

（深圳）

实验报告

开课学期： 2022秋季

课程名称： 数据库系统

实验名称： 校园猫管理系统设计与实现

实验性质： 设计型

实验学时： 6 地点： T2210

学生班级： 5班

学生学号： 200110513

学生姓名： 宗晴

评阅教师：

报告成绩：

实验与创新实践教育中心制

2022年11月

# 实验环境

*请填写用到的操作系统和主要开发工具。*

实验操作系统：Windows

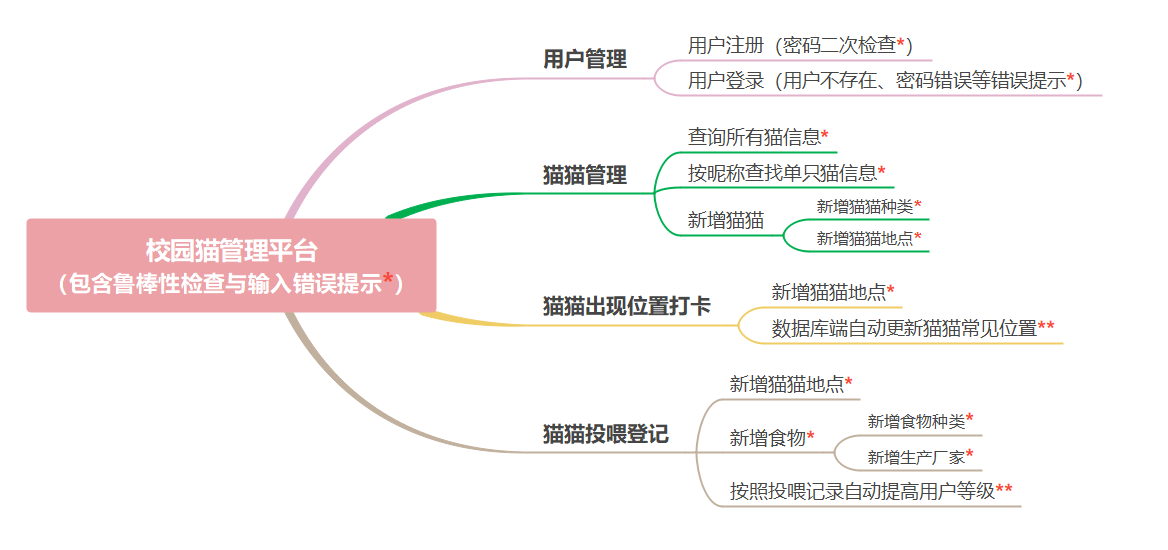
主要开发工具：后端数据库使用MySQL8.0关系数据库管理系统、PowerDesigner。前端界面使用python语言实现，环境为python3.9。前端界面框架为pyqt5，使用pymysql进行前后端连接，开发工具为pycharm、Qt Designer。

# 实验过程

1. 1672316765051

## 系统功能

*请结合文字、图表等方式清晰描述系统的功能。如有亮点功能请用\*标志。*



如上图所示，校园猫管理平台可分为用户管理、猫猫管理、猫猫出现位置打卡和猫猫投喂登记这四部分。整个系统进行了全面的鲁棒性检查，并会按照错误类型弹出相应的提示信息*\**。

1. 用户管理：包括所要求的用户注册和登录功能。其中在用户注册时，必填的信息是用户名和密码，可以选填的信息是性别、年龄、联系电话。此外，设置了密码的二次检查，保证用户输入所希望的密码*\**，并且用户可以按照自己的意愿选择独一无二的用户名*\**（该用户名是用户表的主键），若用户名已被注册，也会进行提示，同时数据库端设置了trigger，在新增用户数据时，设置用户等级为0*\*\**。在用户登录时，输入用户名和密码进行登录，可以直接按回车或者按登录按键均可，会提示用户不存在或者密码错误等信息*\**。如下图所示：



登录界面

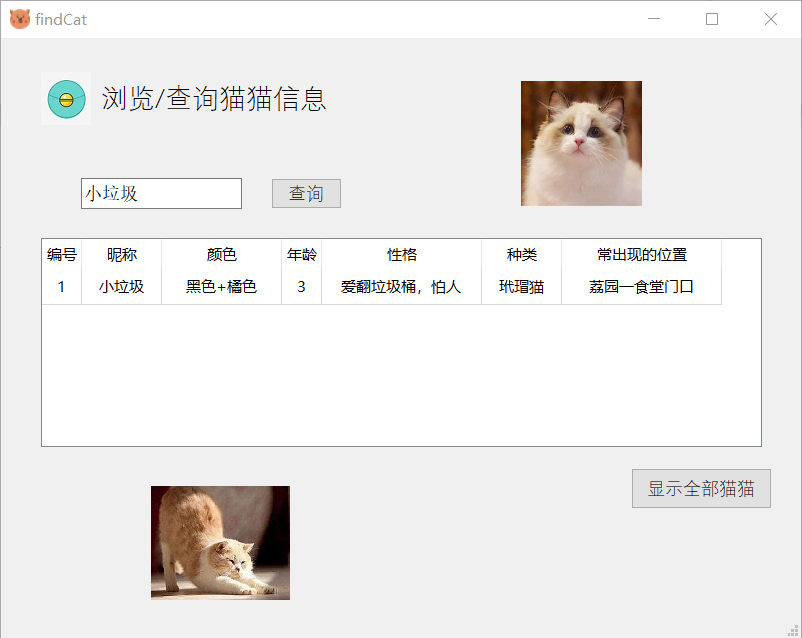


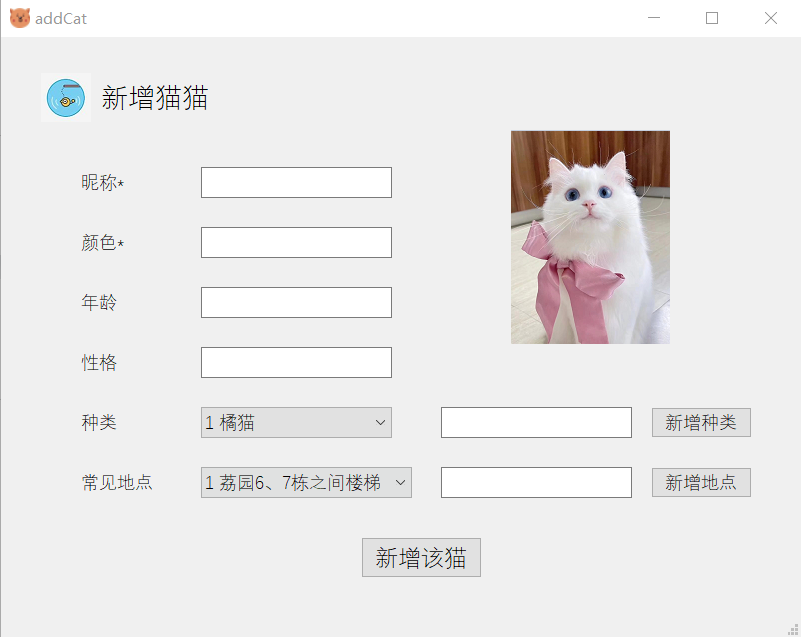
注册界面

（2）猫猫管理：可以一键查询、维护所有的猫信息，包括编号、昵称、颜色、年龄、性格、种类、常出现的位置*\**。也可以按照猫猫昵称查找单只猫的信息*\**（在数据库层面添加了猫猫昵称的索引，可以加快查找速度）。还可以进行猫猫信息的新增*\**，新增时必填的是昵称和颜色，选填的是年龄、性格、种类和常见地点。若种类和常见地点不存在，可以进行新增*\**。如下图所示：



查询界面可一键查询全部猫猫



查询界面也可按昵称查找某一只猫猫

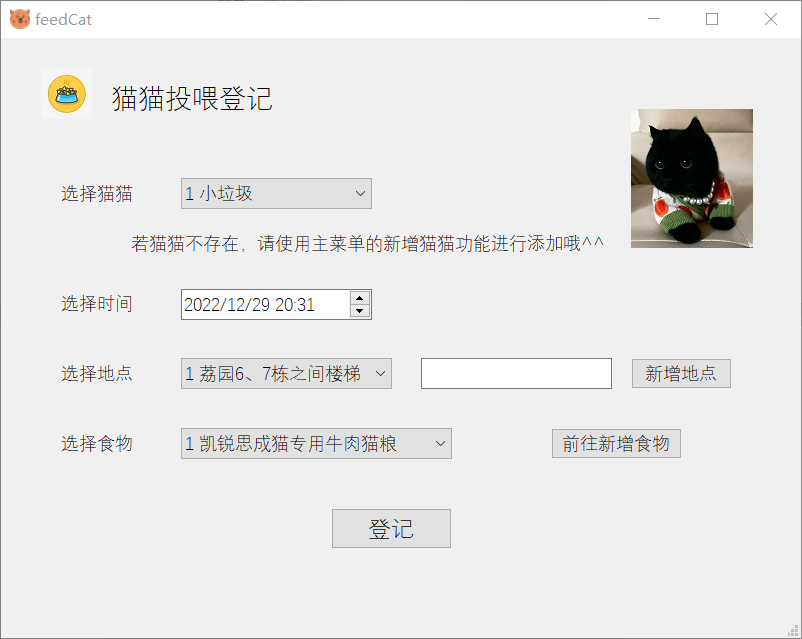
新增猫猫界面

（3）猫猫出现位置打卡：需要进行猫猫、出现时间、出现地点的选择。若地点不存在，则可以进行新增*\**。同时数据库端设计了trigger，会根据打卡情况，自动更新猫猫常见位置*\*\**。

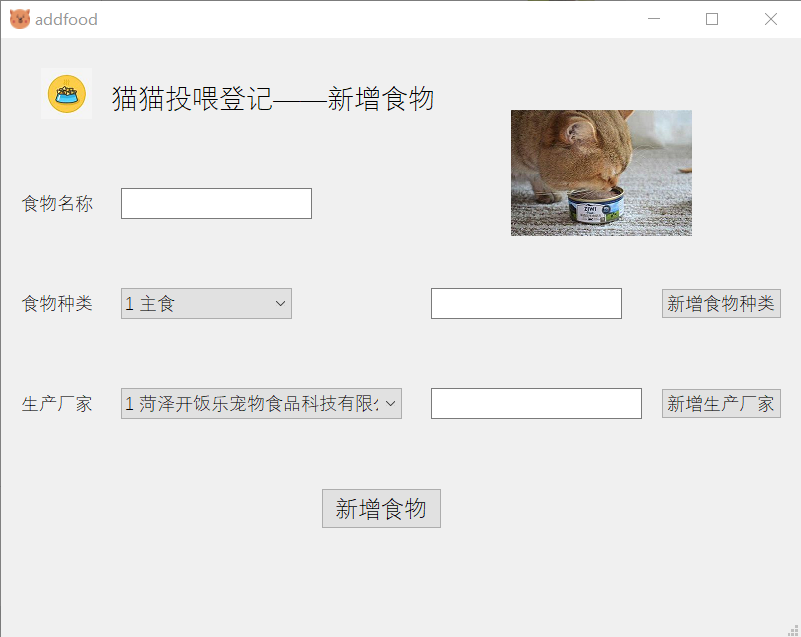


猫猫出现位置打卡界面

（4）猫猫投喂登记：需要进行猫猫、投喂时间、投喂地点、投喂食物的选择。若地点和食物不存在，则可以进行新增*\**。若食物种类和生产厂家不存在，也可以进行新增*\**。当用户登记投喂后，数据库端设计了trigger，自动提高用户等级，并实时显示*\*\**。



猫猫投喂登记界面



猫猫投喂登记——新增食物界面

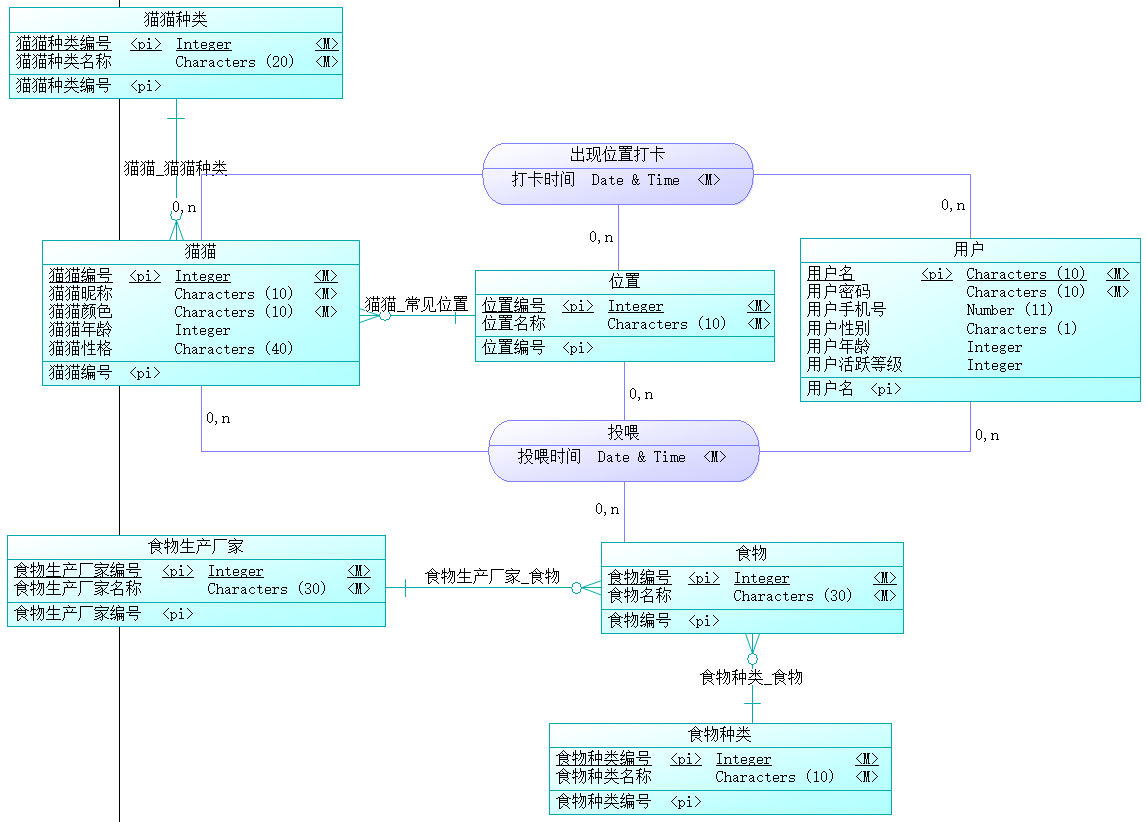


主菜单界面（显示用户当前等级）

## 数据库设计

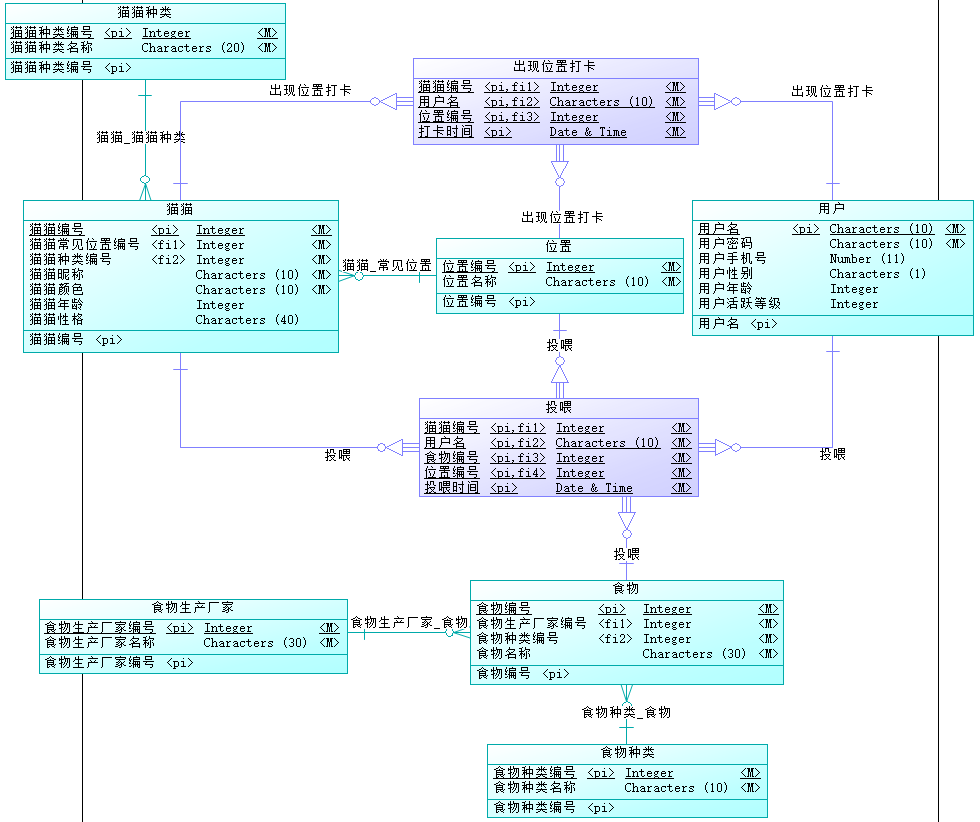
### ER图

*要求：截图务必清晰，如果图太大可截图一个总图，然后再分块截图。如果看不清截图会影响成绩。*



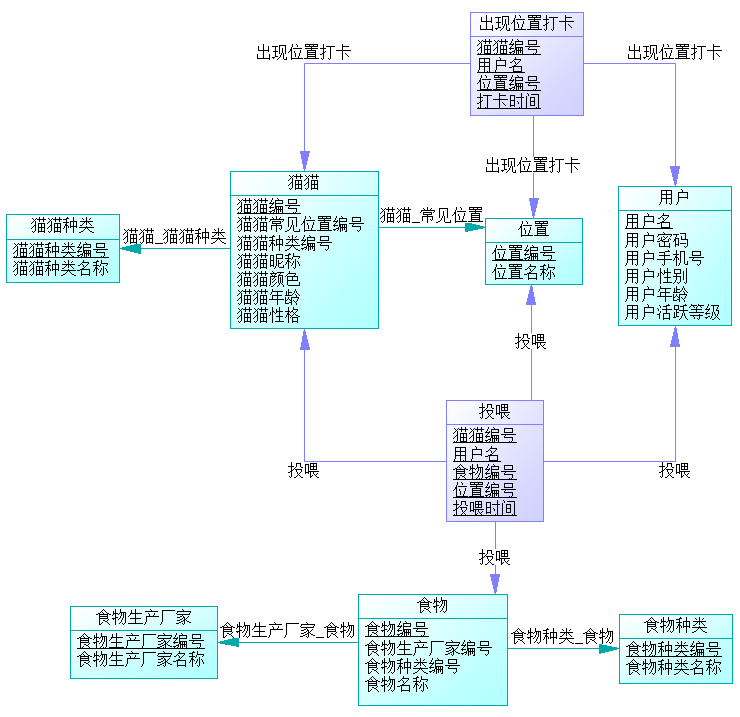
### LDM图

*要求：截图务必清晰，如果图太大可截图一个总图，然后再分块截图。如果看不清截图会影响成绩。*



### PDM图

*要求：截图务必清晰，如果图太大可截图一个总图，然后再分块截图。如果看不清截图会影响成绩。*



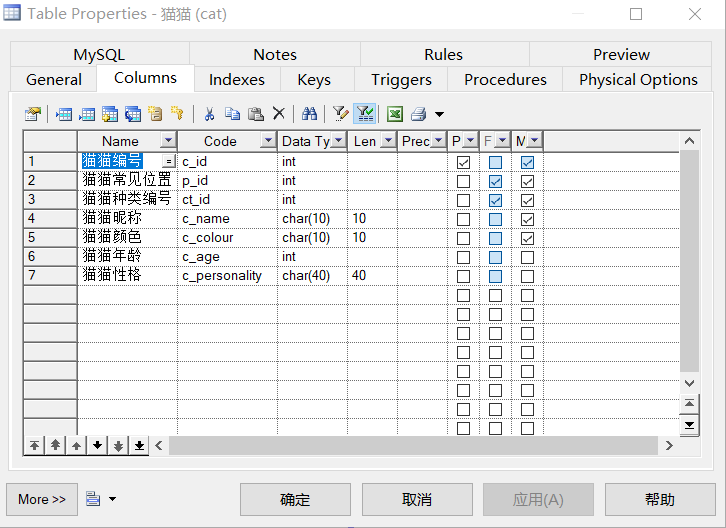
### 数据库表结构

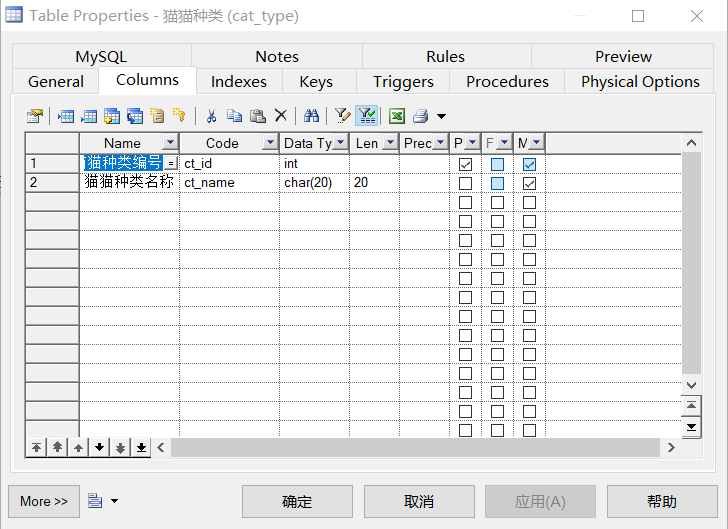
1. **表结构**

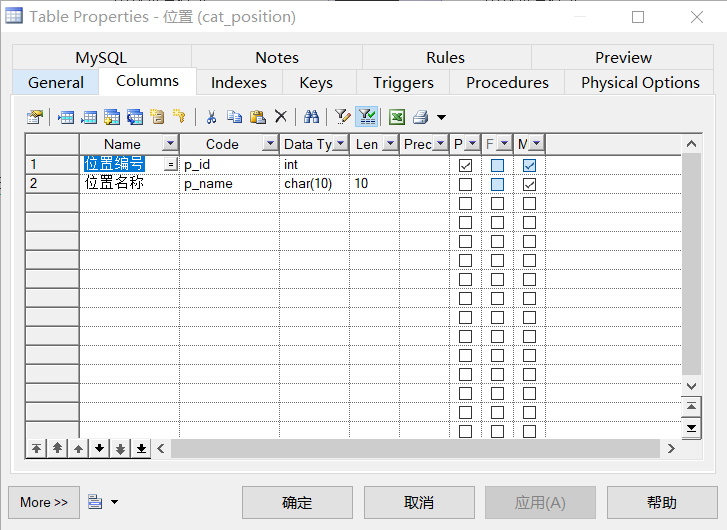
*选取2-3个比较有代表性的表结构截图，体现主键约束、外键约束、空值约束等。*

1. 猫猫（cat）+猫猫种类（cat\_type）+位置（cat\_position）

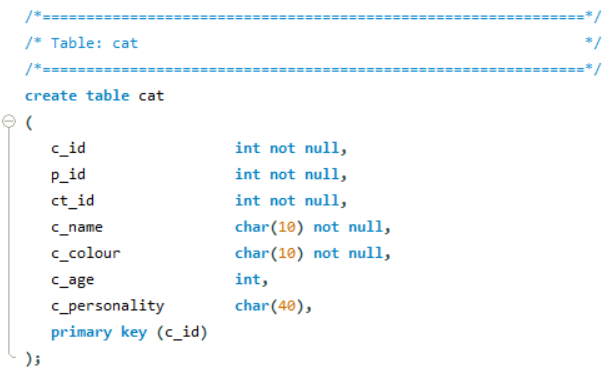
各表结构如下图所示：

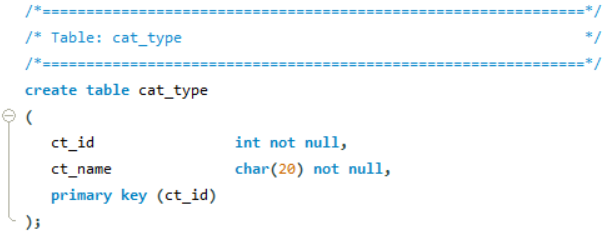


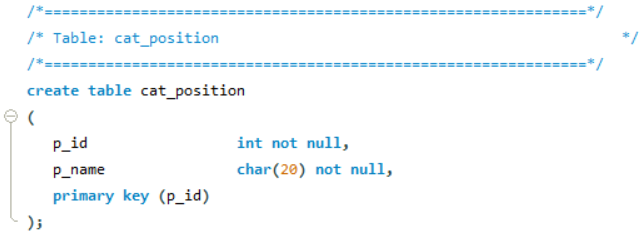


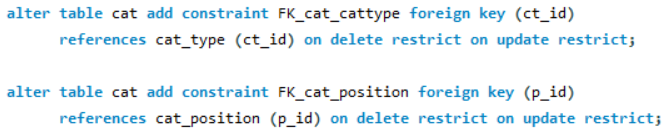


sql代码如下：









如上图所示，在猫猫（cat）表中：

具有主键约束（primary key）的是猫猫编号（c\_id）。

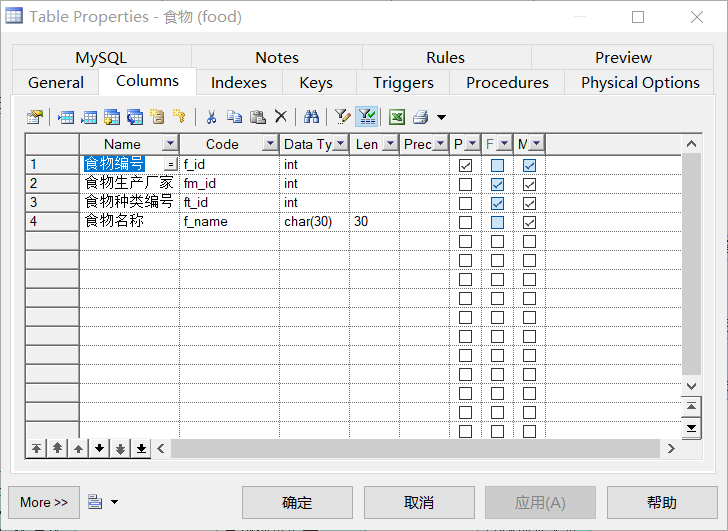
具有外键约束（foreign key）的是猫猫常见位置编号（p\_id）和猫猫种类编号（ct\_id），其中p\_id对应到位置（cat\_position）表中的主键——位置编号（p\_id），ct\_id对应到猫猫种类（cat\_type）表中的主键——猫猫种类编号（ct\_id）。

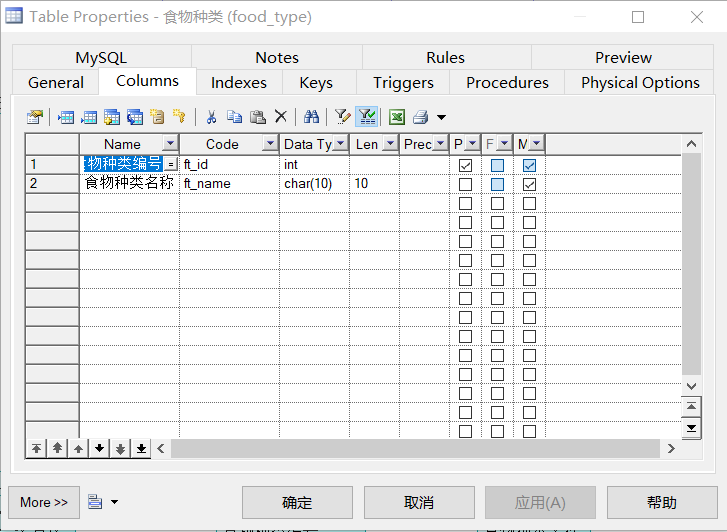
具有空值约束（not null）的是猫猫编号（c\_id）、猫猫常见位置编号（p\_id）、猫猫种类编号（ct\_id）、猫猫昵称（c\_name）和猫猫颜色（c\_colour）。

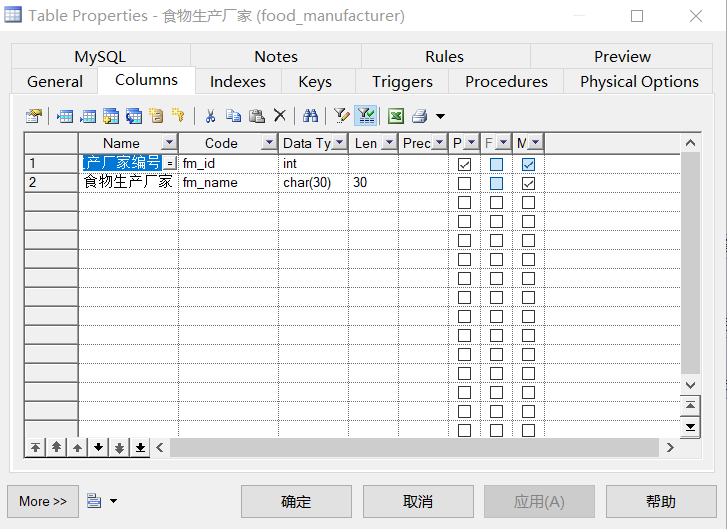
在猫猫种类（cat\_type）表中，具有主键约束（primary key）的是猫猫种类编号（ct\_id），具有空值约束（not null）的是猫猫种类编号（ct\_id）和猫猫种类名称（ct\_name），无外键约束。

在位置（cat\_position）表中，具有主键约束（primary key）的是位置编号（p\_id），具有空值约束（not null）的是位置编号（p\_id）和位置名称（p\_name），无外键约束。

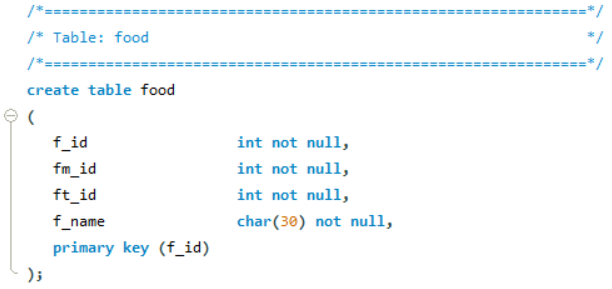
（2）食物（food）+食物种类（food\_type）+食物生产厂家（food\_manufacturer）

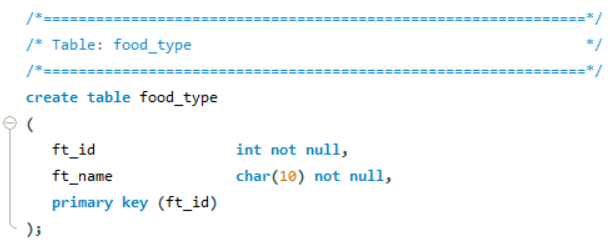


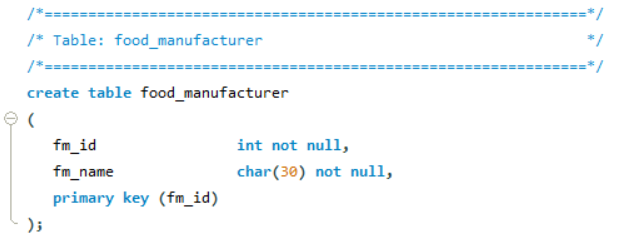


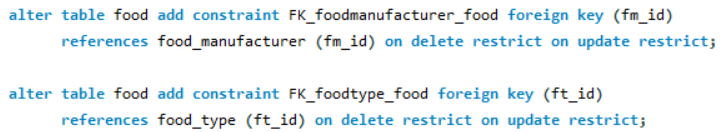


sql代码如下：









如上图所示，在食物（food）表中：

具有主键约束（primary key）的是食物编号（f\_id）。

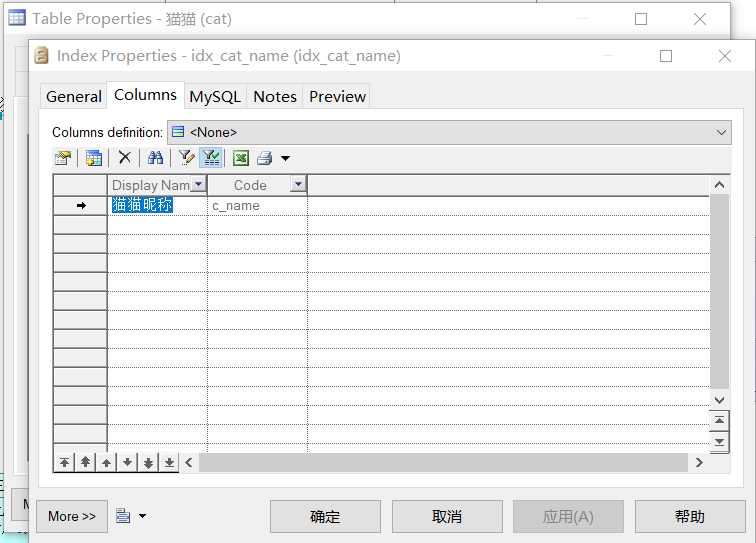
具有外键约束（foreign key）的是食物种类编号（ft\_id）和食物生产厂家编号（fm\_id），其中ft\_id对应到食物种类（food\_type）表中的主键——食物种类编号（ft\_id），fm\_id对应到食物生产厂家（food\_manufacturer）表中的主键——食物生产厂家编号（fm\_id）。

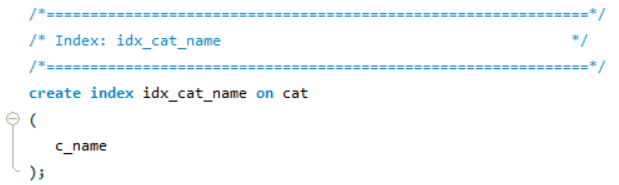
具有空值约束（not null）的是食物编号（f\_id）、食物种类编号（ft\_id）、食物生产厂家编号（fm\_id）和食物名称（f\_name）。

在食物种类（food\_type）表中，具有主键约束（primary key）的是食物种类编号（ft\_id），具有空值约束（not null）的是食物种类编号（ft\_id）和食物种类名称（ft\_name），无外键约束。

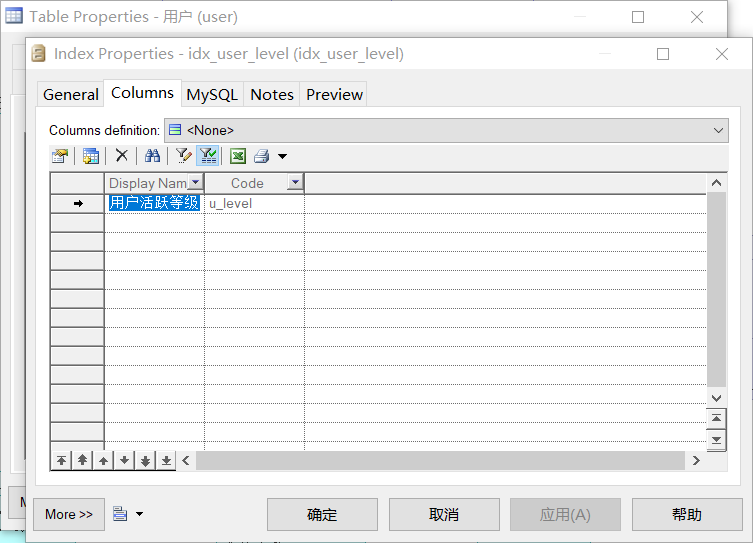
在食物生产厂家（food\_manufacturer）中，具有主键约束食物生产厂家编号（fm\_id），具有空值约束（not null）的是食物生产厂家编号（fm\_id）和食物生产厂家名称（fm\_name），无外键约束。

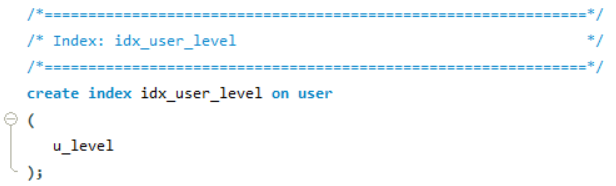
1. **索引**
2. 索引截图





猫猫昵称（c\_name）索引





用户活跃等级（u\_level）索引

1. 使用场景（用途）

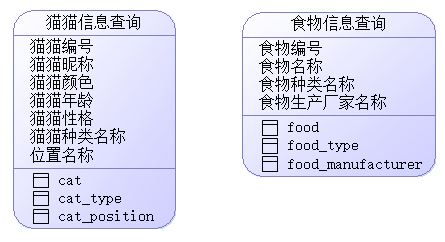
（1）猫猫昵称（c\_name）索引

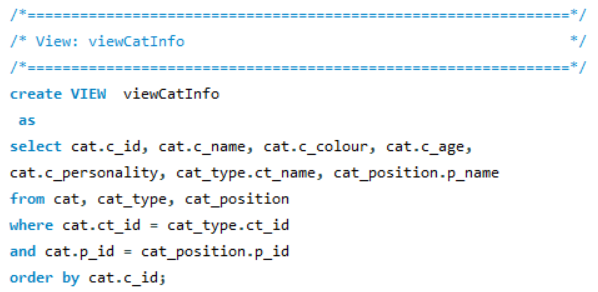
用户想要了解某只猫，在按照猫猫昵称（用户并无法记住猫猫表的主键——猫猫编号c\_id）查找某只猫，从而获取其编号、昵称、颜色、年龄、性格、种类、常出现的位置等信息时，需要给猫猫昵称（c\_name）建立索引，从而加快查找速度。

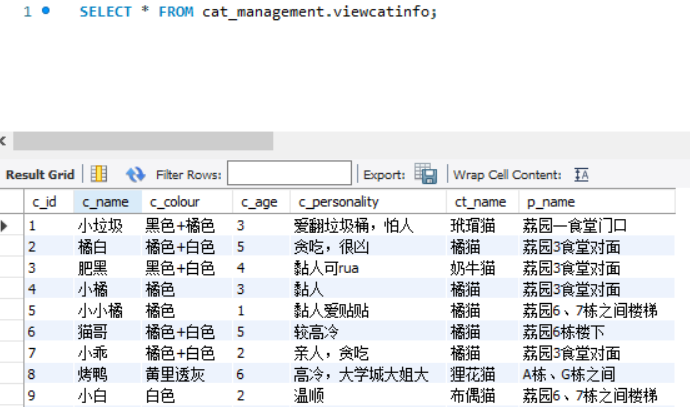
（2）用户活跃等级（u\_level）索引

平台可能需要定期回馈新老用户，给那些经常活跃、等级较高（说明投喂登记次数多）的用户赠送猫粮、猫罐头等小礼品，需要找出用户活跃等级的前几名，因此需要给用户活跃等级（u\_level）建立索引，从而加快查找速度。

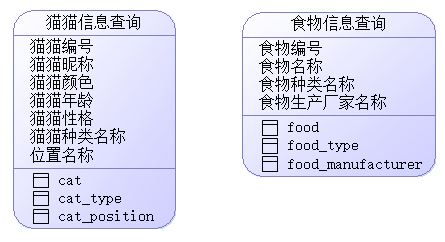
1. **视图**
2. 视图截图

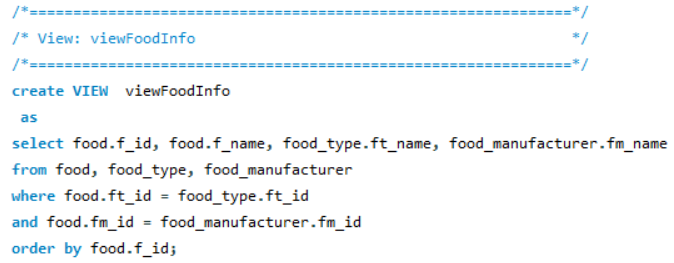


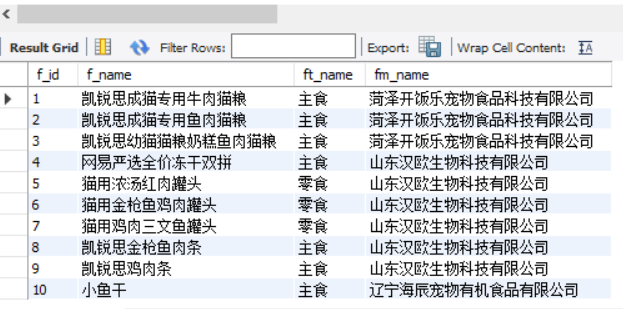




猫猫信息查询视图（viewCatInfo）







食物信息查询视图（viewFoodInfo）

1. 使用场景（用途）

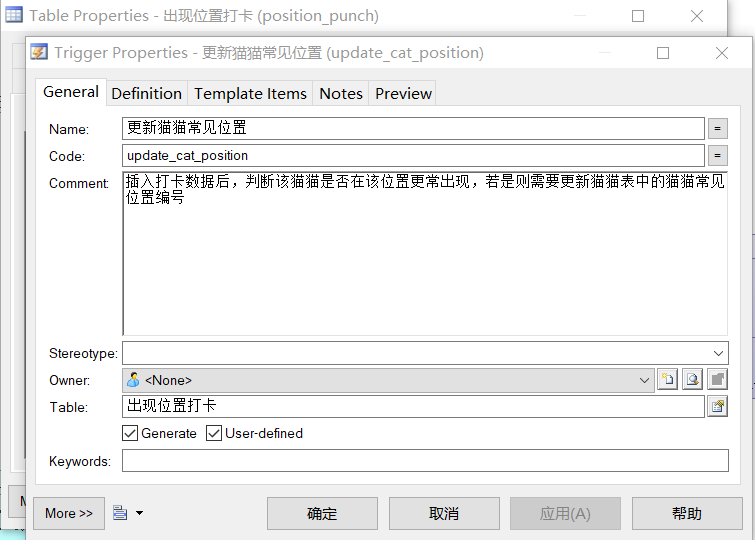
（1）猫猫信息查询视图（viewCatInfo）

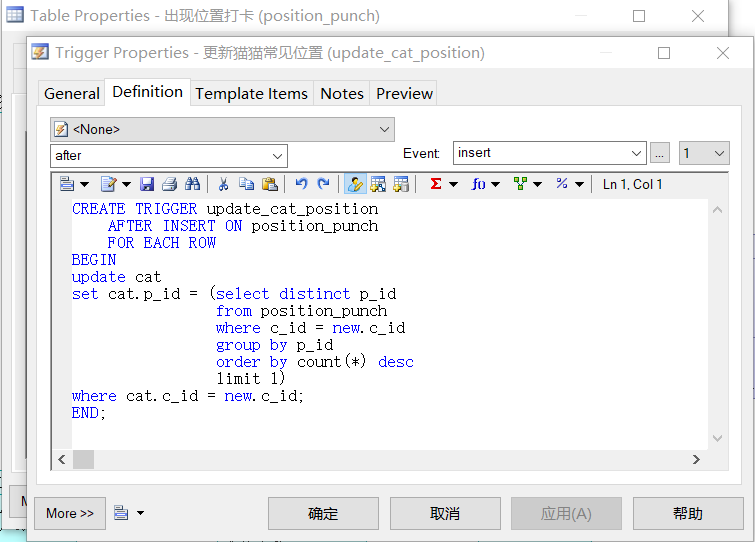
用户想要了解某只猫，在按照猫猫昵称查找某只猫，从而获取其编号、昵称、颜色、年龄、性格、种类、常见位置等信息时，由于猫猫种类和猫猫常见位置并不直接存储在猫猫（cat）表中，而是通过编号使用外键连接到猫猫种类（cat\_type）表和位置（cat\_position）表中，因此需要创建猫猫信息查询视图，便于每次查找信息。

（2）食物信息查询视图（viewFoodInfo）

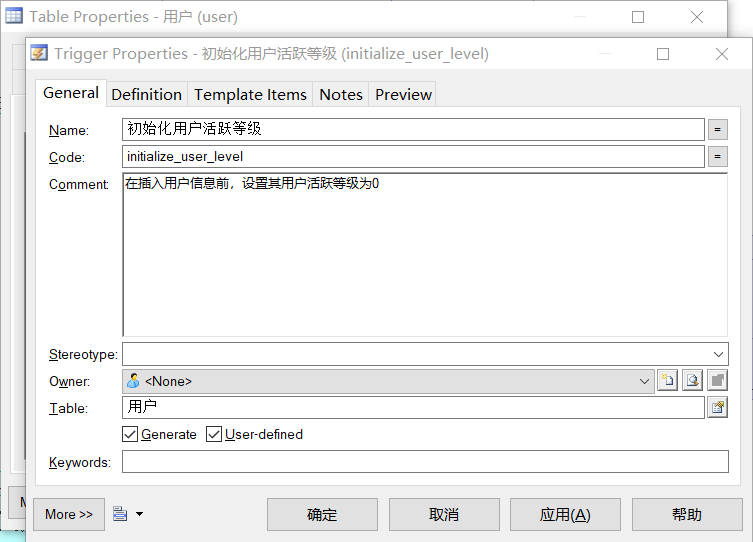
当某只猫猫吃坏肚子生病时，管理员可能需要查看这只猫最近被投喂的所有食物，需要获取食物名称、食物种类名称、食物生产厂家名称等信息，但由于食物种类名称和食物生产厂家名称并不直接存储在食物（food）表中，而是通过编号使用外键连接到食物种类（food\_type）和食物生产厂家（food\_manufacturer）表中，因此需要创建食物信息查询视图，便于每次查找信息。

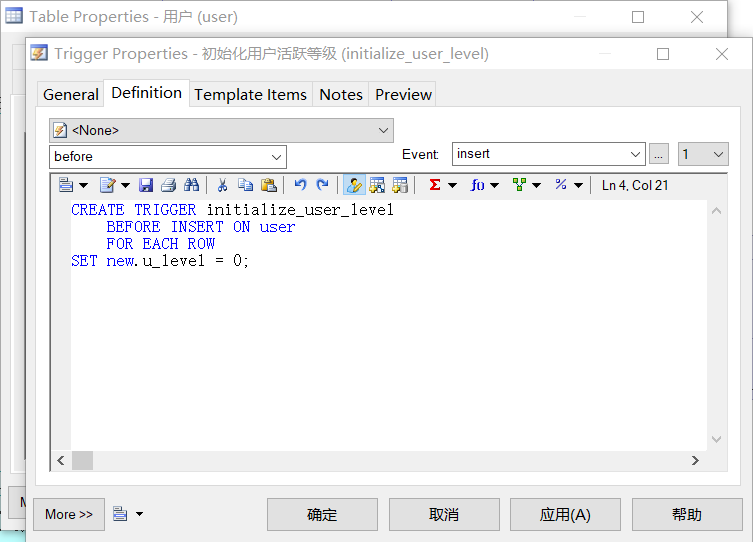
1. **触发器**
2. 触发器截图



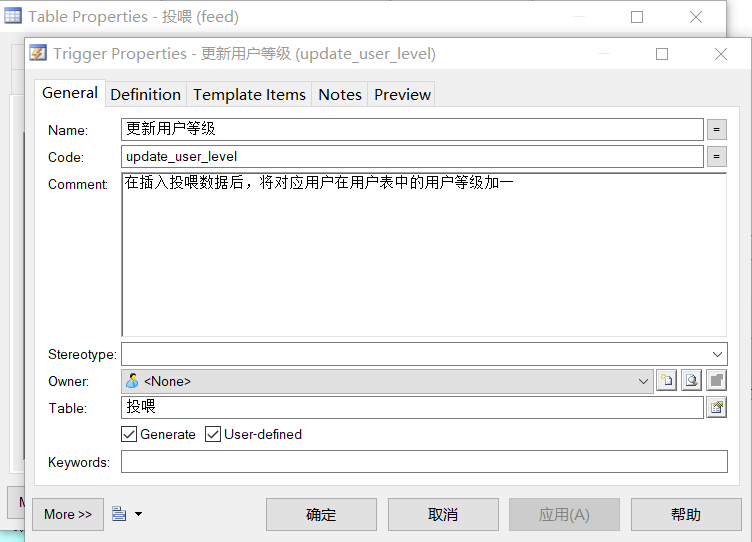


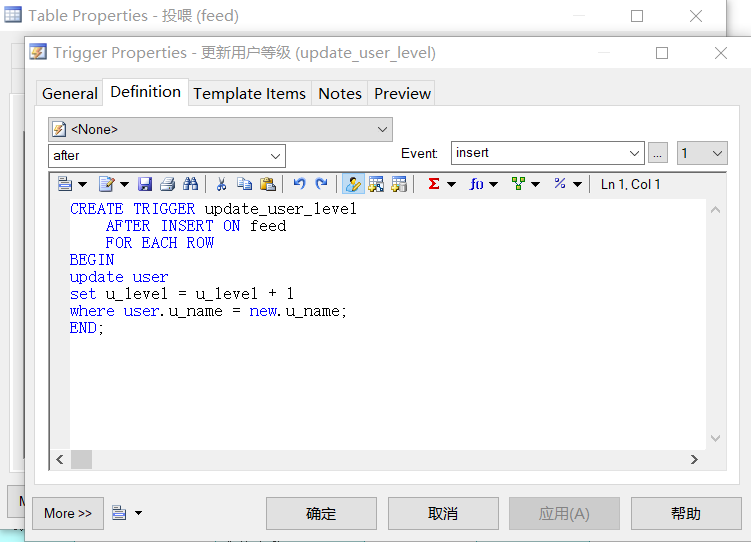
出现位置打卡（position\_punch）表上的更新猫猫常见位置（update\_cat\_position）触发器





用户（user）表上的初始化用户活跃等级（initialize\_user\_level）触发器





投喂（feed）表上的更新用户等级（update\_user\_level）触发器

1. 使用场景（用途）

（1）出现位置打卡（position\_punch）表上的更新猫猫常见位置（update\_cat\_position）触发器

该trigger在往出现位置打卡（position\_punch）表中插入新数据后，判断此时该猫猫在出现位置打卡表中被打卡次数最多的位置，从而更新猫猫表中对应的猫猫的常见位置编号。以用户打卡次数作为依据，使得猫猫的常见位置得到实时真实的更新。

（2）用户（user）表上的初始化用户活跃等级（initialize\_user\_level）触发器

该trigger在插入用户数据之前，设置该用户的用户活跃等级为0，避免用户活跃等级的接口暴露在平台上。从而使得用户的活跃等级不可被用户自行修改，而只能在系统中自动初始化。

（3）投喂（feed）表上的更新用户等级（update\_user\_level）触发器

该trigger在用户插入投喂数据后，将对应用户在用户表中的用户活跃等级加一。从而通过用户活跃等级可以得知投喂猫猫次数较多的用户，便于平台对这些用户给予支持。

1. 验证触发器

（1）出现位置打卡（position\_punch）表上的更新猫猫常见位置（update\_cat\_position）触发器

初始时各猫的常见位置如下图中最后一列所示：



在对猫猫“小垃圾”在“荔园6、7栋之间楼梯”进行多次打卡后，（如下图）



可以发现，猫猫“小垃圾”的常见位置从原来的“荔园一食堂门口”变成了“荔园6、7栋之间楼梯”，如下图所示：



可见，更新猫猫常见位置（update\_cat\_position）触发器验证成功。

（2）用户（user）表上的初始化用户活跃等级（initialize\_user\_level）触发器

首先，通过平台注册一个叫“me”的新用户，注册时，用户并无法自己填写用户活跃等级：



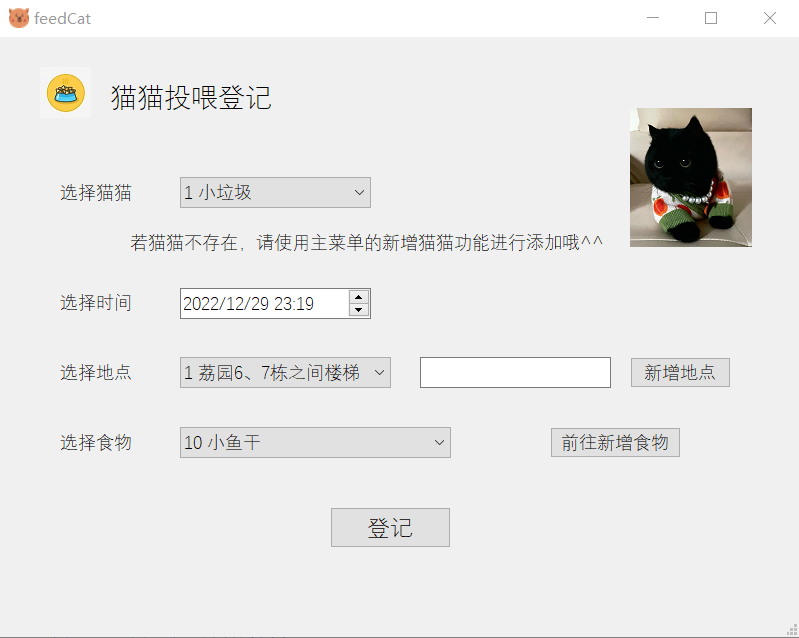
然后，我们使用该账号登录平台，在主菜单的左下角显示了从数据库中获取的该用户的活跃等级，当前为0：



可见，初始化用户活跃等级（initialize\_user\_level）触发器验证成功。

（3）投喂（feed）表上的更新用户等级（update\_user\_level）触发器

在上一个触发器的基础上，使用系统的猫猫投喂登记功能进行登记：



返回主菜单，左下角显示了从数据库中获取的该用户的活跃等级，当前已经变为了1：



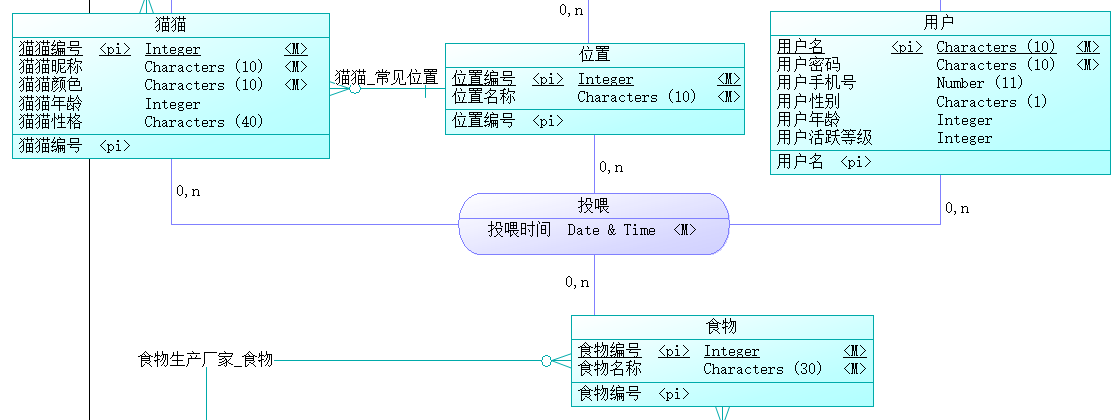
可见，更新用户等级（update\_user\_level）触发器验证成功。

### 分析

*选择较为有代表性的实体和联系进行分析，图文讲解如何从ER图到LDM图、再到PDM图，最后变成数据库表结构的转换过程（参考PPT里“补充内容 ER图🡪LDM图🡪PDM图”中老师讲的例子）。*

选取“投喂”这一联系，和与其关联的四个实体：猫猫（cat）、位置（cat\_position）、用户（user）、食物（food），以及“猫猫\_常见位置”这一联系，和与其关联的两个实体：猫猫（cat）、位置（cat\_position）进行分析。

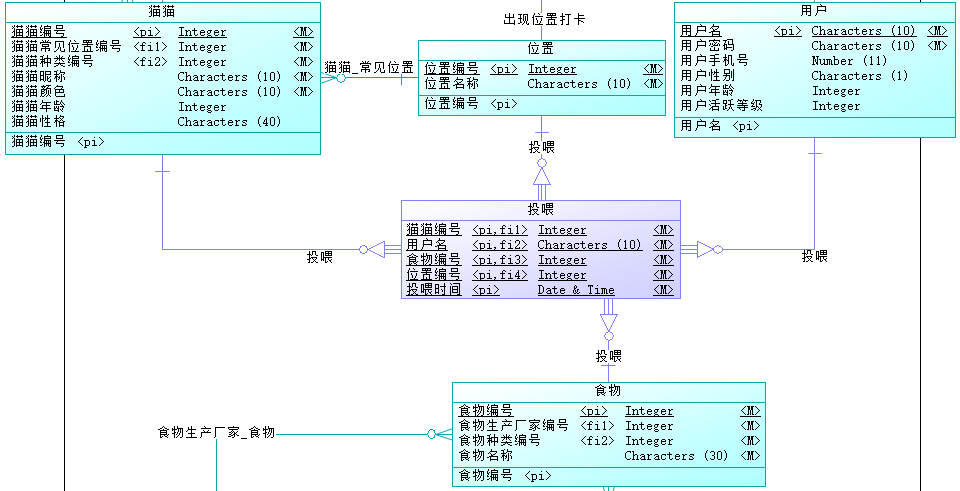
ER图如下所示：



在上图中，猫猫、位置、用户、食物这四个实体与投喂这一联系均是一对多的关系，即一只猫猫可能被多次投喂，一个位置可能被多次作为投喂地点，一个用户可能多次投喂猫猫，一个食物也可能被多次投喂。此时在投喂这一联系中，只有投喂时间（fe\_time）这一属性。

猫猫与常见位置也是一对多的关系，即一只猫猫只有一个常见位置，而一个位置可能被作为多只猫的常见位置。

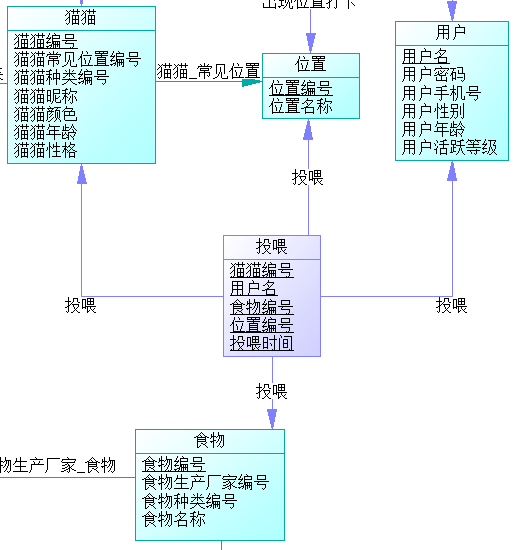
转换成LDM图如下：



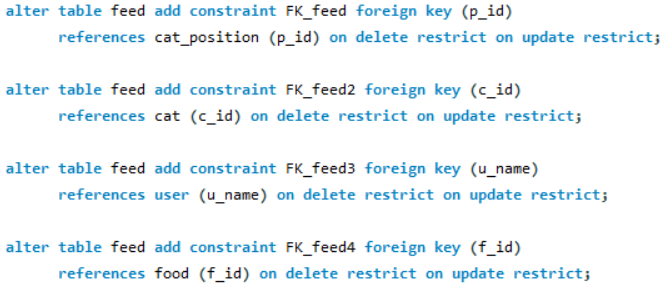
在LDM图中，投喂这一联系中产生了猫猫编号（c\_id）、用户名（u\_name）、食物编号（f\_id）和位置编号（p\_id）这四个属性作为外键，分别连接到猫猫表、用户表、食物表和位置表，作为这四张表的主键。同时猫猫编号、用户名、食物编号、位置编号和投喂时间这五个属性一起作为投喂表的主键，这五个属性将共同决定一条投喂记录。

在猫猫这一实体中也出现了猫猫常见位置编号（p\_id）这一属性，它作为外键连接到位置表中，作为位置表的主键。

转换为PDM图如下：



在PDM图中，在投喂这一联系中，同样产生了猫猫编号（c\_id）、用户名（u\_name）、食物编号（f\_id）和位置编号（p\_id）这四个属性，作为外键分别连接到猫猫表、用户表、食物表和位置表，并且是这四张表的主键：



在猫猫实体中同样产生了猫猫常见位置编号（p\_id）这一属性，它作为外键连接到位置表中，并且是位置表的主键：

1672381968278

# 收获和反思

请填写本次实验的收获，记录实验过程中出现的值得反思的问题及你的思考。

在这次实验中，我实现了一个校园猫管理平台，从零开始设计该平台的数据库，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。然后，从零开始设计平台的前端界面，用代码实现各个控件的功能，实现前后端的连接。整个平台完成之后我非常有成就感，通过这次实验，我全面了解了前后端开发的全过程，也知道了数据库设计比我以为的复杂很多，它需要充分考虑到平台所需要提供的功能，需要进行多次的迭代修改才能最终完成。

在设计数据库时，一开始我并没有分出猫猫种类表、位置表、食物种类表和食物生产厂家表，但在向表里增加了多条数据后，我发现一张表里存储了很多相同的信息，尤其是位置名称、食物生产厂家名称等需要存储的字符串都很长，这样造成了存储空间的损失。因此我后来将这些属性各自抽取出来存储在一张单独的表里，在原表里建立外键连接，这样大大节省了存储的空间。

在实现前端时，由于我之前用html、JavaScript写过网页版的前端，一度觉得排版非常复杂，而且这次所需实现的平台功能较少，所以我选择了学长推荐的pyqt5来实现前端，使用pymysql进行前后端连接，借助Qt Designer，较为方便地完成的前端的设计。

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，给了同学们较大的自由发挥的空间，过程也非常有趣。