数据库案例设计以及实现要求（课程项目）

总体要求：

(1) 按照要求实现一个完整的系统

(2) 提示：可以使用 Navicat/ PowerDesigner 等工具完成对数据的逻辑视图和物理视图的设计

(3) 对于自拟题目，内容需要合理，工作量不能过小（可能酌情扣分），也不宜过大（否则可能完不成），可以提前跟老师沟通

(4) 预计第 9 或 10 周进行项目的期中展示（幻灯片汇报）

a) 介绍题目、需求分析、数据库设计、模型图、功能设计、模块划分等，每组展示时间 10 分钟（报告 8 分钟，提问 2 分钟）

需要在期中展示前一天将汇报 PPT上传至eLearning。PPT中需要包含题目、需求分析、数据库设计、模型图、功能设计、模块划分**等内容。**

(5) 完整系统的演示和提交（包括相关分析报告和设计文档、代码、使用说明书），具体时间再进行通知（一般是学期末）。期末的详细评分标准请参加评分标准文档。

(6) 可以参考现有项目的框架和实现，但严禁代码抄袭。

工具使用：

1. 数据库选择：关系型数据库，如MySQL（MariaDB）、SQL Server、PostgreSQL等。使用H2 Database、SQLite3等轻量级数据库需要说明理由。
2. 编程语言选择：**不限**编程语言、前端库和连接数据库的套件。你可以使用Flask、Django等组件直接生成前端页面，也可以使用HTML、WinForm等技术自行搭建。常见的连接器有：
   a) Python
      i. 可参见https://pymysql.readthedocs.io/en/latest/
   b) Golang
      i. 可参见https://pkg.go.dev/github.com/go-sql-driver/mysql
   c) C++ (Qt)
      i. 可参见https://doc.qt.io/qt-6/sql-driver.html
   d) Php
      i. 可参见http://docs.php.net/manual/zh/class.mysqli.php
   e) Java
      i. 可参见https://dev.mysql.com/downloads/connector/j/

可选题目目录：

一、 数据库应用系统

a) 文献管理系统
b) 微信扫码点名系统
c) 网络数据爬取管理系统
d) 学生成绩数据库
e) 电影行业数据库
f) 网上综合书店销售数据库
g) 航空票务数据库
h) SQL课程在线测评系统

二、 数据库管理系统

i) MiniSQL

可选题目

第一类 数据库应用系统

一、文献管理系统

项目目标：设计并实现一个文献管理系统，该系统能够存储、管理和检索学术文献信息，并支持文献的上传、自动信息解析、信息检索、批量处理以及数据的导入导出。

数据库设计

1. 设计一个符合第三范式（3NF）的数据库模型，至少应包含以下实体：
    - 论文（Paper）
    - 会议/期刊（Conference/Journal）
    - 作者（Author）
    - 单位（Institution）
    - 关键字（Keyword）
2. 实体属性应包括但不限于：
    - 论文：标题、摘要、发布时间、PDF 文件路径等。
    - 会议/期刊：名称、时间、地点等。
    - 作者：姓名、联系信息等。
    - 单位：名称、地址等。
    - 关键字：词汇等。
3. 设计必要的关系表以表示实体间的多对多关系，如：
    - 论文-作者关系
    - 论文-关键字关系
    - 作者-单位关系
4. 确定并实现适当的完整性约束，如外键约束、唯一性约束等。

功能实现（包括但不限于，下同）

1. 上传与解析：
    - 实现一个上传接口，允许用户上传 PDF 格式的论文文件。
    - 设计并实现一个解析模块，能够自动提取上传 PDF 中的论文标题、作者、单位、会议/期刊、时间、地点等信息。
2. 信息检索：
    - 提供一个搜索接口，允许用户通过作者姓名、单位等字段进行检索。
    - 支持高级搜索功能，例如范围搜索、关键字搜索等。
3. 批量处理：
    - 设计批量上传和解析的功能，允许用户一次性上传多篇论文并处理。
    - 实现批量导入和导出数据的功能，支持常见的数据格式，如 CSV、JSON 等。
4. 用户界面：
    - 设计一个用户友好的界面，使用户能够轻松地上传文献、进行检索和管理数据。
    - 确保界面简洁、直观，提供清晰的用户指引。

推荐采用的技术栈

1. 数据库：选择一个现代的数据库系统，如 MySQL、PostgreSQL 等。
2. 编程语言：可以使用任何适合的编程语言，如 Python、Java 等。
3. 自动解析：可以考虑使用文本挖掘技术、OCR 技术或其他机器学习算法来实现自动化解析。

可能的任务分解

1. 数据库设计与实现
2. PDF 上传与自动解析模块的开发或调用
3. 检索功能的实现
4. 批量处理的实现
5. 用户界面的设计与实现
6. 测试与调试

项目文档

1. 提交一个详细的数据库设计文档，包括 ER 图和表结构定义。
2. 编写代码文档，包括模块描述、接口说明和使用示例。
3. 准备一个用户手册，指导用户如何使用系统。

评估标准：项目将根据数据库设计的规范性、功能实现的完整性、用户界面的易用性和文档的详尽程度进行评估。

二、微信扫码点名系统

项目目标：设计并实现一个基于微信扫码的点名系统，用于自动化考勤管理。系统应能够处理学生的选课信息，管理学生基本信息，并进行出勤统计和分析。

数据库设计

1. 设计一个符合至少第二范式（2NF）的数据库模型，涵盖以下实体：
    - 学生（Student）
    - 课程（Course）
    - 出勤记录（Attendance）
2. 实体属性应包括但不限于：
    - 学生：姓名、学号、院系、专业、性别等。
    - 课程：课程名称、课程代码、开课院系等。
    - 出勤记录：日期、出勤状态（出勤、缺勤、请假）、扫码时间等。
3. 设计必要的关系表以表示实体间的多对多关系，如：
    - 学生-课程关系
4. 确定并实现适当的完整性约束，如外键约束、唯一性约束等。
5. 思考：老师的角色应当如何在数据库中体现？上述数据库模型是否需要增加？第二范式能满足要求吗？

功能实现

1. 考勤管理：
    - 实现一个扫码接口，允许学生使用微信扫描二维码进行签到。
    - 设计一个算法来确定扫码时间的有效性，并将出勤状态更新到数据库。
2. 学生信息管理：
    - 提供功能用于增加、减少和修改学生的基本信息。
    - 实现批量导入和导出学生信息的功能，支持常见的数据格式。
3. 出勤统计与查询：
    - 开发出勤统计功能，能够按照院系、性别等维度进行统计和分析。
    - 提供查询界面，允许查询学生的出勤记录和缺勤次数。
4. 系统设置：
    - 实现一个后台管理界面，允许管理员进行扫码有效时间的设置、学生信息的管理等。

推荐采用的技术栈

1. 数据库：选择一个合适的数据库系统，如 MySQL 等。
2. 编程语言：可以使用任何适合的编程语言，如 Python、JavaScript（Node.js）等。
3. 微信接入：熟悉微信开发者工具和 API，实现微信扫码功能。

可能的任务分解

1. 数据库设计与实现
2. 微信扫码接口的集成
3. 学生信息管理功能的实现
4. 出勤统计与查询功能的开发
5. 后台管理功能的实现
6. 系统测试与调试

项目文档

1. 提交一个详细的数据库设计文档，包括 ER 图和表结构定义。
2. 编写代码文档，包括模块描述、接口说明和使用示例。
3. 准备一个用户手册，指导用户如何使用系统，包括管理员和学生。

评估标准：项目将根据数据库设计的规范性、功能实现的完整性、系统易用性和文档的详尽程度进行评估。

三、网络数据爬取管理系统

项目目标：设计并实现一个网络数据爬取管理系统，该系统能够从互联网上抓取信息，并对这些信息进行存储、管理和检索。

数据库设计

1. 设计一个满足至少第三范式（3NF）的数据库模型，包含如下实体：
    - 网站（Website）
    - 网页（Webpage）
    - 内容（Content）
    - 图片（Image）
    - 数据源信息（DataSource）
2. 实体属性应包括但不限于：
    - 网站：域名、所属公司或组织、联系信息等。
    - 网页：网页地址、所属网站、抓取时间等。
    - 内容：文本内容、所属网页、关键字段等。
    - 图片：URL、所属网页、图片描述等。
    - 数据源信息：数据发布者、发布时间、原数据链接等。
3. 设计必要的关系表以表示实体间的一对多和多对多关系。
4. 确定并实现适当的完整性约束，如外键约束、唯一性约束等。

功能实现

1. 爬虫模块：
    - 设计并实现数据爬取模块，能够根据预定的规则抓取网站信息。
    - 支持爬取文本内容和图片，并将其分类存储。
2. 数据管理：
    - 实现数据的增加、删除、修改和查询功能。
    - 提供后台管理功能，对爬取策略进行配置和调整。
3. 检索功能：
    - 开发内容检索功能，支持通过关键字、发布者、时间等多种方式进行高级搜索。
    - 设计高效的索引策略，优化检索性能。
4. 用户界面：
    - 提供一个用户友好的界面，允许用户设定爬虫参数、查看爬取结果和进行数据检索。
    - 界面应该直观、易操作，并且能够显示爬取的统计信息。

推荐采用的技术栈

1. 数据库：选择现代的数据库系统，如 MySQL，可以结合 MongoDB、Elasticsearch 等非关系型数据库引擎，以支持大量数据的高效存储和检索。
2. 编程语言：推荐使用 Python，利用其强大的爬虫框架如 Scrapy。
3. 数据处理：对抓取的数据进行必要的清洗和格式化处理。

可能的任务分解

1. 数据库设计与实现。
2. 爬虫模块的开发与实现。
3. 数据管理后台的开发。
4. 检索功能与优化的实现。
5. 用户界面的设计与实现。
6. 系统的测试与调试。

项目文档

1. 提交一个详细的数据库设计文档，包括 ER 图和表结构定义。
2. 编写代码文档，包括模块描述、接口说明和使用示例。
3. 准备一个用户操作手册，指导用户如何使用系统进行数据爬取和检索。

评估标准：项目将根据数据库设计的规范性、爬虫模块的功能性、检索功能的有效性、用户界面的易用性和文档的详尽程度进行评估。

四、学生成绩数据库

项目目标：设计并实现一个用于管理学生成绩的数据库系统，包括学生、院系、专业、课程、教师及成绩等信息的存储、管理和查询。

数据库设计

1. 设计一个满足第三范式（3NF）的数据库模型，包含以下实体：
    - 学生（Student）
    - 院系（Department）
    - 专业（Major）
    - 课程（Course）
    - 教师（Teacher）
    - 成绩（Grade）
2. 实体属性应包括但不限于：
    - 学生：姓名、学号、身份证号、宿舍、家庭地址、电话、出生日期、性别、年级、专业、主修院系、辅修院系、学位等级、已修学分。
    - 院系：名称、代码、办公地点、电话。
    - 专业：名称、所属院系、各学位等级要求的学分。
    - 课程：课程名、课程说明、课程编号、学时、学分、学位等级、开课院系。
    - 教师：姓名、编号、所属院系。
    - 成绩：学生、课程、分数、学期。
3. 实体间关系：
    - 学生与院系：多对一关系，每个学生属于一个主修院系，可能属于一个辅修院系。
    - 院系与专业：一对多关系，每个院系开设多个专业。
    - 课程与院系：多对一关系，每门课程由一个院系开设。
    - 教师与院系：多对一关系，每名教师属于一个院系。
    - 教师与课程：多对多关系，每门课程可以由多名教师共同开设。
    - 学生与课程（成绩）：多对多关系，每名学生可以选修多门课程。
4. 定义各实体间的完整性约束和业务规则。

功能实现

1. 主页面：
    - 设计一个主页面，提供导航到所有功能页面的链接。
2. 院系和专业管理：
    - 实现院系信息的增加、删除、修改和查询功能。
    - 实现专业信息的增加、删除、修改和查询功能。
3. 课程管理：
    - 实现课程信息的增加、删除、修改和查询功能。
    - 提供课程列表，包括每门课程的学时、学分和开课院系。
4. 教师管理：
    - 实现教师信息的增加、删除、修改和查询功能。
    - 管理教师所授课程信息。
5. 学生管理：
    - 实现学生信息的增加、删除、修改和查询功能。
    - 管理学生的选课信息和成绩。
6. 成绩管理：
    - 实现成绩信息的录入、修改和查询功能。
    - 提供成绩统计和分析。

推荐采用的技术栈

1. 数据库：选择合适的数据库系统，如 MySQL 等。
2. 编程语言：推荐使用如 Python、Java、C#等，以支持 Web 端或桌面端的交互式功能。
3. 前端设计：使用现有框架，或 HTML、CSS、JavaScript 等用于设计用户友好的交互式界面。

可能的任务分解

1. 数据库架构设计与实现。
2. 后端逻辑的编写与实现。
3. 前端界面的设计与实现。
4. 系统集成和测试。

项目文档

1. 提交一个详尽的数据库设计文档，包括 ER 图、表结构定义及关系说明。
2. 编写代码文档，包括后端 API 说明、数据库访问逻辑等。
3. 提供用户操作文档，包括如何通过各页面管理和查询信息。

评估标准：项目将基于数据库设计的规范性、系统功能的完整性、用户界面的实用性以及文档的完备性进行评估。

注意事项：确保合理的用户权限管理，保护学生的个人信息。同时，系统应具备良好的数据验证机制，防止无效或错误数据的输入。

五、电影行业数据库

项目目标：设计并实现一个电影行业数据库系统，用于存储和管理电影、导演、演员、出品公司以及电影分类等信息。

数据库设计

1. 设计一个满足第三范式（3NF）的数据库模型，包含以下实体：
    - 电影（Movie）
    - 演员（Actor）
    - 导演（Director）
    - 出品公司（Production Company）
    - 电影类别（Genre）
    - 角色（Role）
    - 旁白（Narration）
2. 实体属性应包括但不限于：
    - 电影：名称、编号、发行年份、长度、出品公司、情节概要。
    - 演员：姓名、出生日期。
    - 导演：姓名、出生日期。
    - 出品公司：编号、名称、城市。
    - 电影类别：类别名称。
    - 角色：角色名称、关联演员、关联电影。
    - 旁白：内容、关联演员、关联电影。
3. 实体间关系：
    - 电影与出品公司：多对一关系，每部电影有一个出品公司。
    - 电影与类别：多对多关系，每部电影可以属于多个类别。
    - 电影与导演：多对多关系，每部电影可以由多名导演指导。
    - 电影与演员（通过角色）：多对多关系，每部电影有多名演员，每名演员可以出现在多部电影中。
    - 电影与旁白：一对多关系，每部电影可以有多个旁白。
4. 定义各实体间的完整性约束和业务规则。

功能实现

1. 主页面：
    - 设计一个主页面，提供导航到所有功能页面的链接。
2. 电影管理：
    - 实现电影信息的增加、删除、修改和查询功能。
    - 管理电影的类别、导演、演员以及旁白信息。
3. 导演和演员管理：
    - 实现导演和演员信息的增加、删除、修改和查询功能。
    - 管理导演和演员参与的电影及其角色。
4. 出品公司管理：
    - 实现出品公司信息的增加、删除、修改和查询功能。
    - 管理公司出品的电影列表。

推荐采用的技术栈

1. 数据库：选择合适的数据库系统，如 MySQL、PostgreSQL 等。
2. 编程语言：推荐使用如 PHP、Java、Node.js 等，以支持 Web 端的交互式功能。
3. 前端设计：使用现有框架，或 HTML、CSS、JavaScript 等用于设计用户友好的交互式界面。

可能的任务分解

1. 数据库架构设计与实现。
2. 后端逻辑的编写与实现。
3. 前端界面的设计与实现。
4. 系统集成和测试。

项目文档

1. 提交一个详尽的数据库设计文档，包括 ER 图、表结构定义及关系说明。
2. 编写代码文档，包括后端 API 说明、数据库访问逻辑等。
3. 提供用户操作文档，包括如何通过各页面管理和查询信息。

评估标准：项目将基于数据库设计的规范性、系统功能的完整性、用户界面的实用性以及文档的完备性进行评估。

注意事项：在设计数据库时，可以考虑未来可能的扩展性，比如增加在线观看、评论、评分等社交功能。同时，系统应具备良好的数据验证机制，防止无效或错误数据的输入。

六、网上综合书店销售数据库

项目目标：设计并实现一个综合性的数据库系统，用于管理连锁销售公司的门店、商品（包括图书）、供货商、客户（包括会员）、销售记录及统计数据。

数据库设计

1. 设计一个满足第三范式（3NF）的数据库模型，包含以下实体：
    - 门店（Store）
    - 供货商（Supplier）
    - 商品（Product）
    - 图书（Book），作为商品的一个子类别
    - 客户（Customer）
    - 会员（Member），作为客户的一个子类别
    - 销售记录（Sale）
    - 出版社（Publisher）
2. 实体属性应包括但不限于：
    - 门店：编号、名称、地点、电话、负责人。
    - 供货商：编号、名称、电话、e-mail。
    - 商品：条码、名称、计量单位、销售价格、类别（食品、服装、图书等）。
    - 图书：书号、书名、作者、定价、出版社、出版时间、版本号、译者。
    - 客户：类型（会员、非会员）、联系信息。
    - 会员：编号、姓名、联系电话、e-mail、地址。
    - 销售记录：日期、数量、金额、关联门店、关联商品、关联客户。
    - 出版社：编号、名称、联系电话、联系人、e-mail、地址。
3. 实体间关系：
    - 门店与商品：多对多关系，通过销售记录关联。
    - 供货商与商品：多对多关系，反映不同供货商可供应同一商品。
    - 商品与类别：一对多关系，每种商品属于一个类别。
    - 客户与销售记录：一对多关系，每个客户可以有多条销售记录。
    - 图书作为商品的特殊类别，与出版社、作者、译者具有多对多关系。
4. 定义各实体间的完整性约束和业务规则。

功能实现

1. 主页面：
    - 设计一个主页面，提供导航到所有功能页面的链接。
2. 门店与供货商管理：
    - 实现门店和供货商信息的增加、删除、修改和查询功能。
3. 商品与图书管理：
    - 实现商品和图书信息的增加、删除、修改和查询功能。
    - 管理图书的分类、出版社及作者信息。
4. 客户与会员管理：
    - 实现客户信息及会员信息的增加、删除、修改和查询功能。
5. 销售记录管理：
    - 实现销售记录的录入、修改和查询功能。
    - 提供销售数据的汇总和分析。

推荐采用的技术栈

1. 数据库：选择合适的数据库系统，如 MySQL、PostgreSQL 等。
2. 编程语言：推荐使用如 PHP、Java、Python 等，以支持 Web 端的交互式功能。
3. 前端设计：使用现有框架，或 HTML、CSS、JavaScript 等用于设计用户友好的交互式界面。

可能的任务分解

1. 数据库架构设计与实现。
2. 后端逻辑的编写与实现。
3. 前端界面的设计与实现。
4. 系统集成和测试。

项目文档

1. 提交一个详尽的数据库设计文档，包括 ER 图、表结构定义及关系说明。
2. 编写代码文档，包括后端 API 说明、数据库访问逻辑等。
3. 提供用户操作文档，包括如何通过各页面管理和查询信息。

评估标准：项目将基于数据库设计的规范性、系统功能的完整性、用户界面的实用性以及文档的完备性进行评估。

注意事项：确保合理的用户权限管理，保护客户和会员的个人信息。同时，系统应具备良好的数据验证机制，防止无效或错误数据的输入。

七、航空票务数据库

项目目标：设计并实现一个用于管理航空票务的数据库系统，包括航班、机场、票价、售票信息，以及航班销售的特殊需求如特价票和并发售票处理。

数据库设计

1. 设计一个满足第三范式（3NF）的数据库模型，包含以下实体：
    - 航班（Flight）
    - 机场（Airport）
    - 舱位定价（Cabin Pricing）
    - 售票记录（Ticket Sale）
    - 城市（City）
2. 实体属性应包括但不限于：
    - 航班：航班号、飞机机型、头等舱数、经济舱数、出发机场、经停机场、终到机场、每周飞行日。
    - 机场：机场代码、名称、所在城市。
    - 舱位定价：航班号、舱位等级（头等舱、经济舱等）、价格。
    - 售票记录：身份证号、姓名、出发城市、到达城市、日期、仓位、价格、航班号。
    - 城市：城市名称。
3. 实体间关系：
    - 航班与机场：多对多关系（考虑到经停机场），一个航班可以通过多个机场，一个机场可以服务多个航班。
    - 机场与城市：一对多关系，一个城市可以有多个机场。
    - 售票记录与航班：多对一关系，一个航班可以对应多条售票记录。
4. 定义各实体间的完整性约束和业务规则，如航班的容量限制和售票记录的一致性检查。

功能实现

1. 主页面：
    - 设计一个主页面，提供导航到所有功能页面的链接。
2. 机场和城市管理：
    - 实现机场信息的增加、删除、修改和查询功能。
    - 实现城市信息的增加、删除、修改和查询功能。
3. 航班和舱位定价管理：
    - 实现航班信息及其舱位定价的增加、删除、修改和查询功能。
    - 设计航班编排功能，允许为航班设置每周飞行日。
4. 售票信息管理：
    - 实现售票记录的录入、修改和查询功能。
    - 实现特价票和并发售票处理机制。
    - 实现一个购票系统。

推荐采用的技术栈

1. 数据库：选择合适的数据库系统，如 MySQL、PostgreSQL 等。
2. 编程语言：推荐使用如 Java、Ruby、Python 等，以支持 Web 端的交互式功能。
3. 前端设计：使用现有框架，或 HTML、CSS、JavaScript 等用于设计用户友好的交互式界面。

可能的任务分解

1. 数据库架构设计与实现。
2. 后端逻辑的编写与实现。

3. 前端界面的设计与实现。
4. 系统集成和测试。

项目文档

1. 提交一个详尽的数据库设计文档，包括 ER 图、表结构定义及关系说明。
2. 编写代码文档，包括后端 API 说明、数据库访问逻辑等。
3. 提供用户操作文档，包括如何通过各页面管理和查询信息。

评估标准：项目将基于数据库设计的规范性、系统功能的完整性、用户界面的实用性以及文档的完备性进行评估。

注意事项：在设计数据库时，确保考虑到航班的并发售票处理（锁），以及如何应对特价票等销售策略。同时还需要注意数据的安全性和隐私保护，特别是乘客的个人信息和支付信息。

八、SQL课程在线测评系统

项目目标：设计并实现一个 SQL 课程在线测评系统，用于管理学生和老师信息、题库、判题机制、竞赛和考试，以及题目的完成情况统计。

数据库设计

1. 设计一个满足第三范式（3NF）的数据库模型，包含以下实体：
    - 学生（Student）
    - 教师（Teacher）
    - 题目（Question）
    - 答案（Answer）
    - 测试用例（Test Case）
    - 考试（Exam）
    - 提交（Submission）
2. 实体属性应包括但不限于：
    - 学生：ID、姓名、邮箱、注册信息。
    - 教师：ID、姓名、邮箱、权限级别。
    - 题目：ID、描述、样例输入、样例输出、建表语句、题目难度。
    - 答案：题目 ID、正确 SQL 语句。
    - 测试用例：题目 ID、测试输入、预期输出。
    - 考试：ID、开始时间、结束时间、涉及的题目、总分。
    - 提交：ID、学生 ID、题目 ID、提交的 SQL 语句、提交时间、执行结果、得分。
3. 实体间关系应包括但不限于：
    - 教师与学生：一对多关系，一个老师可以管理多个学生。
    - 题目与答案：一对多关系，一个题目可以有多个答案（考虑到多种可能的正确答案）。
    - 考试与题目：多对多关系，一个考试可以包含多个题目，一个题目可以出现在多个考试中。
    - 学生与提交：一对多关系，一个学生可以有多次提交。
4. 定义各实体间的完整性约束和业务规则。

功能实现

1. 用户管理：
    - 实现学生和教师的注册、登录、信息管理。
    - 为教师和助教分配相应权限来管理题目和考试。
2. 题目管理：
    - 实现题目的增加、删除、修改和查询功能。
    - 管理题目的测试用例和标准答案。
3. 判题机制：
    - 实现自动判题功能，比对学生提交的 SQL 语句执行结果与标准答案的结果。
    - 设计机制判断非标准答案的正确性。
    - 采用安全措施以防止学生提交的 SQL 语句破坏数据库。
    - 设计超时防护机制，确保判题过程不会因长时间运行查询而受阻。
4. 考试和竞赛系统：
    - 实现考试创建、管理功能。
    - 为学生提供在线参与考试的界面，记录答题时间和分数。
    - 统计和展示考试成绩，实现成绩排名。
5. 统计分析：
    - 实现题目完成情况的统计分析功能。
    - 提供题目完成率和学生通过率的报告。

推荐的技术要求

1. 数据库：选择合适的数据库系统，如 MySQL、PostgreSQL 等。
2. 编程语言：推荐使用如 Python、Java 等，结合框架如 Django、Spring 等。
3. 前端设计：使用现有框架，或 HTML、CSS、JavaScript 等用于设计用户友好的交互式界面。
4. 安全性（不做强制要求）：确保 SQL 代码的安全执行，考虑 SQL 注入防护等安全措施。

可能的任务分解

1. 数据库架构设计与实现。
2. 后端逻辑的编写与实现。
3. 前端界面的设计与实现。
4. 判题逻辑的实现。
5. 系统集成和测试。

项目文档

1. 提交一个详尽的数据库设计文档，包括 ER 图、表结构定义及关系说明。
2. 编写代码文档，包括后端 API 说明、判题逻辑、数据库访问逻辑等。
3. 提供用户操作文档，包括如何通过界面进行题目解答、考试参与和结果查看。

评估标准：项目将基于数据库设计的规范性、系统功能的完整性、用户界面的实用性、判题逻辑的正确性以及文档的完备性进行评估。

第二类 数据库管理系统

九、MiniSQL

项目概述

MiniSQL 项目旨在从基础了解并实现数据库基本操作和底层优化。该项目设计让学生展现他们在数据库设计、优化及底层操作方面的知识，同时不依赖于现有的任何数据库管理系统，包括关系型数据库，如 MySQL 和 SQLite3 等，或者非关系型数据库，如 Redis 等。

目标

- 开发一个简单的 MiniSQL 数据库系统。
- 可以通过命令行界面（CLI）而非图形用户界面实现。
- 强调数据库的后端操作、存储结构、缓存和索引机制。

核心功能

1. **数据库创建和存储：**能够通过 CLI 命令或 SQL 语句创建新数据库，且不依赖于现有的 DBMS 平台。你可以使用任何的数据结构来管理你的数据库。
2. **表管理：**
    - 能够在数据库中创建带有至少两种数据类型（数值和字符串）的表。
    - 支持使用特定 CLI 或 SQL 命令删除表。
3. **数据操作：**
    - 插入：能够向表中插入有效数据。
    - 查询：从表中检索数据。
    - 过滤：根据列值过滤表数据的能力。
    - 连接：执行表间连接操作，并查询结果数据集。
4. **数据库删除：**实现使用特定命令删除整个数据库的功能。

额外目标

- **优化和结构**：通过正确实现存储结构、缓存设计和索引机制（如 B+树或哈希索引）获得额外分数。
- **文档和演示**：准备全面的文档并展示系统的功能。

性能评估

- 最初根据完成核心功能来评分（40%）。
- 根据优化技术和存储结构的实现获得额外分数，即正确实现课程中涉及的一个或多个存储结构、缓存设计和索引数据结构（如 B+树、哈希索引等），或者效率更高的结构。（10%）
- 通过与 MySQL 8 在类似条件下的效率比较来评估性能。根据相对于 MySQL 8 的性能达到特定阈值来分配分数。（20%）
    - 我们会用统一的数据和语句测试。当你的系统在相同平台达到 MySQL 8 运行效率的 10%（即时间不超过 MySQL 执行相同语句的 10 倍），你可以获得 5

分；当达到 MySQL 8 运行效率的 50%时，你可以再获得 10 分；当达到 MySQL 8 运行效率的 80%时，你可以再获得 5 分。

- 根据效率排名与选择同一项目的其他组竞争部分分数。（15%）
    - (选择本项目的小组数+1-你的排名)*15/选择本项目的小组数.

需要实现的具体命令

1. 创建数据库功能。
2. 创建包含各种数据类型的表。
3. 向表中插入数据。
4. 查询表数据。
5. 根据列条件过滤数据。
6. 实施表间的 JOIN 操作。
7. 删除表。
8. 删除数据库。

对应一下 SQL 语句:

```
CREATE DATABASE 数据库名;
CREATE TABLE 表名 (
  列 1 数据类型,
  列 2 数据类型,
  ...
);
INSERT INTO 表名 (列 1, 列 2, ...)
VALUES (值 1, 值 2, ...);
SELECT 列 1 FROM 表名;
SELECT 列 1 FROM 表名 WHERE 列 2=某值;
SELECT 表 1.列 1 FROM 表 1 INNER/ LEFT JOIN 表 2 ON 表 1.列 1=表 2.列 2;
DROP TABLE IF EXISTS table_name;
DROP DATABASE IF EXISTS database_name;
```

项目交付物

1. **基于 CLI 的数据库系统**：一个通过命令行界面访问的功能性 MiniSQL 数据库系统。（85%）
2. **文档**：涵盖系统设计、**优化技术**和用户指南的全面文档。（8%）
3. **演示**：突出展示关键功能、**优化方法**和性能指标的演示。（7%）

注意事项：当你使用脚本型语言如Python（不含PyPy和任何由C++、Rust包装的性能库）

来实现本系统时，我们认为你的参评时间是你的实际运行时间的一半。与MySQL类似，单个查询应当是单线程的。你可以使用相关的解析库来解析SQL语句，也可以不使用SQL语句但实现命令行相同的功能。