哈尔滨工业大学

<<数据库系统>>

实验报告

(2020 年度春季学期)

姓名:	吴昊
学号:	1170300527
学院:	计算机
教师:	史建焘

实验一 数据库应用系统的开发

一、实验目的

熟练掌握关系数据库系统的使用、SQL 语言；掌握在高级语言中通过嵌入式 SQL 对数据库进行操作，学习简单数据库系统的设计方法，包括数据库概要设计、逻辑设计。

二、实验环境

系统：Windows 10
数据库：MySQL
开发语言：JAVA
IDE：Intellij IDEA
其他框架：SpringBoot | Bootstrap | JdbcTemplate

三、实验过程及结果

系统 E-R 图，库表设计,包括完整性约束、索引、视图

1. 系统说明

该系统为学生信息管理系统，实体包括学院，系，教师，班级，学生，课程，教室等实体，学院设置有多个系，是一对一关系，系雇佣教室，设置班级，都是一对多关系，教师讲授课程为多对多关系，学生选课也为多对多关系，关系中包含分数。课程设置考点为一对一关系，考点与楼为一对多关系。数据库中还包含 user 表，保存登录用户名和密码。

可以进行添加修改教师学生信息，查看所有学生信息，查看学生平均成绩以及查看选课信息

实体如下：

学院(学院编号，学院名称)
系(系号，系名称，学院编号)
教师(教师编号，姓名，系号，年龄，工资，手机号，性别)
课程(课号，学分，名称)
班级（班号，系号，年级）
学生(学号，班号，姓名，年龄，性别，手机号，入学时间)

教室(教室编号，楼号)
楼(楼号，校区，楼名)

2. 表设计

对象 building @db_lab (mysql) - 表

保存 添加字段 插入字段 删除字段 主键 上移 下移

字段 索引 外键 触发器 选项 注释 SQL 预览

名	类型	长度	小数点	不是 null	虚拟	键
buildingId	int	11	0	☑	☐	1
buildingName	varchar	255	0	☐	☐	
campus	varchar	255	0	☐	☐	

对象 class @db_lab (mysql) - 表

保存 添加字段 插入字段 删除字段 主键 上移 下移

字段 索引 外键 触发器 选项 注释 SQL 预览

名	类型	长度	小数点	不是 null	虚拟	键
classId	int	1	0	☑	☐	1
departmentId	int	1	0	☑	☐	
grade	int	1	0	☑	☐	

对象 college @db_lab (mysql) - 表

保存 添加字段 插入字段 删除字段 主键 上移 下移

字段 索引 外键 触发器 选项 注释 SQL 预览

名	类型	长度	小数点	不是 null	虚拟	键
collegeId	int	11	0	☑	☐	1
collegeName	varchar	32	0	☑	☐	

对象 course @db_lab (mysql) - 表

保存 添加字段 插入字段 删除字段 主键 上移 下移

字段 索引 外键 触发器 选项 注释 SQL 预览

名	类型	长度	小数点	不是 null	虚拟	键
courseId	int	11	0	☑	☐	1
courseName	varchar	255	0	☑	☐	
credit	int	1	0	☑	☐	

对象 department @db_lab (mysql) - 表

保存 添加字段 插入字段 删除字段 主键 上移 下移

字段 索引 外键 触发器 选项 注释 SQL 预览

名	类型	长度	小数点	不是 null	虚拟	键
departmentId	int	11	0	☑	☐	1
departmentName	varchar	32	0	☑	☐	
collegeId	int	11	0	☑	☐	

对象 lecture @db_lab (mysql) - 表

保存 添加字段 插入字段 删除字段 主键 上移 下移

字段 索引 外键 触发器 选项 注释 SQL 预览

名	类型	长度	小数点	不是 null	虚拟	键
lectureId	int	11	0	☑	☐	1
teachId	int	11	0	☑	☐	
courseId	int	11	0	☑	☐	
roomId	int	11	0	☐	☐	

对象 room @db_lab (mysql) - 表

保存 添加字段 插入字段 删除字段 主键 上移 下移

字段 索引 外键 触发器 选项 注释 SQL 预览

名	类型	长度	小数点	不是 null	虚拟	键
roomId	int	11	0	☑	☐	1
buildingId	int	11	0	☐	☐	

对象 score @db_lab (mysql) - 表

保存 添加字段 插入字段 删除字段 主键 上移 下移

字段 索引 外键 触发器 选项 注释 SQL 预览

名	类型	长度	小数点	不是 null	虚拟	键
stuId	int	11	0	☑	☐	1
lectureId	int	11	0	☑	☐	2
score	int	11	0	☑	☐	

对象 student @db_lab (mysql) - 表

保存 添加字段 插入字段 删除字段 主键 上移 下移

字段 索引 外键 触发器 选项 注释 SQL 预览

名	类型	长度	小数点	不是 null	虚拟	键
stuId	int	10	0	☑	☐	1
stuName	char	32	0	☑	☐	
classId	int	1	0	☑	☐	
gender	char	3	0	☑	☐	
stuAge	int	1	0	☐	☐	
stuPhone	varchar	32	0	☐	☐	
entertime	timestamp	0	0	☑	☐	

对象 teacher @db_lab (mysql) - 表

保存 添加字段 插入字段 删除字段 主键 上移 下移

字段 索引 外键 触发器 选项 注释 SQL 预览

名	类型	长度	小数点	不是 null	虚拟	键
teachId	int	1	0	☑	☐	1
teachName	varchar	32	0	☑	☐	
teachAge	int	1	0	☐	☐	
teachBirth	timestamp	0	0	☐	☐	
teachSalary	int	1	0	☑	☐	
departmentId	int	1	0	☑	☐	
teachPhone	varchar	32	0	☐	☐	
gender	char	3	0	☐	☐	

3. 完整性约束

a) 实体完整性：主码属性必须非空且唯一。属性值非空。除 score 外实

体主码 ID 唯一，整数类型，插入数据时自增保证不会重复，score 中学号和课号共同作为主键，各实体全部主要属性属性非空。

b) 参照完整性：外键约束。例：当删除学生信息时，该学号对应的分数信息也会删除，删除教师信息时，授课，成绩信息会一起删除。

```
mysql> alter table score add CONSTRAINT stu_score FOREIGN KEY(stuId) REFERENCES student(stuId) on DELETE CASCADE;
Query OK, 5 rows affected (2.07 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> alter table lecture add CONSTRAINT teacher_course FOREIGN KEY(teachId) REFERENCES teacher(teachId) on DELETE CASC
ADE;
Query OK, 2 rows affected (1.38 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> alter table score add CONSTRAINT score_lecture FOREIGN KEY(lectureId) REFERENCES lecture(lectureId) on DELETE CAS
CADE;
Query OK, 5 rows affected (1.32 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

c) 范式要求：

i. 关系中每个分量都不可再分，详情看上面表，索引满足第一范式

ii. 每一非主属性完全函数依赖于候选键，除 score 外其他均只有编号一个候选键，通过编号可以确定每个属性，在 score 中需要需要学号和课号共同确定分数，没有其他属性，页满足第二范式

iii. 其中不包含传递函数依赖，所以满足第三范式

iv. 函数依赖都包含候选键，所以满足 Boyce-Codd 范式

4. 索引

每个表中主键均为索引，并为学生和教师中常用的班号和系号创建索引

信息 结果 1 剖析 状态

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
student	0	PRIMARY	1	stuId	A	6	(Null)	(Null)		BTREE
student	1	stu_classId	1	classId	A	3	(Null)	(Null)		BTREE
teacher	0	PRIMARY	1	teachId	A	8	(Null)	(Null)		BTREE
teacher	1	teach_departn	1	departmentId	A	2	(Null)	(Null)		BTREE

5. 视图

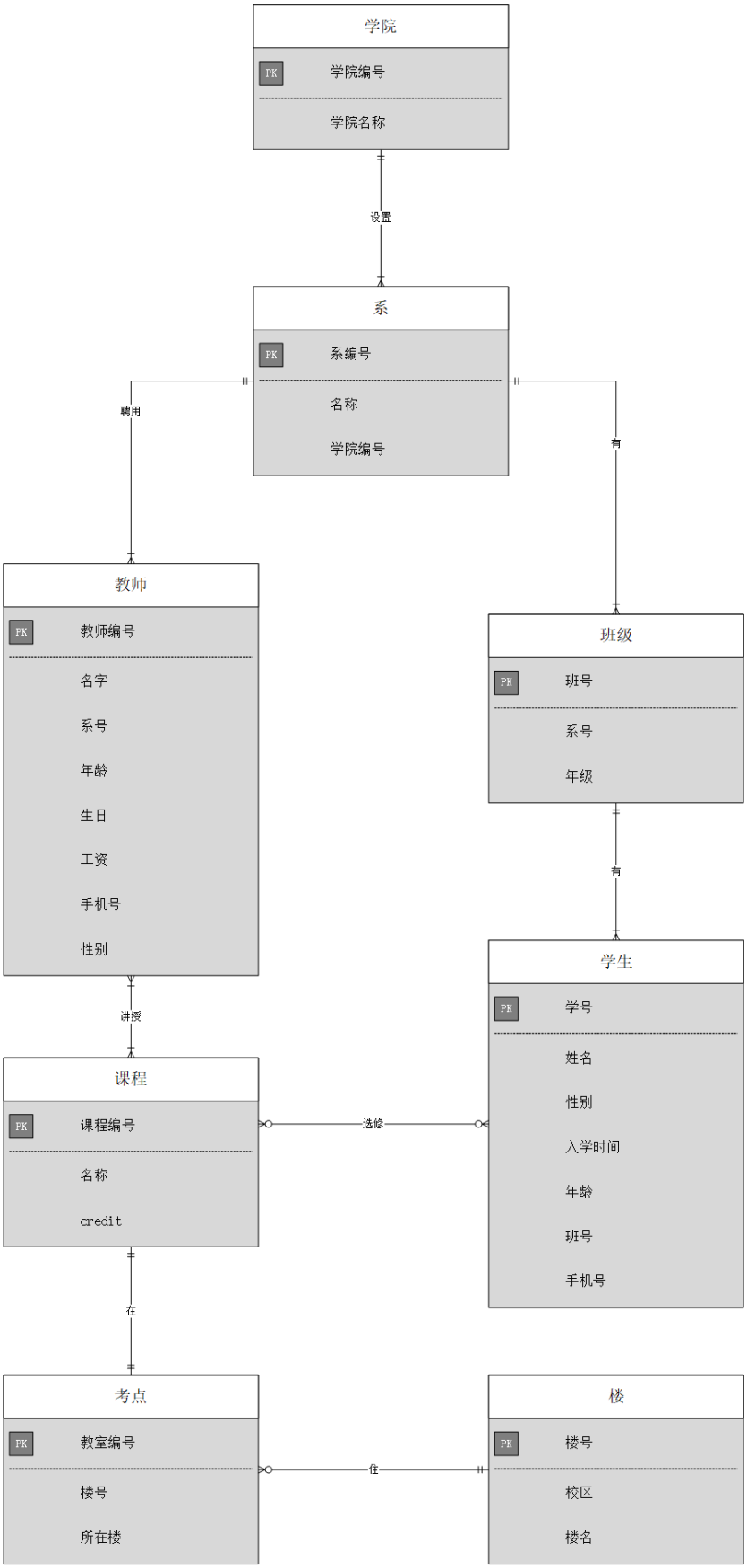
为常用的学生查询创建视图

```
select `student`.`stuId` AS `stuId`,`student`.`stuName` AS `stuName`,`student`.`classId` AS `classId`,`college`.`collegeName` AS `collegeName`,
`department`.`departmentName` AS `departmentname`,`student`.`gender` AS `gender`,`student`.`stuAge` AS `stuAge`,`student`.`entertime` AS `entertime`,
`class`.`grade` AS `grade`,`student`.`stuPhone` AS `stuPhone` from (((`student` join `class` on((`student`.`classId` = `class`.`classId`))) join
`department` on((`class`.`departmentId` = `department`.`departmentId`))) join `college` on((`department`.`collegeId` = `college`.`collegeId`)))
```

开始事务 文本 筛选 排序 导出

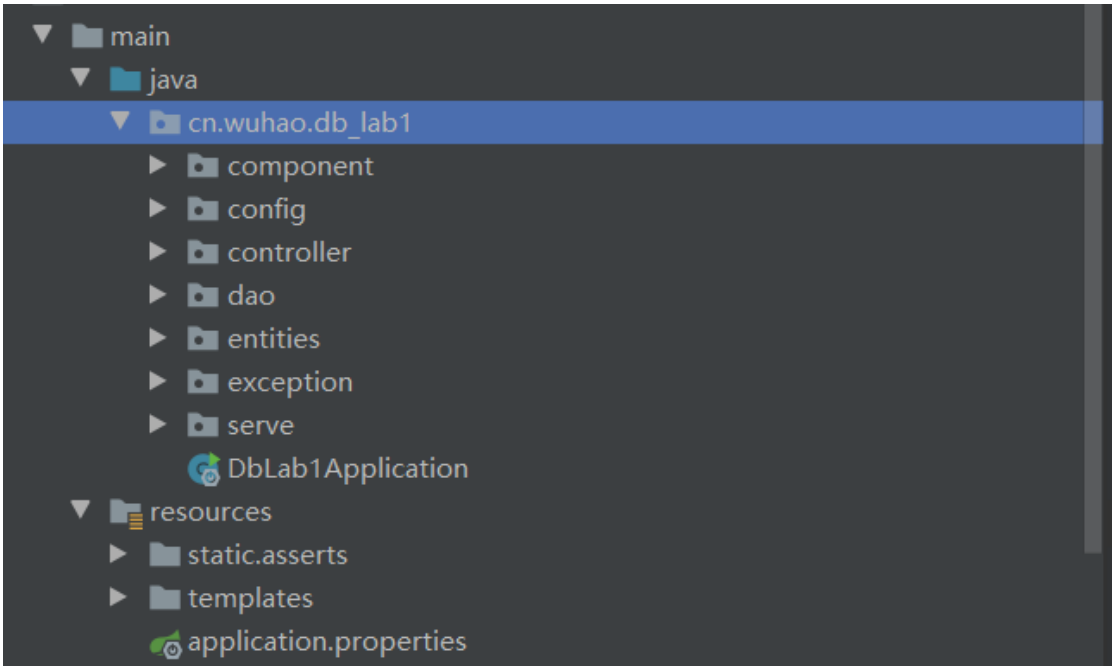
stuId	stuName	classId	collegeName	departmentname	gender	stuAge	entertime	grade	stuPhone
1	吴昊	1	计算机	计算机工程	男	22	2020-04-14 20:36:12	3	13144516882
11	empty	1	计算机	计算机工程	男	15	2020-04-17 09:32:58	3	13144516666
6	new	2	计算机	计算机工程	女	19	2020-04-17 11:43:35	3	13144516666
7	hello	2	计算机	计算机工程	男	22	2020-04-02 11:43:35	3	110112119114
10	new	2	计算机	计算机工程	女	22	2020-04-18 03:55:16	3	15045166541
12	new	2	计算机	计算机工程	女	20	2020-04-18 03:54:05	3	13144516666
3	小陈	3	计算机	信息安全	女	24	2020-04-14 20:37:06	3	165423542354
4	冯帅	3	计算机	信息安全	男	20	2020-04-14 11:59:55	3	15045166541
8	new	3	计算机	信息安全	女	16	2020-04-02 11:43:35	3	15045166541

6. E-R 图


学院
PK
学院编号
学院名称
设置
系
PK
系编号
名称
学院编号
聘用
有
教师
PK
教师编号
名字
系号
年龄
生日
工资
手机号
性别
班级
PK
班号
系号
年级
有
讲授
学生
PK
学号
姓名
性别
入学时间
年龄
班号
手机号
课程
PK
课程编号
名称
credit
选修
在
考点
PK
教室编号
楼号
所在楼
住
楼
PK
楼号
校区
楼名

简述系统设计（插入、删除、查询、其他功能），有必要的解图。

1. 类分布：



Component 为组件包，其中包括登录拦截器

Config 为配置包，包括重定向，添加拦截器等信息

Controler 负责请求转发，接受页面过来的参数， 传给 Service 处理，接到返回值，再传给页面

DAO 层实现数据访问，和数据库直接操作，实现的都为原子操作

Serve 层作为服务器，对 dao 层进行封装，实现满足需求的复杂的操作

Entities 包含实体类，与表对应

Exception 为异常包，遇到空，或相同数据时抛出异常，进入 error 界面显示错误信息

2. 登录与注销：

配置拦截器，进入除登录界面外都需要经过拦截器，判断 session 中是否有用户信息，没有用户信息则重定向到登录界面

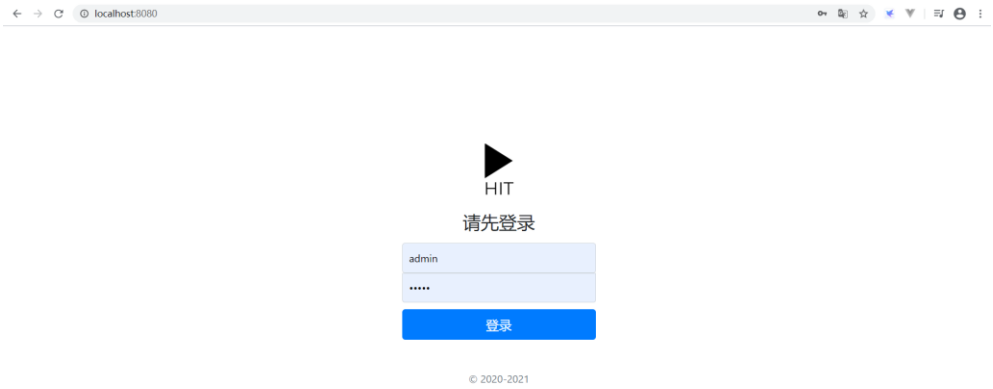
```
@Override
public void addInterceptors(InterceptorRegistry registry) {
    registry.addInterceptor(new LoginHandlerInterceptor()).addPathPatterns("/**")
            .excludePathPatterns("/index.html", "/", "/login", "/webjars/**", "/asserts/**");
}
```

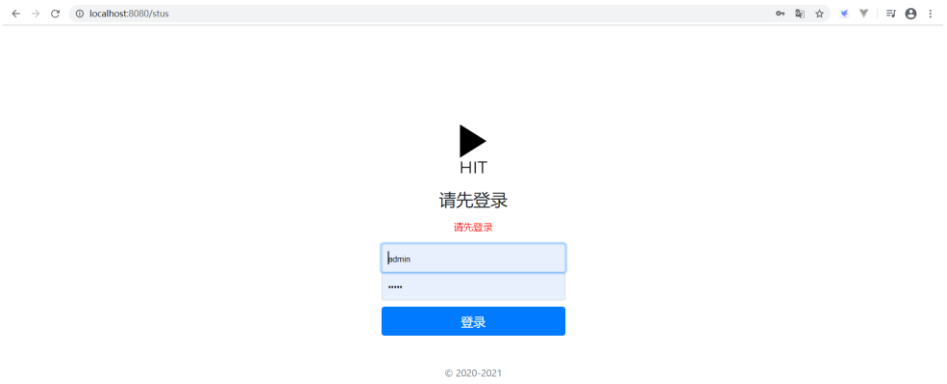
```
@Override
public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throw
    Object user = request.getSession().getAttribute( s: "loginUser");
    if (user == null) {
        request.setAttribute( s: "msg",  o: "请先登录");
        request.getRequestDispatcher( s: "/index.html").forward(request, response);
        return false;
    } else {
        return true;
    }
```

查询 user 表中所有用户，判断是否包含登录用户（重写 equls，hashcode），如果正确重定向到主界面，都在添加提示信息

```
List<User> users = jdbcTemplate.query( sql: "select * from user", new BeanPropertyRowMapper<User>(U
if (users.contains(user)) {
    session.setAttribute( s: "loginUser", username);
    return "redirect:/main.html";
}
else {
    map.put("msg", "用户名或密码错误");
    return "index";
}
```

运行截图：





3. 教师信息管理：

查询：通过自然连接将教师，系学院连接起来以获得教师的所有信息

```
List<Teacher> teachers = jdbcTemplate.query( sql: "SELECT teachId, teachName, collegeName," +
                " departmentName, teachSalary, teachAge, teachBirth, teachPhone, gender, departmentId FROM te
                " NATURAL JOIN department NATURAL JOIN college;",
        new BeanPropertyRowMapper<Teacher>(Teacher.class));
```

添加：将 teacher 中对应 teacher 表中的所有信息插入到表中。

```
ic void add(Teacher teacher) {
jdbcTemplate.update( sql: "insert into teacher(teachName, teachAge, teachBirth, teachSalary, departmentId," +
                " teachPhone, gender) values (?,?,?,?,?,?,?)",teacher.getTeachName(), teacher.getTeachAge(),
        teacher.getTeachBirth(), teacher.getTeachSalary(), teacher.getDepartmentId(), teacher.getTeachPhone()
        teacher.getGender());
```

查出所有系的信息，并将其传入添加界面以实现选择专业的下拉列表

```
@GetMapping("/teacher")
public String toAddPage(Map<String, List<Department>> map) {
    List<Department> departments = departmentServe.getAll();
    map.put("departments", departments);
    return "/teacher/add";
}
```

```
List<Department> departments = jdbcTemplate.query( sql: "select departmentId, departmentName, collegeName from
        "department natural join college", new BeanPropertyRowMapper<Department>(Department.class));
return departments;
```

首先判断想要添加的教师姓名，性别等信息是否为空，如果是空抛出错误信息，再判断输入的教师是否已经存在（重写 equals 和 hashcode）并抛出错误信息，遇到运行时异常会自导跳到请求码为 500 的错误页面

```
@PostMapping("/teacher")
public String addTeacher(Teacher teacher) {
    List<Teacher> teachers = teacherServe.getAll();
    if (teacher.getTeachName().isEmpty())
        throw new EmptyException("请填写教师姓名");
    else if (teacher.getGender() == null)
        throw new EmptyException("请勾选教师性别");
    else if (teachers.contains(teacher))
        throw new SameException("添加教师已存在");
    teacherServe.add(teacher);
    return "redirect:/teachers";
}
```

运行界面：

查询

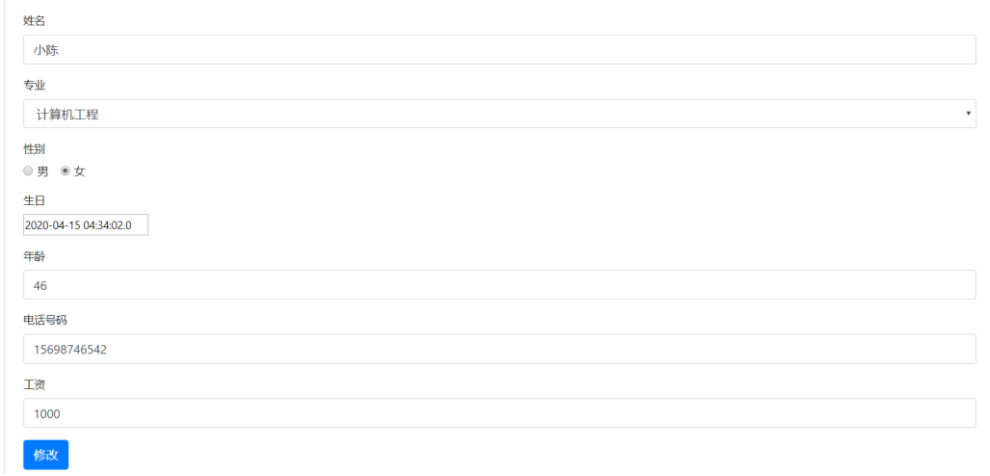
教师信息

ADD TEACHER

工号	姓名	学院	专业	性别	年龄	生日	手机号	工资	操作
1	战神	计算机	计算机工程	男	50	2020-04-15 04:35:15.0	15798453215	66666	编辑 删除
2	小陈	计算机	计算机工程	女	46	2020-04-15 04:34:02.0	15698746542	1000	编辑 删除
4	小孙	计算机	计算机工程	男	45	2020-04-14 20:02:45.0	15789456324	100000	编辑 删除
5	王天	计算机	计算机工程	女	45	2020-04-14 20:02:45.0	15945632154	233333	编辑 删除
12	小陈	计算机	计算机工程	男	45	2020-04-17 18:57:57.0	15789456324	100000	编辑 删除
13	小李	计算机	计算机工程	男	45	2020-04-18 11:26:09.0	15789456324	100000	编辑 删除
3	小李	计算机	信息安全	女	47	2020-04-15 04:34:30.0	15846528761	32000	编辑 删除
7	小李	计算机	信息安全	男	56	2020-04-17 18:57:57.0	15478965412	123456	编辑 删除
8	小李	计算机	信息安全	男	56	2020-04-17 18:57:57.0	15478965412	123456	编辑 删除
10	小李	计算机	信息安全	男	45	2020-04-18 11:26:09.0	15789456324	100000	编辑 删除
11	小苏	计算机	信息安全	女	45	2020-04-18 11:26:09.0	15789456324	100000	编辑 删除

添加

修改：根据要修改的 id 自动回显当前属性



添加相同教师

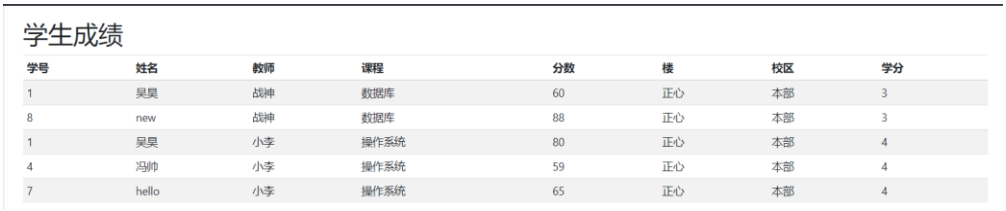
添加教师已存在

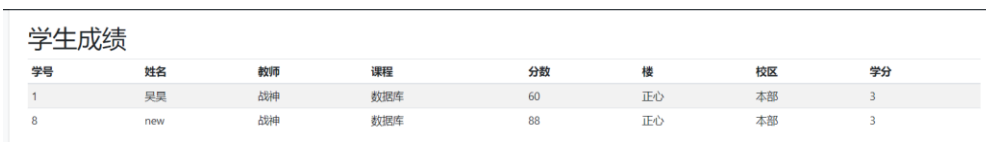
姓名为空

性别为空

删除：点击删除按钮进行删除 11 号

删除教师对应学生分数选课也会被删除：删除 3 号教师小李





4. 学生信息管理：

主要实现与教师信息相似，相似部分就不再进行说明

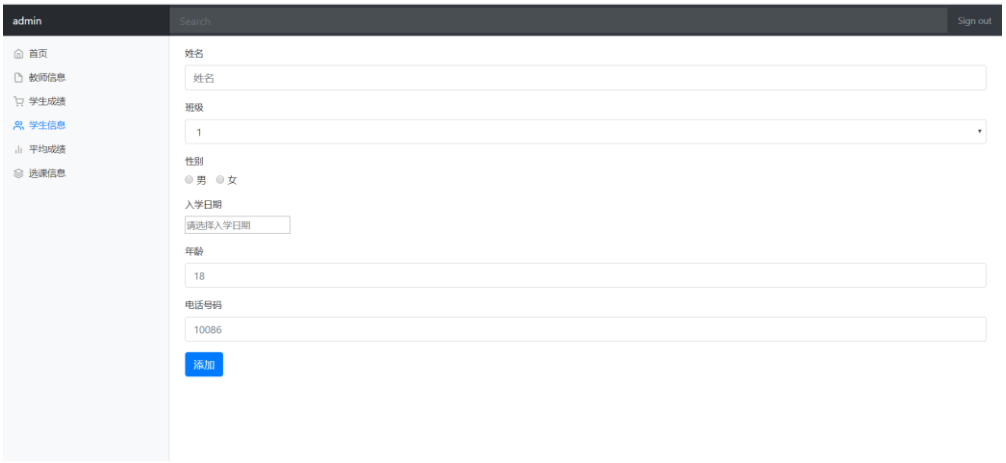
查询；由于创建了视图，这里直接查询视图

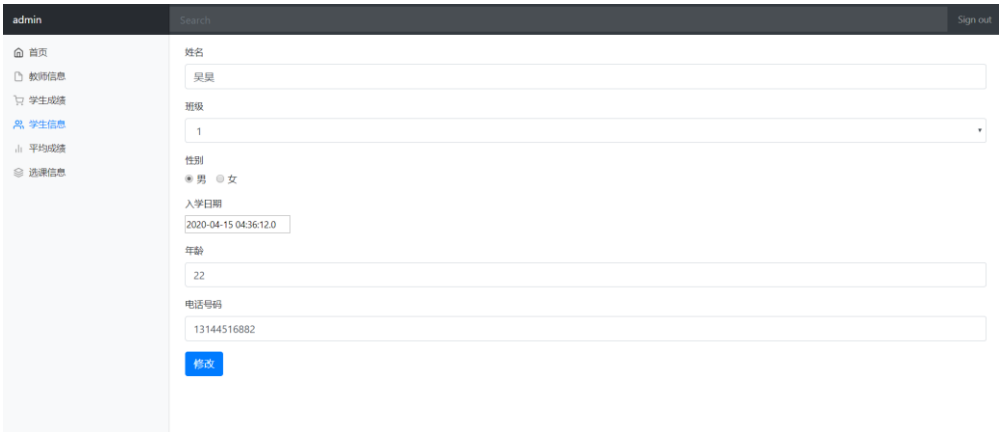
```
public List<Student> getAll() {
    List<Student> students = jdbcTemplate.query( sql: "SELECT * FROM stuInfo;",
            new BeanPropertyRowMapper<Student>(Student.class));
    return students;
}
```

添加：选择班级的下拉列表，需要查询班级信息

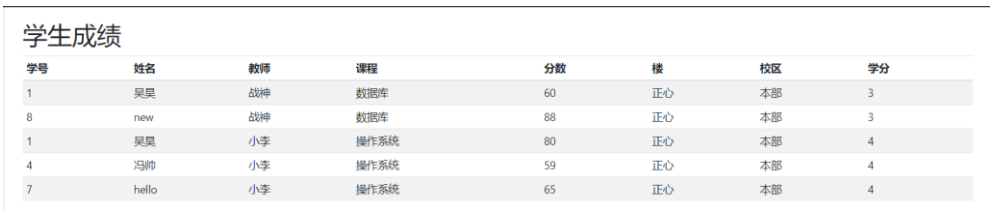
```
lic List<Class> getAll() {
 List<Class> classes = jdbcTemplate.query( sql: "SELECT classId, departmentname, collegeName, grade" +
         " FROM class NATURAL JOIN department, college", new BeanPropertyRowMapper<Class>(Class.class));
 return classes;
```

运行截图：





学生删除后，对应的选课分数等信息会一起删除：删除学号 6 的学生



5. 所有成绩：

通过自然连接，再学生，课程，教师，授课，考点，楼中查询成绩信息

```
List<Score> scores = jdbcTemplate.query( sql: "SELECT stuId, stuName, teachName, courseName, score, buildingName
        " campus, credit FROM score NATURAL JOIN student NATURAL JOIN lecture INNER JOIN teacher ON" +
        " lecture.teachId = teacher.teachId NATURAL JOIN course NATURAL JOIN room NATURAL JOIN" +
        "  building ", new BeanPropertyRowMapper<Score>(Score.class));
```

6. 学生平均成绩：

这里基于学号进行分组查询，并使用 AVG 函数获取每个学生的平均分，对平均分用 AS 进行重命名为 score 便于封装回平均分的类中，并按分数降序排序

```
List<StuScore> stuScores = jdbcTemplate.query( sql: "SELECT stuId, stuName, classId, departmentName," +
        " collegeName,  AVG(score)  AS score FROM score NATURAL JOIN student NATURAL JOIN class NATURAL JOIN"
        " department  NATURAL JOIN college GROUP BY stuId ORDER BY AVG(score) DESC",
        new BeanPropertyRowMapper<StuScore>(StuScore.class));
```

学生成绩

学号	姓名	班级	专业	学院	平均分
8	new	3	信息安全	计算机	88.0
6	new	2	计算机工程	计算机	80.0
1	吴昊	1	计算机工程	计算机	70.0
7	hello	2	计算机工程	计算机	65.0
4	冯帅	3	信息安全	计算机	59.0

7. 选课信息：

右上角为课程的下拉列表，通过 CourseDao 中的课程查询来实现

直接查询所有课程并传到 list 界面即可

```
lic List<Course> getAll() {
 List<Course> courses = jdbcTemplate.query( sql: "select * from course", new BeanPropertyRowMapper<Course>(Cours
 return courses;
```

使用了子查询，在学生的信息中查询学号在（分数连接讲授进一步连接到课程的编号，查询选了该课程的学号）中的学生信息

```
lic List<Elective> getElective(Integer id) {
List<Elective> electives = jdbcTemplate.query( sql: "SELECT stuId, stuName, classId, departmentName," +
        " collegeName,stuAge, stuPhone FROM student  NATURAL JOIN class  NATURAL JOIN department" +
        "  NATURAL JOIN college WHERE stuId in (SELECT stuId FROM score NATURAL JOIN lecture" +
        " NATURAL JOIN course WHERE courseId = ?)", new BeanPropertyRowMapper<Elective>(Elective.class), id);
return electives;
```

选课信息

操作系统 查询

学号	姓名	班号	专业	学院	年龄	手机号
1	吴昊	1	计算机工程	计算机	22	13144516882
7	hello	2	计算机工程	计算机	22	110112119114
4	冯帅	3	信息安全	计算机	20	15045166541

四、实验心得

本次实验实现了一个 web 应用的数据库管理系统，完成的对学生，教师等信息的增删改查等操作，通过具体时间对数据库有了新的认识，主要为了界面花费了不少时间，页更见便于操作和演示（最主要时好看）实际实现的数据库操作要比页面上显示的查询做的要多，比如添加学生，教师，查询选课信息时看到的的下拉列表，实际也是通过查询的结果。

总的来说，通过本次实验，对数据库的操作不只停留在表面上，虽然过程中遇到很多问题，但当问题得到解决时还是收获巨大的。