"/>	<None	<input type="checkbox"/>	<input type="checkbox"/>					
年龄	u_age	smallint	<input type="checkbox"/>	<None	<input type="checkbox"/>	<input type="checkbox"/>					
所在城市	u_city	varchar(20)	<input type="checkbox"/>	<None	<input type="checkbox"/>	<input type="checkbox"/>					
联系电话	u_phone	numeric(11,0)	<input type="checkbox"/>	<None	<input type="checkbox"/>	<input type="checkbox"/>					
邮箱	u_email	varchar(20)	<input type="checkbox"/>	<None	<input type="checkbox"/>	<input type="checkbox"/>					
申请编号	ap_id	int	<input checked="" type="checkbox"/>	<None	<input type="checkbox"/>	<input checked="" type="checkbox"/>					
管理员编号	a_id	int	<input type="checkbox"/>	<None	<input checked="" type="checkbox"/>	<input type="checkbox"/>					
用户编号	u_id	int	<input type="checkbox"/>	<None	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					
编号	c_id	int	<input type="checkbox"/>	<None	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					
家庭人数	u_fnum	numeric(2,0)	<input type="checkbox"/>	<None	<input type="checkbox"/>	<input type="checkbox"/>					
是否全同意领	is_agree	bool	<input type="checkbox"/>	<None	<input type="checkbox"/>	<input type="checkbox"/>					
申请理由	reason	varchar(300)	<input type="checkbox"/>	<None	<input type="checkbox"/>	<input type="checkbox"/>					
申请宠物编号	ap_pet	int	<input type="checkbox"/>	<None	<input type="checkbox"/>	<input type="checkbox"/>					
审核状态	a_state	char(5)	<input type="checkbox"/>	<None	<input type="checkbox"/>	<input type="checkbox"/>					
审核时间	a_time	datetime	<input type="checkbox"/>	<None	<input type="checkbox"/>	<input type="checkbox"/>					
审核结果	a_result	char(5)	<input type="checkbox"/>	<None	<input type="checkbox"/>	<input type="checkbox"/>					
申请时间	ap_time	datetime	<input type="checkbox"/>	<None	<input type="checkbox"/>	<input type="checkbox"/>					

More >> 确定 取消 应用(A) 帮助

```

/*=====*/
/* Table: AdoptionApplication */
/*=====*/
create table AdoptionApplication
(
    u_name          varchar(10) not null comment '',
    u_sex           char(1) comment '',
    u_age           smallint comment '',
    u_city          varchar(20) comment '',
    u_phone         numeric(11,0) comment '',
    u_email         varchar(20) comment '',
    ap_id           int auto_increment not null comment '',
    a_id            int comment '',
    u_id            int not null comment '',
    c_id            int not null comment '',
    u_fnum          numeric(2,0) comment '',
    is_agree        bool comment '',
    reason          varchar(300) comment '',
    a_state         char(5) comment '',
    a_time          datetime comment '',
    a_result        char(5) comment '',
    ap_time         datetime comment '',
    primary key (ap_id)
);

```

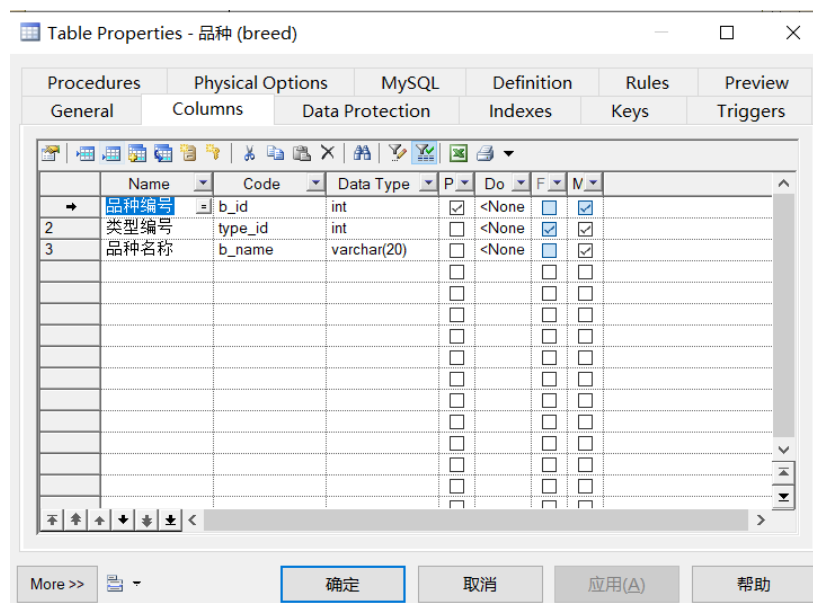
```

28 • alter table AdoptionApplication add constraint FK_ADOPTION_AUDIT_ADMINIST foreign key (a_id)
29     references administrator (a_id) on delete restrict on update restrict;
30
31 • alter table AdoptionApplication add constraint FK_ADOPTION_APPLICATI_ADOPTER foreign key (u_id)
32     references adopter (u_id) on delete restrict on update restrict;
33
34 • alter table AdoptionApplication add constraint FK_ADOPTION_APPLICATI_CITY foreign key (c_id)
35     references city (c_id) on delete restrict on update restrict;
36

```

3. breed 表——宠物的品种

主键是品种编号，设置成 Int 型自增，外键包括类型编号，记录该品种属于的类型，主键控制不能为空值。

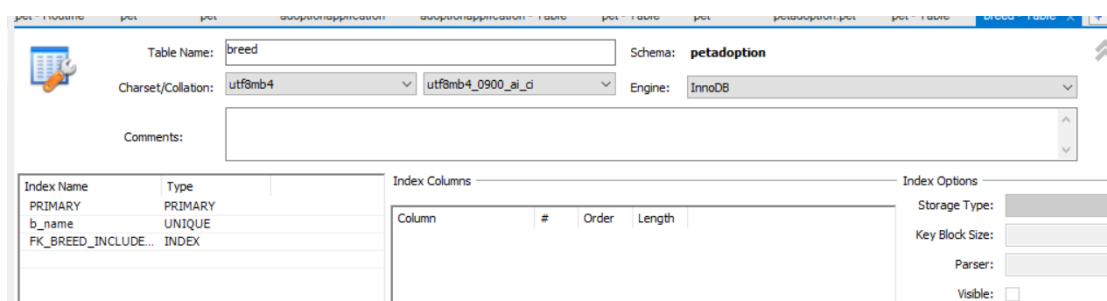


```
/*=====*/
/* Table: breed */
/*=====*/
create table breed
(
    b_id          int auto_increment not null comment '',
    type_id       int not null comment '',
    b_name        varchar(20) not null comment '',
    primary key (b_id)
);
```

```
alter table breed add constraint FK_BREED_INCLUDE_TYPE foreign key (type_id)
references type (type_id) on delete restrict on update restrict;
```

2、索引

1) 索引截图



2) 使用场景（用途）

查询一种品种的宠物时，加速查找的速度。

3、 视图

1) 视图截图

待领养宠物信息图：

The screenshot displays a SQL script in a text editor and its execution result in a grid below.

```
1 • CREATE
2     ALGORITHM = UNDEFINED
3     DEFINER = `root`@`localhost`
4     SQL SECURITY DEFINER
5     VIEW `petinfo` AS
6     SELECT
7         `pet`.`p_id` AS `p_id`,
8         `pet`.`p_name` AS `p_name`,
9         `pet`.`p_age` AS `p_age`,
10        `pet`.`p_sex` AS `p_sex`,
11        `pet`.`p_character` AS `p_character`,
12        `pet`.`is_adopted` AS `is_adopted`,
13        `breed`.`b_name` AS `b_name`,
14        `type`.`type_name` AS `type_name`
15    FROM
16        ((`pet`
17        JOIN `breed`)
18        JOIN `type`)
19    WHERE
20        ((`pet`.`b_id` = `breed`.`b_id`)
21        AND (`breed`.`type_id` = `type`.`type_id`))
```

The result grid shows the following data:

	p_id	p_name	p_age	p_sex	p_character	is_adopted	b_name	type_name
▶	21	鱼仔	3	公	调皮活泼，聪明	0	边牧	狗
	22	花花	2	母	安静，温顺	0	田园猫	猫
	23	胖虎	1	公	有点胆小，怕生人	0	橘猫	猫

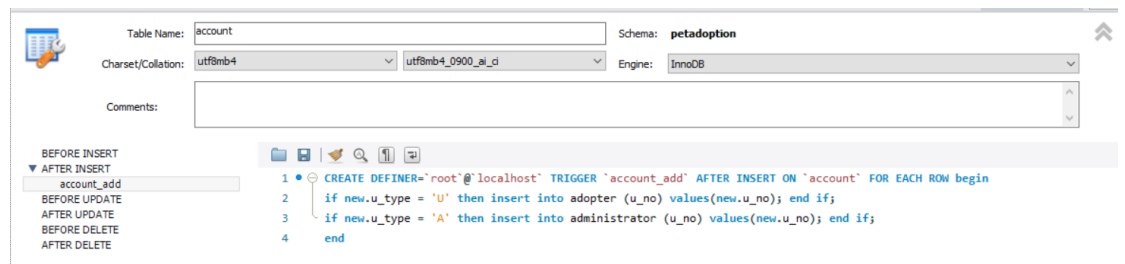
2) 使用场景（用途）

用户可以看到待领养宠物的所有信息，获取其编号、昵称、年龄、性格、种类、是否已经被领养等信息时，由于宠物种类和品种并不直接存储在宠物信息（pet）表中，而是通过编号使用外键连接到宠物品种（breed）表和宠物种类（type）表中，因此需要创建宠物信息查询视图，便于每次查找信息。

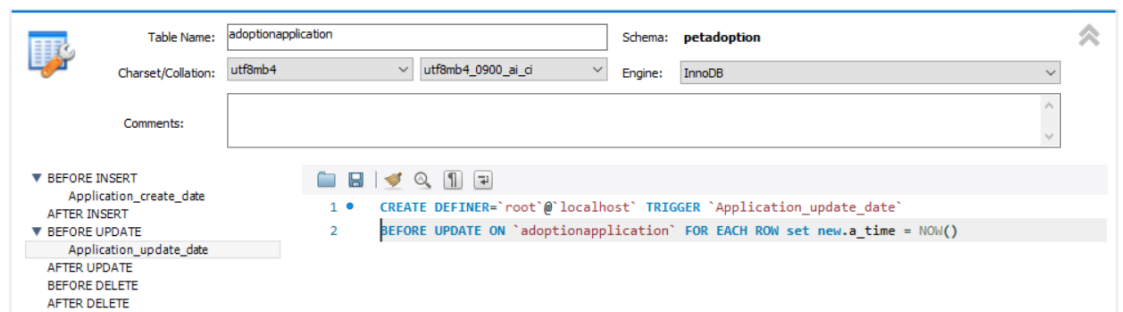
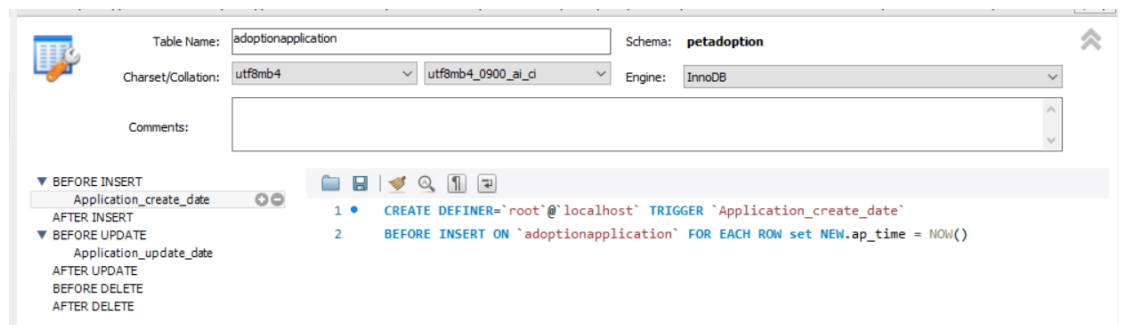
4、 触发器

1) 触发器截图

1. 账户（account）表上的 account_add 触发器



2. 申请信息表



2) 使用场景（用途）

1. 账户（account）表上的 account_add 触发器

在注册账户时，可以选择注册用户还是管理员，在插入账户时就要把用户和管理员加到不同的表里，因此需要用一个触发器，在插入账户信息后，判断该条信息是用户账户还是管理员账户，从而插入不同的表中。

2. 申请表信息（adoptionapplication）表上的 application_create_date 触发器

用户提交申请表信息后，需要获取当前的时间记录在表中，如果要求用户端获取太繁琐，因此使用触发器，每次插入新申请时自动获取当前时间插入表中。

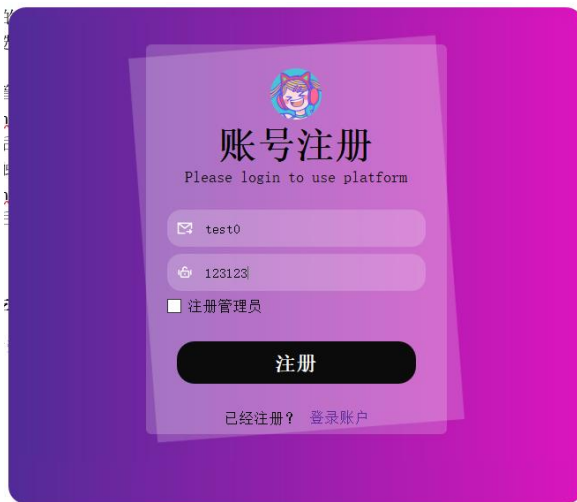
3. 申请表信息（adoptionapplication）表上的 application_update_date 触发器

管理员在处理完申请后需要提供一个审核结束的时间，使用触发器自动添加。

3) 验证触发器

1. 验证 account_add 触发器：

注册一个普通用户



```
1 • SELECT * FROM petadoption.account;
```

	u_no	u_state	u_password	u_type
▶	123	1	12345	U
	212	1	212	A
	a12	1	123	A
	a123	1	123456	A
	aba	1	df	U
	abb	1	1212	U
	test0	1	123123	U
	xff	1	123123	U
	xly	1	123123	U
	xxf	1	123123	U
	xyf	1	123123	U
	zlh	1	123123	U
*	NULL	NULL	NULL	NULL

查看用户表发现已经插入，即生效

```
1 • SELECT * FROM petadoption.adopter;
```

	u_id	u_no	adopt_date	adopt_state
▶	1	123	NULL	NULL
	2	xxf	NULL	NULL
	3	abb	NULL	NULL
	4	aba	NULL	NULL
	5	xff	NULL	NULL
	6	xyf	NULL	NULL
	7	xly	NULL	NULL
	8	zlh	NULL	NULL
	9	test0	NULL	NULL
*	NULL	NULL	NULL	NULL

2.验证 application_create_date 触发器
提交领养申请



USER: test0

- 待领养
- 申请表
- 我的申请

Exit

姓名	ttt0	性别	男	年龄	34
家庭总人数	3	家庭住址	吉林市xx街道xx号		
联系电话	1234234242	邮箱	1232123@qq.com	申请宠物编号	21
申请理由 有养宠经验，家里有一只，想找一只陪伴					
审核状态	未处理	审核结果	未处理		

驳回申请


可以看到能更新插入时间:

```
1 • SELECT ap_time FROM petadoption.adoptionapplication where u_name = 'ttt0'
```

ap_time
2023-10-30 16:44:10

3.验证 application_update_date 触发器

登录管理员账户看到刚刚的申请，批准申请



USER: 212

- 待领养管理
- 待处理申请
- 我已处理

Exit

姓名	zlh	性别	女	年龄	22
家庭总人数	4	家庭住址	深圳市xx街道		
联系电话	3212123243	邮箱	123123@123.com	申请宠物编号	19
申请理由 热爱小动物					
批准			驳回		

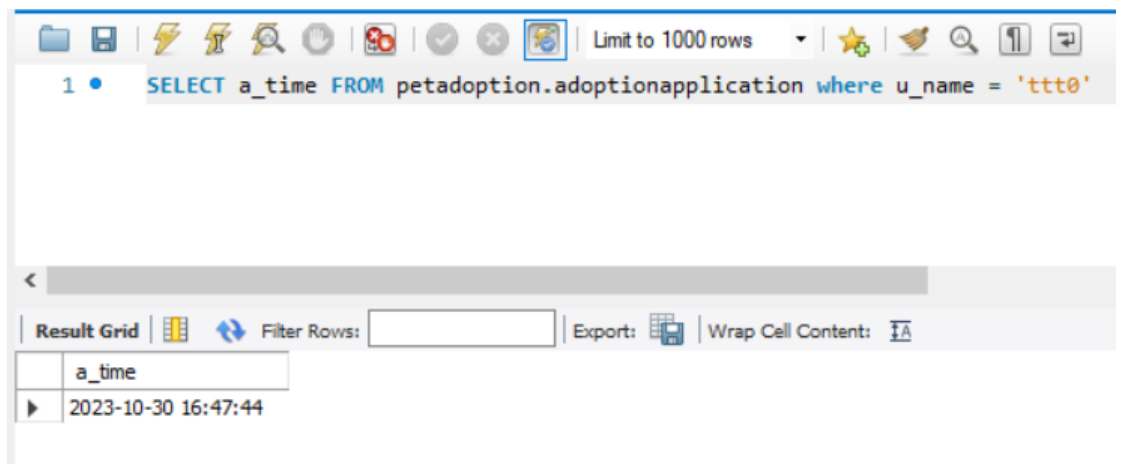
姓名	ttt0	性别	男	年龄	34
家庭总人数	3	家庭住址	吉林市xx街道xx号		
联系电话	1234234242	邮箱	1232123@qq.com	申请宠物编号	21
申请理由 有养宠经验，家里有一只，想找一只陪伴					
批准			驳回		

USER: 212

待领养管理
待处理申请
我已处理
Exit

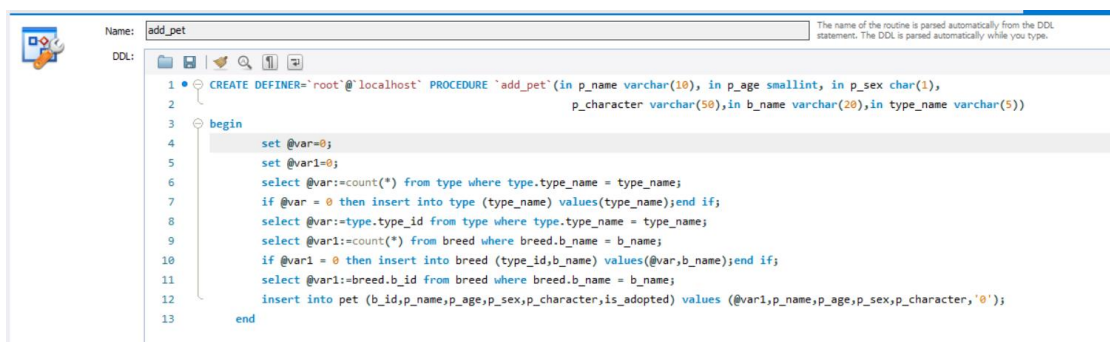
姓名	性别	年龄	家庭总人数	家庭住址	联系电话	邮箱	申请宠物编号	申请理由	审核状态	审核结果
喵喵犯了	男	32	3	深圳市xx街道xx号	1235141212	87989000@qq.com	8	想要小猫咪吸吸~	已处理	批准
乐乐	女	26	4	珠海市xx区xx街道	4542134242	898909@123.com	17	摸摸狗头，万事不愁~	已处理	驳回
ttt0	男	34	3	吉林市xx街道xx号	1234234242	1232123@qq.com	21	有养宠经验，家里有一只，想找一只陪伴	已处理	批准

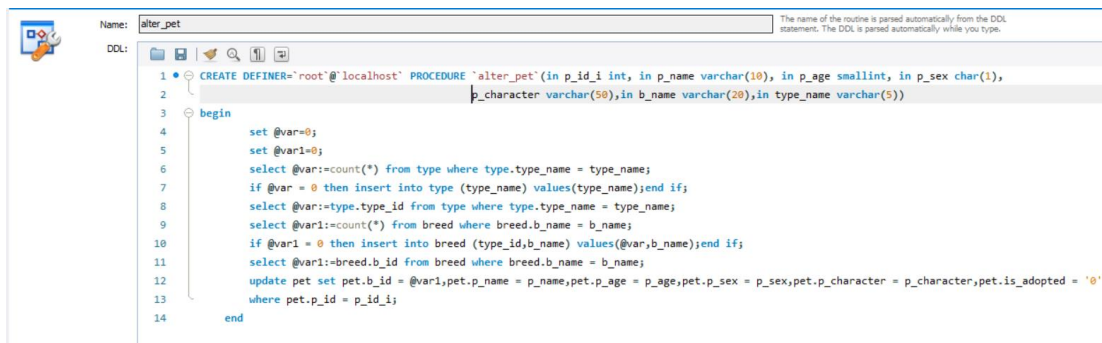
可以看到审核的时间，触发成功：



5、 存储过程或存储函数

1) 存储过程或存储函数截图





```
1 CREATE DEFINER='root'@'localhost' PROCEDURE `alter_pet`(in p_id_i int, in p_name varchar(10), in p_age smallint, in p_sex char(1),
2 p_character varchar(50),in b_name varchar(20),in type_name varchar(5))
3
4 begin
5     set @var=0;
6     set @var1=0;
7     select @var:=count(*) from type where type.type_name = type_name;
8     if @var = 0 then insert into type (type_name) values(type_name);end if;
9     select @var:=type.type_id from type where type.type_name = type_name;
10    select @var1:=count(*) from breed where breed.b_name = b_name;
11    if @var1 = 0 then insert into breed (type_id,b_name) values(@var,b_name);end if;
12    select @var1:=breed.b_id from breed where breed.b_name = b_name;
13    update pet set pet.b_id = @var1,pet.p_name = p_name,pet.p_age = p_age,pet.p_sex = p_sex,pet.p_character = p_character,pet.is_adopted = '0'
14    where pet.p_id = p_id_i;
15 end
```

2) 使用场景（用途）

1. 储存过程 add_pet:

在新添加宠物信息的时候，不仅需要操作插入 pet 表，还要检查记录宠物品种和种类的 breed、type 表有没有当前要插入的值，如果没有则要先插入品种和种类，再插入宠物信息，同时将种类编号和品种编号给宠物信息；如果有也需要先找到对应的编号，再插入宠物信息，因此使用一个储存过程能更简洁的实现这个过程。

2. 储存过程 alter_pet:

与新添加宠物信息的情况类似，在修改宠物信息时，也会遇到上述这种情况，因此又创建了一个储存过程实现修改宠物信息时，维护品种表和类型表。

3 收获和反思

在这次实验中，我实现了一个宠物领养管理平台。整个开发过程非常有挑战性，但也让我收获了很多。从零开始设计数据库，从 ER 图开始，逐步转换成 LDM 图，再转换成 PDM 图，再生成 sql 脚本，实现平台的后端。这个过程让我深刻认识到数据库设计的复杂性。

一开始，我并没有分出宠物种类表，而是将宠物的种类信息直接存储在主表中。然而，随着向表中增加了多条数据，我发现同一种类的宠物信息在多个记录中重复出现，造成了存储空间的浪费。为了解决这个问题，我决定将宠物种类信息抽取出来，单独存储在一个表中，并在原表中建立外键连接。这样一来，不仅有效节省了存储空间，还提高了数据的一致性和准确性。

在实现平台的后端时，我选择使用了 SQL 语言来操作数据库。通过编写 SQL 脚本，我能够对数据库进行增、删、改、查等操作。同时，我还实现了一些复杂的功能，比如宠物领养申请的处理和宠物信息的更新等。通过这些操作，我能够对数据库中的数据进行灵活的管理和控制。

接着，我开始设计平台的前端界面。考虑到之前有过使用经验，我选择使用了 pyqt5 作为前端开发工具。借助 Qt Designer，我能够快速而方便地设计出各个控件的布局和样式。在开发过程中，我更加熟悉了一些常用控件的功能和应用，并尽可能使整个界面简洁美观。通过与后端的连接，我能够实现前后端的数据交互和信息传递，使整个平台能够正常运行。

整个平台完成之后，我感到非常有成就感。通过这次实验，我全面了解了前后端开发的全过程，从数据库设计到后端实现再到前端开发，每个环节都需要仔细思考和调试。我也意识到了数据库设计的重要性，它需要充分考虑到平台所需要提供的功能，并进行多次的迭代修改才能最终完成。

总的来说，这次实验与以往的较为不同，前后端都由自己设计实现，给了我们同学较大的自由发挥的空间。虽然过程中遇到了一些困难和挑战，但通过不断努力和学习的，我成功地完成了这个宠物领养管理平台。这次实验不仅让我掌握了前后端开发的技能，还锻炼了我的问题解决能力。我相信这些经验将对我的未来职业发展产生积极的影响。