教师教学科研登记系统-系统设计与实现报告

来泽远 PB21000164

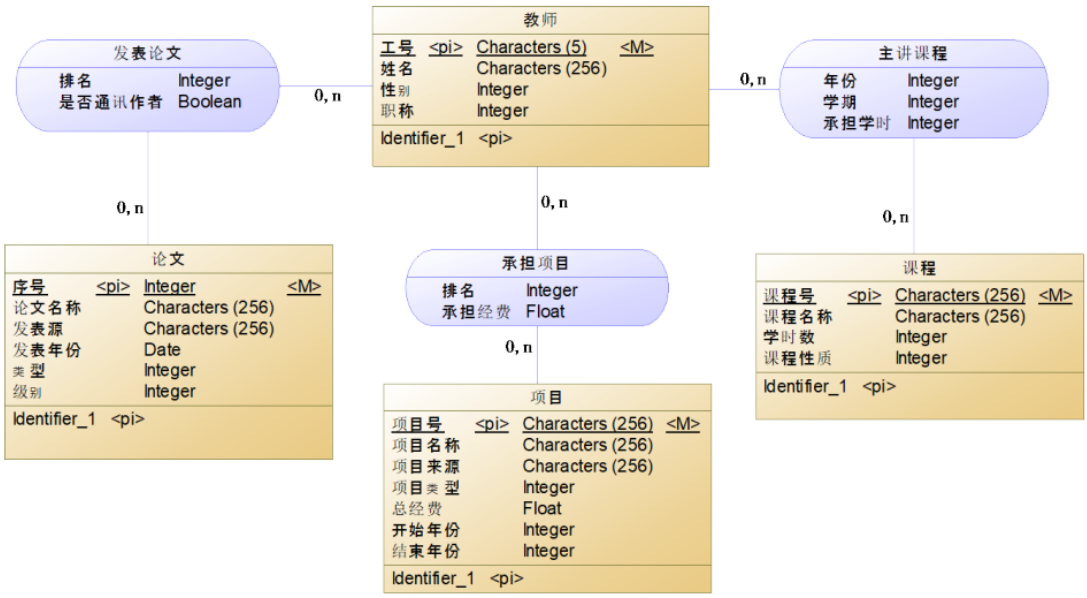
1 需求分析

1.1 系统目标

本系统的开发目标是实现一个教师教学科研登记系统，其中包括教师信息、论文信息、项目信息、教学信息，以及他们之间的关系。旨在正确并方便地添加与管理教师教学科研信息。采用了B/S架构，前端使用基于python的flask以及html，后端使用MySQL。

1.2 需求说明

数据需求：



- **老师**：包含工号、姓名、性别、职称四个属性，其中工号是唯一标识，也即主键。
- **论文**：包含序号、论文名称、发表期刊、发表年份、论文类型、论文级别六个属性，其中序号为主键。
- **项目**：包含项目号、项目名称、项目来源、项目类型、总经费、开始年份、结束年份七个属性，其中项目号为主键。
- **课程**：包含课程号、课程名称、学时数、课程性质四个属性，其中课程号为主键。
- **老师-论文**：为多对多关系，每个老师在其发表的论文中有一个唯一排名，并且可以是唯一的通讯作者。
- **老师-项目**：为多对多关系，每个老师在其参与的项目中有一个唯一排名与承担的经费。
- **老师-课程**：为多对多关系，每个老师会在某一年某一学期开某一节课，并且承担一部分或全部学时。

数据说明

1. 性别为整数，1-男，2-女
2. 教师职称为整数：1-博士后，2-助教，3-讲师，4-副教授，5-特任教授，6-教授， 7-助理研究员，8-特任副研究员，9-副研究员，10-特任研究员，11-研究员。
3. 论文类型为整数：1-full paper，2-short paper，3-poster paper，4-demo paper。
4. 论文级别为整数：1-CCF-A，2-CCF-B，3-CCF-C，4-中文 CCF-A，5-中文 CCF□B，6-无级别。
5. 项目类型为整数：1-国家级项目，2-省部级项目，3-市厅级项目，4-企业合作项 目，5-其它类型项目。
6. 发表论文和承担项目中的排名：1-表示排名第一，以此类推。论文排名第一即为第 一作者，承担项目排名第一即为项目负责人。
7. 主讲课程中的学期取值为：1-春季学期，2-夏季学期，3-秋季学期。
8. 课程性质为整数：1-本科生课程，2-研究生课程。

功能需求：

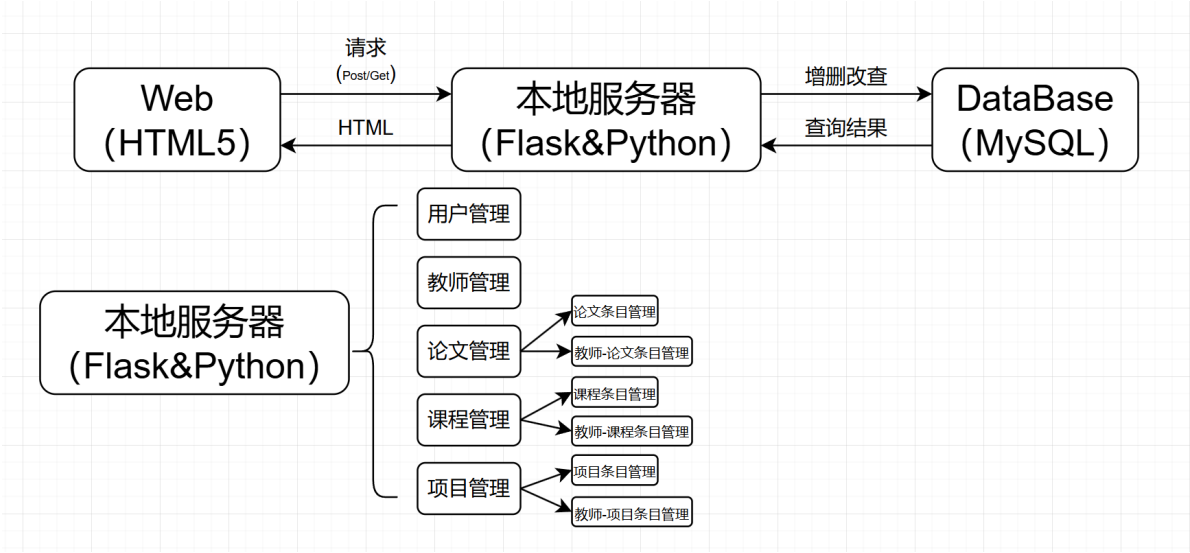
- **登记发表论文情况**：提供教师论文发表信息的的增、删、改、查功能；输入时要求检 查：一篇论文只能有一位通讯作者，论文的作者排名不能有重复，论文的类型和级别只 能在约定的取值集合中选取。
- **登记承担项目情况**：提供教师承担项目信息的增、删、改、查功能；输入时要求检查： 排名不能有重复，一个项目中所有教师的承担经费总额应等于项目的总经费，项目类型 只能在约定的取值集合中选取。
- **登记主讲课程情况**：提供教师主讲课程信息的增、删、改、查功能；输入时要求检查： 一门课程所有教师的主讲学时总额应等于课程的总学时，学期。
- **查询统计**：实现按教师工号和给定年份范围汇总查询该教师的教学科研情况的功能；例如输入 工号“01234”，“2023-2023”可以查询 01234 教师在 2023 年度的教学科研工 作情况。实现按教师工号和给定年份范围生成教学科研工作量统计表并导出文档的 功能，导出文档格式为Markdown。

附加需求：

- 额外实现对于老师、课程、项目、论文的增删改查。即对以上七个表都可以进行增删改查。
- 多样化查询：用户可以选择缺省查询条件已跳过该条件的查询；或者填入多个查询条件，查询同时满足的条目信息。
- 简便查询：用户可以查询某一个课程、项目、论文对应的所有老师，亦可查询一个老师对应的所有的课程、项目、论文，无需精确至某一条目的查询。
- 可扩展性：程序拥有很强的可扩展性，对于课程、论文、项目的实现十分相似，可以轻松扩展出类似的功能。同时程序也保留了增加其他各式请求的空间，包括各种自定义操作，只需要拓展if块即可。同时网页端的html也具有高度相似性，便于扩展。
- 安全性：实现了用户登录与注册界面，管理了用户名与密码。

2 总体设计

2.1系统模块结构



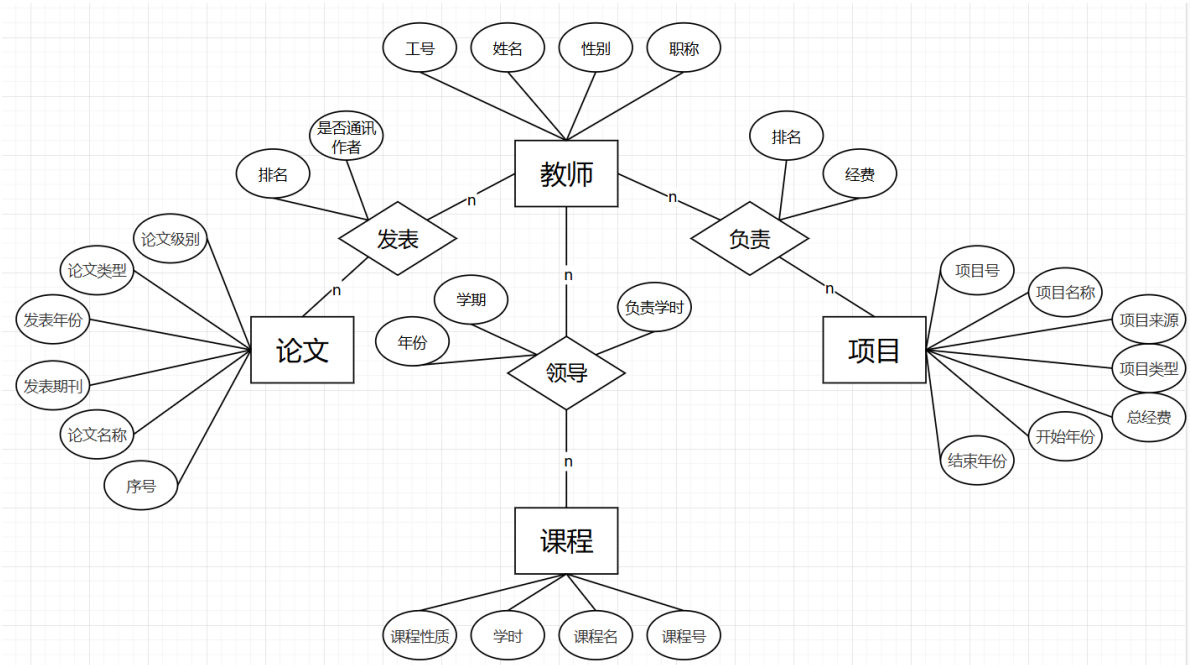
- 前端使用HTML，使用网页与用户进行交互。用户可以在网页端看到查询结果，并且进行对应增删改。
- 服务器使用flask架构，在对应的route下编写函数，响应网页端的对应请求。
    - 用户管理模块，管理用户的登录以及注册请求。
    - 教师管理模块，管理老师条目的增删改查。
    - 论文管理模块，管理论文以及老师-论文条目的增删改查。
    - 项目管理模块，管理项目以及老师-项目条目的增删改查。
    - 课程管理模块，管理课程以及老师-课程条目的增删改查。
- 后端使用MySQL数据库管理数据的存取与修改。

2.2系统工作流程

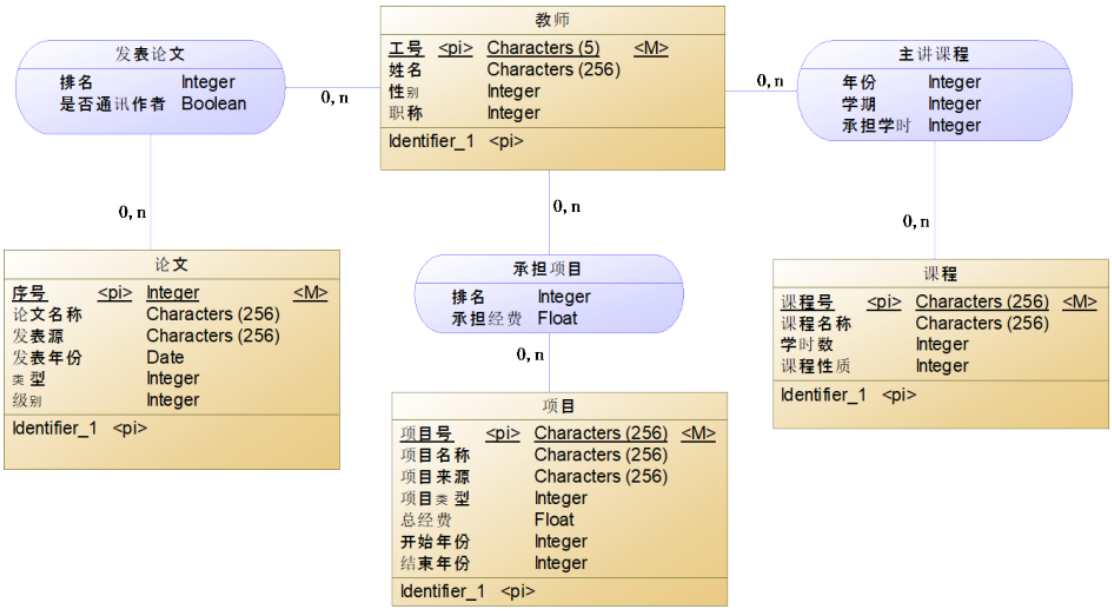
如2.1图中所示。Web端响应用户的请求并将数据（如有）一并发送给服务器。服务器对不同的请求调用不同的模块进行处理。模块内部调用python与MySQL的接口完成对数据库的增删改查。若为查询则数据库会返回对应查询结果，服务器将数据结果内嵌在html中并发送给Web，Web对html进行解析后便可将包含数据的网页呈现给用户。

2.3数据库设计

ER图



CMD



PDM

由于Powerdesigner的license过期，且未找到解决方法，所以没有跑物理模型。

3 详细设计

数据库实现

```
-- Table for user
CREATE TABLE user (
    name VARCHAR(256) PRIMARY KEY,
    password VARCHAR(256) NOT NULL
);

-- Table for Teachers
CREATE TABLE Teachers (
    工号 CHAR(5) PRIMARY KEY,
    姓名 VARCHAR(256) NOT NULL,
    性别 INT CHECK (性别 IN (1, 2)),
    职称 INT CHECK (职称 IN (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11))
);

-- Table for Papers
CREATE TABLE Papers (
    序号 INT PRIMARY KEY,
    论文名称 VARCHAR(256) NOT NULL,
    发表期刊 VARCHAR(256) NOT NULL,
    发表年份 INT NOT NULL,
    论文类型 INT CHECK (论文类型 IN (1, 2, 3, 4)),
    级别 INT CHECK (级别 IN (1, 2, 3, 4, 5, 6))
);

-- Table for Projects
CREATE TABLE Projects (
    项目号 CHAR(5) PRIMARY KEY,
    项目名称 VARCHAR(256) NOT NULL,
```

```
    项目来源 VARCHAR(256) NOT NULL,
    项目类型 INT CHECK (项目类型 IN (1, 2, 3, 4, 5)),
    总经费 FLOAT NOT NULL,
    起始年份 INT NOT NULL,
    结束年份 INT
);

-- Table for Courses
CREATE TABLE Courses (
    课程号 VARCHAR(256) PRIMARY KEY,
    课程名称 VARCHAR(256) NOT NULL,
    学时数 INT NOT NULL,
    课程性质 INT CHECK (课程性质 IN (1, 2))
);

-- Table for Publishing Papers
CREATE TABLE PublishingPapers (
    工号 CHAR(5),
    序号 INT,
    排名 INT CHECK (排名 > 0),
    是否通讯作者 BOOLEAN,
    PRIMARY KEY (工号, 序号),
    FOREIGN KEY (工号) REFERENCES Teachers(工号),
    FOREIGN KEY (序号) REFERENCES Papers(序号)
);

-- Table for Leading Courses
CREATE TABLE LeadingCourses (
    工号 CHAR(5),
    课程号 VARCHAR(256),
    年份 INT,
    学期 INT CHECK (学期 IN (1, 2, 3)),
    承担学时 INT,
    PRIMARY KEY (工号, 课程号, 年份, 学期),
    FOREIGN KEY (工号) REFERENCES Teachers(工号),
    FOREIGN KEY (课程号) REFERENCES Courses(课程号)
);

-- Table for Taking Projects
CREATE TABLE TakingProjects (
    工号 CHAR(5),
    项目号 CHAR(5),
    排名 INT CHECK (排名 > 0),
    承担经费 FLOAT,
    PRIMARY KEY (工号, 项目号),
    FOREIGN KEY (工号) REFERENCES Teachers(工号),
    FOREIGN KEY (项目号) REFERENCES Projects(项目号)
);
```

按照上文中的需求实现即可。

各模块实现

登录

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    global lasttemplate
    lasttemplate = 'login'
    if request.method == 'POST':
        if request.form.get('type') == 'signup':
            name = request.form.get('name')
            password = request.form.get('password')
            new = User(
                name=name,
                password=password,
            )
            db.session.add(new)
            db.session.commit()
            return render_template('login.html')
        elif request.form.get('type') == 'login':
            name = request.form.get('name')
            key = request.form.get('password')
            if not exist('user', name): return render_template('error.html',
message1='缺少条目', message2='用户不存在', message3='请先注册')
            user_result = db.session.query(User).filter_by(name=name).first()
            if user_result.password == key:
                return render_template('index.html')
            else:
                return render_template('error.html', message1='密码错误',
message2='请重新输入')
    return render_template('login.html')
```

用户打开网站时，会重定向到登录页面。登录页面有两者访问类型，分别是注册与登录。注册则向数据库中写入用户名与密码；登录则先验证密码是否正确，如否则跳转到报错信息，正确则正式进入系统。

首页

```
@app.route('/index', methods=['GET', 'POST'])
def index():
    global lasttemplate
    lasttemplate = 'index'
    return render_template('index.html')
```

教师

```
@app.route('/teacher', methods=['GET', 'POST'])
def teacher():
    global lasttemplate
    lasttemplate = 'teacher'
    labels = ['工号', '姓名', '职称', '性别']
    result_query = db.session.query(Teachers)
    if request.method == 'POST':
        if request.form.get('type') == 'query':
            for i in range(len(labels)):
                idx = request.form.get(labels[i])
                if idx != "":
```

```
                    result_query =
result_query.filter(Teachers.__table__.columns[labels[i]] == idx)
        elif request.form.get('type') == 'delete':
            idx = request.form.get('key')
            teacher_result = db.session.query(Teachers).filter_by(工号
=idx).first()
            db.session.delete(teacher_result)
            db.session.commit()
        elif request.form.get('type') == 'update':
            idx = request.form.get('key')
            teacher_result = db.session.query(Teachers).filter_by(工号
=idx).first()
            for i in range(len(labels)):
                update_value = request.form.get(labels[i])
                if update_value != "":
                    teacher_result.__setattr__(labels[i], update_value)
            db.session.commit()
        elif request.form.get('type') == 'insert':
            new = Teachers(
                工号=request.form.get('工号'),
                姓名=request.form.get('姓名'),
                性别=request.form.get('性别'),
                职称=request.form.get('职称')
            )
            db.session.add(new)
            db.session.commit()
    result = display(result_query.all())
    return render_template('teacher.html', labels=labels, content=result)
```

实现四种增删改查功能，后文相同则不再赘述。

- 查询：我们在网页中实现了若干个输入框，每一个对应一个列，即属性的查询。若有输入则去查询，若干次查询是相关联的，取交集。由此实现了多条件筛选。
- 删除：删除是在查询的基础上，对于一个显示的条目可以进行删除。我们从网页获取了对于条目的id，并进行删除。如此也可以保证被删除的条目是一定存在于数据库中的，所以不需要进行检查。
- 更新：逻辑与删除类似，对于缺省值我们的策略是不更新。
- 插入：从网页端获取属性，新建条目并插入。

论文

```
@app.route('/paper', methods=['GET', 'POST'])
def paper():
    global lasttemplate
    lasttemplate = 'paper'
    labels = ['序号', '论文名称', '发表期刊', '发表年份', '论文类型', '级别']
    result_query = db.session.query(Papers)
    labels1 = ['工号', '姓名', '序号', '论文名称', '排名', '是否通讯作者']
    result_query1 = db.session.query(None)
    labels2 = ['工号', '序号', '排名', '是否通讯作者']
    print(f"\n{request.form.get('type')}\n")
    if request.method == 'POST':
        if request.form.get('type') == 'query':
            ...
        elif request.form.get('type') == 'query_1':
            teacher = request.form.get('工号')
            num = request.form.get('序号')
            if teacher != "" or num != "":
```

```
                result_query1 = db.session.query(Teachers.工号, Teachers.姓名,
Papers.序号, Papers.论文名称, PublishingPapers.排名, PublishingPapers.是否通讯作
者).join(PublishingPapers, Teachers.工号 == PublishingPapers.工号).join(Papers,
Papers.序号 == PublishingPapers.序号)
                if teacher != "": result_query1 = result_query1.filter(Teachers.
工号 == teacher)
                if num != "": result_query1 = result_query1.filter(Papers.序号 ==
num)
        elif request.form.get('type') == 'delete_1':
            idx = request.form.get('key')
            paper_result = db.session.query(PublishingPapers).filter_by(工号
=idx).first()
            db.session.delete(paper_result)
            db.session.commit()
        elif request.form.get('type') == 'insert_1' and
request.form.get('choose') == '修改':
            paramlen = request.form.get('cnt')
            num = request.form.get('序号')
            if not exist('paper', num):
                return render_template('error.html', message1='缺少条目',
message2='论文不存在', message3='请先添加论文')
            for i in range(int(paramlen)):
                contact = request.form.get('是否通讯作者'+str(i+1))
                if contact == 'True': contact = True
                if contact == 'False': contact = False
                rank = request.form.get('排名'+str(i+1))
                idx = request.form.get('工号'+str(i+1))
                publish_result = db.session.query(PublishingPapers).filter_by(序
号=num, 工号=idx).first()
                publish_result.__setattr__('排名', rank)
                publish_result.__setattr__('是否通讯作者', contact)
            # check
            publish_result = db.session.query(PublishingPapers).filter_by(序号
=num)
            contact_cnt = 0
            rank_set = {}
            for i in publish_result.all():
                if i.是否通讯作者: contact_cnt += 1
                if contact_cnt > 1:
                    db.session.rollback()
                    return render_template('error.html', message1='数据检查失败',
message2='一篇论文只能有一位通讯作者', message3=f'{i.工号}为重复通讯作者')
                if i.排名 in rank_set:
                    db.session.rollback()
                    return render_template('error.html', message1='数据检查失败',
message2='论文的作者排名不能有重复', message3=f'{i.工号}的排名重复')
                rank_set[i.排名] = True
            db.session.commit()
        elif request.form.get('type') == 'insert_1' and
request.form.get('choose') == '添加':
            paramlen = request.form.get('cnt')
            num = request.form.get('序号')
            if not exist('paper', num):
                return render_template('error.html', message1='缺少条目',
message2='论文不存在', message3='请先添加论文')
            for i in range(int(paramlen)):
                contact = request.form.get('是否通讯作者'+str(i+1))
                if contact == 'True': contact = True
```

```
                if contact == 'False': contact = False
                rank = request.form.get('排名'+str(i+1))
                pub = PublishingPapers(
                    工号 = request.form.get('工号'+str(i+1)),
                    序号 = num,
                    排名 = rank,
                    是否通讯作者 = contact
                )
                db.session.add(pub)
            # check
            publish_result = db.session.query(PublishingPapers).filter_by(序号
=num)
            contact_cnt = 0
            rank_set = {}
            for i in publish_result.all():
                if i.是否通讯作者: contact_cnt += 1
                if contact_cnt > 1:
                    db.session.rollback()
                    return render_template('error.html', message1='数据检查失败',
message2='一篇论文只能有一位通讯作者', message3=f'{i.工号}为重复通讯作者')
                if i.排名 in rank_set:
                    db.session.rollback()
                    return render_template('error.html', message1='数据检查失败',
message2='论文的作者排名不能有重复', message3=f'{i.工号}的排名重复')
                rank_set[i.排名] = True
            db.session.commit()

    result = display(result_query.all())
    result1 = display(result_query1.all())
    return render_template('paper.html', labels=labels, content=result,
labels1=labels1, content1=result1)
```

基本的增删改查与老师是相同的，其中有下标1的方式是对于关系进行的，也即对于论文发表进行增删改查。

- 查询：我们实现了对于一个工号查询所有的发表论文，以及对于一个论文序号查询所有的作者。
- 删除：删除的逻辑与上文相同。
- 更新：为了维护约束，我们实现了批量更新，并且在网页端实现了动态个数的输入框。我们在更改后再进行约束检测，若不正确则回滚。
- 插入：与更新类似，我们实现了批量插入。在插入完成后进行错误检测。

剩余课程与项目的部分与论文部分的实现大同小异，在此不展开。

统计

```
@app.route('/statistic', methods=['GET', 'POST'])
def statistic():
    labels = {}
    content = {}
    if request.method == 'POST':
        if request.form.get('type') == 'query':
            num = request.form.get('工号')
            start = request.form.get('起始年份')
            if start == '': start = 0
            end = request.form.get('结束年份')
            if end == '': end = 9999
            # 查询基本信息
```

```
            labels[0] = ['工号', '姓名', '职称', '性别']
            result_query = db.session.query(Teachers).filter(Teachers.工号 ==
num)
            content[0] = display(result_query.all())
            # 查询课程信息
            labels[1] = ['课程号', '课程名称', '学时数', '课程性质', '年份', '学期',
'承担学时']
            result_query = db.session.query(Teachers.工号, Courses.课程号,
Courses.课程名称, Courses.学时数, Courses.课程性质, LeadingCourses.年份,
LeadingCourses.学期, LeadingCourses.承担学时).join(LeadingCourses, Teachers.工号 ==
LeadingCourses.工号).join(Courses, Courses.课程号 == LeadingCourses.课程
号).filter(Teachers.工号 == num).filter(LeadingCourses.年份 >= start,
LeadingCourses.年份 <= end)
            content[1] = display(result_query.all())
            # 查询论文信息
            labels[2] = ['序号', '论文名称', '发表期刊', '发表年份', '论文类型', '级
别', '排名', '是否通讯作者']
            result_query = db.session.query(Teachers.工号, Papers.序号, Papers.论文
名称, Papers.发表期刊, Papers.发表年份, Papers.论文类型, Papers.级别,
PublishingPapers.排名, PublishingPapers.是否通讯作者).join(PublishingPapers,
Teachers.工号 == PublishingPapers.工号).join(Papers, Papers.序号 ==
PublishingPapers.序号).filter(Teachers.工号 == num).filter(Papers.发表年份 >=
start, Papers.发表年份 <= end)
            content[2] = display(result_query.all())
            # 查询项目信息
            labels[3] = ['项目号', '项目名称', '项目来源', '项目类型', '总经费', '起始年
份', '结束年份', '排名', '承担经费']
            result_query = db.session.query(Teachers.工号, Projects.项目号,
Projects.项目名称, Projects.项目来源, Projects.项目类型, Projects.总经费, Projects.起
始年份, Projects.结束年份, TakingProjects.排名, TakingProjects.承担经
费).join(TakingProjects, Teachers.工号 == TakingProjects.工号).join(Projects,
Projects.项目号 == TakingProjects.项目号).filter(Teachers.工号 == num)
            t1 = result_query.filter(Projects.起始年份 >= start, Projects.起始年份
<= end)
            t2 = result_query.filter(Projects.结束年份 >= start, Projects.结束年份
<= end)
            t3 = result_query.filter(Projects.起始年份 <= start, Projects.结束年份
>= end)
            result_query = t1.union(t2).union(t3)
            content[3] = display(result_query.all())
            output(content, path + '\\teach.md', (start, end))
    return render_template('statistic.html', labels=labels, content=content,
path=path + '\\teach.md')
```

统计模块本质上做的就是查询的功能，只不过多了年份的条件查询。照搬之前的查询范式即可。需要指出的是python与sql的接口不支持逻辑变量的与或非运算，所以我们使用filter和join代替与或运算。其中对于项目查询的解释是：一个项目没有准确的年份，而是从开始到结束。我们的策略是只要该项目的周期与查询的周期有交集，我们便将其计入查询结果。

导出Markdown

```
def output(content, path, year):
    with open(path, 'w') as f:
        year0 = year[0]
        year1 = year[1]
        if year[0] == 0: year0 = ''
        if year[1] == 9999: year1= ''
```

```
        f.write(f'# 教师科学研究统计工作 （{year0}-{year1}）\n')
        f.write('\n')

        f.write('## 教师基本信息\n')
        f.write('\n')
        f.write(f'#### 工号:{content[0][0].工号}\t\t姓名:{content[0][0].姓名}\t\t职称:{content[0][0].职称}\t\t性别:{content[0][0].性别}\n')
        f.write('\n')

        f.write('## 教学情况\n')
        f.write('\n')
        f.write('| 课程号 | 课程名称 | 学时数 | 课程性质 | 年份 | 学期 | 承担学时 |\n')
        f.write('| --- | --- | --- | --- | --- | --- | --- |\n')
        for i in range(len(content[1])):
            f.write(f'| {content[1][i].课程号} | {content[1][i].课程名称} | {content[1][i].学时数} | {content[1][i].课程性质} | {content[1][i].年份} | {content[1][i].学期} | {content[1][i].承担学时} |\n')
        f.write('\n')

        f.write('## 发表论文情况\n')
        f.write('\n')
        for i in range(len(content[2])):
            content[2][i].是否通讯作者 = ', 通讯作者' if content[2][i].是否通讯作者 else ''
            f.write(f'{i+1}. {content[2][i].论文名称}, {content[2][i].发表期刊}, {content[2][i].发表年份}, {content[2][i].论文类型}, {content[2][i].级别}, 排名第{content[2][i].排名}{content[2][i].是否通讯作者}\n')
        f.write('\n')

        f.write('## 承担项目情况\n')
        f.write('\n')
        for i in range(len(content[3])):
            f.write(f'{i+1}. {content[3][i].项目名称}, {content[3][i].项目来源}, {content[3][i].项目类型}, 总经费:{content[3][i].总经费}, {content[3][i].起始年份}-{content[3][i].结束年份}, 排名第{content[3][i].排名}, 承担经费:{content[3][i].承担经费}\n')
    print(f'output style: markdown\noutput path:{path}')
```

我们额外实现了统计部分的结果导出markdown的功能。课程按照表格的形式输出，其余部分的格式保持与实验文档中的示例一致，结果会展示在下一部分。

4 实现与测试

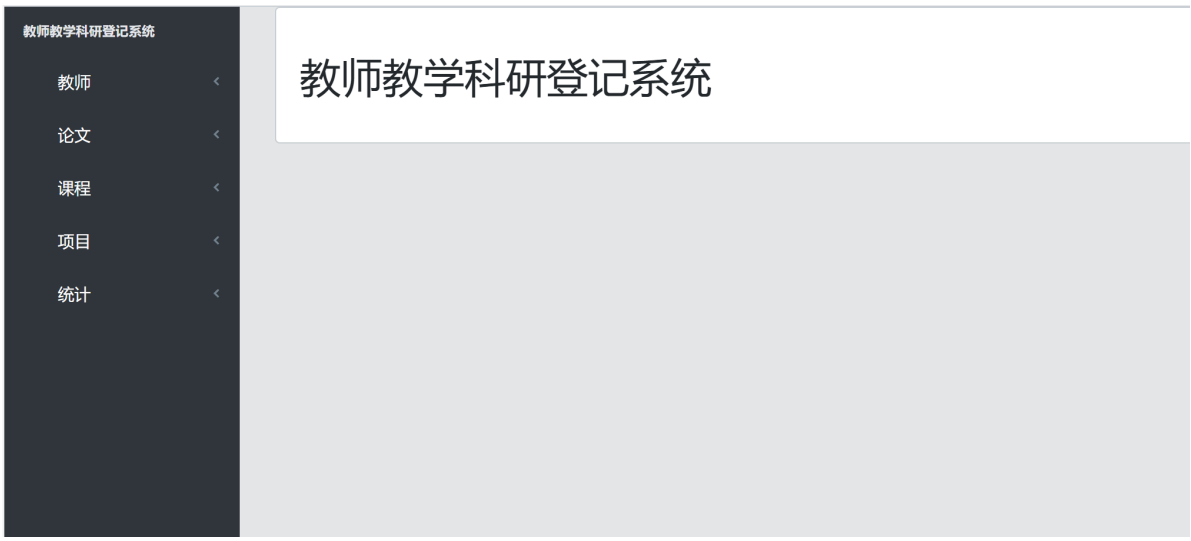
4.1 实现结果

注明：网页端的css文件， 地址如下：https://github.com/hehaha68/USTC_2021Spring_An-Introduction-to-Database-System/tree/master/%E5%AE%9E%E9%AA%8C/lab3/src/static

登录界面


教师教学科研登记系统
登录
教师教学科研登记系统
用户名:
密码:
注册
登录

首页


教师教学科研登记系统
教师
论文
课程
项目
统计
教师教学科研登记系统

教师

查询

工号 | 姓名 | 职称 | 性别 | 查询

教师

工号	姓名	职称	性别	操作
00001	张三	博士后	男	更新 删除
00002	李四	助教	男	更新 删除
00003	王五	讲师	男	更新 删除
00004	赵六	副教授	女	更新 删除

新建

工号 | 姓名 | 职称 | 性别 | 新建

论文

查询

序号 | 论文名称 | 发表期刊 | 发表年份 | 论文类型 | 级别 | 查询

论文

序号	论文名称	发表期刊	发表年份	论文类型	级别	操作
1	基于Transformer的Diffusion Model	ICLR	2019	full paper	CCF-A	更新 删除
2	层剪枝Transformer	AAAI	2020	short paper	CCF-B	更新 删除
3	Video Diffuison Model	NIPS	2021	poster paper	CCF-C	更新 删除

新建

序号 | 论文名称 | 发表期刊 | 发表年份 | 论文类型 | 级别 | 新建

教师-论文

查询论文发表情况

老师工号 | 论文序号 | 查询

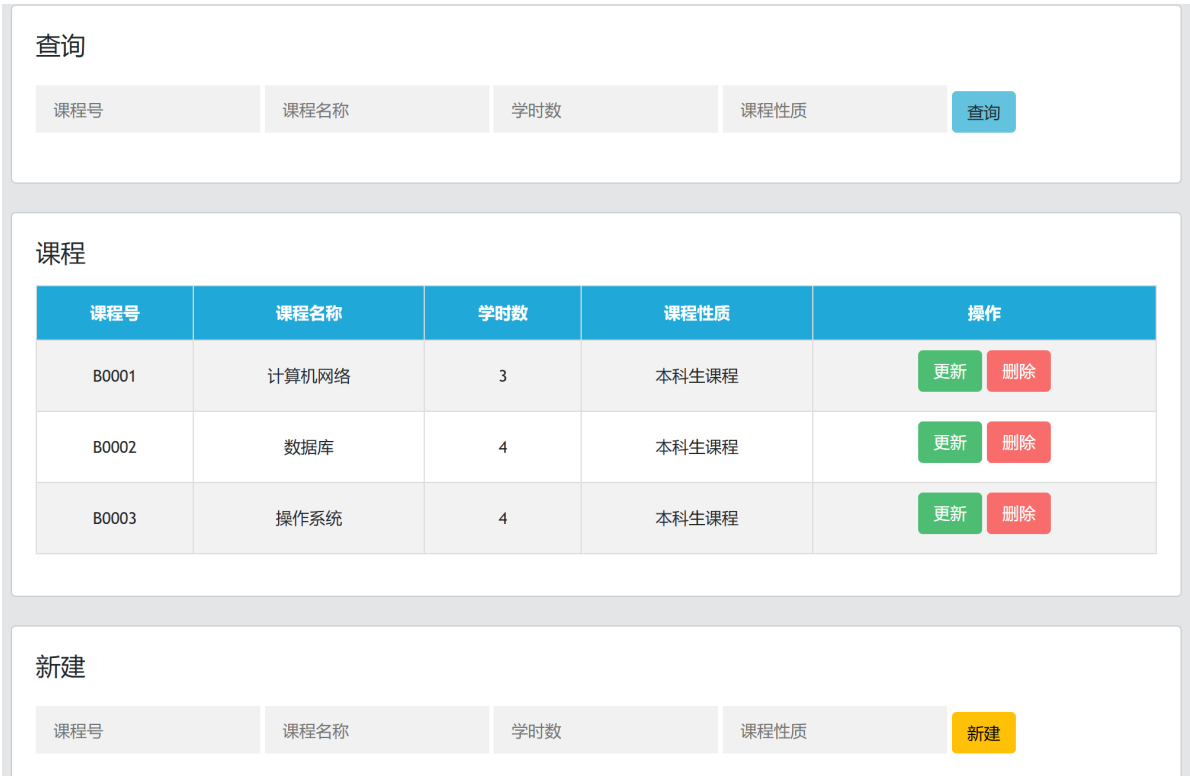
论文发表

工号	姓名	序号	论文名称	排名	是否通讯作者	操作
00002	李四	1	基于Transformer的Diffusion Model	2	False	删除

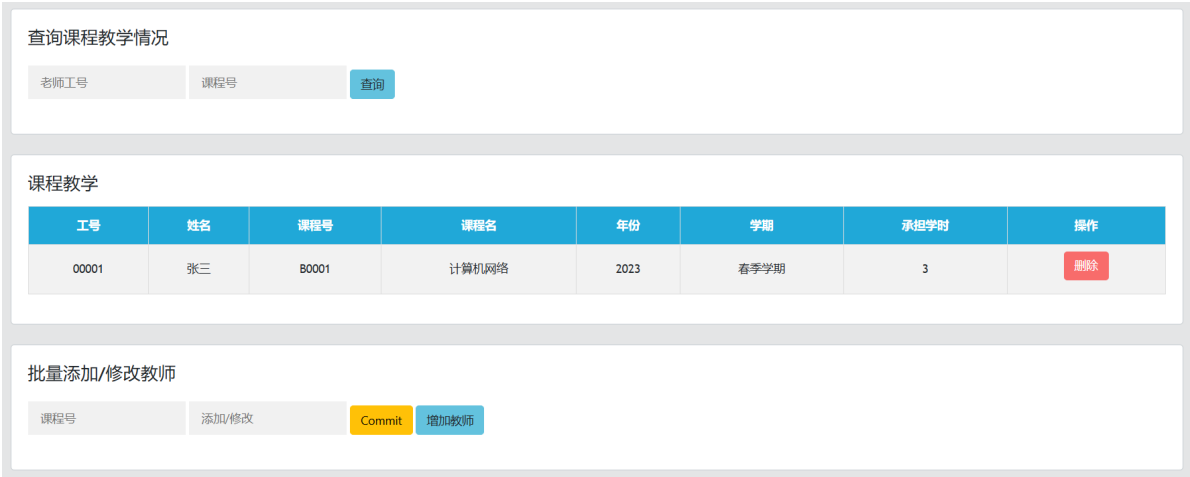
批量添加/修改作者

论文序号 | 添加/修改 | Commit | 增加作者

课程


查询
课程号
课程名称
学时数
课程性质
查询
课程
课程号
课程名称
学时数
课程性质
操作
B0001
计算机网络
3
本科生课程
更新
删除
B0002
数据库
4
本科生课程
更新
删除
B0003
操作系统
4
本科生课程
更新
删除
新建
课程号
课程名称
学时数
课程性质
新建

教师-课程


查询课程教学情况
老师工号
课程号
查询
课程教学
工号
姓名
课程号
课程名
年份
学期
承担学时
操作
00001
张三
B0001
计算机网络
2023
春季学期
3
删除
批量添加/修改教师
课程号
添加/修改
Commit
增加教师

项目


查询
项目号
项目名称
项目来源
项目类型
总经费
起始年份
结束年份
查询
项目
项目号
项目名称
项目来源
项目类型
总经费
起始年份
结束年份
操作
A0001
Mate60研发
华为
国家级项目
100000.0
2018
2020
更新
删除
A0002
小米Su7研发
小米
省部级项目
200000.0
2021
2023
更新
删除
A0003
中科大高新区建设
中科大
市厅级项目
300000.0
2020
2024
更新
删除
新建
项目号
项目名称
项目来源
项目类型
总经费
起始年份
结束年份(可缺省)
新建

教师-项目

查询项目承担情况

老师工号 项目号 查询

项目承担

工号	姓名	项目号	项目名称	排名	承担经费	操作
00001	张三	A0001	Mate60研发	1	50000.0	删除

批量添加/修改参与者

项目号 添加/修改 Commit 增加参与者

统计

查询

00001 起始年份 结束年份 查询

输出格式：markdown

输出路径：d:\git\2024spring-Database\lab3\mine\teach.md

基本信息

工号	姓名	职称	性别
00001	张三	博士后	男

教学情况

课程号	课程名称	学时数	课程性质	年份	学期	承担学时
B0001	计算机网络	3	本科生课程	2023	春季学期	3

发表论文情况

序号	论文名称	发表期刊	发表年份	论文类型	级别	排名	是否通讯作者
1	基于Transformer的Diffusion Model	ICLR	2019	full paper	CCF-A	1	True

项目承担情况

项目号	项目名称	项目来源	项目类型	总经费	起始年份	结束年份	排名	承担经费
A0001	Mate60研发	华为	国家级项目	100000.0	2018	2020	1	50000.0

4.2 测试结果

登录

输入不存在的用户名：

用户名： 2

密码： 2

注册 登录

报错

缺少条目

用户不存在

请先注册

返回

注册：

密码错误:

正确登录:

教师

教师

工号	姓名	职称	性别	操作
00001	张三	博士后	男	更新 删除
00002	李四	助教	男	更新 删除
00003	王五	讲师	男	更新 删除
00004	赵六	副教授	女	更新 删除

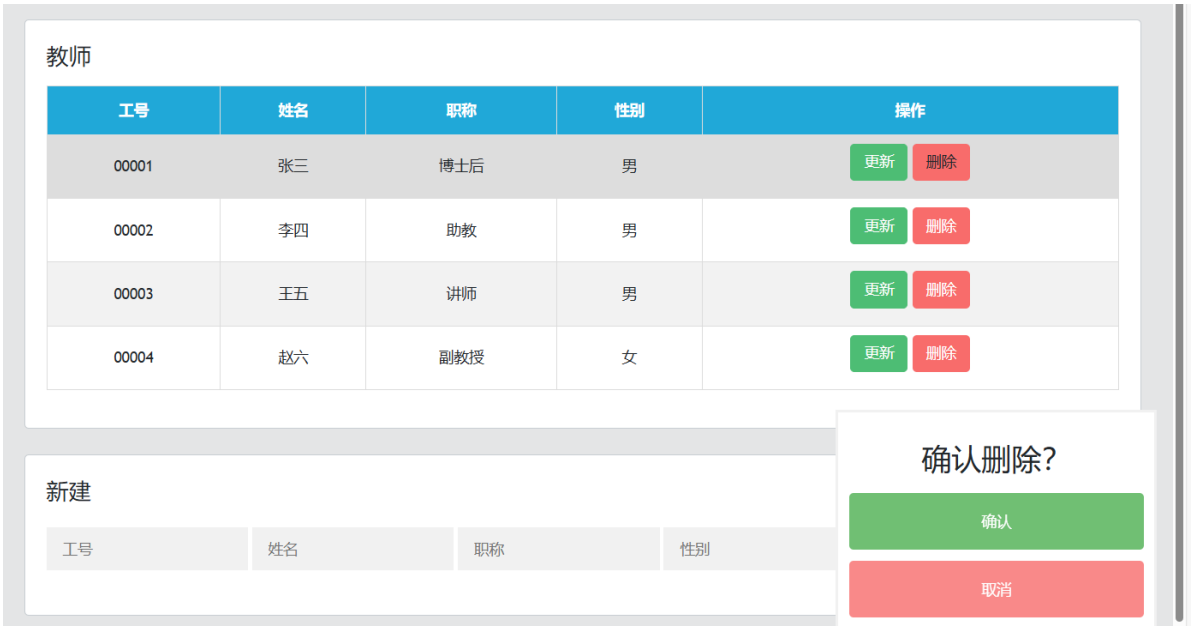
查询工号:

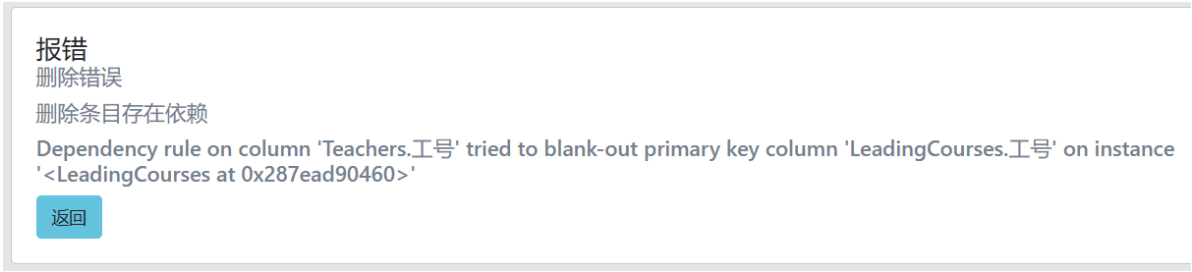
查询职称：

查询性别：

多条件查询：

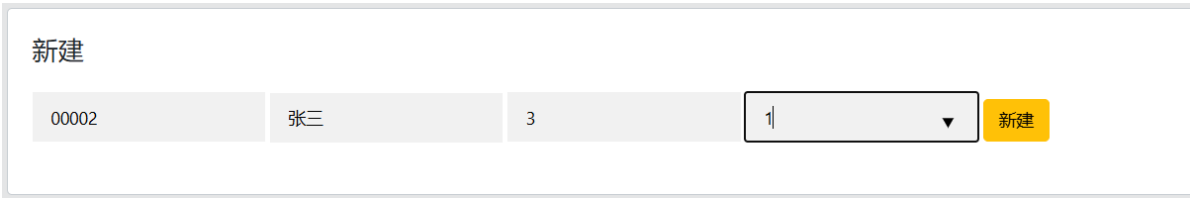
删除：

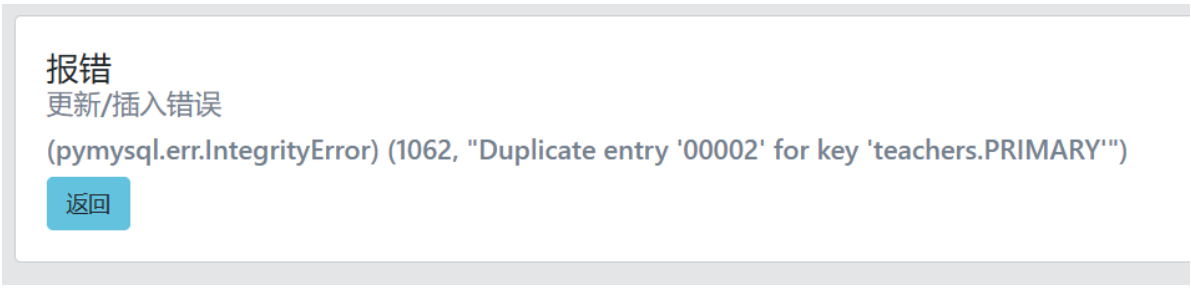




由于其他表中存在外键依赖无法删除。清空其他表后删除：

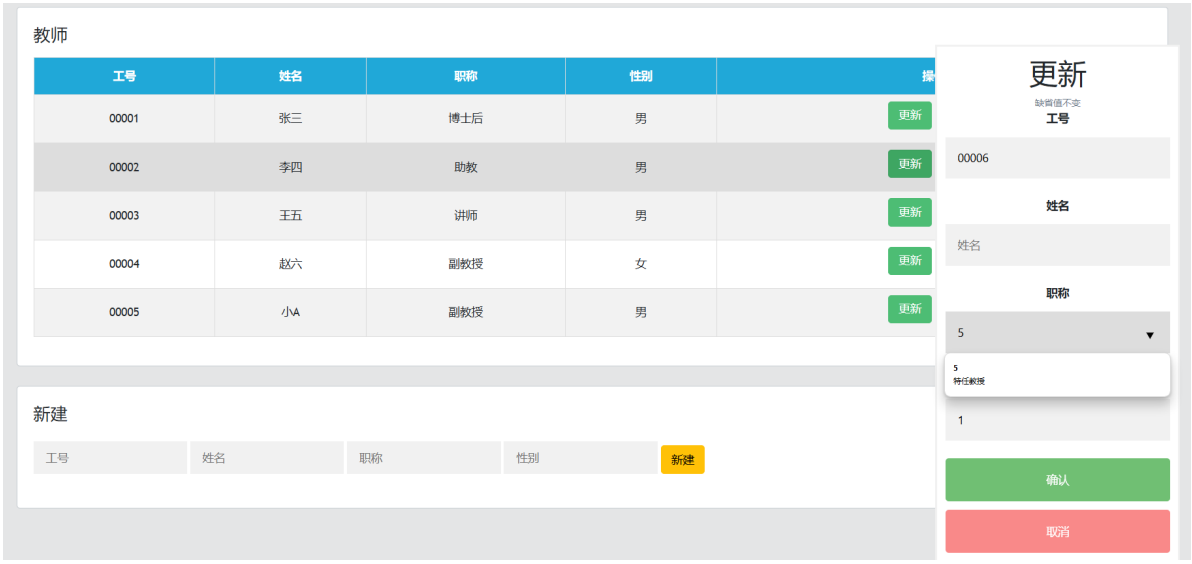
插入重复主键：





正确插入：

对李四进行更新：



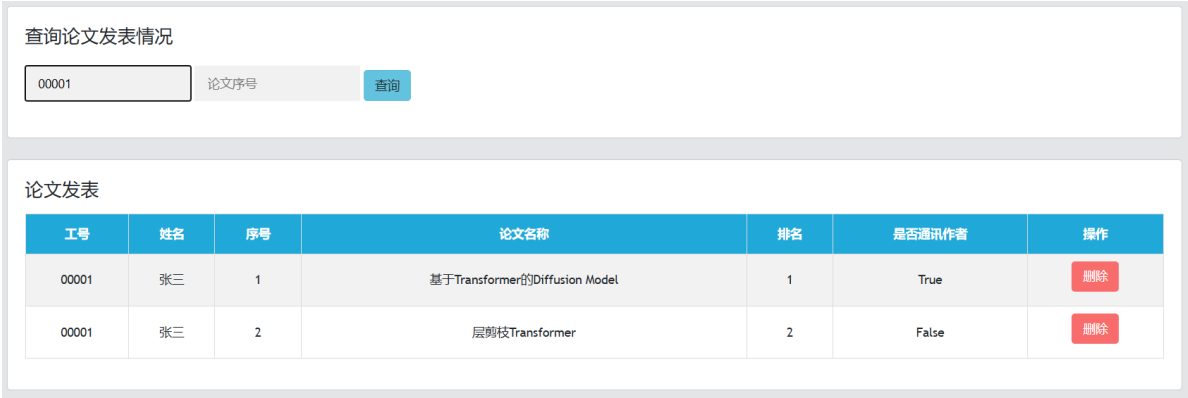
其中缺省值不变。若主键冲突则正常报错。

对于论文、课程、项目三表基本类似，不再演示。

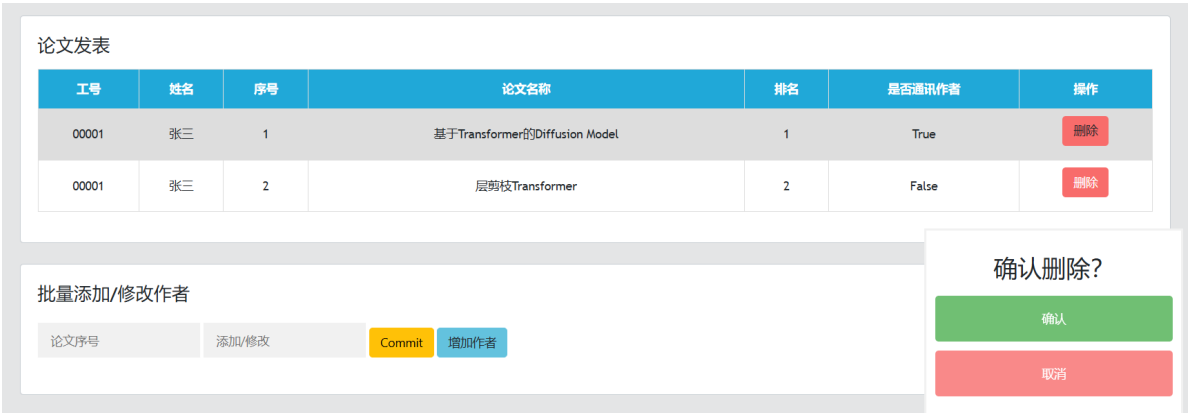
论文-教师

以论文号查询：

以老师工号查询：



删除：



正确删除了张三对于论文1的发表。

插入：

点击两次“增加作者”，网页端增加两个输入框

排名重复：

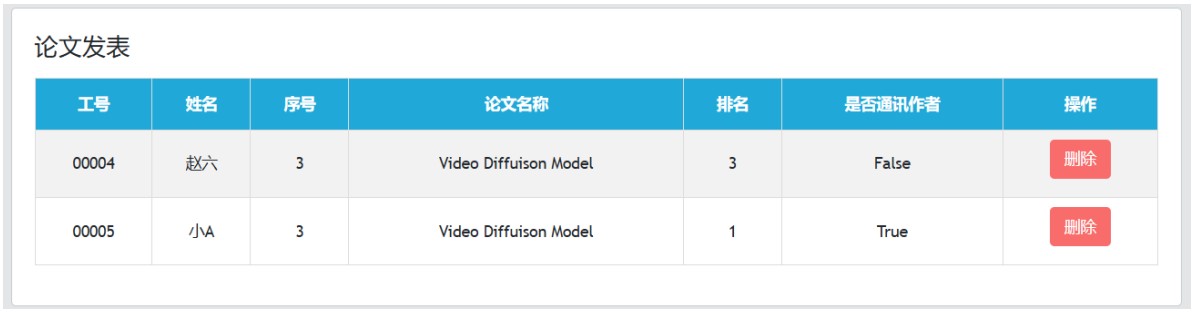
通讯作者重复：

正确插入：

再插入违法数据，正常报错：

对赵六论文3的排名进行更改：

正确更新了排名。



论文发表

工号	姓名	序号	论文名称	排名	是否通讯作者	操作
00004	赵六	3	Video Diffuison Model	3	False	删除
00005	小A	3	Video Diffuison Model	1	True	删除

教师-课程

教师-课程功与前相似，其他功能不再展示，仅展示在此演示文档中要求的约束合法性的判断。

操作系统课程4学分，目前为一名老师担任全部。

查询课程教学情况

老师工号 B0003 查询

课程教学

工号	姓名	课程号	课程名	年份	学期	承担学时	操作
00003	王五	B0003	操作系统	2023	春季学期	4	删除

我们插入一位教师再承担一学时，程序报错。

我们先让他承担0学时。

再进行修改：

教师-项目

排名在论文中已经展示对应功能，经费与承担学时的逻辑一致，在此也略过。

4.3 实现中的难点问题及解决

- html和python的接口对应问题：通过控制变量调试以及打印变量相关信息解决。
- python与数据库接口对应问题：查询函数文档，上网寻找相关例子学习。
- 若干类型错误：html上get的类型一致为string，需要进行类型转换。
- 数据库的连接与相关参数配置：参考网络解决方法。
- 测试不方便，无法批量测试：无解，手动慢慢测试。

5 总结与讨论

我在大一时曾参与过使用flask架构的前后端程序实现，在本项目中我也选择以此为架构进行实现。整个项目的工程量很大，我从中学习到许多经验与技能：

- 模块化实现与模块化调试，避免bug的耦合与堆积。
- 善用网络与文档，很多接口的用法不需要摸索，有标准示例。
- 进一步学习了flask框架开发与前后端协同。
- 强化了系统性思考，对于用户输入错误的解决方法与编程。
- 对于数据库应用有了更深的理解与实现能力。