校园课程管理系统

周雨扬 罗润冬

(北京大学信息科学技术学院，北京 100871);

E-mail: {2000013061, rundong\_luo}@stu.pku.edu.cn

**摘要**

本课程设计旨在创建一个校园课程管理系统，涉及数据库、后端和前端技术。主要工作包括设计数据库结构、开发后端API和实现前端界面。特色包括角色权限管理、学生作业上传和评分查看等功能。完成这个项目，我们获得了深入的数据库设计和管理经验，提升了后端开发和前端设计能力，还学会了角色权限管理和团队协作。这份工作让我们全面了解了数据库应用程序开发，对职业发展和项目工作具有重要意义。

# 引言

本项目旨在创建一个校园课程管理系统，提供给超管、教师和学生使用。系统的功能如下：

1. 用户注册：用户可以注册账号，成为系统的一员。

2. 用户和超管登录：注册后的用户可以使用其凭据登录系统，并根据其角色进行权限验证。

3. 超管的课程信息编辑：超管可以查看和编辑所有课程的信息，包括确定课程的老师和学生。

4. 教师用户的课程信息查看和作业编辑：教师用户可以查看自己作为教师的课程信息，包括学生列表和作业列表。他们可以编辑作业和上传课件。

5. 学生用户的课程信息查看和作业上传：学生用户可以查看自己参加的课程信息，包括教师和作业列表。他们可以上传作业，并查看课件。

6. 学生用户的作业查看和评分查看：学生可以查看自己上传的作业，并查看教师给出的评分。

这个系统的目的是为了实现高效的校园课程管理，促进教师和学生之间的交流和合作。通过提供清晰的界面和功能，系统可以帮助超管管理课程，教师进行作业管理和评分，学生方便地获取课程和作业信息，以及提交和查看作业成绩。

完成这个项目后，我们获得了丰富的经验和技能提升。我们深入了解了数据库设计和管理的流程，包括表结构定义、关系建立和数据操作。通过修复后端API以适应前端设计，我们提高了后端开发能力，处理请求和参数、用户验证以及与数据库的交互。在前端方面，我们学会了使用Vue框架构建用户友好的界面，与后端API通信、展示数据和实现交互操作。角色权限管理使我掌握了设计和实现权限系统的能力，保障了数据安全和系统可靠性。此外，项目还培养了我们的团队协作和项目管理技巧，如需求分析、任务分配和进度跟踪。这个项目让我们全面了解了数据库应用程序开发，提升了数据库设计、后端开发、前端开发、权限管理和团队协作能力，对我们的职业发展和实际项目工作带来巨大价值。

本文的组织结构如下：在第二部分中，我们将简要介绍本课程设计需要的背景知识，包括前端框架Vue、后端框架Flask、以及数据库系统MySQL；第三部分描述了课程设计的系统框架；第四部分给出相关的关系数据库模式和相应的操作；第五部分总结了全文，并且描述了完成本课程设计的收获。

# 相关工作

本项目使用MySQL数据库，后端采用Flask框架，前端采用vue框架，前后端交互使用axios库。

MySQL是一种开源关系型数据库管理系统，它使用结构化查询语言（SQL）来管理和操作数据。MySQL被广泛应用于各种规模的应用程序，从小型网站到大型企业级解决方案。它提供了可靠、高性能和可伸缩的数据库解决方案。

Flask是一个轻量级的Python Web框架，用于构建灵活且可扩展的Web应用程序。它提供了简单易用的工具和库，帮助开发人员快速构建功能完整的后端。Flask具有模块化设计和灵活的插件机制，允许开发人员根据需求定制应用程序的各个方面。

Vue是一种流行的JavaScript框架，用于构建用户界面。它通过提供可重用的组件和直观的开发模式，简化了构建交互式前端应用程序的过程。Vue具有响应式数据绑定和组件化的特性，使得开发人员可以更轻松地管理和更新应用程序的状态。

在本项目中，MySQL用于存储和管理数据，提供可靠的数据持久化解决方案。Flask作为后端框架，处理来自前端的请求并与数据库进行交互，执行业务逻辑并返回响应。同时，Flask还负责处理用户认证、授权和路由管理等任务。Vue作为前端框架，负责构建用户界面并与后端进行交互。它可以通过向后端发送HTTP请求获取数据，并将数据动态地展示在网页上。Vue的组件化特性使得开发人员可以将界面拆分成独立的可复用组件，提高代码的可维护性和复用性。

# 课程设计的系统框架

## 后端框架

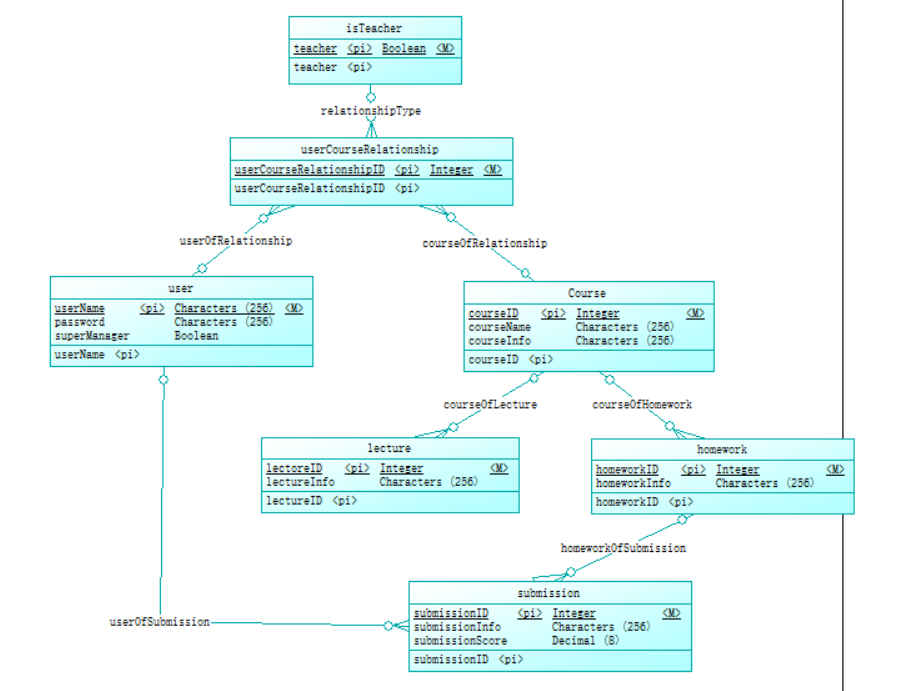
后端的主要功能实现在backend/app.py中。后端开始运行后，会首先建立与数据库之间的交互通道，之后便会监听前端发来的请求并根据请求内容进行相应。通常前端的请求需要访问数据库。我们采用python的flask\_sqlalchemy 库实现后端与数据库间的交互，其可以使用和正常终端交互一样的指令字符串作为输入，并且将数据库的输出以 list 的方式返回到 python 中供后端进行处理。由于我们设计的关系数据库模式有着较好的性质，因此我们只需要利用非常基础的数据库操作便可以实现数据库的修改与查询。

## 前端框架

前端的网页设计部分实现在frontend/course/src/components/中，后端交互实现在frontend/course/src/api.js中，网页路由实现在frontend/course/src/router/router.js中。每一个frontend/course/src/components/中的vue文件对应一个挂载在router中的网页。点击网页中的组件可以执行网页间的跳转。api.js负责实现前后端交互功能，其中的每个函数对应一个请求事件。在收到网页发来的请求（即函数调用）后，api.js将请求打包，并使用get或post方法发送给后端，随后等待后端相应。

# 课程设计对应的关系数据库模式

## ER图

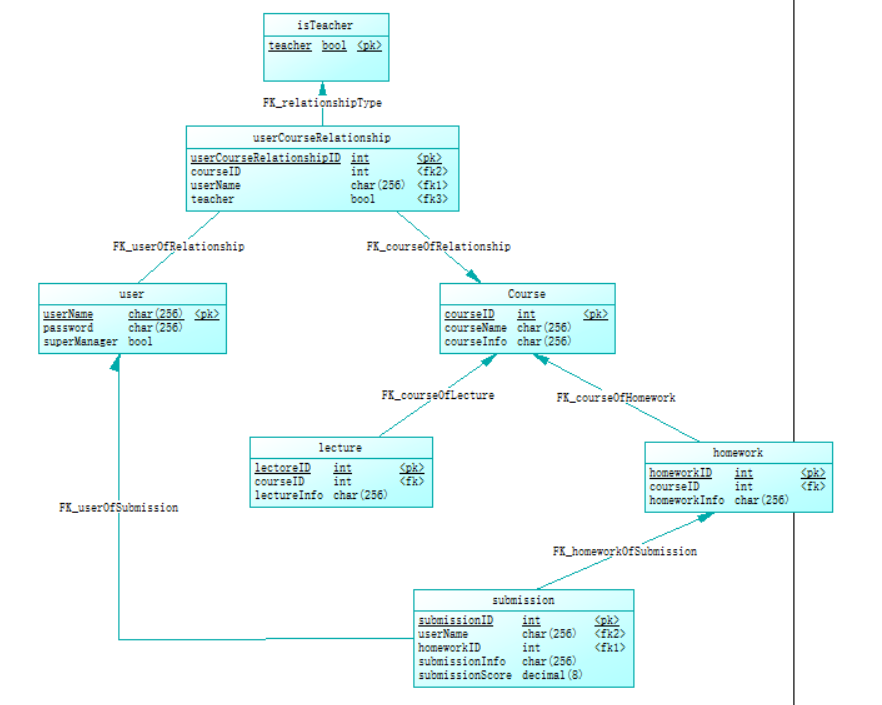
我们利用 powerdesigner 进行的关系数据库模式设计。下图为我们设计的 ER 图。

我们的用户权限总共有三级：学生，教师超级管理员。由于一个用户可能同时是一门课的学生，也是另外一门课的老师(助教，这里我们将其与老师归为一类)，但是不能同时是超级管理员和教师，因此我们选择在user表内记录该用户是超级管理员，还是一般用户（包括老师与学生）。

用户与课程之间存在多对多的关系，且我们需要在该组关系内记录该用户是课程中的学生，还是授课教师。由于我们不希望相同的课程与学生之间存在多个关系，因此我们采用了user, course, teacher 之间的三元关系，采用 userID 与 courseId 的双主码保证关系的唯一性。但与之不同的是，相同的作业，相同的用户可能有多次提交，因此我们便采用了一个独立的 submission 表，采用 submissionID 作为主键，并通过额外的两个键记录其所属的用户与作业。

课程与课时，课程与作业均为简单的一对多关系，不需要特殊处理。

上述 ER 图转换到物理模型后结果如下图：



由于我们在进行前后端交互时候不会对 isTeacher 表进行任何操作，因此我们去除了该表，并且去除 teacher 键值的 foreign key 依赖关系。不难发现删除后的关系模式更为合理。

同时为了方便查询，我们在 submission 中额外记录了该次提交所属的课程的编号 courseID. 由于用户需要从课程中选取作业再进行提交，因此该键值能够在提交请求时非常方便地获得，不会增加获取信息的成本。记录 courseID 同时也可以简化用户界面下获取其在本课程提交的所有作业的信息的过程。

## 关系数据库模式描述

本课程管理系统共有6个表，具体如下：

1. 课程表course，主键为 courseID

| 列名 | 数据类型 | 约束条件 |
| --- | --- | --- |
| courseID | int | 非空，主键 |
| courseName | char(128) | 非空 |
| courseInfo | char(128) |  |

2. 作业表assignment，主键为 (assignmentID, courseID)

| 列名 | 数据类型 | 约束条件 |
| --- | --- | --- |
| assignmentID | int | 非空，主键 |
| courseID | int | 非空，外键参考course表， |
| assignmentInfo | char(128) |  |

3. 讲座表lecture，主键为 (lectureID, courseID)

| 列名 | 数据类型 | 约束条件 |
| --- | --- | --- |
| lectureID | int | 非空，自动递增，也可以手动输入，主键 |
| courseID | int | 非空，外键参考course表，主键 |
| lectureInfo | char(128) |  |

4. 作业提交信息表submission, 主键为submissionID

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 约束条件 |
| submissionID | int | 自动递增，主键 |
| userID | char(128) | 非空，外键参考user表的userID列 |
| assignmentID | int | 非空，外键参考assignment表 |
| courseID | int | 非空，外键参考course表 |
| submissionInfo | char(128) | 非空 |
| submissionScore | decimal(8) |  |

5. 用户表user, 主键为userID注意一名user可以同时为教师和学生

| 列名 | 数据类型 | 约束条件 |
| --- | --- | --- |
| userID | char(128) | 非空，主键 |
| password | char(128) | 非空，主键 |
| superManager | bool | 非空 |

6. 教学信息表userCourseRelationship, 主键为 (courseID, userID)

| 列名 | 数据类型 | 约束条件 |
| --- | --- | --- |
| courseID | int | 非空，主键 |
| userID | char(128) | 非空，主键 |
| teacher | bool | 非空 |

如果不考虑 submission 表中的 courseID 项，则我们的设计符合4NF范式。

## 关系数据库模式之上的操作

由于我们比较良好的模式设计，我们所有的编辑操作，修改操作，查询操作均仅仅涉及到对于单张表的条件操作，且均不涉及到复杂的嵌套查询，递归查询等。这极大地降低了我们在设计SQL表达式时候的难度。

但是与之相对的，4NF范式与部分冗余信息项的增加也增加了后端在检查操作人是否有对应权限时候的成本。我们针对submission 表中的冗余项 courseID，在进行编辑与查询时均进行了一次额外的检查，防止数据库中出现自相矛盾的数据，产生数据不一致的问题。

针对外键的处理，我们统一将其设置为了：不存在对应的主键时候删除表格中的对应项目。因此删除对应的课程便会删除课程下的所有提交，作业，讲座，以及该课程与用户之间的所有关系。不难发现这种处理方式符合场景需求。

# 总结

在这个数据库课程项目中，我们设计并实现了一个校园课程管理系统，包括前端、后端和数据库三个部分。项目采用Vue.js进行前端设计，用Flask框架编写后端代码，同时使用MySQL作为数据库。前端Vue.js主要负责界面的展示，包括课程信息、作业信息、课件等内容。我们根据用户的角色（学生或教师）展示不同的操作界面，满足了用户查看课程信息、作业信息和课件的需求，学生还可以上传作业，助教则可以编辑作业、查看并评分学生提交的作业，以及上传课件。后端Flask主要负责业务逻辑处理，包括用户登录、课程信息的编辑、作业的编辑、作业的提交和评分等功能。我们使用了Flask提供的路由机制，根据用户的请求，返回相应的数据。数据库部分，我们使用MySQL设计了user（用户）表、course（课程）表、assignment（作业）表等。在用户表中，我们记录了用户的角色，是超级管理员还是普通用户。在课程表和作业表中，我们根据用户的角色和课程关系，设计了相应的权限。

这个项目不仅让我们深入理解了数据库、前端和后端的设计和实现，还让我们理解到了在实际项目中如何进行协作，以及如何将理论知识应用到实际问题中。我们学会了如何使用Vue.js进行前端设计，如何使用Flask编写后端代码，以及如何设计和使用MySQL数据库。同时，我们也学会了如何根据用户的角色和需求，设计不同的用户界面和业务逻辑。这个项目让我们收获颇丰，对今后的学习和工作都会有很大的帮助。