实验要求

1. 按照给出的关系数据库应用系统开发实践的指导书内容，实现第二部分（X1）到第六部分（X5）的开发任务。
2. 应用场景和数据库表，可选择实验指导书给出的教学管理系统和SCT数据库；也可以选择建立其他的应用、数据库和数据库表，但完成的任务需参照指导书或与指导书任务内容相当。

实验目的

本实验将以关系数据库的定义与操纵为目标，使学生更好地体验关系数据库系统管理大规模数据的基本思维。掌握SQL语言以及基于SQL语言的数据库应用程序的开发和运用。

实验过程

主要的数据结构

库表设计

模仿实验指导书的sct数据库，构造了以下的数据表。

```
course表
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| C#     | char(4)     | NO   | PRI | NULL    |       |
| Cname  | varchar(255) | YES  |     | NULL    |       |
| Credit | float       | YES  |     | NULL    |       |
| Chours | int         | YES  |     | NULL    |       |
| T#     | char(3)     | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
sc表
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| Score | float   | YES  |     | NULL    |       |
| S#    | char(8) | YES  | MUL | NULL    |       |
| C#    | char(4) | YES  | MUL | NULL    |       |
+-------+---------+------+-----+---------+-------+
student表
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| S#          | char(8)     | NO   | PRI | NULL    |       |
| Sname       | char(10)    | YES  |     | NULL    |       |
| Ssex        | char(2)     | YES  |     | NULL    |       |
| Sage        | int         | YES  |     | NULL    |       |
| SClass      | varchar(255) | YES  |     | NULL    |       |
| Saddr       | varchar(255) | YES  |     | NULL    |       |
| Sdepartment | varchar(255) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
```

与指导书相比student表多了SClass、Saddr、Sdepartment三个字段，基本与指导书一致。

记录生成方式

本实验我使用python的`fake`库来生成数据。

这是student表的数据生成函数，通过随机生成班级数来随机并有一定规律的生成学号，至于学生姓名、地址之类的则由fake函数自行生成

```python
def generate_student():
    student = ''

    for year in range(2018, 2024):
        annual_year_class = random.randint(95, 99)

        annual_year_student = 0
        for i in range(1, annual_year_class + 1):
            annual_year_class_student = random.randint(27, 30)
            # 编号前缀
            for j in range(1, annual_year_class_student + 1):
                student += f"INSERT INTO student VALUES('{year}{add_zero(annual_year_student, 4)}','" \
                           f"{fake.name()}','" \
                           f"{['男', '女'][i % 2]}'," \
                           f"{random.randint(15, 35)},'" \
                           f"{year}{add_zero(i, 2)}','" \
                           f"{fake.address()}','{major[random.randint(0, len(major) - 1)]}');\n"
                annual_year_student += 1
    with open("student.sql", mode="w+", newline="", encoding="utf-8", errors="ignore") as w:
        w.write(student)
```

这是课程表的生成方式，通过使用事先生成的课程数据，循环使用并根据规律生成对应的课号。

```python
def generate_class_data():
    #"INSERT INTO course VALUES('0000','数据库',3,48,'001');"
    course = []
    data = ''
    with open("select.txt", mode="r+", newline="", encoding="utf-8", errors="ignore") as r:
        course += eval(r.read())
    index = 0
    for i in course:
        index += 1
        num = 0
        for j in i:
            course_num = random.randint(1,24) * 8
            if num > 99:
                break
            data += f"INSERT INTO course VALUES('{add_zero(index, 2)}{add_zero(num,2)}','" \
                    f"{j}'," \
                    f"{course_num/16}," \
                    f"{course_num},'" \
                    f"{add_zero(random.randint(0,999),3)}');\n"
            num += 1
    with open("class.sql", mode="w+", newline="", encoding="utf-8", errors="ignore") as w:
        w.write(data)
    return data

```

最后的sc表则是通过读取导出的student表文件，按照学号规律，无重复的循环组合，生成数据。大概有60w左右。

```python
    with open("student.txt", mode="r+", newline="", encoding="utf-8", errors="ignore") as r2:
        data = r2.read().replace('\r','').split('\n')
        for i in data:
            if int(i.split(',')[0][:4]) <= 2020:
                c_num = random.sample(data_course,64)
                for j in range(64):
                    sql += f"INSERT INTO sc VALUES({random.uniform(30.0, 100.0)},'" \
                           f"{i.split(',')[0]}','{c_num[j][0]}');\n"
            else:
                length = int(i.split(',')[0][:4])-2020
                c_num = random.sample(data_course, length * 16)
                for j in range(length * 16):
                    sql += f"INSERT INTO sc VALUES({random.uniform(30.0, 100.0)},'" \
                           f"{i.split(',')[0]}','{c_num[j][0]}');\n"
            with open("sc.sql", mode="a+", newline="", encoding="utf-8", errors="ignore") as w:
                w.write(sql)
```

主要算法和系统实现

前端

我是使用前后端分离的技术，前端使用vue3进行页面的展示，而后端使用java实现。

前端主要的任务是与后端通信，获得数据并展示。通信的方式是ajax，用了axios。

```
import http from "@/utils/request";


export const getStudents =  (param)=>{
    return http.post('/students',param)
}

export const getCourses =  (param)=>{
    return http.post('/courses',param)
}
```

关于页面的展示，主要是将获取到的数据信息保存起来，给所有组件使用，当按下按钮时变执行更新，重新获取数据。

```
  </el-select>
  <el-container class="bottom">
    <table class="show" border="1px" cellspacing="0px" style="border-collapse:collapse">
      <tr>
        <td v-for="item in table.header" :key="item">{{ item }}</td>
      </tr>
      <tr v-for="item in table.data" :key="item[0]">
        <td v-for="item1 in table.header" :key="item1">{{ item[item1] }}</td>
      </tr>
    </table>
  </el-container>
</div>
```

后端

通过jdbc连接和使用数据库，主要有以下几步：

1. 注册驱动

   通过创建驱动对象告知JDBC，我们即将连接哪个厂商的数据库。

2. 获取数据库连接

   通过DriverManager.getConnection(url,user,pwd)获取连接

3. 获取数据库操作对象

   一个数据库连接对象可以创建多个数据库操作对象，通过conn.createStatement()获取数据库操作对象

4. 执行SQL语句

   通过数据库操作对象`statement.executeUpdate(sql)`编译执行SQL，JDBC编写SQL语句不需要以分号结尾。

5. 处理查询结果集

   第一种方式：根据字段下标获取，不管数据库表中的字段是什么类型，都以字符串方式取出，JDBC所有下标都是以1开始

   第二种方式：通过结果集中字段名称获取数据，该方式的程序更加健壮

   第三种方式：通过特定类型获取数据，该方式，明确知道字段的类型，可以节省类型转换花费的性能，该方式的程序更加健壮，性能更高

6. 释放资源

   JDBC驱动对象的创建、连接、获取数据库操作对象，执行sql、处理结果集等，都需要消耗时间，其中，这里面涉及到的ResultSet、Statement、Connection对象，使用完了需要释放，否则，造成资源浪费，严重的，服务器宕机。需要关闭ResultSet、Statement、Connection。

再就是通过jdbc获取数据库**元数据**，也就是sqlda：

```
Connection con ;
con = DriverManager.getConnection(url,userName,password);
DatabaseMetaData dbmd = con.getMetaData();
```

比如表的元数据

原型：

```
ResultSet DatabaseMetaData.getTables(String catalog,String schema,String
tableName,String []type)
```

此方法可返回结果集合ResultSet ， 结果集中有5列， 超出会报越界异常

功能描述：得到指定参数的表信息

参数说明：

- 参数:catalog:目录名称，一般都为空.
- 参数：schema:数据库名，对于oracle来说就用户名
- 参数：tablename:表名称
- 参数：type :表的类型(TABLE | VIEW)

最后就是利用springboot框架，将查询数据库的功能整合成接口的形式，暴露给前端使用。

```java
@PostMapping("/courses")
public R getCourses(@RequestBody CourseDto courseDto) {
    return new R(databaseService.getCourses(courseDto));
}

@PostMapping("/sc")
public R getSC(@RequestBody SCdto sc) {
    return new R(databaseService.getSC(sc));
}

@PostMapping("/any")
public R directInput(@RequestBody Map<String,String> map){
    return databaseService.Any(map);
}
```

实验感想

在进行该实验时，我发现以下几点感受：

1. 软件开发过程是一个复杂的过程。在开发该数据库管理系统时，我们需要进行需求分析、架构设计、编码实现、测试和部署等各个阶段的工作，这需要我们具备良好的软件开发和项目管理能力。
2. 数据库设计和实现是该实验的核心。在该实验中，我们需要设计和实现一个完整的数据库管理系统，包括数据库的建立、表的创建、数据的插入、查询和修改等功能。这需要我们掌握数据库设计和实现技术，并熟悉数据库管理系统和工具的使用。
3. 用户体验和界面设计是非常重要的。在该实验中，我们需要考虑用户体验和界面设计，以便用户能够轻松地使用该数据库管理系统。这需要我们具备良好的用户体验和界面设计能力，并了解用户的需求和使用习惯。

总之，该实验是一项非常有价值的数据库实践项目，它可以帮助我们深入理解数据库的设计、实现和管理技术，以及软件开发和用户体验等方面的知识和技能，为我们未来的工作和研究提供了良好的基础。