DB Project 1 Document

武柯昊 21307140016

May 19, 2023

1 功能概要

本数据库的主题是学生成绩管理系统，其主要包含的功能是：

- 学生查询所修课程的成绩、查询所有课程的均分和均分排名
- 教师录入、查询所教授课程各学生成绩和均分

2 详细设计

2.1 数据库概念与结构设计

根据所要设计的数据库的功能，我们发现，此数据库应当包含三个主体，分别是学生、课程、教师；两个关系，即教师教授课程、学生修读课程。此中，一位教师可以教授数门课程、一位学生也可以修读数门课程、一门课程有数位学生修读，分别对应“教授”关系是一个一对多关系，而“修读”关系是一个多对多关系。因此，我们可以画出此结构对应的 ER 图如下：

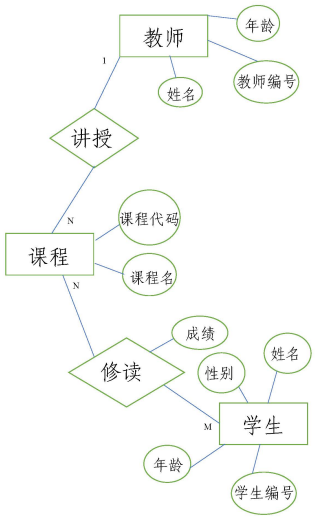

Figure 1: 数据库设计的 E-R 图

根据如上 ER 图，我们很容易地将此转化为关系模型：

- 教师（姓名，教师编号）
- 课程（课程代码、课程名、教师编号）
- 学生（姓名，学生编号，性别，年龄）
- 选课（课程代码，学生编号，成绩）

注意到这样的关系模型满足第三范式，是一个结构上较为合理的设计。

2.2 重要功能和部署细节

在本系统中，我们期望实现如下功能：通过一个登陆界面，教师、学生、管理员可以进入不同的界面。对于学生而言，他仅能查询自己所修读课程和对应的成绩、所修读课程的时间，以及均分、排名等；对于老师，他可以增删改自己所授课程、学生成绩，查看班级均分等；对于管理员，它具有所有权限，相当于 Django 的超级用户 SuperUser。然而，在实际操作过程中，我本想利用 Django 自带的 Authentication 功能实现登陆，但这意味着 Teacher，Student，Admin 这些 class 需要设立一个向 User（Django 自带模型）的外键，而 User 类在注册过程中，系统会自动分配唯一的 user_id，这一点在使用网页 fetch 信息传至后端注册时无法实现。若不采用这种做法，明文存储密码也存在安全问题。因此，实际上正确的方法应当是将用户注册时使用的密码 Hash（或用其他方式加密）并存储在对应的类中。当然这里同样存在一个问题，对于 Django 自带的认证功能，它对于密码的长度、复杂程度都有要求，但我们在自定义认证时如何做到这一点仍需要在前后端进一步设计。

对于本系统而言，最重要的功能即是对于不同关系模型的增删查改。我采用了 django 构建前后端交互渠道，利用 python 内置的 sqlite3 数据库进行设计。对于数据库的使用而言，实际上也可以采用 MySQL，PostgreSQL 等常用的数据库，但 sqlite3 和 python 配合得更好，也不需要再设计数据库和后端 python 之间的数据通路，因此我最后采用了 sqlite3 数据库，虽然这并不是一个比较标准的 SQL 数据库。对于前端页面，我通过 html 界面不同的表格实现四种功能，同时展示用户查询的内容、允许用户查看其余操作的结果。当然，在这里也可以用 React 框架和 css 实现更加美观的效果，但受限于时间，我仅采用了最简单的 HTML。

3 部署方法

3.1 模型

```
class Student(models.Model):
    name = models.CharField(verbose_name="姓名", max_length=20)
    student_id = models.CharField(verbose_name="学号", max_length=15,unique=True)
    sex_choices = (
        (1, "男"),
        (2, "女"),
    )
    sex = models.SmallIntegerField(verbose_name="性别", choices=sex_choices)
    age = models.IntegerField(verbose_name="年龄")
    major = models.CharField(verbose_name="专业", max_length=50)
```

```
    #password = models.CharField(verbose_name="密码", max_length=50)
    #user = models.OneToOneField(User, on_delete=models.CASCADE)

    class Meta:
        verbose_name = "学生"
        verbose_name_plural = verbose_name
        ordering = ['student_id']

    def __str__(self):
        return self.name
```

3.2 视图和网页

在视图层面，我以函数式编程为指导，通过定义不同函数进行对应的增删查改操作。为实现用户在 HTML 界面输入指定内容以得到预想结果，我利用 POST 和 render 实现前后端数据和信号交互。在这里我以删除学生信息操作为例展示 POST 和 render 的作用。

```
ddef delete_student(request):
    if request.method == 'POST':
        sid = request.POST['student_id']
        my_model = Student.objects.filter(student_id=sid)
        my_model.delete()
    return render(request, 'student.html')
```

此外，我为每一个操作设置了对应的子 url，即是/database/student/下的三级域名。在网页界面，我利用 HTML 标识符进行设计，通过 form 格式让用户填写指定的信息，并通过一个按键传递‘POST’信号，实现前后端数据交换。因此在实际运行中，用户能在主页实施增删查改操作，完成后会跳转到子域名，用户可以通过点击“返回主页”按钮回到主页并查看除查询以外的结果（查询结果直接在新的域名中展示）。