学籍管理系统设计文档

[文档副标题]

作者：申现文

版本：1.2

**版本修订历史**

|  |  |  |  |
| --- | --- | --- | --- |
| 修订日期 | 修订人 | 修订内容描述 | 备注 |
| 2020.12.10 | 申现文 | 初始化文档 |  |
| 2020.12.12 | 申现文 | 修改数据模型 | 加入字段，明确语义 |
| 2020.12.15 | 申现文 | 载入测试数据，添加总结 |  |

目录

[一． 前言 1](#_Toc58941450)

[1． 编写目的 1](#_Toc58941451)

[2． 需求背景 1](#_Toc58941452)

[3． 需求定位 1](#_Toc58941453)

[4． 参考资料 1](#_Toc58941454)

[二． 设计目的 1](#_Toc58941455)

[1． 目标 1](#_Toc58941457)

[2． 系统功能 2](#_Toc58941458)

[3． 外部需求 2](#_Toc58941459)

[外部程序 2](#_Toc58941460)

[相关约定 2](#_Toc58941461)

[软件支持 2](#_Toc58941462)

[三． 需求分析 2](#_Toc58941463)

[1． 确定数据流、控制流 2](#_Toc58941465)

[（2） 登记查询系统第二层数据流图 3](#_Toc58941466)

[（3） 登记查询系统第三层查询图 4](#_Toc58941467)

[2． 数据字典设计 5](#_Toc58941468)

[（1） 数据元素定义 5](#_Toc58941469)

[数据结构定义 5](#_Toc58941470)

[数据流定义 5](#_Toc58941471)

[数据处理定义 6](#_Toc58941472)

[3． 系统模块设计 7](#_Toc58941473)

[四． 概念结构设计 7](#_Toc58941474)

[五． 逻辑结构设计 8](#_Toc58941475)

[六． 物理结构设计 10](#_Toc58941476)

[七． 数据库系统实施 10](#_Toc58941477)

[1． 数据库建表 11](#_Toc58941478)

[2． 数据录入 11](#_Toc58941479)

[3． 视图、索引的建立 11](#_Toc58941480)

[八． 学籍管理系统运行 11](#_Toc58941485)

[1． 学生基本信息查询 12](#_Toc58941486)

[（1） 按学号查询 12](#_Toc58941487)

[（2） 按姓名查询 12](#_Toc58941488)

[2． 学生成绩信息查询 13](#_Toc58941489)

[3． 快被开除学生信息查询 14](#_Toc58941490)

[九． 总结 14](#_Toc58941491)

[1． 遇到的主要问题与解决方案 14](#_Toc58941492)

[2． 其他 15](#_Toc58941493)

[附录一 16](#_Toc58941494)

# 前言

## 编写目的

本文档基于系统的需求分析的基础上，主要解决实现该系统的程序模块设计，包括如何将系统划分为多个子模块，子模块间如何进行通讯，以及数据结构、模块结构的设计。该文档旨在向系统分析人员以及软件开发人员了解本系统的基本功能，在此基础上进行进一步的开发工作。本文档的预读作者为开发小组成员、项目管理员以及数据库管理人员。

## 需求背景

学籍管理系统是学校信息管理的重要组成部分，它的内容对于一个学校的学生学籍管理起着至关重要的作用。如今，随着学生数量的增加，传统的学生管理模式不但不能满足现代教育的要求，更无法适应时代的发展。因此我认为有必要建立一个学籍管理系统，使学生的信息管理工作系统化，程序化，提高信息处理的速度和准确性，能够及时、准确、有效的查询和修改学生资料，并保证学生学籍系统的安全性。本项目采用python的dijango框架以及mysql数据库管理系统进行系统设计。本系统的建成将大大提高学校学生管理工作者的工作效率。

## 需求定位

* 解决学生管理人员查询、处理学生学籍、成绩效率低的痛点
* 解决学生需要快速查询自己成绩并获取相关提示的需求
* 在保证内部数据安全性的前提下，提升数据管理，数据校核效率

## 参考资料

1. 数据库系统概论

# 设计目的



## 目标

根据教务处业务需求，制定设计全面、高效的学籍管理系统，以便于解决教师、学籍管理人员对于学生信息管理的需求，也同样解决学生查询自己培养方案、各科成绩的需求，以此提高各用户的办公、学习效率。

## 系统功能

本系统设计功能模块如下：

1. 学生基本信息模块
2. 学生成绩管理模块
3. 教师信息模块
4. 管理员管理模块
5. 系统运维模块

## 外部需求

### 外部程序

本系统主要采用python3.7作为后端应用开发语言，DBMS选型为Mysql8，Windows10作为系统平台，应用开发环境为pycharm2020.3.

### 相关约定

* 所有命名皆明确、具有描述性，不采用拼音或混杂的命名方式
* 字符集编码采用UTF-8
* 除特殊说明外，日期格式均采用date格式
* 除特别说明外，所有字段默认都设置不充许为空， 需要设置默认值。

### 软件支持

操作系统：Windows10

DBMS：Mysql

数据库查询浏览工具：Navicat

# 需求分析

进行数据库设计首先必须准确了解和分析用户需求（包括数据与处理）。需求分析是整个设计过程的基础，也是最困难，最耗时的一步。对此采用SA(Structured Analysis)结构化分析方法，从最上层的系统组织结构入手，采用自顶向下，逐层分解的方式分析系统，绘制数据流图，随后依据分解完成的数据，制定数据字典。



## 确定数据流、控制流

1. 整体数据流图

首先，根据用户的需求，设计总体的数据流与控制流，已知该系统用户的用户分为三类，管理员（教务人员），教师与学生。其中管理人员享用数据管理系统的最高权限，负责全量数量的管理，可进行数据的录入与删除，对应的系统模块为管理员管理模块。教师与学生可对相关数据进行查询，教师可查询所教授班级成绩情况，学生可查询自己所有成绩情况，且可导出成绩单，因此数据访问的权限均有所不同。整体数据流图如下：



图一 整体数据流图

### 登记查询系统第二层数据流图

第二层查询图为第一层总体图的进一步细化，成绩管理系统的主要板块为成绩管理板块，其中教务人员负责录入的主要信息为学生信息，课程信息，成绩信息。而学生对成绩管理系统操作仅为发起查询请求。具体数据图如图二。



图二 成绩管理数据流图

根据业务需求，系统需要根据学生的考试成绩、总体的成绩状况，判断其是否需要补考，是否需要发送退学预警等信息，另外成绩分析系统能够根据学生的总体成绩反馈其一些具有概括性的指数，例如平均绩点，故因此设计的成绩分析数据流图如图三。



图三 成绩分析数据流图

### 登记查询系统第三层查询图

对第二层DFD图的“查询成绩”进行进一步展开，详细说明成绩查询的数据流与操作流，进行查询时，需要进行两种合法性检查，第一种为查询合法性检查，主要检查用户所在的用户组是否拥有该查询的查询权限和数据的访问权限；第二种为条件合法性检查，主要检查查询是否可以从内部数据源中提取信息，且提取信息是否满足相关约束条件与数据库安全条件。



图四 成绩分析数据流图

## 数据字典设计

### 数据元素定义

表一 数据定义表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 数据元素类 | 内部名 | 数据类型 | 注释 |
| E01 | 学生学号 | student\_id | Int | 学生实体唯一编号 |
| E02 | 教师编号 | teacher\_id | Int | 教师实体唯一编号 |
| E03 | 班级编号 | class\_id | Int | 班级实体唯一编号 |
| E04 | 课程编号 | course\_id | Char | 课程实体唯一编号 |
| E05 | 成绩 | score\_id | Tinyint |  |
| E06 | 院系 | college\_name | Varchar | 院系名称 |
| E07 | 专业名 | major\_name | Varchar |  |
| E08 | 姓名 | name | Varchar |  |
| E09 | 性别 | gender | char | 个人性别（‘男’‘女’） |
| E10 | 出生日期 | birthday | Date |  |
| E11 | 开课日期 | course\_date | Date | 课程开课时间 |
| E12 | 学分 | credit | Tinyint |  |
| E13 | 课程类型 | course\_type | Char | 必修or选修 |
| E14 | 授课学期 | semester | Char | 教学计划中的授课学期 |
| E15 | 课程名 | course\_name | Varchar |  |

### 数据结构定义

表二 数据结构定义表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 数据结构名 | 内部名 | 数据组成 | 组织方式 |
| S01 | 学生资料 | Student | E01+E07+E04+E10+E08+E09 | E01升序 |
| S02 | 教师资料 | Teacher | E02+E06+E08+E09+E10 | E02 升序 |
| S03 | 教学计划 | Plan | E06+E07+E13+E15 | E07,E06 |
| S04 | 学生成绩 | Score | E01+E04+E05+E12+E13+E15+E14 | E01 升序 |

### 数据流定义

表三 数据流定义表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 数据流名称 | 来源 | 去向 | 数据流组成 | 说明 |
| F01 | 成绩录入 | 管理人员客户端 | 成绩管理模块 | S04 |  |
| F02 | 教师信息录入 | 管理人员客户端 | 教师信息管理模块 | S02 | 仅教师入职时录入 |
| F03 | 学生信息录入 | 管理人员客户端 | 学生信息管理模块 | S01+S03 | 仅学生入学时录入 |
| F04 | 学生成绩查询 | 成绩管理模块、学生信息模块 | 学生客户端 | S04+S03+S01 |  |
| F05 | 教师授课成绩查询 | 成绩管理模块 | 教师客户端 | S04 | 仅返回所教授课程学生成绩 |
| F06 | 成绩预警信息 | 成绩管理模块 | 学生客户端、管理员客户端 | S01+S04 |  |
| F07 | 综合查询 | 所有数据源 | 管理员客户端 | S01+S02+S03+S04 | 综合查询教务信息 |

### 数据处理定义

表四 数据处理定义表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 数据处理名称 | 输入数据 | 输出数据 | 处理逻辑 | 相关数据 |
| P1.1 | 添加修改 | F02 、F03 | - | IF input not duplicate  DO P1.1  END IF  添加修改学生、教师信息 |  |
| P1.2 | 添加修改 | F01 |  | IF input NOT NULL  DO P1.2  END IF  添加修改成绩 |  |
| P2.1 | 删除 | F07 | - | IF input EXIST  do P2.1  END IF  删除学生、教师信息 | S02 S01 |
| P2.2 | 删除 | F07 | - | IF input EXIST  do P2.1  END IF  删除成绩信息 | S01 |
| P3.1 | 统计分析 | F03 F04 | F03 F04 | 对查询结果进行统计计算再输出，如班级平均分 |  |
| P3.2 | 学业预警 | F03 F07 | F03 F06 F07 | IF DO3.2 满足预警条件  输出 F07  END IF | S03 S01 |

## 系统模块设计

以下为功能模块的描述：

表五 功能模块表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 功能名称 | 启动条件 | 输入 | 功能 | 输出 |
| 登录 | 无 | 账号密码 | 登录相关账号 | 登录超过（失败）信息 |
| 查询 | 任意账号登录成功 | 查询条件，点击查询bar | 对目标信息进行查询 | 显示查询结果 |
| 修改 | 管理员账号登录成功 | 需修改的目标信息 | 修改目标信息 | 修改成功or失败message |
| 删除 | 管理员账号登录成功 | 需删除的信息 | 删除目标信息 | 删除成功or失败message |
| 打印 | 任意账号登录成功 | 需打印的成绩信息 | 打印成绩 | 打印后的学生成绩结果 |

# 概念结构设计

概念结构是各种数据模型的共同基础，它比数据模型更独立于机器、更抽象，从而更加稳定概念结构设计是整个数据库设计的关键，本份文档采用ER图进行设计，采用混合策略设计全局概念框架，并严格遵循BCNF范式与完整性规则。具体ER图如图三。

根据BCNF范式约定，将课程实体划分为开课实体与课程实体，开课实体依赖于老师与课程存在，对于教学计划，根据原有方案做出了一个假设-不同院系拥有相同名称的专业，因此专业-教学计划不为1-1映射关系，（教学计划）函数依赖于（专业名+院系名）,故将教学计划实体分解为教学计划实体与教学计划详情实体，后者依赖前者存在。



图三 学籍管理系统ER图

# 逻辑结构设计

根据第四部分的概念结构设计，设计学籍管理系统的逻辑结构，转换为Mysql数据库所支持的数据模型，并继续优化，本部分主要集中与各部分ER图存在的三类冲突，即属性冲突，命名冲突，和结构冲突，并确定关系模式的数据和码。设计的模型结构如下：

表六 学生表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名 | 元数据命名 | 数据类型 | 宽度 | 是否有空 | 外键 |
| 学号 | student\_id | Int | 8 | 无 | 无 |
| 姓名 | name | Varchar | 5 | 无 | 无 |
| 性别 | gender | char | 1 | 有 | 无 |
| 出生日期 | birthday | Date | - | 有 | 无 |
| 班级号 | class\_id | Int | 10 | 无 | classes(‘id’) |

表七 班级表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名 | 元数据命名 | 数据类型 | 宽度 | 是否有空 | 外键 |
| 班级号 | id | Int | 10 | 无 | 无 |
| 班级名 | class\_name | Varchar | 10 | 有 | 无 |
| 专业号 | major\_id | Int | 10 | 无 | majors(‘id’) |

表八 分数表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名 | 元数据命名 | 数据类型 | 宽度 | 是否有空 | 外键 |
| 学号 | student\_id | Int | 10 | 无 | students(‘id’) |
| 课程号 | course\_id | Int | 10 | 无 | courses(‘id’) |
| 分数 | score | Int | 3 | 有 | 无 |
| 是否为补考 | is\_make\_up | Int | 1 | 无 | 无 |

表九 开课表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名 | 元数据命名 | 数据类型 | 宽度 | 是否有空 | 外键 |
| 开课号 | id | Int | 10 | 无 | 无 |
| 课程名 | course\_name | Varchar | 20 | 无 | course\_type(‘name’) |
| 班级号 | class\_id | Int | 10 | 无 | classes(‘id’) |
| 授课教师 | teacher\_id | Int | 10 | 无 | teachers(‘id’) |
| 开课时间 | class\_time | Date | - | 无 | 无 |

表十 教师表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名 | 元数据命名 | 数据类型 | 宽度 | 是否有空 | 外键 |
| 教职工号 | id | Int | 10 | 无 | 无 |
| 姓名 | name | Varchar | 5 | 无 | 无 |
| 性别 | gender | char | 1 | 有 | 无 |
| 出生日期 | birthday | Date | - | 有 | 无 |
| 所属院系 | college\_id | Int | 10 | 无 | colleges(‘id’) |

表十一 教学计划详情表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名 | 元数据命名 | 数据类型 | 宽度 | 是否有空 | 外键 |
| 计划号 | id | Int | 10 | 无 | 无 |
| 课程名 | course\_name | Varchar | 10 | 无 | course\_sort(‘name’) |
| 课程类型 | course\_type | Char | 2 | 五 | 无 |
| 授课学期 | semester | Varchar | 5 | 无 | 无 |
| 所属专业 | major | Int | 10 | 无 | major（‘id’） |

表十三 专业表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名 | 元数据命名 | 数据类型 | 宽度 | 是否有空 | 外键 |
| 专业号 | id | Int | 10 | 无 | 无 |
| 专业名 | name | Varchar | 10 | 无 | 无 |
| 所属院系 | college\_name | Int | 10 | 无 | colleges(‘name’) |

表十四 课程类型表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名 | 元数据命名 | 数据类型 | 宽度 | 是否有空 | 外键 |
| 课程名 | couse\_type\_name | Varchar | 20 | 无 | 无 |
| 学分 | credit | TinyInt | 1 | 无 | 无 |

表十五 院系表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名 | 元数据命名 | 数据类型 | 宽度 | 是否有空 | 外键 |
| 院系名 | name | Varchar | 10 | 无 | 无 |

# 物理结构设计

对于关系型数据库，即OLTP系统，所有数据存储在单一节点之中，故在物理结构的设计上仅需考虑存储系统OI的效率与数据检索的效率，决定存储结构的主要因素是存取时间、存储空间以及维护代价。

对于制定的逻辑数据模型，选取一个最适合应用环境的物理结构。数据库的物理结构设计分为两步：

A、确定物理结构：存取方法和存储结构

B、评价物理结构：评价重点是时间和空间效率

因为数据库管理系统所提供的多种存储结构和存取方法等依赖于具体计算机结构的各项物理设计措施，本文档不做详细讨论，因此对学籍管理系统选定最合适的物理存储结构(数据类型、索引与主键)为主要的设计目标，对于索引和主键的设置遵循以下的规则：

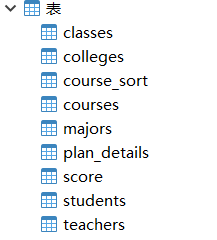
* 对于高频查询时输入的字段，例如“学号”，优先添加唯一索引
* 对于链接操作常访问的字段，优先添加普通索引
* 对于复合主键，优先添加多列索引
* 主键必须符合实体完整性约束
* 所有的索引的命名采用表面加设置索引的字段名组合，例如student\_name\_index
* 对于查询频率低的表和相关字段，不考虑添加索引
* 主键的设定根据数据逻辑模型与所选型的DBMS（Mysql）而定

# 数据库系统实施

建表阶段采用python的mysql.connect库，链接Mysql的接口完成通讯，并进行的建表，建立索引，建立视图与插入语句操作。

## 数据库建表

部分建表代码于附录一，具体模式如下图



图四 模式详情图

## 数据录入

将已有的全量数据录入数据库，并根据业务要求假设部分数据并录入，验证了完整性约束，部分数据录入代码于附录一。

## 视图、索引的建立

为方便查询，建立了学生-成绩视图与学生-老师视图，具体代码于附录一。



图四 外模式详情图

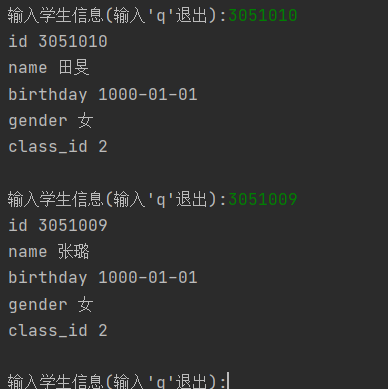


# 学籍管理系统运行

根据业务要求，进行了以下查询的测试。

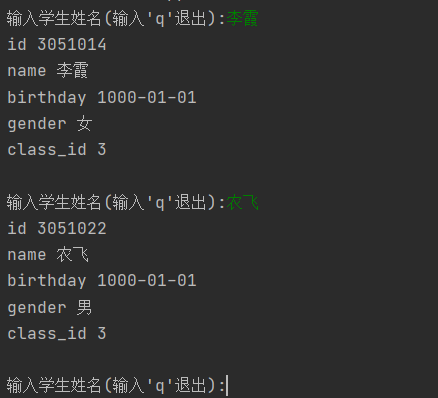
## 学生基本信息查询

### 按学号查询



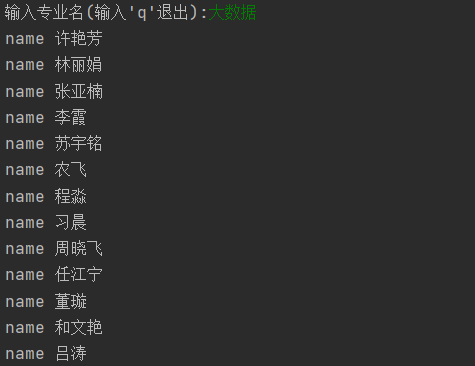
图五 学号查询结果图

### 按姓名查询



图六 学生姓名查询结果图

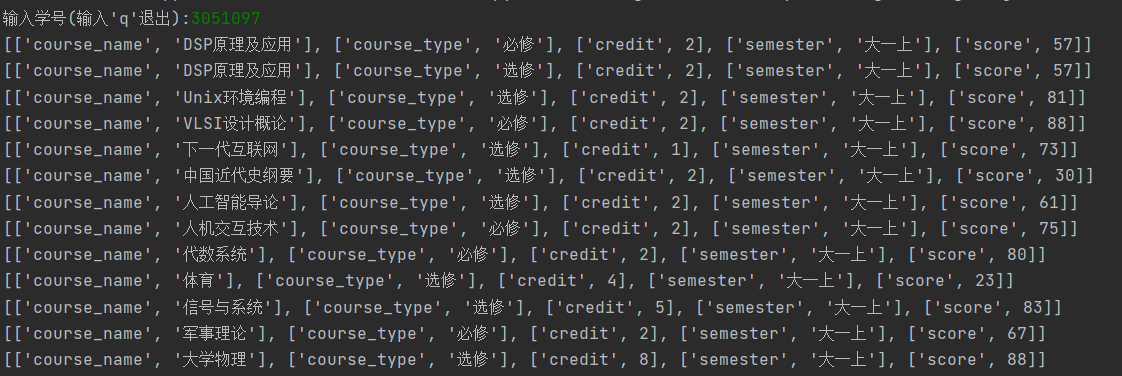
1. 按专业查询



图七 按专业信息查询结果图

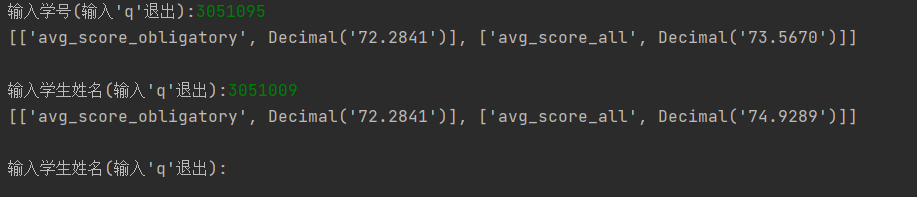
## 学生成绩信息查询

1. 查询一位学生所修的课程、性质、学期、学分及成绩



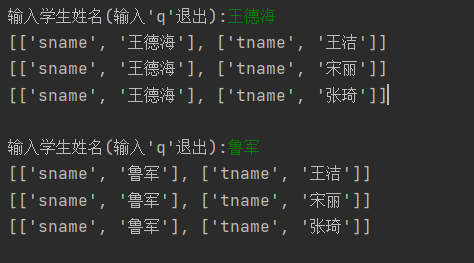
图八 学生成绩查询结果图

1. 查询学生必修课平均成绩、所有课程平均成绩



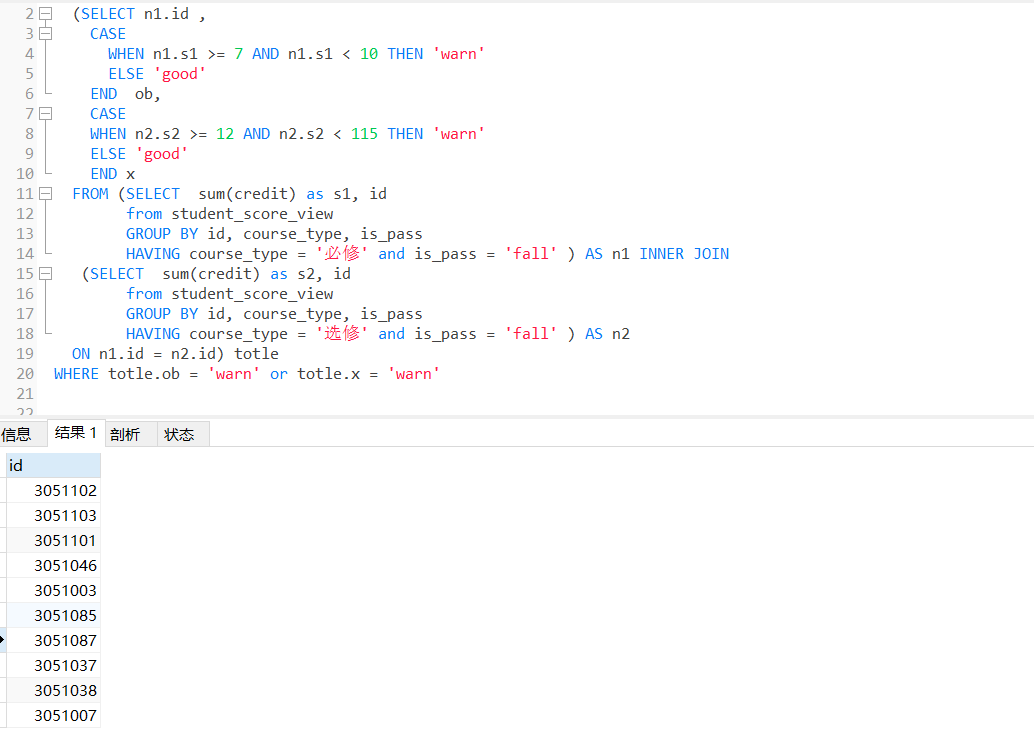
图九 学生平均成绩查询结果图

1. 被哪些老师授过课



图九 学生-老师查询结果图

## 快被开除学生信息查询



图十 查询快要被开除学生学号

# 总结

## 遇到的主要问题与解决方案

此次数据库设计的问题来源主要来源以下三方面，分别是概念模型设计阶段、逻辑模型设计以及数据载入阶段。

1. 概念模型设计阶段

概念模型设计时，经过分析、聚集和概况三阶段后，进行ER图设计，发现ER图1.0版本仅符合1NF，为了消除部分函数依赖与传递依赖达到BCNF范式，原始的4个视图被分解为多个实体，其主要问题存在于如何分解上，为了解决该问题，严格按照范式分解的步骤进行，最终结果较满意。

1. 逻辑模型设计阶段

该阶段的设计主要存在于完整性约束所引起的一系列情况，例如修改某一字段的字符长度，可能会触发约束条件导致无法修改，解决方案为删除约束进行修改，然后重新加入约束条件。

1. 数据载入阶段

在此阶段遇到一个较为棘手的问题，即数据库表被死锁的问题。一个长事务对某个表进行DDL操作时，mysql会将这个表填上Metadata lock，但如果线程任务转换为其他事务，该事务将继续休眠，该表将被死锁。在插入score数据时，因为一些不当操作导致以上情况发生，通过查阅相关资料，了解到可以通过配置长事务自动回滚时间，解决死锁，也可以直接监控线程，回滚引起死锁的事务，进而消除死锁。

## 其他

# 附录一

部分sql代码：

(1)教学计划表

CREATE TABLE IF NOT EXISTS plan\_detials(

`id` INT(10) PRIMARY KEY AUTO\_INCREMENT ,

`course\_name` VARCHAR(20) NOT NULL REFERENCES course\_sort(`name`) ,

`course\_type` CHAR(2) NOT NULL DEFAULT '选修',

`semester` CHAR(3) NOT NULL DEFAULT '大一上',

`major\_id` INT(10) REFERENCES majors(`id`)

)ENGINE=InnoDB DEFAULT CHARSET=UTF8

1. 学生表

CREATE TABLE IF NOT EXISTS students(

`id` INT(10) PRIMARY KEY AUTO\_INCREMENT,

`name` VARCHAR(5) NOT NULL DEFAULT '',

`birthday` DATE DEFAULT '1000-10-10',

`gender` CHAR(1) CHECK((`gender` = '男') or (`gender` = '女')) DEFAULT '男,

`class\_id` INT(10) REFERENCES classes(`id`)

)ENGINE=InnoDB DEFAULT CHARSET=UTF8

1. 成绩索引

CREATE INDEX std\_id\_index ON students(`id`)",

"CREATE INDEX score\_id\_index ON score(`id`)

1. 学生-成绩视图

CREATE VIEW student\_score\_view AS(

SELECT DISTINCT s.id, s.name , sc.score, cos.name as course\_name, cos.credit,

cs.course\_id, pd.course\_type, pd.semester,

CASE

WHEN sc.score < 60 THEN 'fall'

WHEN sc.score >= 60 THEN 'pass'

END is\_pass

FROM students s INNER JOIN

score sc ON s.id = sc.student\_id

INNER JOIN courses cs

ON cs.course\_id = sc.course\_id

INNER JOIN course\_sort cos

ON cos.name = cs.course\_name

INNER JOIN classes cl

ON cl.id = s.class\_id

INNER JOIN plan\_details pd

ON pd.major\_id = cl.major\_id AND pd.course\_name = cs.course\_name

)

1. 学生信息插入代码

def exe\_sql\_queries(query, cur, connect):

try:

cur.execute(query)

except ValueError as v:

print(v)

except :

print ("不明错误", sys.exc\_info()[0])

else:

connect.commit()

print("插入成功")

def insert\_student\_data(data, cur, connect):

"""

向数据库中插入学生信息

Parm: data, DataFrame, 学生信息

cur, 游标

connect,链接

Return: null

"""

query = "INSERT students (id, name, gender, class\_id, birthday) values ('{}', '{}', '{}', '{}', '{}')".format(\*values)

exe\_sql\_queries(query, cur, connect)

with open(road, encoding='UTF-8') as f:

file = pandas.read\_csv(f)

print(file.head())

for i, value in file.iterrows():

#insert\_student\_data(value, cur, conn)

insert\_tudent\_data(value, cur, conn)