基于恰烂钱设计风格的推箱子游戏信息管理系统

真的在恰烂钱

华东理工大学

数据库

[1. 数据库设计概述 2](#_Toc26292794)

[2. 需求分析 2](#_Toc26292795)

[2.1. 功能需求 2](#_Toc26292796)

[2.2. 数据需求 2](#_Toc26292797)

[2.3. 数据流图 2](#_Toc26292798)

[3. 概念结构设计 2](#_Toc26292799)

[3.1. 实体与联系 2](#_Toc26292800)

[3.2. E-R图 2](#_Toc26292801)

[4. 逻辑结构设计 2](#_Toc26292802)

[4.1. 数据库模型 2](#_Toc26292803)

[4.2. 数据库模型函数依赖集 2](#_Toc26292804)

[5. 物理结构设计（程序设计） 2](#_Toc26292805)

[5.1. 游戏用户信息管理系统的的功能模块 2](#_Toc26292806)

[5.2. 主要模块的功能描述 2](#_Toc26292807)

[5.3. 数据库源代码 3](#_Toc26292808)

[5.3.1. 数据库表设计 3](#_Toc26292809)

[5.3.2. 数据初始化 3](#_Toc26292810)

[6. 数据库的实施与维护 3](#_Toc26292811)

[6.1. 开始界面 3](#_Toc26292812)

[6.1.1. 运行截图 3](#_Toc26292813)

[6.2. 数据库连接 3](#_Toc26292814)

[6.2.1. 源代码 3](#_Toc26292815)

[6.3. 登录界面 3](#_Toc26292816)

[6.3.1. 源代码 3](#_Toc26292817)

[6.3.2. 运行截图 3](#_Toc26292818)

[6.4. 商店界面 3](#_Toc26292819)

[6.4.1. 源代码 3](#_Toc26292820)

[6.4.2. 运行截图 3](#_Toc26292821)

[6.5. 游戏界面 3](#_Toc26292822)

[6.5.1. 运行截图 4](#_Toc26292823)

[6.6. 积分上传与更新 4](#_Toc26292824)

[6.6.1. 源代码 4](#_Toc26292825)

[6.6.2. 运行截图 4](#_Toc26292826)

[6.7. 查看排行榜 4](#_Toc26292827)

[6.7.1. 源代码 4](#_Toc26292828)

[6.7.2. 运行截图 4](#_Toc26292829)

[7. 运行环境描述 4](#_Toc26292830)

[7.1. 选择语言 4](#_Toc26292831)

[7.1.1. SQL语言的特点 4](#_Toc26292832)

[7.1.2. SQL功能与特性 4](#_Toc26292833)

[7.1.3. C++语言的特点 4](#_Toc26292834)

[7.2. 选择数据库 4](#_Toc26292835)

# 数据库设计概述

我们在python程序设计课程中开发设计了一款推箱子小游戏，当时通过在推箱子小游戏中利用txt文本进行游戏数据记录,

1. 游戏记录无法保存，无法比较各玩家之间的记录高低，游戏获胜也无法感受到成就感
2. 玩家希望能够增加游戏资金从而为炮塔提供升级功能

为此我们考虑通过建立数据库系统以完善该游戏的游戏体验。

# 需求分析

## 功能需求

1. 用户
2. 用户注册：首次进入游戏的玩家需要输入用户名和密码进行注册
3. 用户登录：已注册过的玩家可以通过登录已有的用户名和密码进入游戏
4. 积分
5. 积分上传：在结束游戏以后玩家可以选择将自己当前所获积分记录和昵称上传到自己的用户名下
6. 积分更新：当上传的积分记录高于该用户之前的游戏记录时，该用户的最高分将被覆盖更新
7. 积分查询：用户在结束游戏以后可以查询积分表，查看当前各用户的最高分排行榜，同时调出该积分所对应的用户的昵称
8. 资金
9. 资金充值：在游戏开始前可以通过点击不同金额的充值按钮来提升用户在游戏中可使用的资金
10. 资金查询：调用该用户当前的资金金额并即时地反馈在商店界面上

## 数据需求

1. 用户基本数据
2. 用户名（主键，长度不超过50个字符）
3. 密码（长度不超过50个字符）
4. 积分基本数据
5. 昵称（长度不超过255个字符）
6. 分值（取整的数字，长度不超过10个字符）
7. 资金基本数据
8. 金额（取整的数字，长度不超过10个字符）

## 数据流图

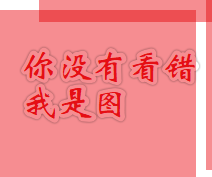


图 1游戏用户数据的处理流程图

# 概念结构设计

## 实体与联系

（1）用户（用户名，密码）

（2）积分（用户名，昵称，分值）

（3）资金（用户名，金额）

## E-R图

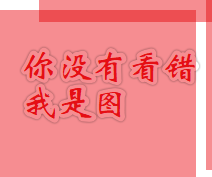


图 2 ER图

# 逻辑结构设计

## 数据库模型

数据字典：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表中列名 | 数据类型 | 数据长度 | 可否为空 | 说明 |
| UID | int | 10 | Not null（主键） | 用户id |
| UserName | varchar | 55 |  | 用户名 |
| PassWord | varchar | 55 |  | 密码 |
| NickName | varchar | 55 |  | 昵称 |

表格 1 login用户信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表中列名 | 数据类型 | 数据长度 | 可否为空 | 说明 |
| GID | int | 10 | Not null（主键） | 场次id,每局游戏一个id |
| DID | int | 10 | Not null（外键） | DLCid |
| UID | int | 10 | Not null（外键） | 用户id |
| EndTime | DATE | 3 |  | 游戏结束时间 |
| Score | int | 10 |  | 分数 |

表格 2 gamerecord游戏记录表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表中列名 | 数据类型 | 数据长度 | 可否为空 | 说明 |
| DID | int | 10 | Not null（主键） | DLCid |
| Dtype | int | 10 | Not null | DLC类型 |
| UID | int | 10 | Not null（外键） | 用户id |
| Money | int | 10 |  | 金额 |
| Date | DATE | 3 |  | 购买日期 |

表格 3 DLC购买记录表

## 数据库模型函数依赖集

（1）用户关系中主键为用户名，其中函数依赖有：用户名→密码

（2）积分关系中主键为用户名，其中函数依赖有：用户名→昵称，用户名→分值

（3）资金关系中主键为用户名，其中函数依赖有：用户名→金额

# 物理结构设计（程序设计）

## 游戏用户信息管理系统的的功能模块

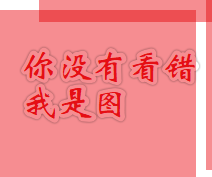


图 3 功能模块

## 主要模块的功能描述

本系统分为3个模块。包括用户管理模块、积分管理模块和资金管理模块。

1. 用户管理模块

用户管理模块的功能主要包括用户注册和用户登录。

1. 用户注册：首次进入游戏的玩家需要输入用户名和密码进行注册
2. 用户登录：已注册过的玩家需通过登录已有的用户名和密码进入游戏
3. 积分管理模块

积分管理模块的主要功能包括积分上传、积分更新和积分查询。

1. 积分上传：在结束游戏以后玩家可以选择将自己当前所获积分记录和昵称上传到自己的用户名下（也可以选择不上传）
2. 积分更新：当上传的积分记录高于该用户之前的游戏记录时，该用户的最高分将被覆盖更新（当低于或等于之前的游戏记录时，则不会更新）
3. 积分查询：用户在结束游戏以后可以查询积分表，查看当前各用户的最高分排行榜，同时会显示该积分所对应的用户的昵称
4. 资金管理模块

资金管理模块的主要功能包括资金充值和资金查询。

1. 资金充值：在游戏开始前可以通过点击不同金额的充值按钮来提升用户在游戏中可使用的资金
2. 资金查询：在充值时该用户当前的资金金额会即时地显示在商店界面上

## 数据库源代码

### 数据库表设计

/\*  
Navicat MySQL Data Transfer  
  
Source Server : my  
Source Server Version : 50725  
Source Host : 182.254.217.138:3306  
Source Database : box\_vs\_sql  
  
Target Server Type : MYSQL  
Target Server Version : 50725  
File Encoding : 65001  
  
Date: 2019-12-21 09:53:03  
\*/  
  
SET FOREIGN\_KEY\_CHECKS=0;  
  
-- ----------------------------  
-- Table structure for `dlc`  
-- ----------------------------  
DROP TABLE IF EXISTS `dlc`;  
CREATE TABLE `dlc` (  
 `DID` int(10) NOT NULL,  
 `Dtype` int(10) NOT NULL,  
 `UID` int(10) NOT NULL,  
 `money` int(10) NOT NULL,  
 `date` datetime DEFAULT NULL,  
 PRIMARY KEY (`DID`),  
 KEY `dlcforeignUID` (`UID`),  
 CONSTRAINT `dlcforeignUID` FOREIGN KEY (`UID`) REFERENCES `login` (`UID`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;  
  
-- ----------------------------  
-- Records of dlc  
-- ----------------------------  
  
-- ----------------------------  
-- Table structure for `gamerecord`  
-- ----------------------------  
DROP TABLE IF EXISTS `gamerecord`;  
CREATE TABLE `gamerecord` (  
 `GID` int(10) NOT NULL,  
 `DID` int(10) NOT NULL,  
 `UID` int(10) NOT NULL,  
 `EndTime` datetime DEFAULT NULL,  
 `Score` int(10) DEFAULT NULL,  
 PRIMARY KEY (`GID`),  
 KEY `loginforeignUID` (`UID`),  
 KEY `loginforeignDID` (`DID`),  
 CONSTRAINT `loginforeignDID` FOREIGN KEY (`DID`) REFERENCES `dlc` (`DID`),  
 CONSTRAINT `loginforeignUID` FOREIGN KEY (`UID`) REFERENCES `login` (`UID`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;  
  
-- ----------------------------  
-- Records of gamerecord  
-- ----------------------------  
  
-- ----------------------------  
-- Table structure for `login`  
-- ----------------------------  
DROP TABLE IF EXISTS `login`;  
CREATE TABLE `login` (  
 `UID` int(10) NOT NULL,  
 `UserName` varchar(55) NOT NULL,  
 `PassWord` varchar(55) NOT NULL,  
 `NickName` varchar(55) NOT NULL,  
 PRIMARY KEY (`UID`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;  
  
-- ----------------------------  
-- Records of login  
-- ----------------------------

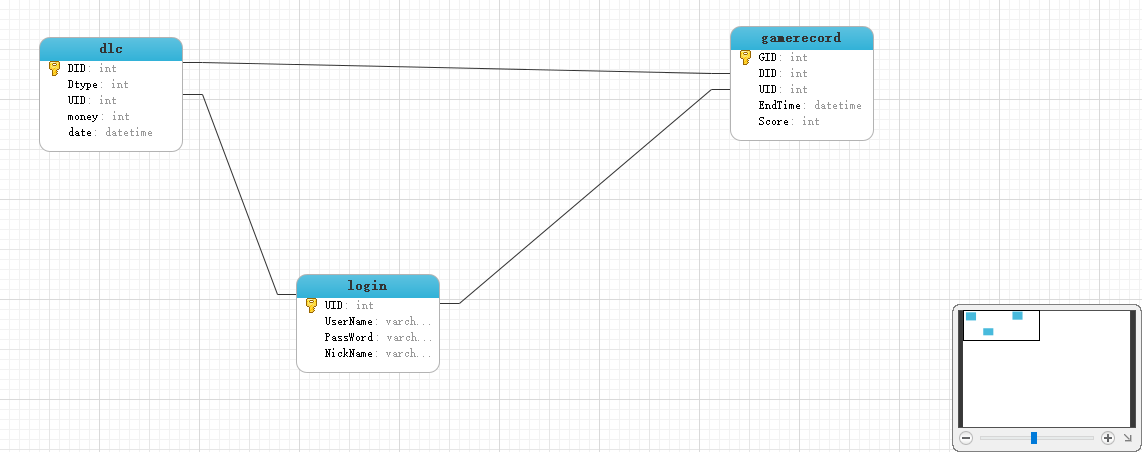


图 4 数据表创建结果

### 数据初始化

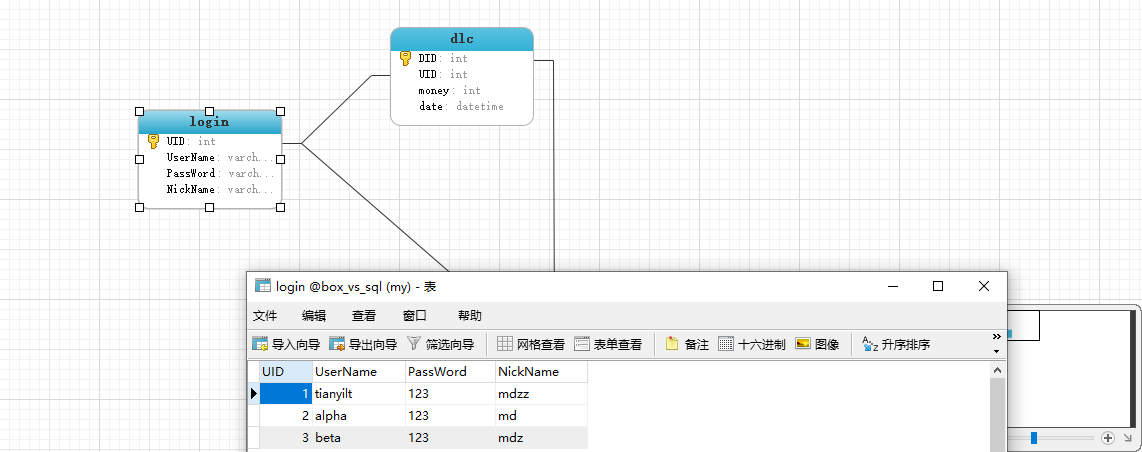
1. 初始化用户表login

insert into login values

(1,'tianyilt','123','mdzz'),

(2,'alpha','123','md'),

(3,'beta','123','mdz');



1. 初始化dlc购买记录表

insert into dlc values

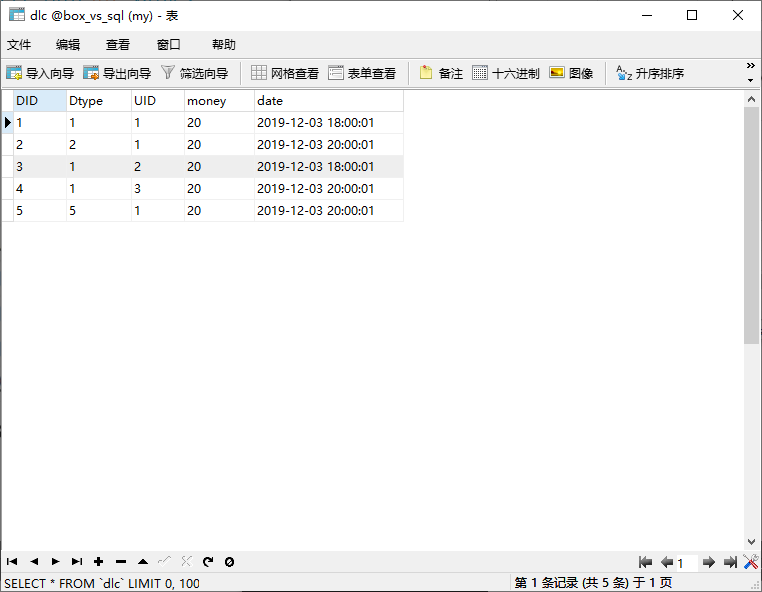
(1,1,1,20,'2019-12-03 18:00:01'),

(2,2,1,20,'2019-12-03 20:00:01'),

(3,1,2,20,'2019-12-03 18:00:01'),

(4,1,3,20,'2019-12-03 20:00:01'),

(5,5,1,20,'2019-12-03 20:00:01');



1. 初始化gamerecord

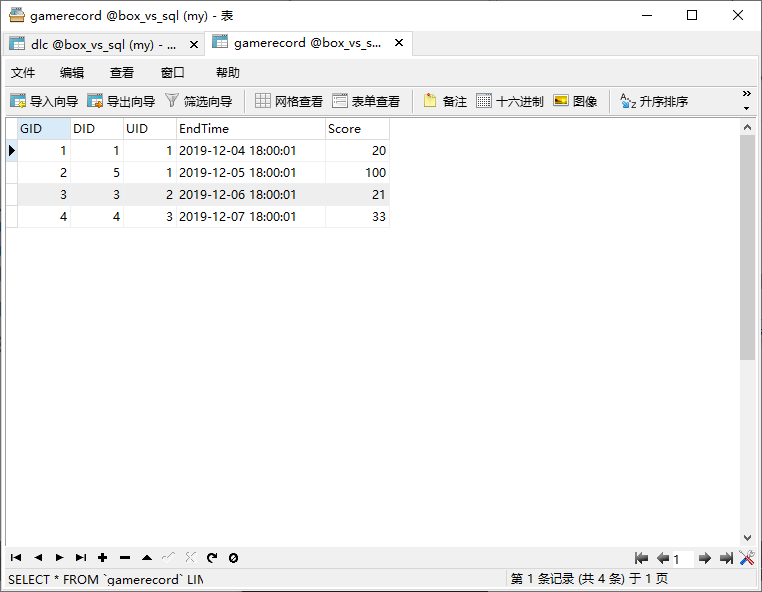
insert into gamerecord values

(1,1,1,'2019-12-04 18:00:01',20),

(2,5,1,'2019-12-05 18:00:01',100),

(3,3,2,'2019-12-06 18:00:01',21),

(4,4,3,'2019-12-07 18:00:01',33);



### 建立视图

1. dlcbuy用户dlc购买视图

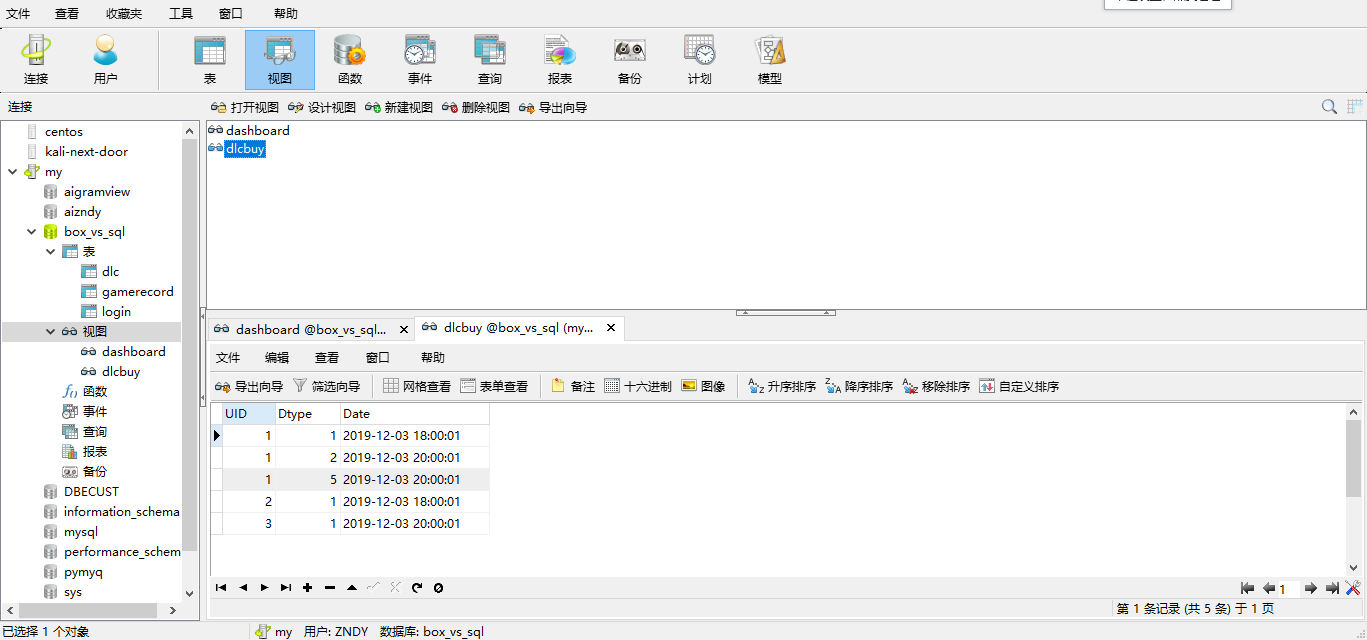
CREATE VIEW dlcbuy

as

SELECT UID, Dtype, Date

FROM dlc

ORDER BY UID;



1. dashboard总分榜查询视图

CREATE VIEW dashboard

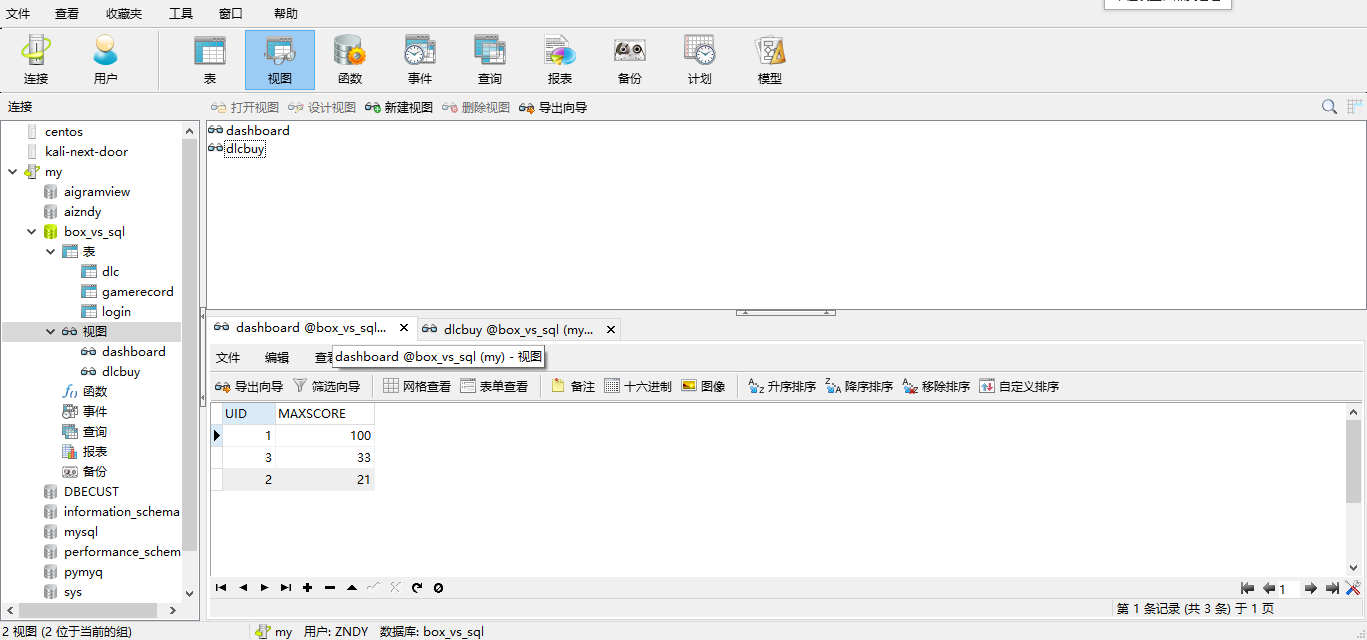
as

SELECT UID, MAX(Score) AS MAXSCORE

FROM gamerecord

GROUP BY UID

ORDER BY MAXSCORE DESC;



# 数据库的实施与维护

游戏介绍

参数

## 开始界面

### 运行截图

## 数据库连接

### 源代码

import pymysql  
"""  
这只是一个链接数据库的demo  
"""  
db = pymysql.connect('182.254.217.138', 'ZNDY', 我是不会告诉你密码的, 'box\_vs\_sql', charset="utf8")  
  
cursor = db.cursor()  
  
# SQL 插入语句  
sql = "INSERT INTO EMPLOYEE(FIRST\_NAME, \  
 LAST\_NAME, AGE, SEX, INCOME) \  
 VALUES ('%s', '%s', %s, '%s', %s)" % \  
 ('Mac', 'Mohan', 20, 'M', 2000)  
  
try:  
 # 执行sql语句  
 cursor.execute(sql)  
 # 提交到数据库执行  
 db.commit()  
except:  
 # 如果发生错误则回滚  
 db.rollback()  
  
db.close()

## 登录界面

### 源代码

### 运行截图

## 商店界面

### 源代码

### 运行截图

## 游戏界面

### 运行截图

## 积分上传与更新

### 源代码

### 运行截图

## 查看排行榜

### 源代码

### 运行截图

# 运行环境描述

## 开发语言

Python神教,千秋万载,一统江湖

## 开发平台

Microsoft Windows 10

Anaconda集成环境

## 数据库

Mysql[[1]](#footnote-1): MySQL是根据GNU通用公共许可证条款的自由和开源软件，也可以在各种专有许可证下使用。MySQL由瑞典公司MySQL AB拥有和赞助，该公司被Sun Microsystems(现在的Oracle公司)收购。2010年，当Oracle收购Sun时，Widenius发起了开源MySQL项目来创建MariaDB。

## 服务器端参数

## 链接数据库工具

Navicat for MySQL

1. <https://www.mysql.com/> [↑](#footnote-ref-1)