“教师教学科研登记系统”

系统设计与实现报告

姓名：蔡孟辛

学号：PB22111695

计算机科学与技术学院

中国科学技术大学

2025 年 5 月

目 录

1 需求分析

1.1 系统目标

本系统主要目标为开发一个面向教师的教学科研登记系统。采用 Python 语言，后台 DBMS 使用 MySQL。

1.2 需求说明

1.2.1 数据需求

教师：多个教师。教师信息包含工号、姓名、性别、职称。

论文：教师发表论文。论文信息包含序号、论文名称、发表源、发表年份、类型、级别。多个教师发表同一篇论文需要记录每个教师的排名，以及该教师是否为通讯作者（一篇论文只有一个通讯作者）。

项目：教师承担项目。项目信息包括项目号、项目名称、项目来源、项目类型、总经费、开始年份、结束年份。多个教师承担同一个项目需要记录教师的排名和承担经费。教师的排名不可重复，项目总经费等于每个教师的承担经费总额。

课程：教师主讲课程。课程信息包括课程号、课程名称、学时数、课程性质。多个教师主讲同一个课程时需记录该教师的主讲年份、学期和承担学时。

1.2.2 功能需求

教师管理：提供教师信息的增、删、改、查功能。

登记发表论文情况：提供教师论文发表信息的增、删、改、查功能；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；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；一门课程所有教师

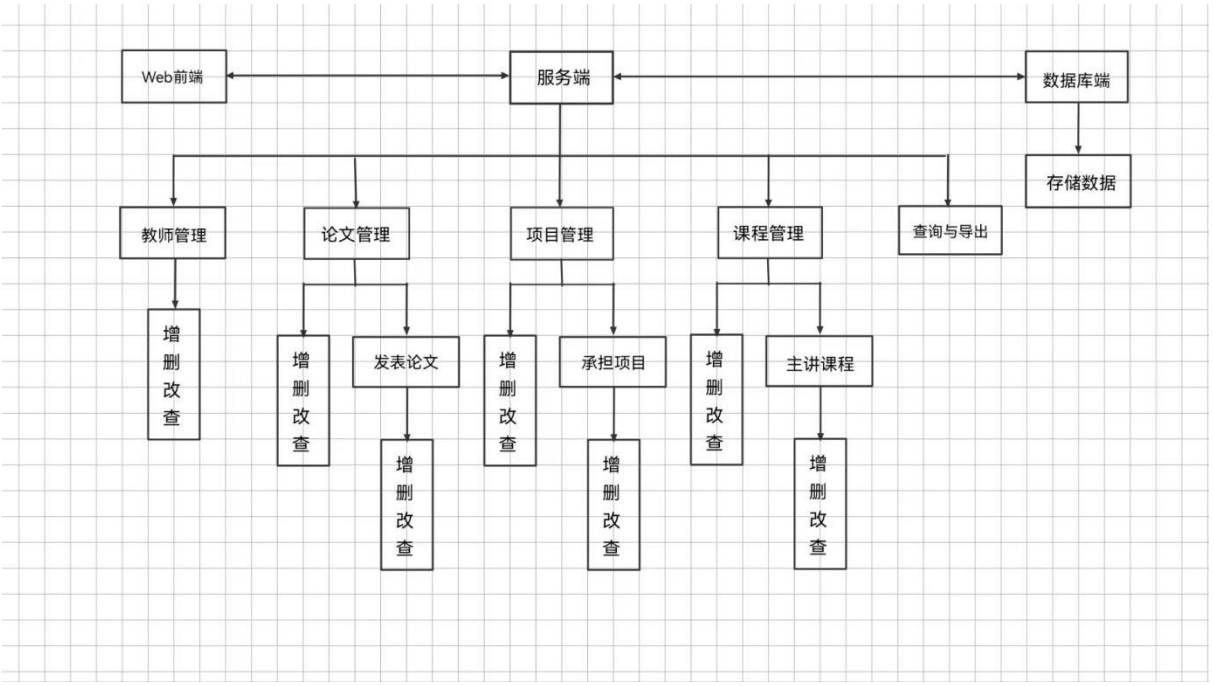
的主讲学时总额应等于课程的总学时，学期。

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。

文档导出：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，导出文档格式是 PDF。

2 总体设计

2.1 系统模块结构



- Web 前端部分实现用户交互界面，提供相关操作接口；
- 服务端处理前端的操作请求，与数据库进行交互，实现下述子模块的功能：

  教师管理模块：实现对教师信息的增删改查；

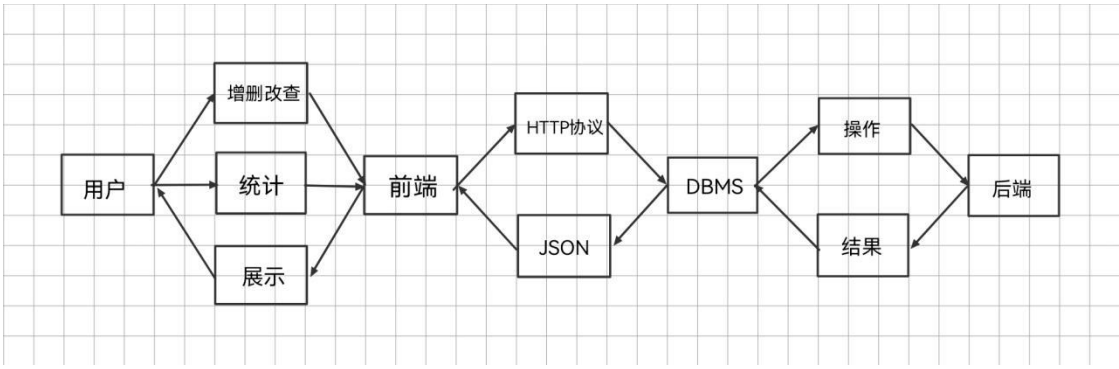
  论文管理模块：实现对论文信息的增删改查，和对发表论文的增删改查；

  项目管理模块：实现对项目信息的增删改查，和对承担项目信息的增删改查；

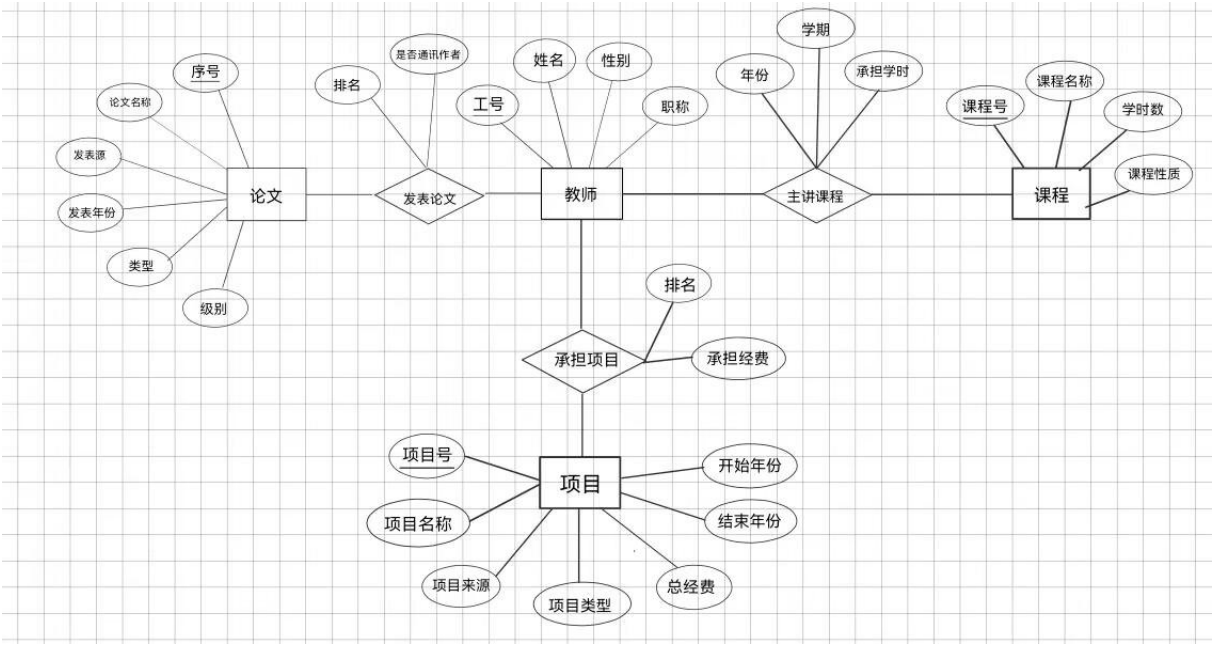
  课程管理模块：实现对课程信息的增删改查，和对主讲课程信息的增删改查；

  查询与导出模块：实现对信息的查询，以及导出 pdf 的功能
- 数据库模块提供了上述信息的存储结构，并且通过存储过程，实现在数据库部分的信息统计功能。

2.2 系统工作流程



2.3 数据库设计



MySQL 数据库建立如下：

• 教师（工号，姓名，性别，职称）

```
create table Teacher (
   T_ID     varchar(5) not null,
   T_Name   varchar(256) default null,
   T_sexual integer default null,
   T_type   integer default null,
   primary key (T_ID)
);
```

• 论文（序号，论文名称，发表源，发表年份，类型，级别）

```
create table Paper (
   P_ID     integer not null,
   P_Name   varchar(256) default null,
   P_Url    varchar(256) default null,
   P_Year   date default null,
   P_Type   integer default null,
   P_Level  integer default null,
   primary key (P_ID)
);
```

• 项目（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）

```
create table Project (
   Pr_ID       varchar(256) not null,
   Pr_Name     varchar(256) default null,
   Pr_Source   varchar(256) default null,
   Pr_Type     integer default null,
   Pr_Summoney float default null,
   Pr_From     integer default null,
   Pr_End      integer default null,
   primary key(Pr_ID)
);
```

• 课程（课程号，课程名称，学时数，课程性质）

```
create table Class (
   C_ID     varchar(256) not null,
   C_Name   varchar(256) default null,
   C_Sum    integer default null,
   C_Type   integer default null,
   primary key (C_ID)
);
```

• 发表论文（工号，序号，排名，是否通讯作者）

```
create table Publish_Paper
(
   P_ID        integer not null,
   T_ID        varchar(5)                    not null,
   P_Rank      integer default null,
   P_Contact   smallint default null,
   primary key (P_ID, T_ID)
);
```

```
alter table Publish_Paper
   add constraint FK_PUBLISH__PUBLISH_P_PAPER foreign key (P_ID)
      references Paper (P_ID)
      on update restrict
      on delete restrict;

alter table Publish_Paper
   add constraint FK_PUBLISH__PUBLISH_P_TEACHER foreign key (T_ID)
      references Teacher (T_ID)
      on update restrict
      on delete restrict;
```

• 承担项目（工号，项目号，排名，承担经费）

```
create table Own_Project
(
   T_ID     varchar(5) not null,
   Pr_ID    varchar(256) not null,
   Pr_Rank  integer default null,
   Pr_money float default null,
   primary key(T_ID, Pr_ID)
);
alter table Own_Project
   add constraint FK_OWN_PROJ_OWN_PROJE_TEACHER foreign key (T_ID)
      references Teacher (T_ID)
      on update restrict
      on delete restrict;

alter table Own_Project
   add constraint FK_OWN_PROJ_OWN_PROJE_PROJECT foreign key (Pr_ID)
      references Project (Pr_ID)
      on update restrict
      on delete restrict;
```

• 主讲课程（工号，课程号，年份，学期，承担学时）

```
create table Teach_Class
(
   T_ID        varchar(5) not null,
   C_ID        varchar(256) not null,
   C_Year      integer default null,
   C_Semester  integer default null,
   C_hours     integer default null,
   primary key (T_ID, C_ID)
```

```
);
alter table Teach_Class
   add constraint FK_TEACH_C_TEACH_CL_TEACHER foreign key (T_ID)
      references Teacher (T_ID)
      on update restrict
      on delete restrict;

alter table Teach_Class
   add constraint FK_TEACH_C_TEACH_CL_CLASS foreign key (C_ID)
      references Class (C_ID)
      on update restrict
      on delete restrict;
```

3 详细设计

3.1 教师模块

目标：实现教师信息的增删改查，如果教师存在相关论文，项目，课程，则拒接删除。允许将数据库中的数字数据显示为文字。

URL：http://127.0.0.1:8000/teacher

代码部分：

model.py：

```
# 教师
class Teacher(db.Model):
    __tablename__ = 'teacher'
    T_ID = db.Column(db.String(5), primary_key=True)
    T_Name = db.Column(db.String(256))
    T_sexual = db.Column(db.Integer)
    T_type = db.Column(db.Integer)
```

app.py：

```
@app.route('/teacher', methods=['GET', 'POST'])
def teacher():
    labels = ['工号', '姓名', '性别', '职称']
    result_query = db.session.query(Teacher)
    result = result_query.all()

    # 定义性别映射关系
    gender_mapping = {
        1: '男',
        2: '女'
    }
    # 定义职称映射关系
    Type_mapping = {
        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'
    }
    # 定义一个函数来转换性别数字为文字
    def format_teacher_data(teachers):
        formatted = []
        for t in teachers:
            gender = gender_mapping.get(t.T_sexual, '未知')
            Type = Type_mapping.get(t.T_type, '未知')
            formatted.append({
                'T_ID': t.T_ID,
                'T_Name': t.T_Name,
                'T_sexual': gender,
                'T_type': Type
            })
        return formatted
    formatted_result = format_teacher_data(result)
    if request.method == 'GET':
        return render_template('teacher.html', labels=labels,
content=formatted_result)
    else:
        if request.form.get('type') == 'query':
            teacher_id = request.form.get('id')
            teacher_name = request.form.get('name')
            teacher_sexual = request.form.get('sexual')
            teacher_type = request.form.get('t_type')
            if teacher_id != "":
                result_query = result_query.filter(Teacher.T_ID == teacher_id)
            if teacher_name != "":
                result_query = result_query.filter(Teacher.T_Name ==
teacher_name)
            if teacher_sexual != "":
                result_query = result_query.filter(Teacher.T_sexual ==
teacher_sexual)
            if teacher_type != "":
                result_query = result_query.filter(Teacher.T_type ==
teacher_type)
            result = result_query.all()
            formatted_result = format_teacher_data(result)
            return render_template('teacher.html', labels=labels,
content=formatted_result)
        elif request.form.get('type') == 'update':
            old_num = request.form.get('key')
            teacher_name = request.form.get('teacher_name')
            teacher_sexual = request.form.get('teacher_sexual')
```

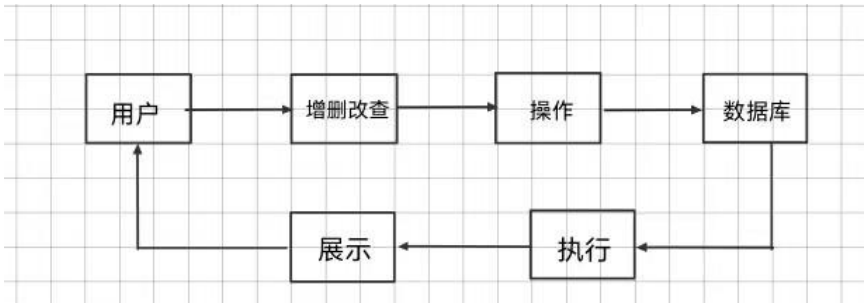
```
            teacher_type = request.form.get('teacher_type')
            teacher_result =
db.session.query(Teacher).filter_by(T_ID=old_num).first()
            teacher_result.T_Name = teacher_name
            teacher_result.T_sexual = teacher_sexual
            teacher_result.T_type = teacher_type
            db.session.commit()
        elif request.form.get('type') == 'delete':
            old_num = request.form.get('key')
            teacherNotExist =
db.session.query(PublishPaper).filter_by(T_ID=old_num).scalar() is None
            if teacherNotExist != 1:
                error_title = '删除错误'
                error_message = '教师在存在关联论文'
                return render_template('404.html', error_title=error_title,
error_message=error_message)
            teacherNotExist =
db.session.query(OwnProject).filter_by(T_ID=old_num).scalar() is None
            if teacherNotExist != 1:
                error_title = '删除错误'
                error_message = '教师在存在关联项目'
                return render_template('404.html', error_title=error_title,
error_message=error_message)
            teacherNotExist =
db.session.query(TeachClass).filter_by(T_ID=old_num).scalar() is None
            if teacherNotExist != 1:
                error_title = '删除错误'
                error_message = '教师在存在主讲课程'
                return render_template('404.html', error_title=error_title,
error_message=error_message)
            teacher_result =
db.session.query(Teacher).filter_by(T_ID=old_num).first()
            db.session.delete(teacher_result)
            db.session.commit()
        elif request.form.get('type') == 'insert':
            teacher_id = request.form.get('id')
            teacher_name = request.form.get('name')
            teacher_sexual = request.form.get('sexual')
            teacher_type = request.form.get('t_type')
            newteacher = Teacher(
                T_ID=teacher_id,
                T_Name=teacher_name,
                T_sexual=teacher_sexual,
                T_type=teacher_type
```

```
            )
            db.session.add(newteacher)
            db.session.commit()
    result = db.session.query(Teacher).all()
    formatted_result = format_teacher_data(result)
    return render_template('teacher.html', labels=labels,
content=formatted_result)
```

输入：工号，姓名，性别，职称

输出：增删改查结果

程序流程图：



3.2 论文模块

目的：提供教师论文发表信息的增、删、改、查功能；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。

URL：http://127.0.0.1:8000/paper

代码部分：

model.py

```
class Paper(db.Model):
    __tablename__ = 'paper'

    P_ID = db.Column(db.Integer, primary_key=True)
    P_Name = db.Column(db.String(256))
    P_Url = db.Column(db.String(256))
    P_Year = db.Column(db.Date)
    P_Type = db.Column(db.Integer)
    P_Level = db.Column(db.Integer)
class PublishPaper(db.Model):
    __tablename__ = 'publish_paper'
```

```
    P_ID = db.Column(db.ForeignKey('paper.P_ID', ondelete='RESTRICT',
onupdate='RESTRICT'), primary_key=True, nullable=False)
    T_ID = db.Column(db.ForeignKey('teacher.T_ID', ondelete='RESTRICT',
onupdate='RESTRICT'), primary_key=True, nullable=False)
    P_Rank = db.Column(db.Integer)
    P_Contact = db.Column(db.SmallInteger)
```

app.py

```
@app.route('/paper', methods=['GET', 'POST'])
def paper():
    labels1 = ['序号', '论文名称', '发表源', '发表年份', '类型', '级别']
    labels2 = ['序号', '作者 ID', '排名', '是否通讯作者']
    result_query1 = db.session.query(Paper)
    result_query2 = db.session.query(Paper, PublishPaper).filter(Paper.P_ID ==
PublishPaper.P_ID)
    result1 = result_query1.all()
    result2 = result_query2.all()

    # 定义论文类型映射关系
    Type_mapping = {
        1: 'full paper',
        2: 'short paper',
        3: 'poster paper',
        4: 'semo paper'
    }
    # 定义论文级别映射关系
    Level_mapping = {
        1: 'CCF-A',
        2: 'CCF-B',
        3: 'CCF-C',
        4: '中文 CCF-A',
        5: '中文 CCF-B',
        6: '无级别'
    }
    # 定义是否为通讯作者映射关系
    contact_mapping = {
        0: '否',
        1: '是'
    }
    def format_paper_data(papers):
        formatted = []
        for p in papers:
            Type = Type_mapping.get(p.P_Type, '未知')
            Level = Level_mapping.get(p.P_Level, '未知')
            formatted.append({
```

```
                'P_ID': p.P_ID,
                'P_Name': p.P_Name,
                'P_Url': p.P_Url,
                'P_Year':p.P_Year,
                'P_Type':Type,
                'P_Level':Level
            })
        return formatted

    def format_publishpaper_data(publishpapers):
        formatted = []
        for p in publishpapers:
            contact = contact_mapping.get(p.PublishPaper.P_Contact, '未知')
            formatted.append({
                'P_ID': p.PublishPaper.P_ID,
                'T_ID': p.PublishPaper.T_ID,
                'P_Rank': p.PublishPaper.P_Rank,
                'P_Contact':contact
            })
        return formatted
    formatted_result1 = format_paper_data(result1)
    formatted_result2 = format_publishpaper_data(result2)
    if request.method == 'GET':
        return render_template('paper.html', labels1=labels1, labels2=labels2,
content1=formatted_result1, content2=formatted_result2)
    else:
        if request.form.get('type') == 'query1':
            paperId = request.form.get('paperId')
            paperName = request.form.get('name')
            paperUrl = request.form.get('url')
            paperYear = request.form.get('year')
            paperType = request.form.get('p_type')
            paperLevel = request.form.get('level')
            if paperId != '':
                result_query1 = result_query1.filter(Paper.P_ID == paperId)
            if paperName != '':
                result_query1 = result_query1.filter(Paper.P_Name == paperName)
            if paperUrl != '':
                result_query1 = result_query1.filter(Paper.P_Url == paperUrl)
            if paperYear:
                result_query1 = result_query1.filter(Paper.P_Year == paperYear)
            if paperType:
                result_query1 = result_query1.filter(Paper.P_Type == paperType)
            if paperLevel:
```

```
                result_query1 = result_query1.filter(Paper.P_Level == paperLevel)
            result1 = result_query1.all()
            formatted_result1 = format_paper_data(result1)
            formatted_result2 = format_publishpaper_data(result2)
            return render_template('paper.html', labels1=labels1,
labels2=labels2, content1=formatted_result1, content2=formatted_result2)
        elif request.form.get('type') == 'query2':
            paperId = request.form.get('paperId')
            teacherId = request.form.get('teacherId')
            rank = request.form.get('rank')
            is_contact = request.form.get('is_contact')
            if paperId != '':
                result_query2 = result_query2.filter(PublishPaper.P_ID==
paperId)
            if teacherId != '':
                result_query2 = result_query2.filter(PublishPaper.T_ID ==
teacherId)
            if rank:
                result_query2 = result_query2.filter(PublishPaper.P_Rank == rank)
            if is_contact:
                result_query2 = result_query2.filter(PublishPaper.P_Contact ==
is_contact)
            result2 = result_query2.all()
            formatted_result1 = format_paper_data(result1)
            formatted_result2 = format_publishpaper_data(result2)
            return render_template('paper.html', labels1=labels1,
labels2=labels2, content1=formatted_result1, content2=formatted_result2)
        elif request.form.get('type') == 'update1':
            paperId = request.form.get('key')
            paperName = request.form.get('name')
            paperUrl = request.form.get('url')
            paperYear = request.form.get('year')
            paperType = request.form.get('p_type')
            paperLevel = request.form.get('level')
            Paper_result =
db.session.query(Paper).filter_by(P_ID=paperId).first()
            Paper_result.P_Name = paperName
            Paper_result.P_Url = paperUrl
            Paper_result.P_Year = paperYear
            Paper_result.P_Type = paperType
            Paper_result.P_Level = paperLevel
            db.session.commit()
        elif request.form.get('type') == 'update2':
            paperId = request.form.get('key')
```

```
            teacherId = request.form.get('teacherId')
            rank = request.form.get('rank')
            is_contact = request.form.get('is_contact')
            # 检查同一论文下是否存在相同排名（排除当前记录自身）
            same_rank = db.session.query(PublishPaper).filter(
                PublishPaper.P_ID == paperId,
                PublishPaper.P_Rank == rank,
                PublishPaper.P_ID != paperId  # 排除当前记录（更新时）
            ).first()
            if same_rank:
                return render_template('404.html', error_message='同一论文中排名
不能重复！')

           # 2. 验证通讯作者唯一性
            if is_contact == '1' and PublishPaper.P_Contact != '1':
                # 如果从非通讯作者改为通讯作者，检查是否已有通讯作者
                has_contact = db.session.query(PublishPaper).filter(
                    PublishPaper.P_ID == paperId,
                    PublishPaper.P_Contact == 1,
                    PublishPaper.id != PublishPaper.id  # 排除当前记录
                ).first()
                if has_contact:
                    return render_template('error.html', error_message='一篇论文
只能有一位通讯作者！')
            PublishPaper_result =
db.session.query(PublishPaper).filter_by(P_ID=paperId).first()
            PublishPaper_result.T_ID = teacherId
            PublishPaper_result.P_Rank = rank
            PublishPaper_result.P_Contact = is_contact
            db.session.commit()
        elif request.form.get('type') == 'delete1':
            paperId = request.form.get('key')
            paper_result =
db.session.query(Paper).filter_by(P_ID=paperId).first()
            publishpaper_result =
db.session.query(PublishPaper).filter_by(P_ID=paperId).first()
            publishpaperNotExist =
db.session.query(PublishPaper).filter_by(P_ID=paperId).scalar() is None
            if publishpaperNotExist != 1:
                db.session.delete(publishpaper_result)
                db.session.commit()
            db.session.delete(paper_result)
            db.session.commit()
        elif request.form.get('type') == 'delete2':
```

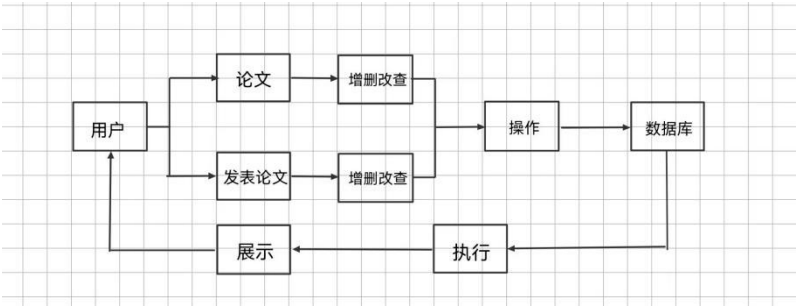
```
            paperId = request.form.get('key')
            publishpaper_result =
db.session.query(PublishPaper).filter_by(P_ID=paperId).first()
            db.session.delete(publishpaper_result)
            db.session.commit()
        elif request.form.get('type') == 'insert2':
            paperId = request.form.get('paperId')
            teacherId = request.form.get('teacherId')
            rank = request.form.get('rank')
            is_contact = request.form.get('is_contact')
            # 1. 验证排名唯一性
            existing_rank = db.session.query(PublishPaper).filter(
                PublishPaper.P_ID == paperId,
                PublishPaper.P_Rank == rank
            ).first()
            if existing_rank:
                return render_template('404.html', message='同一论文中排名不能重复!
')

            # 2. 验证通讯作者唯一性
            if is_contact == '1':
                has_contact = db.session.query(PublishPaper).filter(
                    PublishPaper.P_ID == paperId,
                    PublishPaper.P_Contact == 1
                ).first()
                if has_contact:
                    return render_template('404.html', message='一篇论文只能有一位
通讯作者! ')

            newPublishPaper = PublishPaper(
                P_ID=paperId,
                T_ID=teacherId,
                P_Rank=rank,
                P_Contact = is_contact
            )
            db.session.add(newPublishPaper)
            db.session.commit()
            result2 = db.session.query(Paper, PublishPaper).filter(Paper.P_ID ==
PublishPaper.P_ID).all()
            formatted_result1 = format_paper_data(result1)
            formatted_result2 = format_publishpaper_data(result2)
            return render_template('paper.html', labels1=labels1,
labels2=labels2, content1=formatted_result1, content2=formatted_result2)
        elif request.form.get('type') == 'insert1':
```

```
            paperId = request.form.get('paperId')
            paperName = request.form.get('name')
            paperUrl = request.form.get('url')
            paperYear = request.form.get('year')
            paperType = request.form.get('p_type')
            paperLevel = request.form.get('level')
            newPaper = Paper(
                P_ID=paperId,
                P_Name=paperName,
                P_Url=paperUrl,
                P_Year=paperYear,
                P_Type=paperType,
                P_Level=paperType
            )
            db.session.add(newPaper)
            db.session.commit()
    result1 = db.session.query(Paper).all()
    formatted_result1 = format_paper_data(result1)
    formatted_result2 = format_publishpaper_data(result2)
    return render_template('paper.html', labels1=labels1, labels2=labels2,
content1=formatted_result1, content2=formatted_result2)
```

输入：论文序号，论文名称，发表源，发表年份，类型，级别；发表论文教师工号，论文序号，排名，教师是否通讯作者

输出：增删改查结果

流程图：



3.3 项目模块

目的：提供教师承担项目信息的增、删、改、查功能；排名不能有重复，计算一个项目中所有教师的承担经费总额等于项目的总经费，项目类型只能在约定的取值集合中选取。

URL：http://127.0.0.1:8000/project

代码部分：

model.py

```
class Project(db.Model):
    __tablename__ = 'project'

    Pr_ID = db.Column(db.String(256), primary_key=True)
    Pr_Name = db.Column(db.String(256))
    Pr_Source = db.Column(db.String(256))
    Pr_Type = db.Column(db.Integer)
    Pr_Summoney = db.Column(db.Float, default=0)
    Pr_From = db.Column(db.Integer)
    Pr_End = db.Column(db.Integer)
class OwnProject(db.Model):
    __tablename__ = 'own_project'

    Pr_ID = db.Column(db.ForeignKey('project.Pr_ID', ondelete='RESTRICT',
onupdate='RESTRICT'), primary_key=True, nullable=False)
    T_ID = db.Column(db.ForeignKey('teacher.T_ID', ondelete='RESTRICT',
onupdate='RESTRICT'), primary_key=True, nullable=False)
    Pr_Rank = db.Column(db.Integer)
    Pr_money = db.Column(db.Float)
```

app.py

```
@app.route('/project', methods=['GET', 'POST'])
def project():
    labels1 = ['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']
    labels2 = ['项目号', '教师 ID', '排名', '承担经费']
    result_query1 = db.session.query(Project)
    result_query2 = db.session.query(Project,OwnProject).filter(Project.Pr_ID ==
OwnProject.Pr_ID)
    result1 = result_query1.all()
    result2 = result_query2.all()

    # 定义项目类型映射关系
    Type_mapping = {
        1: '国家级项目',
        2: '省部级项目',
        3: '市厅级项目',
        4: '企业合作级项目',
        5: '其他类型项目'
    }
    def format_Project_data(projects):
        formatted = []
```

```
        for p in projects:
            Type = Type_mapping.get(p.Pr_Type, '未知')
            formatted.append({
                'Pr_ID': p.Pr_ID,
                'Pr_Name': p.Pr_Name,
                'Pr_Source': p.Pr_Source,
                'Pr_Type': Type,
                'Pr_Summoney': p.Pr_Summoney,
                'Pr_From': p.Pr_From,
                'Pr_End': p.Pr_End
            })
        return formatted
    formatted_result = format_Project_data(result1)
    if request.method == 'GET':
        return render_template('project.html', labels1=labels1, labels2=labels2,
content1=formatted_result, content2=result2)
    else:
        # 查询
        if request.form.get('type') == 'query1':
            projectId = request.form.get('projectId')
            projectName = request.form.get('name')
            projectSource = request.form.get('source')
            projectSum = request.form.get('Summoney')
            projectFrom = request.form.get('beginyear')
            projectEnd = request.form.get('endyear')
            projectType = request.form.get('Type')
            if projectId != '':
                result_query1 = result_query1.filter(Project.Pr_ID == projectId)
            if projectName != '':
                result_query1 = result_query1.filter(Project.Pr_Name ==
projectName)
            if projectSource != '':
                result_query1 = result_query1.filter(Project.Pr_Source ==
projectSource)
            if projectSum != '':
                result_query1 = result_query1.filter(Project.Pr_Summoney ==
projectSum)
            if projectFrom != '':
                result_query1 = result_query1.filter(Project.Pr_From ==
projectFrom)
            if projectEnd != '':
                result_query1 = result_query1.filter(Project.Pr_End ==
projectEnd)
            if projectType != '':
```

```
                result_query1 = result_query1.filter(Project.Pr_Type ==
projectType)
            result1 = result_query1.all()
            formatted_result = format_Project_data(result1)
            return render_template('project.html', labels1=labels1,
labels2=labels2, content1=formatted_result, content2=result2)
        elif request.form.get('type') == 'query2':
            projectId = request.form.get('projectId')
            teacherId = request.form.get('teacherId')
            rank = request.form.get('rank')
            money = request.form.get('money')
            if projectId != '':
                result_query2 = result_query2.filter(OwnProject.Pr_ID==
projectId)
            if teacherId != '':
                result_query2 = result_query2.filter(OwnProject.T_ID ==
teacherId)
            if rank != '':
                result_query2 = result_query2.filter(OwnProject.Pr_Rank == rank)
            if money != '':
                result_query2 = result_query2.filter(OwnProject.Pr_money ==
money)
            result2 = result_query2.all()
            formatted_result = format_Project_data(result1)
            return render_template('project.html', labels1=labels1,
labels2=labels2, content1=formatted_result, content2=result2)
        elif request.form.get('type') == 'update1':
            projectId = request.form.get('key')
            projectName = request.form.get('name')
            projectSource = request.form.get('source')
            projectFrom = request.form.get('beginyear')
            projectEnd = request.form.get('endyear')
            projectType = request.form.get('Type')
            Project_result =
db.session.query(Project).filter_by(Pr_ID=projectId).first()
            Project_result.Pr_Name = projectName
            Project_result.Pr_Source = projectSource
            Project_result.Pr_From = projectFrom
            Project_result.Pr_End = projectEnd
            Project_result.Pr_Type = projectType
            db.session.commit()
        elif request.form.get('type') == 'update2':
            projectId = request.form.get('key')
            teacherId = request.form.get('teacherId')
```

```
            rank = request.form.get('rank')
            money = request.form.get('money')
            # 检查同一项目下是否存在相同排名（排除当前记录自身）
            same_rank = db.session.query(OwnProject).filter(
                OwnProject.Pr_ID == projectId,
                OwnProject.Pr_Rank == rank,
                OwnProject.Pr_ID != projectId  # 排除当前记录（更新时）
            ).first()
            if same_rank:
                return render_template('404.html', error_message='同一论文中排名
不能重复！')

            # 更新总经费
            oldmoney =
db.session.query(OwnProject).filter_by(Pr_ID=projectId).first().Pr_money
            projectSum =
db.session.query(Project).filter_by(Pr_ID=projectId).first().Pr_Summoney
            projectSum = projectSum + float(money) - oldmoney
            Project_result =
db.session.query(Project).filter_by(Pr_ID=projectId).first()
            Project_result.Pr_Summoney = projectSum
            OwnProject_result =
db.session.query(OwnProject).filter_by(Pr_ID=projectId).first()
            OwnProject_result.T_ID = teacherId
            OwnProject_result.Pr_Rank = rank
            OwnProject_result.Pr_money = money
            db.session.commit()
        elif request.form.get('type') == 'delete1':
            projectId = request.form.get('key')
            project_result =
db.session.query(Project).filter_by(Pr_ID=projectId).first()
            ownproject_result =
db.session.query(OwnProject).filter_by(Pr_ID=projectId).first()
            ownprojectNotExist =
db.session.query(OwnProject).filter_by(Pr_ID=projectId).scalar() is None
            if ownprojectNotExist != 1:
                db.session.delete(ownproject_result)
                db.session.commit()
            db.session.delete(project_result)
            db.session.commit()
        elif request.form.get('type') == 'delete2':
            projectId = request.form.get('key')
            ownproject_result =
db.session.query(OwnProject).filter_by(Pr_ID=projectId).first()
```

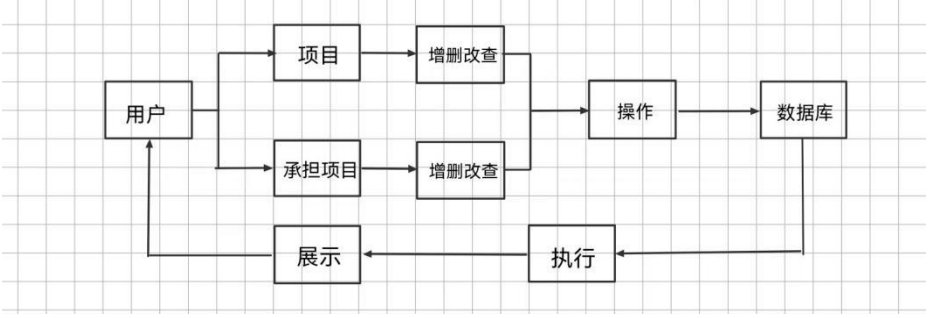
```
            # 删除时更新总经费
            projectSum =
db.session.query(Project).filter_by(Pr_ID=projectId).first().Pr_Summoney
            money =
db.session.query(OwnProject).filter_by(Pr_ID=projectId).first().Pr_money
            projectSum = projectSum - money
            Project_result =
db.session.query(Project).filter_by(Pr_ID=projectId).first()
            Project_result.Pr_Summoney = projectSum
            db.session.delete(ownproject_result)
            db.session.commit()
        elif request.form.get('type') == 'insert2':
            projectId = request.form.get('projectId')
            teacherId = request.form.get('teacherId')
            rank = request.form.get('rank')
            money = request.form.get('money')
            # 1. 验证排名唯一性
            existing_rank = db.session.query(OwnProject).filter(
                OwnProject.Pr_ID == projectId,
                OwnProject.Pr_Rank == rank
            ).first()
            if existing_rank:
                error_title = '更新错误'
                error_message = '排名不能相同'
                return render_template('404.html', error_title=error_title,
error_message=error_message)

            # # 2. 将项目经费添加到经费总额
            projectSum =
db.session.query(Project).filter_by(Pr_ID=projectId).first().Pr_Summoney
            projectSum = projectSum + float(money)
            Project_result =
db.session.query(Project).filter_by(Pr_ID=projectId).first()
            Project_result.Pr_Summoney = projectSum
            newOwnProject = OwnProject(
                Pr_ID=projectId,
                T_ID=teacherId,
                Pr_Rank=rank,
                Pr_money = money
            )
            db.session.add(newOwnProject)
            db.session.commit()
            result2 = db.session.query(Project, OwnProject).filter(Project.Pr_ID
== OwnProject.Pr_ID).all()
```

```
            formatted_result = format_Project_data(result1)
            return render_template('project.html', labels1=labels1,
labels2=labels2, content1=formatted_result, content2=result2)
        elif request.form.get('type') == 'insert1':
            projectId = request.form.get('projectId')
            projectName = request.form.get('name')
            projectSource = request.form.get('source')
            projectFrom = request.form.get('beginyear')
            projectEnd = request.form.get('endyear')
            projectType = request.form.get('Type')
            newProject = Project(
                Pr_ID=projectId,
                Pr_Name=projectName,
                Pr_Source=projectSource,
                Pr_Type=projectType,
                Pr_From=projectFrom,
                Pr_End=projectEnd
            )
            db.session.add(newProject)
            db.session.commit()
    result1 = db.session.query(Project).all()
    formatted_result = format_Project_data(result1)
    return render_template('project.html', labels1=labels1, labels2=labels2,
content1=formatted_result, content2=result2)
```

输入：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；承担项目的教师工号，项目号，排名，承担经费

输出：增删改查结果

流程图：



3.4 课程模块

目的：提供教师承担项目信息的增、删、改、查功能；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。

URL：http://127.0.0.1:8000/class

代码部分：

model.py

```
class Class(db.Model):
    __tablename__ = 'class'

    C_ID = db.Column(db.String(256), primary_key=True)
    C_Name = db.Column(db.String(256))
    C_Sum = db.Column(db.Integer)
    C_Type = db.Column(db.Integer)
class TeachClass(db.Model):
    __tablename__ = 'teach_class'

    C_ID = db.Column(db.ForeignKey('class.C_ID', ondelete='RESTRICT',
onupdate='RESTRICT'), primary_key=True, nullable=False)
    T_ID = db.Column(db.ForeignKey('teacher.T_ID', ondelete='RESTRICT',
onupdate='RESTRICT'), primary_key=True, nullable=False)
    C_Year = db.Column(db.Integer)
    C_Semester = db.Column(db.Integer)
    C_hours = db.Column(db.Integer)
```

app.py

```
@app.route('/class', methods=['GET', 'POST'])
def classes():
    labels1 = ['课程号', '课程名称', '学时数',  '课程性质']
    labels2 = ['课程号', '教师 ID', '年份', '学期', '承担学时']
    result_query1 = db.session.query(Class)
    result_query2 = db.session.query(Class,TeachClass).filter(Class.C_ID ==
TeachClass.C_ID)
    result1 = result_query1.all()
    result2 = result_query2.all()

    # 定义课程性质映射关系
    Type_mapping = {
        1: '本科生课程',
        2: '研究生课程'
```

```
    }
    # 定义学期映射关系
    semester_mapping = {
        1: '春季学期',
        2: '夏季学期',
        3: '秋季学期'
    }
    def format_class_data(classes):
        formatted = []
        for c in classes:
            Type = Type_mapping.get(c.C_Type, '未知')
            formatted.append({
                'C_ID': c.C_ID,
                'C_Name': c.C_Name,
                'C_Sum': c.C_Sum,
                'C_Type':Type
            })
        return formatted

    def format_teachclass_data(teachclasses):
        formatted = []
        for c in teachclasses:
            semester = semester_mapping.get(c.TeachClass.C_Semester, '未知')
            formatted.append({
                'C_ID': c.TeachClass.C_ID,
                'T_ID': c.TeachClass.T_ID,
                'C_Year': c.TeachClass.C_Year,
                'C_Semester': semester,
                'C_hours': c.TeachClass.C_hours
            })
        return formatted
    formatted_result1 = format_class_data(result1)
    formatted_result2 = format_teachclass_data(result2)
    if request.method == 'GET':
        return render_template('class.html', labels1=labels1, labels2=labels2,
content1=formatted_result1, content2=formatted_result2)
    else:
        if request.form.get('type') == 'query1':
            classId = request.form.get('classId')
            className = request.form.get('name')
            classSum = request.form.get('sumhours')
            classType = request.form.get('Type')
            if classId != '':
                result_query1 = result_query1.filter(Class.C_ID == classId)
```

```
            if className != '':
                result_query1 = result_query1.filter(Class.C_Name == className)
            if classSum != '':
                result_query1 = result_query1.filter(Class.C_Sum == classSum)
            if classType != '':
                result_query1 = result_query1.filter(Class.C_Type == classType)
            result1 = result_query1.all()
            formatted_result1 = format_class_data(result1)
            formatted_result2 = format_teachclass_data(result2)
            return render_template('class.html', labels1=labels1,
labels2=labels2, content1=formatted_result1, content2=formatted_result2)
        elif request.form.get('type') == 'query2':
            classId = request.form.get('classId')
            teacherId = request.form.get('teacherId')
            year = request.form.get('year')
            semester = request.form.get('semester')
            hours = request.form.get('hours')
            if classId != '':
                result_query2 = result_query2.filter(TeachClass.C_ID== classId)
            if teacherId != '':
                result_query2 = result_query2.filter(TeachClass.T_ID ==
teacherId)
            if year != '':
                result_query2 = result_query2.filter(TeachClass.C_Year == year)
            if semester != '':
                result_query2 = result_query2.filter(TeachClass.C_Semester ==
semester)
            if hours != '':
                result_query2 = result_query2.filter(TeachClass.C_hours == hours)
            result2 = result_query2.all()
            formatted_result1 = format_class_data(result1)
            formatted_result2 = format_teachclass_data(result2)
            return render_template('class.html', labels1=labels1,
labels2=labels2, content1=formatted_result1, content2=formatted_result2)
        elif request.form.get('type') == 'update1':
            classId = request.form.get('key')
            className = request.form.get('name')
            classSum = request.form.get('sumhours')
            classType = request.form.get('Type')
            Class_result =
db.session.query(Class).filter_by(C_ID=classId).first()
            Class_result.C_Name = className
            Class_result.C_Sum = classSum
            Class_result.C_Type = classType
```

```
            db.session.commit()
        elif request.form.get('type') == 'update2':
            classId = request.form.get('key')
            teacherId = request.form.get('teacherId')
            year = request.form.get('year')
            semester = request.form.get('semester')
            hours = request.form.get('hours')

            # 一个课程中所有教师主讲课程的总额等于总学时
            course = db.session.query(Class).filter_by(C_ID=classId).first()
            if not course:
                return render_template('404.html', error_message='课程不存在！')

            # 查询当前记录
            teach_class_record = db.session.query(TeachClass).filter_by(
                C_ID=classId, T_ID=teacherId, C_Year=year, C_Semester=semester
            ).first()
            if not teach_class_record:
                return render_template('404.html', error_message='记录不存在，无
法更新！')

            # 计算当前学期其他教师已分配的学时（排除当前记录）
            total_assigned_hours =
db.session.query(func.sum(TeachClass.C_hours)).filter(
                TeachClass.C_ID == classId,
                TeachClass.C_Year == year,
                TeachClass.C_Semester == semester,
                db.or_(
                    TeachClass.T_ID != teacherId,
                    db.and_(TeachClass.C_Year != year, TeachClass.C_Semester !=
semester)
                )
            ).scalar() or 0
            total_assigned_hours += int(hours)
            # 判断是否超过每学期的总学时
            # if total_assigned_hours > course.C_Sum:
            if total_assigned_hours != course.C_Sum:
                error_message=f'学期 {year} 第 {semester} 学期教师承担学时总额
({total_assigned_hours})多于课程每学期总学时({course.C_Sum})！'
                return render_template('404.html', error_message = error_message)

            # 更新 TeachClass 记录
            TeachClass_result =
db.session.query(TeachClass).filter_by(C_ID=classId).first()
```

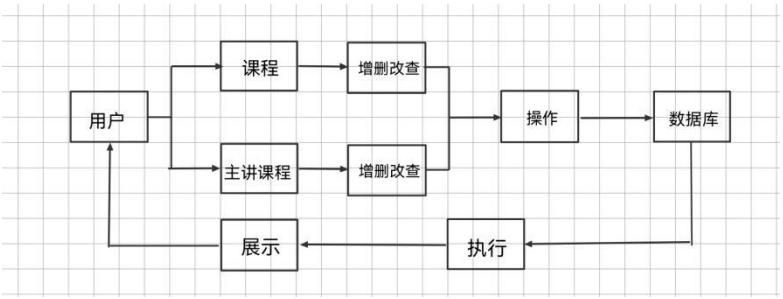
```
            TeachClass_result.T_ID = teacherId
            TeachClass_result.C_Year = year
            TeachClass_result.C_Semester = semester
            TeachClass_result.C_hours = hours
            db.session.commit()
        elif request.form.get('type') == 'delete1':
            classId = request.form.get('key')
            class_result =
db.session.query(Class).filter_by(C_ID=classId).first()
            teachclass_result =
db.session.query(TeachClass).filter_by(C_ID=classId).first()
            teachclassNotExist =
db.session.query(TeachClass).filter_by(C_ID=classId).scalar() is None
            if teachclassNotExist != 1:
                db.session.delete(teachclass_result)
                db.session.commit()
            db.session.delete(class_result)
            db.session.commit()
        elif request.form.get('type') == 'delete2':
            classId = request.form.get('key')
            teachclass_result =
db.session.query(TeachClass).filter_by(C_ID=classId).first()
            db.session.delete(teachclass_result)
            db.session.commit()
        elif request.form.get('type') == 'insert2':
            classId = request.form.get('classId')
            teacherId = request.form.get('teacherId')
            year = request.form.get('year')
            semester = request.form.get('semester')
            hours = request.form.get('hours')
            # # 验证一门课程中所有教师主讲学时总额等于课程总学时
            course = db.session.query(Class).filter_by(C_ID=classId).first()
            if not course:
                return render_template('404.html', error_message='课程不存在！')

            # 计算当前学期已分配的学时
            total_assigned_hours =
db.session.query(func.sum(TeachClass.C_hours)).filter(
                TeachClass.C_ID == classId,
                TeachClass.C_Year == year,
                TeachClass.C_Semester == semester
            ).scalar() or 0
            total_assigned_hours += int(hours)
            # 判断是否超过每学期的总学时
```

```
            # if total_assigned_hours > course.C_Sum:
            if total_assigned_hours != course.C_Sum:
                error_message=f'学期 {year} 第 {semester} 学期教师承担学时总额
({total_assigned_hours})不等于课程每学期总学时({course.C_Sum})! '
                return render_template('404.html', error_message=error_message)
            newTeachClass = TeachClass(
                C_ID=classId,
                T_ID=teacherId,
                C_Year=year,
                C_Semester = semester,
                C_hours = hours
            )
            db.session.add(newTeachClass)
            db.session.commit()
            result2 = db.session.query(Class, TeachClass).filter(Class.C_ID ==
TeachClass.C_ID).all()
            formatted_result1 = format_class_data(result1)
            formatted_result2 = format_teachclass_data(result2)
            return render_template('class.html', labels1=labels1,
labels2=labels2, content1=formatted_result1, content2=formatted_result2)
        elif request.form.get('type') == 'insert1':
            classId = request.form.get('classId')
            className = request.form.get('name')
            classSum = request.form.get('sumhours')
            classType = request.form.get('Type')
            newClass = Class(
                C_ID=classId,
                C_Name=className,
                C_Sum=classSum,
                C_Type=classType
            )
            db.session.add(newClass)
            db.session.commit()
    result1 = db.session.query(Class).all()
    formatted_result1 = format_class_data(result1)
    formatted_result2 = format_teachclass_data(result2)
    return render_template('class.html', labels1=labels1, labels2=labels2,
content1=formatted_result1, content2=formatted_result2)
```

输入：课程号，课程名称，学时数，课程性质；主讲课程的教师工号，对应课程号，年份，学期，承担学时。

输出：增删改查结果。

流程图：



3.5 查询模块

目的：实现对信息的查询，以及导出 pdf 的功能

URL：http://127.0.0.1:8000/search

代码部分：

app.py

```
@app.route('/search', methods=['GET', 'POST'])
def search():
    labels_teacher = ['工号', '姓名', '性别', '职称']
    result_query_teacher = db.session.query(Teacher)
    result_teacher = result_query_teacher.all()

    labels_paper = ['教师 ID', '论文名称', '发表源', '发表年份', '类型', '级别','排
名', '是否通讯作者']
    result_query_paper = db.session.query(Paper, PublishPaper).filter(Paper.P_ID
== PublishPaper.P_ID)
    result_paper = result_query_paper.all()
    labels_project = ['教师 ID', '项目名称', '项目来源',  '项目类型', '总经费', '开
始年份', '结束年份', '承担经费']
    result_query_project =
db.session.query(Project,OwnProject).filter(Project.Pr_ID == OwnProject.Pr_ID)
    result_project = result_query_project.all()
    labels_class = ['课程号', '教师 ID', '课程名称', '学时数',  '课程性质', '年份',
'学期', '承担学时']
    result_query_class = db.session.query(Class,TeachClass).filter(Class.C_ID ==
TeachClass.C_ID)
    result_class = result_query_class.all()
    # 定义性别映射关系
    gender_mapping = {
        1: '男',
        2: '女'
```

```
}
# 定义职称映射关系
Teacher_Type_mapping = {
    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'
}
# 定义论文类型映射关系
Paper_Type_mapping = {
    1: 'full paper',
    2: 'short paper',
    3: 'poster paper',
    4: 'semo paper'
}
# 定义论文级别映射关系
Level_mapping = {
    1: 'CCF-A',
    2: 'CCF-B',
    3: 'CCF-C',
    4: '中文 CCF-A',
    5: '中文 CCF-B',
    6: '无级别'
}
# 定义是否为通讯作者映射关系
contact_mapping = {
    0: '否',
    1: '是'
}
# 定义项目类型映射关系
Project_Type_mapping = {
    1: '国家级项目',
    2: '省部级项目',
    3: '市厅级项目',
    4: '企业合作级项目',
    5: '其他类型项目'
}
```

```
# 定义课程性质映射关系
Class_Type_mapping = {
    1: '本科生课程',
    2: '研究生课程'
}
# 定义学期映射关系
semester_mapping = {
    1: '春季学期',
    2: '夏季学期',
    3: '秋季学期'
}
def format_teacher_data(teachers):
    formatted = []
    for t in teachers:
        gender = gender_mapping.get(t.T_sexual, '未知')
        Type = Teacher_Type_mapping.get(t.T_type, '未知')
        formatted.append({
            'T_ID': t.T_ID,
            'T_Name': t.T_Name,
            'T_sexual': gender,
            'T_type': Type
        })
    return formatted

def format_paper_data(papers):
    formatted = []
    for p in papers:
        Type = Paper_Type_mapping.get(p.Paper.P_Type, '未知')
        Level = Level_mapping.get(p.Paper.P_Level, '未知')
        contact = contact_mapping.get(p.PublishPaper.P_Contact, '未知')
        formatted.append({
            'P_ID': p.Paper.P_ID,
            'T_ID': p.PublishPaper.T_ID,
            'P_Name': p.Paper.P_Name,
            'P_Url': p.Paper.P_Url,
            'P_Year':p.Paper.P_Year,
            'P_Type': Type,
            'P_Level': Level,
            'P_Rank':p.PublishPaper.P_Rank,
            'P_Contact':contact
        })
    return formatted

def format_project_data(projects):
```

```
        formatted = []
        for p in projects:
            Type = Project_Type_mapping.get(p.Project.Pr_Type, '未知')
            formatted.append({
                'Pr_ID': p.Project.Pr_ID,
                'T_ID': p.OwnProject.T_ID,
                'Pr_Name': p.Project.Pr_Name,
                'Pr_Source': p.Project.Pr_Source,
                'Pr_Type': Type,
                'Pr_Summoney': p.Project.Pr_Summoney,
                'Pr_From': p.Project.Pr_From,
                'Pr_End': p.Project.Pr_End,
                'Pr_Rank': p.OwnProject.Pr_Rank,
                'Pr_money': p.OwnProject.Pr_money
            })
        return formatted

    def format_class_data(classes):
        formatted = []
        for c in classes:
            Type = Class_Type_mapping.get(c.Class.C_Type, '未知')
            semester = semester_mapping.get(c.TeachClass.C_Semester, '未知')
            formatted.append({
                'C_ID': c.Class.C_ID,
                'T_ID': c.TeachClass.T_ID,
                'C_Name': c.Class.C_Name,
                'C_Sum': c.Class.C_Sum,
                'C_Type':Type,
                'C_Year': c.TeachClass.C_Year,
                'C_Semester': semester,
                'C_hours': c.TeachClass.C_hours
            })
        return formatted
    formatted_teacher_result = format_teacher_data(result_teacher)
    formatted_paper_result = format_paper_data(result_paper)
    formatted_project_result = format_project_data(result_project)
    formatted_class_result = format_class_data(result_class)
    if request.method == 'GET':
        return render_template('search.html', labels_teacher=labels_teacher,
labels_paper=labels_paper, labels_project=labels_project,
labels_class=labels_class,
                               content1=formatted_teacher_result,
content2=formatted_paper_result, content3=formatted_project_result,
content4=formatted_class_result)
```

```
    else:
        if request.form.get('type') == 'query':
            teacher_id = request.form.get('id')
            fromyear = request.form.get('fromyear')
            endyear = request.form.get('endyear')
            # 教师 id 查询:
            if teacher_id:
                result_query_teacher = result_query_teacher.filter(Teacher.T_ID
== teacher_id)
                result_query_paper = result_query_paper.filter(PublishPaper.T_ID
== teacher_id)
                result_query_project =
result_query_project.filter(OwnProject.T_ID == teacher_id)
                result_query_class = result_query_class.filter(TeachClass.T_ID
== teacher_id)

            # 根据年份范围筛选
            if fromyear and endyear:
                try:
                    # 论文年份查询
                    from_date = f"{int(fromyear)}-01-01"
                    to_date = f"{int(endyear)}-12-31"
                    result_query_paper = result_query_paper.filter(
                        Paper.P_Year.between(from_date, to_date)
                    )
                    # 项目年份查询
                    result_query_project = result_query_project.filter(
                        (Project.Pr_From <= endyear) & (Project.Pr_End >=
fromyear)
                    )
                    # 授课年份查询
                    result_query_class = result_query_class.filter(
                        TeachClass.C_Year.between(fromyear, endyear)
                    )
                except ValueError:
                    pass  # 忽略非法年份输入

            result_teacher = result_query_teacher.all()
            result_paper = result_query_paper.all()
            result_project = result_query_project.all()
            result_class = result_query_class.all()

            formatted_teacher_result = format_teacher_data(result_teacher)
            formatted_paper_result = format_paper_data(result_paper)
```

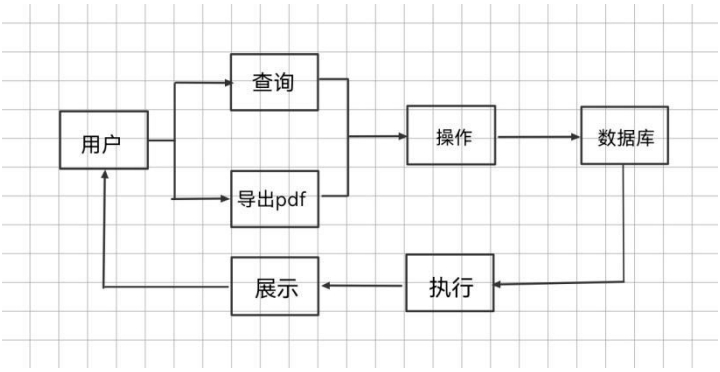
```
            formatted_project_result = format_project_data(result_project)
            formatted_class_result = format_class_data(result_class)
            return render_template('search.html', labels_teacher=labels_teacher,
labels_paper=labels_paper, labels_project=labels_project,
labels_class=labels_class,
                                   content1=formatted_teacher_result,
content2=formatted_paper_result, content3=formatted_project_result,
content4=formatted_class_result)

        elif request.form.get('type') == 'export':
            # 导出查询结果到 pdf 文件
            teacher_id = request.form.get('id')
            fromyear = request.form.get('fromyear')
            endyear = request.form.get('endyear')
            # 教师 id 查询:
            if teacher_id:
                result_query_teacher = result_query_teacher.filter(Teacher.T_ID
== teacher_id)
                result_query_paper = result_query_paper.filter(PublishPaper.T_ID
== teacher_id)
                result_query_project =
result_query_project.filter(OwnProject.T_ID == teacher_id)
                result_query_class = result_query_class.filter(TeachClass.T_ID
== teacher_id)

            # 根据年份范围筛选
            if fromyear and endyear:
                try:
                    # 论文年份查询
                    from_date = f"{int(fromyear)}-01-01"
                    to_date = f"{int(endyear)}-12-31"
                    result_query_paper = result_query_paper.filter(
                        Paper.P_Year.between(from_date, to_date)
                    )
                    # 项目年份查询
                    result_query_project = result_query_project.filter(
                        (Project.Pr_From <= endyear) & (Project.Pr_End >=
fromyear)
                    )
                    # 授课年份查询
                    result_query_class = result_query_class.filter(
                        TeachClass.C_Year.between(fromyear, endyear)
                    )
                except ValueError:
```

```
                pass  # 忽略非法年份输入

            result_teacher = result_query_teacher.all()
            result_paper = result_query_paper.all()
            result_project = result_query_project.all()
            result_class = result_query_class.all()

            formatted_teacher_result = format_teacher_data(result_teacher)
            formatted_paper_result = format_paper_data(result_paper)
            formatted_project_result = format_project_data(result_project)
            formatted_class_result = format_class_data(result_class)

            # 渲染模板
            rendered = render_template('export.html',
                                       content1=formatted_teacher_result,
                                       content2=formatted_paper_result,
                                       content3=formatted_project_result,
                                       content4=formatted_class_result,
                                       fromyear=fromyear,
                                       endyear=endyear)

            # 手动指定 wkhtmltopdf 路径
            config = pdfkit.configuration(wkhtmltopdf=r'C:\Program
Files\wkhtmltopdf\bin\wkhtmltopdf.exe')
            pdf_file_path = 'output.pdf'
            # 生成 PDF
            pdfkit.from_string(rendered, pdf_file_path, configuration=config)
            # 构造响应
            print(f'PDF 文件已生成：{pdf_file_path}')
            return rendered

    formatted_teacher_result = format_teacher_data(result_teacher)
    formatted_paper_result = format_paper_data(result_paper)
    formatted_project_result = format_project_data(result_project)
    formatted_class_result = format_class_data(result_class)
    return render_template('search.html', labels_teacher=labels_teacher,
labels_paper=labels_paper, labels_project=labels_project,
labels_class=labels_class,
                           content1=formatted_teacher_result,
content2=formatted_paper_result, content3=formatted_project_result,
content4=formatted_class_result)
```

输入：教师工号，年份范围；是否打印 pdf

输出：教师信息；pdf

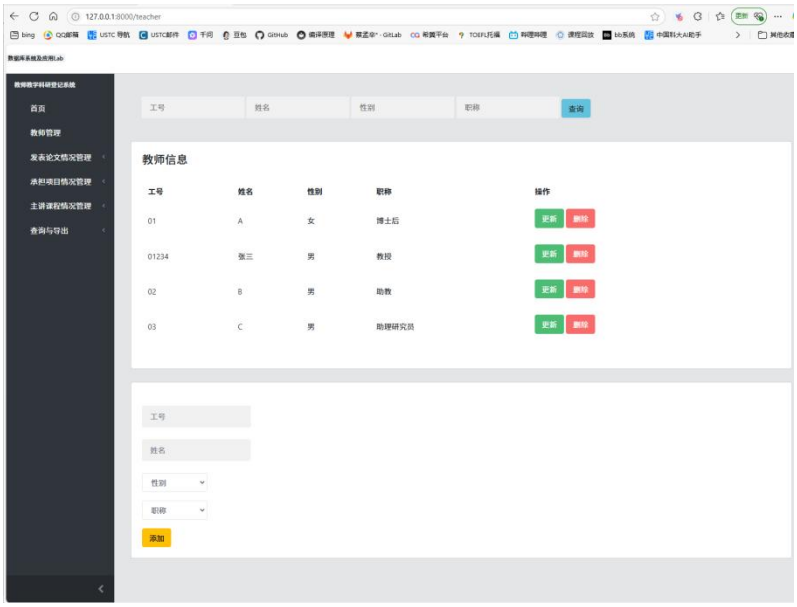
流程图：



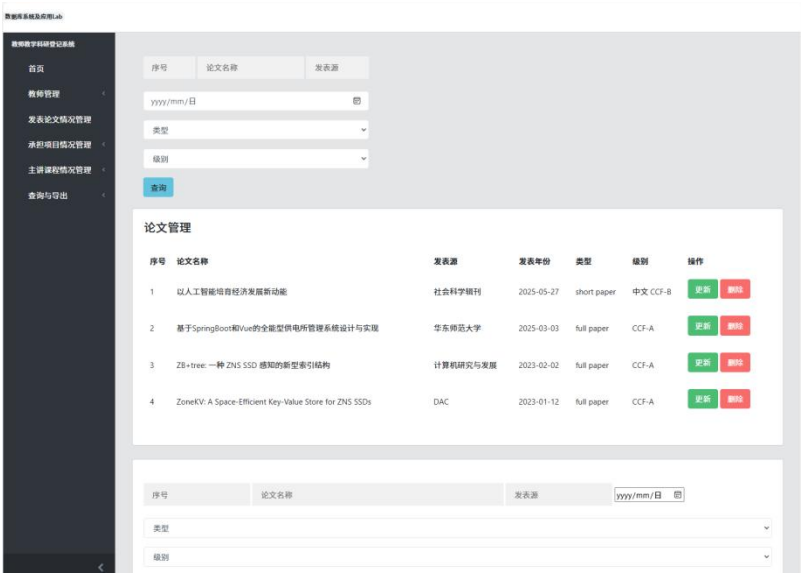
4 实现与测试

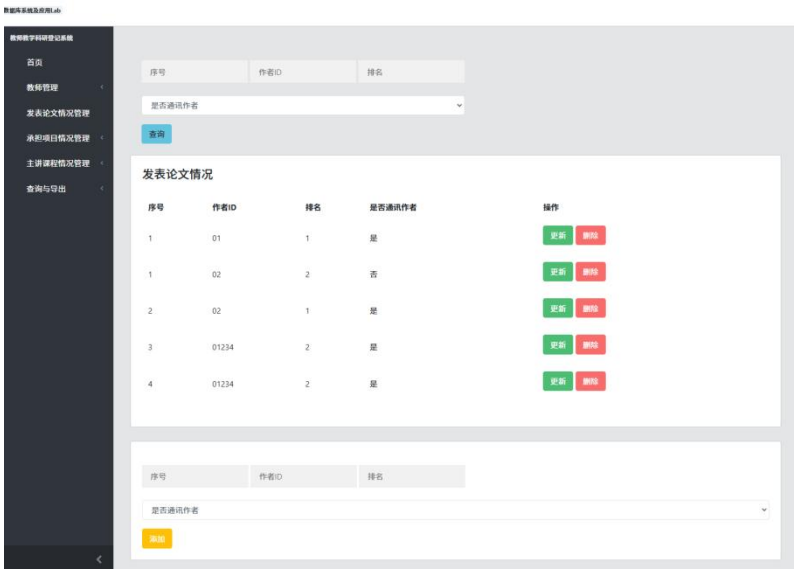
4.1 实现结果

教师：

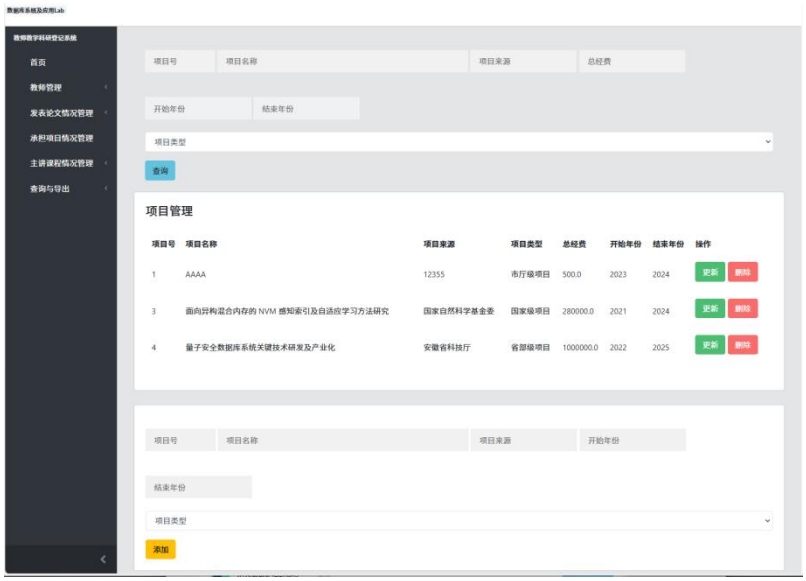


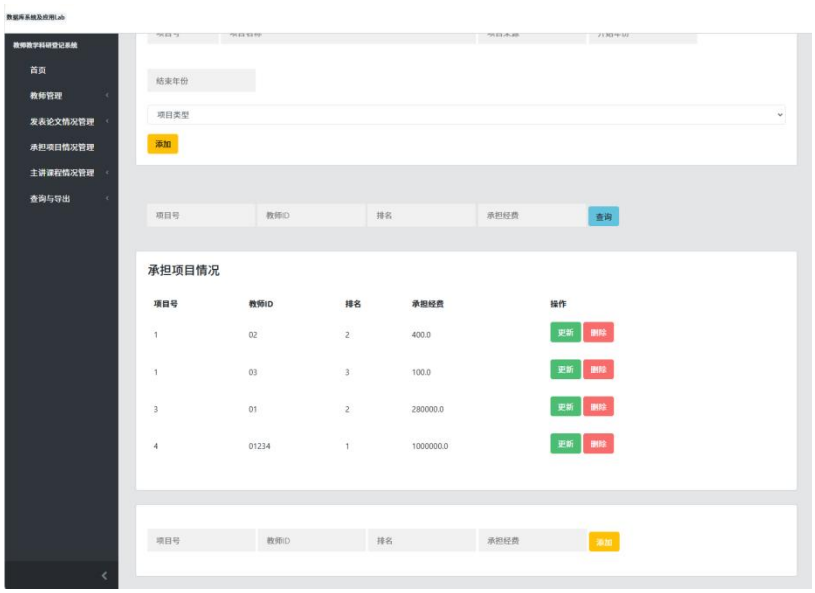
论文：



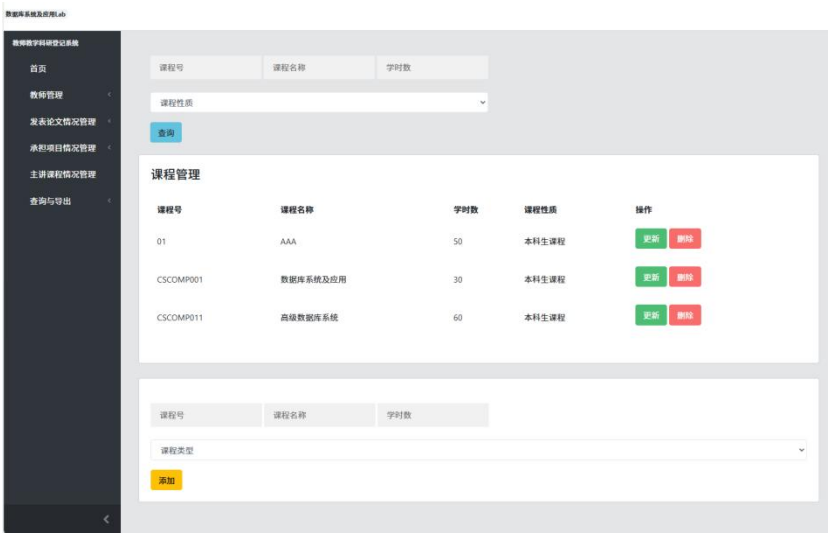

教师教学科研登记系统
首页
教师管理
发表论文情况管理
承担项目情况管理
主讲课程情况管理
查询与导出
序号
作者ID
排名
是否通讯作者
查询
发表论文情况
序号 作者ID 排名 是否通讯作者 操作
1 01 1 是 更新 删除
1 02 2 否 更新 删除
2 02 1 是 更新 删除
3 01234 2 是 更新 删除
4 01234 2 是 更新 删除
序号
作者ID
排名
是否通讯作者
添加

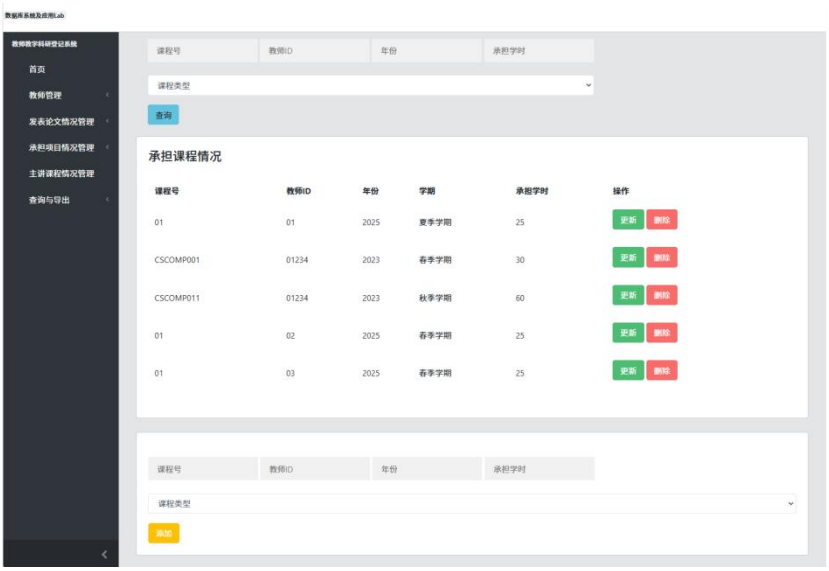
项目：


教师教学科研登记系统
首页
教师管理
发表论文情况管理
承担项目情况管理
主讲课程情况管理
查询与导出
项目号
项目名称
项目来源
总经费
开始年份
结束年份
项目类型
查询
项目管理
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 操作
1 AAAA 12355 市厅级项目 500.0 2023 2024 更新 删除
3 面向异构混合内存的 NVM 感知索引及自适应学习方法研究 国家自然科学基金委 国家级项目 280000.0 2021 2024 更新 删除
4 量子安全数据库系统关键技术研发及产业化 安徽省科技厅 省部级项目 1000000.0 2022 2025 更新 删除
项目号
项目名称
项目来源
开始年份
结束年份
项目类型
添加


教师教学科研登记系统
首页
教师管理
发表论文情况管理
承担项目情况管理
主讲课程情况管理
查询与导出
结束年份
项目类型
添加
项目号
教师ID
排名
承担经费
查询
承担项目情况
项目号 教师ID 排名 承担经费 操作
1 02 2 400.0 更新 删除
1 03 3 100.0 更新 删除
3 01 2 280000.0 更新 删除
4 01234 1 1000000.0 更新 删除
项目号
教师ID
排名
承担经费
添加

课程：


数据库系统及应用Lab
教师教学科研登记系统
首页
教师管理
发表论文情况管理
承担项目情况管理
主讲课程情况管理
查询与导出
课程号
课程名称
学时数
课程性质
查询
课程管理
课程号 课程名称 学时数 课程性质 操作
01 AAA 50 本科生课程 更新 删除
CSCOMP001 数据库系统及应用 30 本科生课程 更新 删除
CSCOMP011 高级数据库系统 60 本科生课程 更新 删除
课程号
课程名称
学时数
课程类型
添加


数据库系统及应用Lab
教师教学科研登记系统
首页
教师管理
发表论文情况管理
承担项目情况管理
主讲课程情况管理
查询与导出
课程号
教师ID
年份
承担学时
课程类型
查询
承担课程情况
课程号 教师ID 年份 学期 承担学时 操作
01 01 2025 夏季学期 25 更新 删除
CSCOMP001 01234 2023 春季学期 30 更新 删除
CSCOMP011 01234 2023 秋季学期 60 更新 删除
01 02 2025 春季学期 25 更新 删除
01 03 2025 春季学期 25 更新 删除
课程号
教师ID
年份
承担学时
课程类型
添加

查询：


数据库系统及应用Lab
教师教学科研登记系统
首页
教师管理
发表论文情况管理
承担项目情况管理
主讲课程情况管理
查询与导出
01234
年份开始
年份结束
查询
工号
年份开始
年份结束
导出PDF
教师信息
工号 姓名 性别 职称
01234 张三 男 教授
论文信息
教师ID 论文名称 发表源 发表年份 类型 级别 排名 是否通讯作者
01234 ZB+tree: 一种 ZNS SSD 感知的新型索引结构 计算机研究与发展 2023-02-02 full paper CCF-A 2 是
01234 ZoneKV: A Space-Efficient Key-Value Store for ZNS SSDs DAC 2023-01-12 full paper CCF-A 2 是
项目信息
教师ID 项目名称 项目来源 项目类型 总经费 开始年份 结束年份 承担经费
01234 量子安全数据库系统关键技术研发及产业化 安徽省科技厅 省部级项目 1000000.0 2022 2025 1000000.0

Pdf：

教师教学科研工作统计（2022-2025）

教师基本信息

工号	01234	姓名	张三	性别	男	职称	教授

教学情况

课程号	课程名	主讲学时	学期
CSCOMP001	数据库系统及应用	30	春季学期
CSCOMP011	高级数据库系统	60	秋季学期

发表论文情况

1. ZB+tree: 一种 ZNS SSD 感知的新型索引结构：计算机研究与发展, 2023-02-02, CCF-A, 排名第2, 是
2. ZoneKV: A Space-Efficient Key-Value Store for ZNS SSDs：DAC, 2023-01-12, CCF-A, 排名第2, 是

承担项目情况

1. 面向异构混合内存的 NVM 感知索引及自适应学习方法研究, 国家自然科学基金委, 国家级项目, 2021-2024, 总经费：580000.0, 承担经费：300000.0
2. 量子安全数据库系统关键技术研发及产业化, 安徽省科技厅, 省部级项目, 2022-2025, 总经费：2000000.0, 承担经费：1000000.0

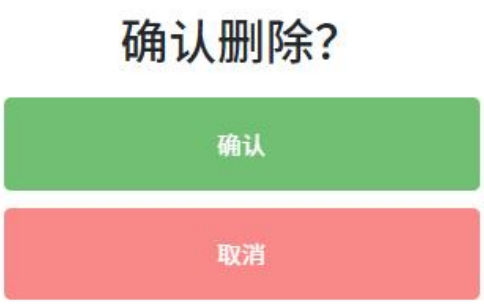
404 界面：

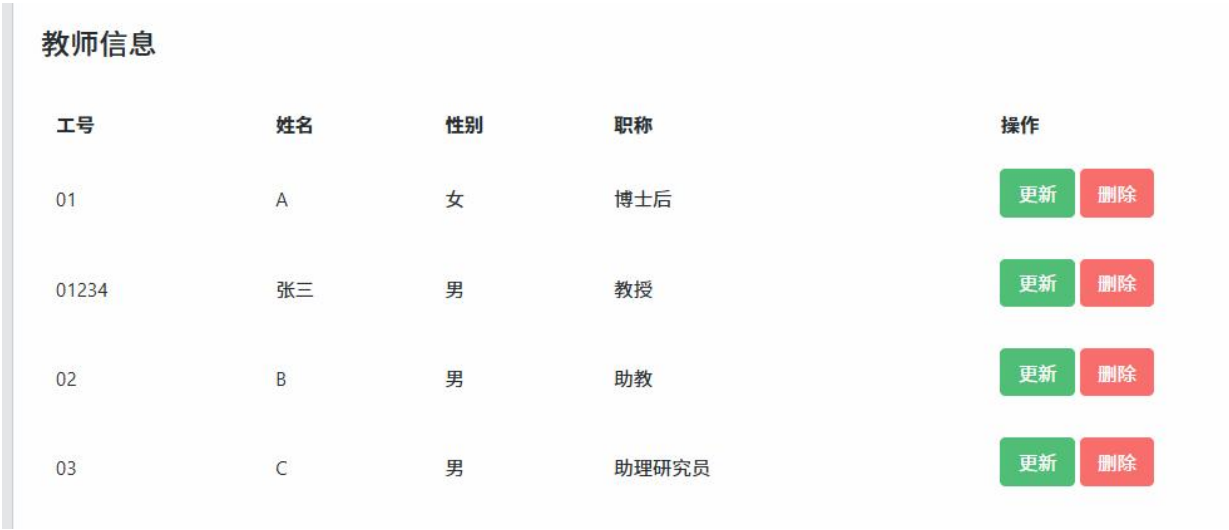
4.2 测试结果

教师：

添加教师：04 小明 男 副教授 结果如下

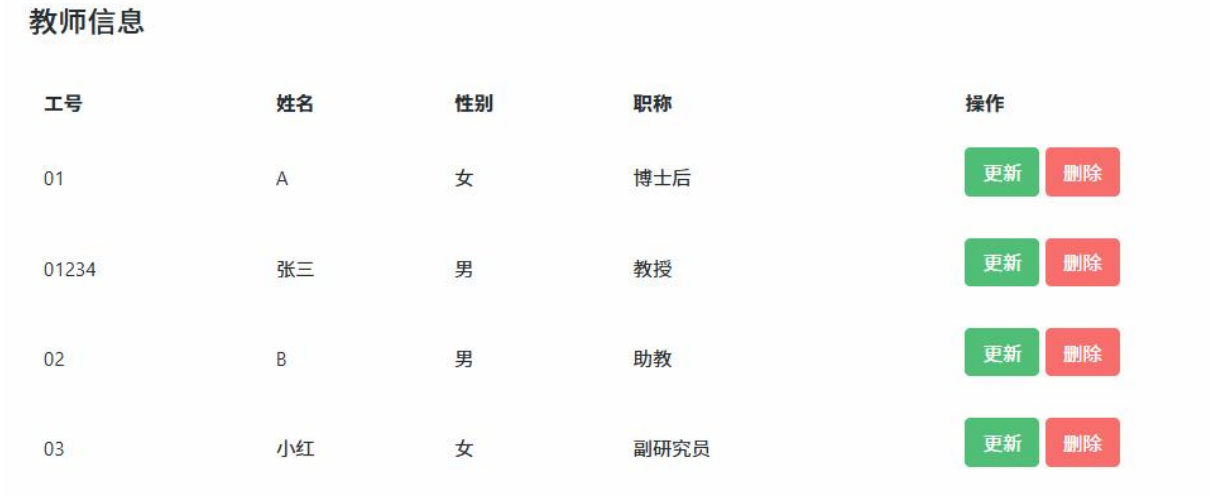
删除 04：




教师信息
工号 姓名 性别 职称 操作
01 A 女 博士后 更新 删除
01234 张三 男 教授 更新 删除
02 B 男 助教 更新 删除
03 C 男 助理研究员 更新 删除

修改 03：


更新表单
姓名
小红
性别
女
职称
副研究员
确认
取消


教师信息
工号 姓名 性别 职称 操作
01 A 女 博士后 更新 删除
01234 张三 男 教授 更新 删除
02 B 男 助教 更新 删除
03 小红 女 副研究员 更新 删除

查询男教师：

其余增删改均一致，查询演示在 4.1 已涉及。
特殊情况：增加项目金额时会改变总金额。

添加新老师：

此时项目变为：

项目管理

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	操作
1	AAAA	12355	市厅级项目	500.0	2023	2024	更新 删除
3	面向异构混合内存的 NVM 感知索引及自适应学习方法研究	国家自然科学基金委	国家级项目	280000.0	2021	2024	更新 删除
4	量子安全数据库系统关键技术研发及产业化	安徽省科技厅	省部级项目	2000000.0	2022	2025	更新 删除

更新和删除时，总金额也会改变。

4.3 实现中的难点问题及解决

难点和问题：

数据库很好实现，但是写实验之前对于前端简直就是一窍不通 QAQ，学习了很多 html 的相关文法，在网上找了一个好看的模版。并学习了是怎么用 python 来调用 html 和数据库。

在前后端传参的时候遇到了很多 bug，解决了很久。。

删除时要考虑很多情况，是否有冲突什么的。

总金额和总学时一开始是直接判断是否相等，导致添加修改分金额，分学时时需要手动修改总学时。。很笨的做法。询问了助教以后修改为通过每个金额来计算总金额。

导出 pdf 的时候，发现可以先实现一个网页，再把网页打印出来来实现 pdf 的导入功能。

5 总结与讨论

收获了相关开发经验，学习了很多前端网页后端数据库连接的相关知识。在实验实现和 debug 时都学到了很多。对我来说这算是一个大项目了，第一次写了大概 2000+行的代码，完成的时候很有成就感。