###### 负责的任务：数据库设计、系统测试

本人参与数据库的设计，并完成系统最终的测试部分，找到部分漏洞，并参与修补漏洞、完善系统功能的环节。

## 实验环境以及技术选型

#### 实验环境：

操作系统: windows 10  
数据库管理系统: SQL server 2018 DEV  
系统设计与开发用到的工具：visual studio 2017

#### C#的窗体程序开发

本系统设计的主题采用了C#的TreeView 组件实现不同界面的导航菜单。

#### SQL server数据库

#### Python获取电影数据

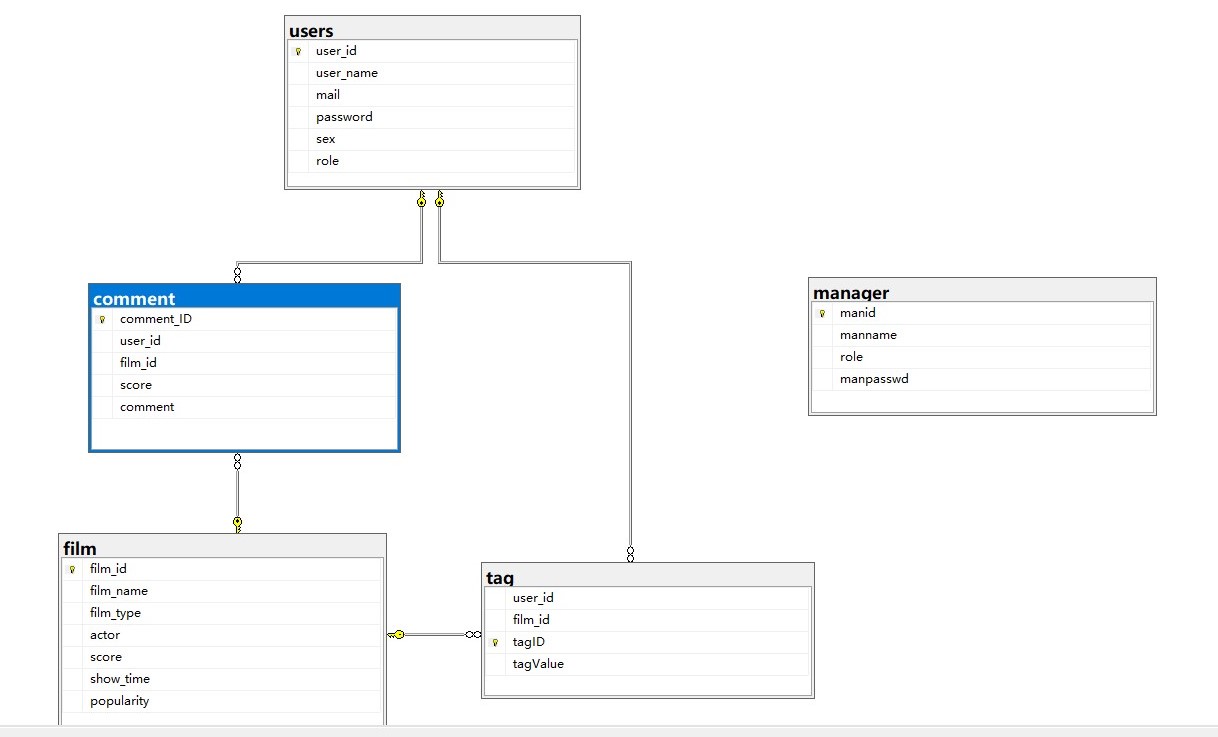
使用python的request、json解析的库，解析猫眼电影网页，获得排名榜前100的电影资讯。

## 需求分析

* 本系统包括两类使用人员：用户、管理员。
* 管理员可以使用的功能：修改用户信息、对电影进行增删查改，以及针对某些用户的不良言论进行删除。
* 用户可以使用的功能：搜索电影（输入电影全名进行精确搜索、根据输入电影名称中的关键字或者关键词进行定位、选取电影类型和设定评分阈值进行模糊搜索、根据评论热度推荐电影）、对已看过的电影进行评论、操作已收藏的电影、根据个人喜好对电影进行相关操作。
* 管理员和用户都拥有基本的注册、登录、修改密码的三个功能。

## 4、数据库设计

### 4.1数据库关系图



### 4.2完成系统中数据表的结构设计

#### 用户信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 可空 | 说明 |
| User\_id | int | 不可 | 主键，用户id，无实际意思，每次自增1 |
| Users\_name | Nvarchar(30) | 不可 | 用户姓名 |
| mail | Nvarchar(30) | 不可 | 用户注册邮箱，增加unique约束条件（规定注册邮箱不可重复）。 |
| password | Nvarchar(30) | 不可 | 用户密码 |
| sex | varchar(2) |  | 用户性别 |
| role | Nvarchar(30) |  | 角色，用户 |

#### 管理员信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 可空 | 说明 |
| manid | int | 不可 | 主键，管理员id，无实际意思，每次自增1 |
| manname | Nvarchar(30) | 不可 | 管理员用户名 |
| role | Nvarchar(30) | 不可 | 角色，管理员 |
| manpasswd | Nvarchar(30) | 不可 | 管理员密码 |

#### 电影信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 可空 | 说明 |
| Film\_id | int | 不可 | 主键，电影id，无实际意思，每次自增1 |
| Film\_name | Nvarchar(30) | 可 | 电影名 |
| film\_type | Nvarchar(30) | 可 | 电影类型 |
| Actor | nvarchar(50) | 可 | 主要演员 |
| show\_time | nvarchar(255) | 可 | 上映时间 |
| popularity | nchar(10) | 可 | 评论热度 |
| score | float | 可 | 评分 |

#### 电影标签表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 可空 | 说明 |
| tagID | int | 不可 | 主键，课程细节id，无实际意思，每次自增1 |
| user\_id | int | 可 | 外键，用户ID，联系到用户表 |
| film\_id | int | 可 | 外键，电影ID |
| tagValue | int | 可 | 用户给予电影一个标签，赋值为1-4，其对应的实际意义：看过且喜欢、未看过但喜欢、看过不喜欢、未看过且不喜欢，之后用户可以根据标签收藏或评论电影 |

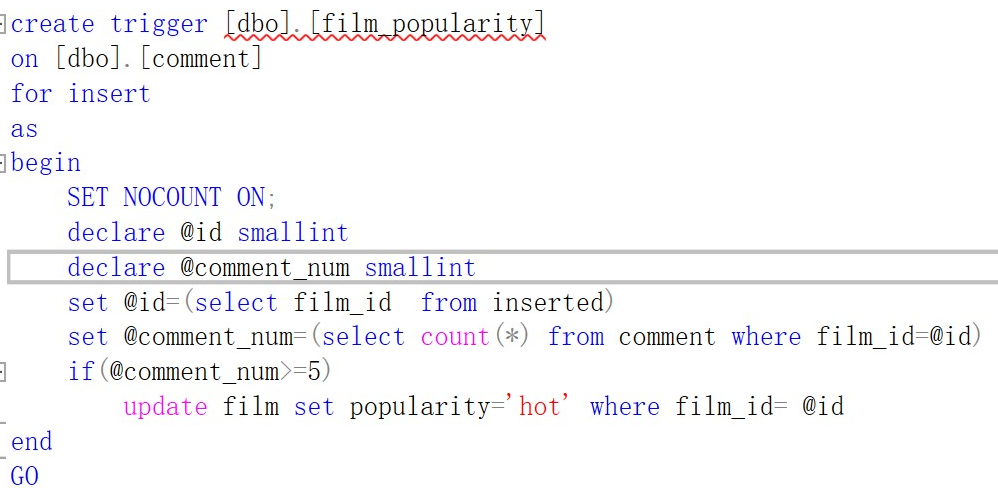
#### 评论表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 可空 | 说明 |
| Commnt\_id | int | 不可 | 主键，评论ID，无实际意思，每次自增1 |
| User\_id | int | 不可 | 外键，用户 id |
| Film\_id | int | 不可 | 外键，电影id |
| score | int | 不可 | 电影评分 |

### 4.3设计触发器

###### 触发器[film\_popularity]：

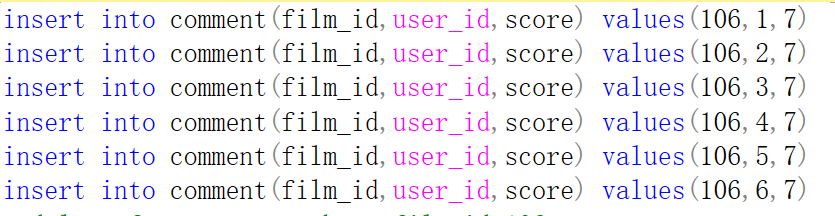
* + 功能：某电影中的评论数量达到一定数量后，该电影的热度属性会被置为”hot”，随后当用户查看评论热度榜单时，可以看到该电影。



* + 运行结果：当多于5个用户对ID为 的电影分别进行评论后，comment表上的触发器检查到insert操作之后，评论数量达到5，因此更新电影表中该电影的评论热度
    - 更新评论热度之前的榜单



* + - 方便起见，直接在SQL server中构造插入语句，模拟不同用户对ID=105的电影进行评论的行为

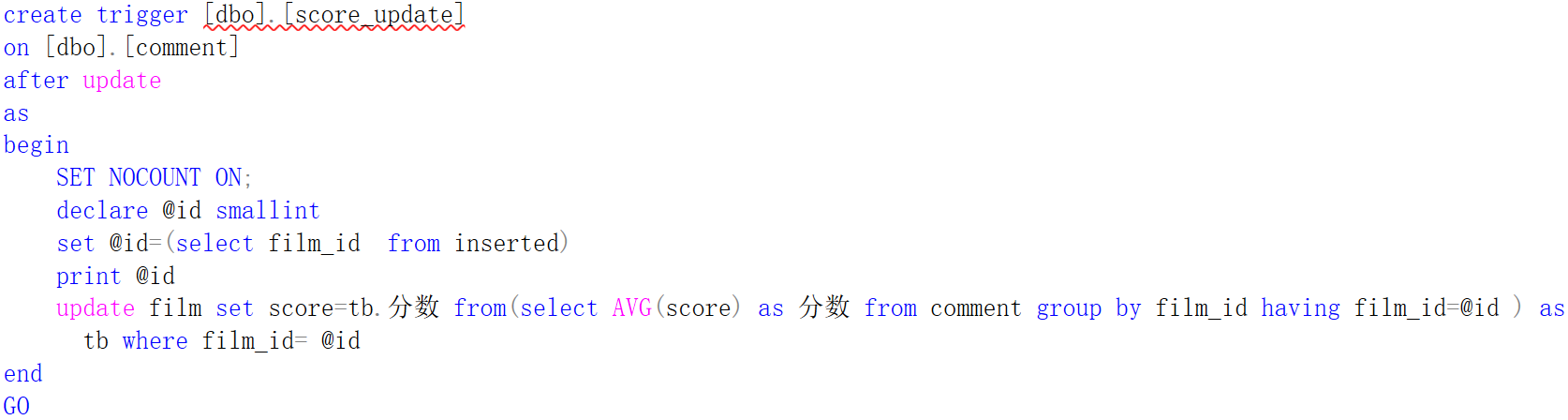


* + - 由于评论数量超过5，因此触发器被激活，评论热度榜单即显示出该电影



###### 触发器[score\_update]：

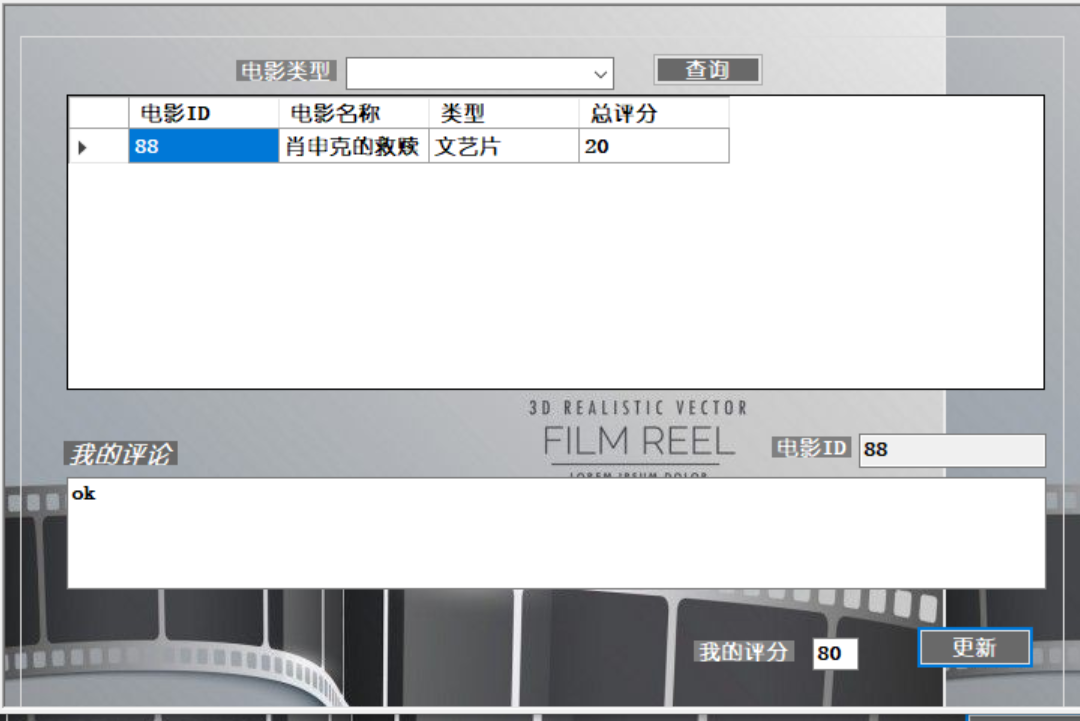
* + 功能：对电影的评分进行实时更新。由于每部电影的评分为所有用户对其评分的均值，因此，每当有用户对某电影进行评分或更新评分之后，该电影的评分就需要进行更新。



* + 运行结果:ID=88的电影原先评分为24，某用户对其评分之后，comment表的触发器检测到update操作，因此更新电影表中该电影的评分



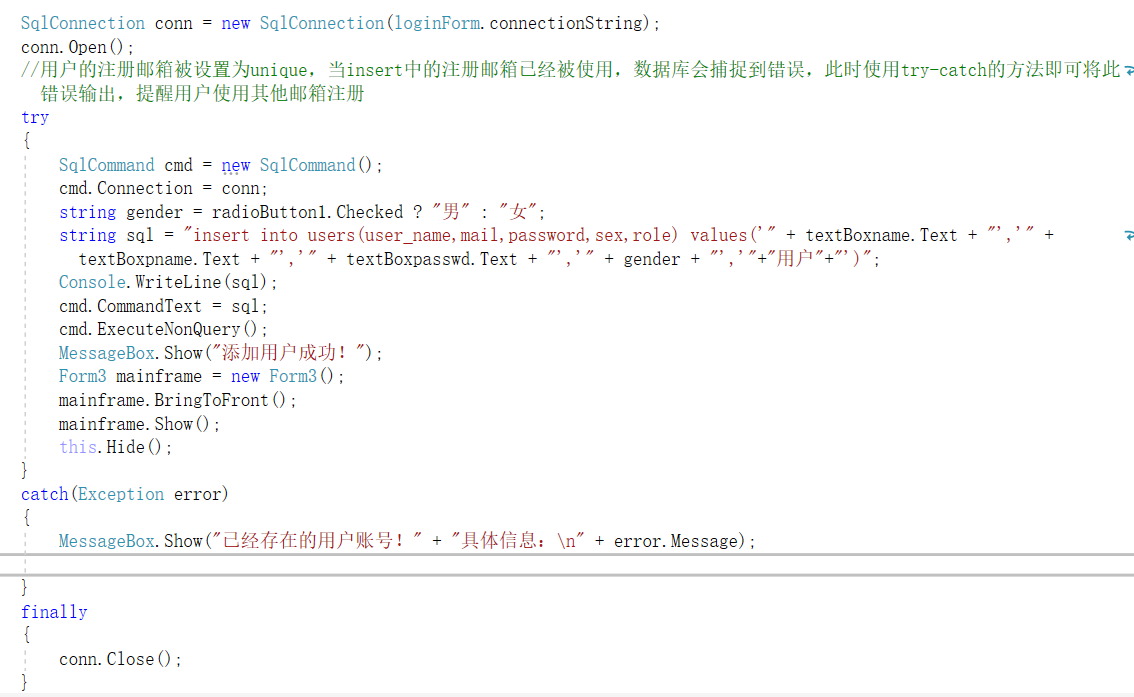
更新后，评分为20



## 5、系统测试以及完善

### 5.1 用户注册时，使用异常捕捉，判断注册邮箱（有unique约束条件）是否已存在

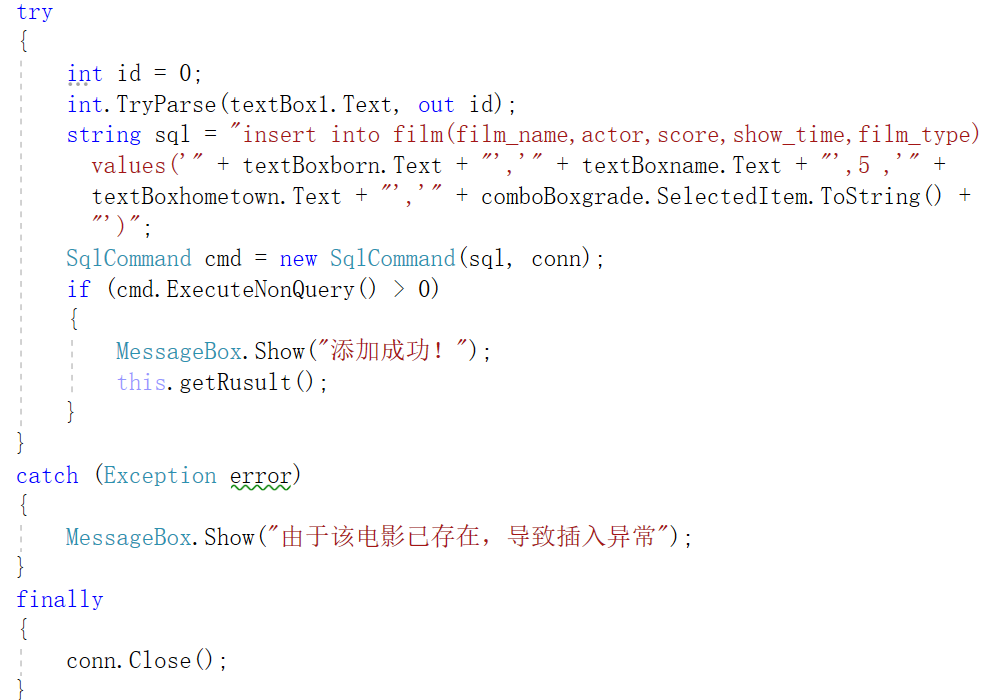
此处用到C#语言的SqlCommand对象，执行封装的方法ExecuteNonQuery()，返回结果为操作影响的行数



* 执行结果：若注册已经存在的账号，应用程序会报错，提醒重新选择注册邮箱



### 5.2 管理员对电影进行相关操作（使用异常捕捉）



###### 添加已存在电影的报错提醒



###### 更新某电影类型



###### 更新后的结果



## 6. 项目总结

本次小组合作完成电影管理系统的过程让我收获颇丰。

首先，作为组长，我深刻体会到了领导小组的不易和小组合作的重要性。从项目选题的构思，到数据库的设计优化，再到项目的实现，我需要不断考虑到小组成员技能上的不同以及项目进度的发展，不断调整分配给组员的任务。同时，组员之间的合作交流、大家发挥各自的专长和观点上的碰撞，让整个项目的开发充满了乐趣。

其次，作为本管理系统开发的主要编程人员，我学习到了很多关于使用C#进行系统开发的技能。通过参考网络上的教程，我从基础部分开始学习项目的配置、构建和调试方法，重新复习面向对象编程的相关知识，并将其实践，大大加深了我对面向对象的理解，领略其编程思想的精髓。

最重要的是，通过此项目的开发，我深刻体会到数据库设计的重要性以及理解数据库设计的理论知识。在项目的起步阶段，我们需要设计符合实际功能的数据库，并且尽可能优化表的结构。在此过程中，我们将课本上所学的范式规则、外键、约束条件、保持表的完整性等性质应用到表的设计中，学会了灵活运用它们，并在不断地尝试、碰壁中更深刻理解相关知识。之后，在编程阶段，以嵌入式SQL语句实现和后台数据库交互时，我逐步熟悉了增删查改语句的格式，以及异常情况的处理。最后，通过设计两个触发器和调试的过程，我才真正理解了触发器的本质和优点。但是，由于现阶段知识和能力的不足，我们尚未运用查询优化和事务执行的相关内容。今后，我们将继续尽力完善该系统，以实现更多功能。

## 7.参考文献

* 博文：触发器deleted 表和 inserted 表详解

<https://www.cnblogs.com/joetao/articles/2250521.html>

* SQL Server 2008 [SQL.Server2008中文版关系数据库基础与实践教程]
* 《Pro C# 5.0 and the .NET 4.5 Framework》