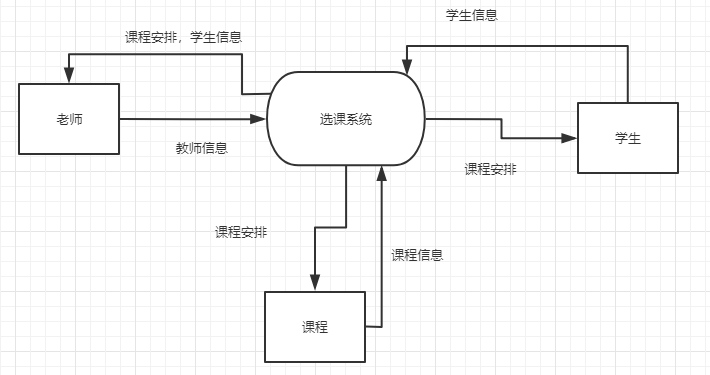
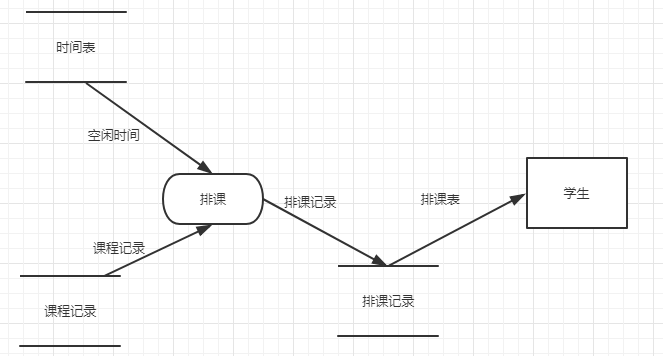
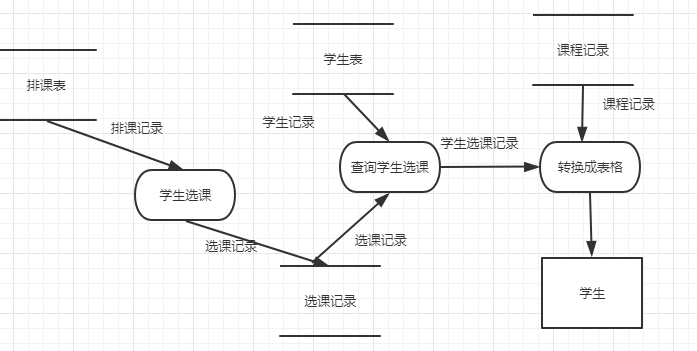
数据库设计



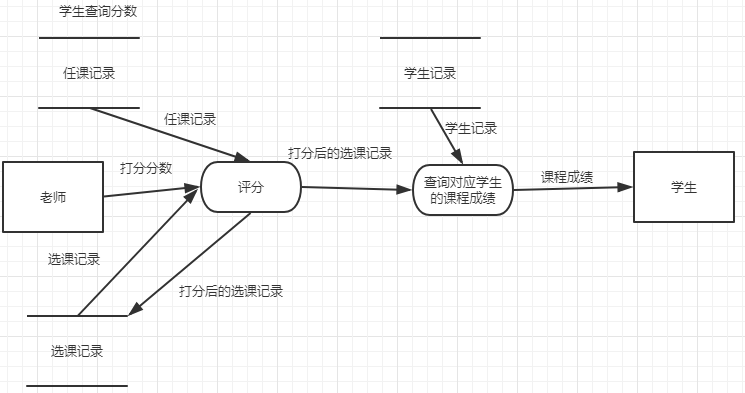
排课数据流图



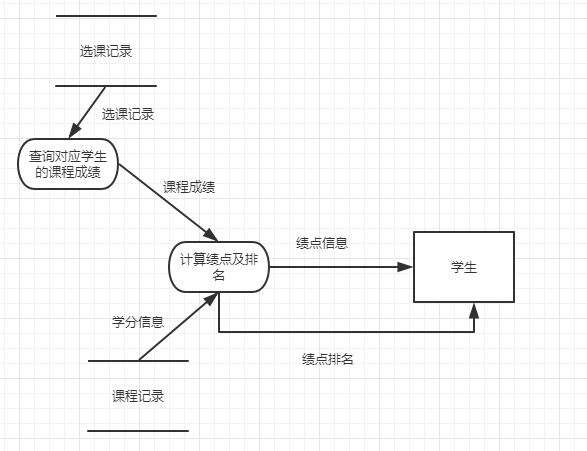
选课数据流图



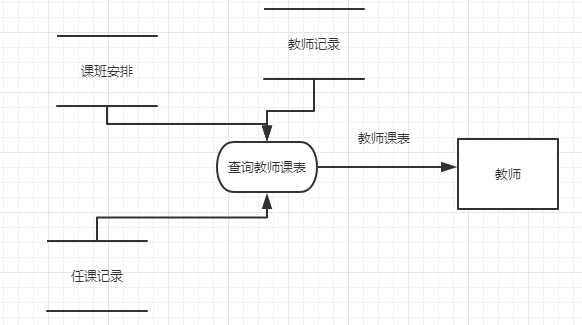
学生查询成绩数据流图



学生查询绩点和排名数据流图



教师查询个人课表数据流图



数据字典：

**Student**

名称：sno

别名：学号

意义简述：学生唯一标识

取值类型：int

长度：10

样式：1806100118

取值范围：11

与其它数据项的关系：

名称：name

别名：姓名

意义简述：学生名字

取值类型：varchar

长度：20以内

样式：小周小舟

取值范围：20

与其它数据项的关系：

名称：class

别名：班级

意义简述：学生所在班级

取值类型：varchar

长度：7

样式：计科186

取值范围：20

与其它数据项的关系：

名称：gpa

别名：绩点

意义简述：评估学习成绩的一种标准

取值类型：float

长度：3

样式：3.45

取值范围：

与其它数据项的关系：

名称：pwd

别名：密码

意义简述：学生登录的密码

取值类型：varchar

长度：20以内

样式：2090578469aa

取值范围：50

与其它数据项的关系：

**Teacher**

名称：tno

别名：教工号

意义简述：教师唯一标识

取值类型：int

长度：8

样式：20210102

取值范围：11

与其它数据项的关系：

名称：name

别名：教师姓名

意义简述：

取值类型：varchar

长度：20以内

样式：老古

取值范围：20

与其它数据项的关系：

名称：pwd

别名：密码

意义简述：教师登录的密码

取值类型：varchar

长度：20以内

样式：2090578469aa

取值范围：50

与其它数据项的关系：

名称：identity

别名：身份

意义简述：老师身份（课程负责人、系主任

取值类型：varchar

长度：5

样式：课程负责人

取值范围：10

与其它数据项的关系：

**Course**

名称：cno

别名：课程号

意义简述：课程唯一标识

取值类型：int

长度：9

样式：181700805

取值范围：11

与其它数据项的关系：

名称：name

别名：名字

意义简述：课程名字

取值类型：varchar

长度：20以内

样式：生命科学与人类生活

取值范围：20

与其它数据项的关系：

名称：credit

别名：学分

意义简述：课程学分

取值类型：tinyint

长度：1

样式：2|3|4三种

取值范围：4

与其它数据项的关系：

名称：type

别名：课程类型

意义简述：指明是选修课还是必修课

取值类型：varchar

长度：2

样式：选修|必修

取值范围：20

与其它数据项的关系：

名称：status

别名：课程状态

意义简述：课程是否结课

取值类型：varchar

长度：2

样式：结课|选课|上课

取值范围：10

与其它数据项的关系：

名称：class\_hours

别名：学时

意义简述：对应学分

取值类型：tinyint

长度：2

样式：32|48|64

取值范围：5

与其它数据项的关系：

**student\_class（选课表**

**要加个选课号（把教工号和课程号替换掉？）因为一门课不同阶段可能有不同的老师来上，**

**查找老师的话通过teacher\_course表的教学班号查询**

**查找课程通过teaching\_class的教学班号查询**

名称：curricula\_variable

别名：教学班号

意义简述：教学班唯一标识

取值类型：varchar

长度：18

样式：181700805-2（cno+开班数

取值范围：20

与其它数据项的关系：

名称：sno

别名：学号

意义简述：学生唯一标识

取值类型：int

长度：10

样式：1806100118

取值范围：11

与其它数据项的关系：

名称：grade

别名：课程成绩

意义简述：学生在该门课取得的成绩

取值类型：tinyint

长度：3

样式：98

取值范围：5

与其它数据项的关系：

名称：capacity

别名：课程容量

意义简述：最大报名人数

取值类型：tinyint

长度：3

样式：100

取值范围：11

与其它数据项的关系：

名称：enrollment

别名：选课人数

意义简述：已选课人数

取值类型：tinyint

长度：3

样式：98

取值范围：11

与其它数据项的关系：

**teacher\_course(任课表**

名称：tno

别名：教工号

意义简述：教师唯一标识

取值类型：int

长度：8

样式：20210102

取值范围：11

与其它数据项的关系：

名称：time\_period

别名：上课时间段

意义简述：说明是哪几周、周几、哪个时间段上课

取值类型：varchar

长度：20

样式：tue|9-11

取值范围：100

与其它数据项的关系：

名称：curricula\_variable

别名：教学班号

意义简述：教学班唯一标识

取值类型：varchar

长度：18

样式：181700805-1（cno+开班数

取值范围：20

与其它数据项的关系：

**course\_class(课程安排表**

名称：cno

别名：课程号

意义简述：课程唯一标识

取值类型：int

长度：9

样式：181700805

取值范围：11

与其它数据项的关系：

名称：curricula\_variable

别名：教学班号

意义简述：教学班唯一标识

取值类型：varchar

长度：11

样式：181700805-2（cno+开班数

取值范围：20

与其它数据项的关系：

名称：time\_period

别名：上课时间段

意义简述：说明是哪几周、周几、哪个时间段上课

取值类型：varchar

长度：20

样式：tue|9-11

取值范围：100

与其它数据项的关系：

**teaching\_class**

名称：curricula\_variable

别名：教学班号

意义简述：教学班唯一标识

取值类型：varchar

长度：11

样式：181700805-2（cno+开班数

取值范围：20

与其它数据项的关系：

**group**

名称：gno

别名：课程组号

意义简述：课程组唯一标识

取值类型：int

长度：4

样式：1806

取值范围：5

与其它数据项的关系：

名称：cno

别名：课程号

意义简述：课程唯一标识

取值类型：int

长度：9

样式：181700805

取值范围：11

与其它数据项的关系：

名称：tno

别名：课程负责人教工号

意义简述：课程负责人的教工号

取值类型：int

长度：8

样式：20210102

取值范围：11

与其它数据项的关系：

**teacher\_group**

名称：gno

别名：课程组号

意义简述：课程组唯一标识

取值类型：int

长度：4

样式：1806

取值范围：11

与其它数据项的关系：

名称：tno

别名：教工号

意义简述：一个课程组有多个教师

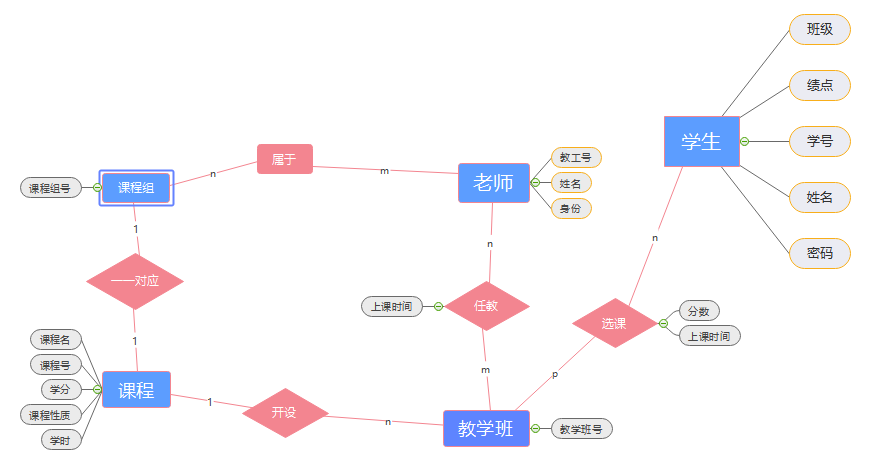
取值类型：int

长度：8

样式：20210102

取值范围：11

与其它数据项的关系：



逻辑设计：

除了建立老师，学生，课程这三个实体类，我们还建立了教学班（不是指学生所在的班级，一门课考虑到教室人数限制会开几个班给学生上，这个班就是教学班），课程组（一门课对应一个课程组，课程组由多个老师组成，这几个老师就负责这门课）这两个实体，目的是为了减少数据的冗余。实体之间的关系如上所示

course表



student表



teacher表



course\_class表（课程与教学班的对应关系）



course\_group表（课程组和课程的对应关系）



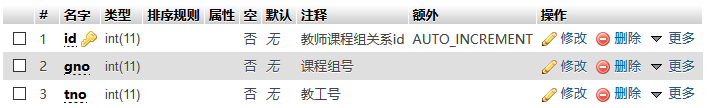
student\_class表（学生选修的是一门课对应的教学班，这里是保存学生选课后的记录，学生和课程的关系是通过教学班建立的）



teacher\_class表（老师与教学班的关系，一个老师可能教多个教学班，还有种特殊情况是多个老师教一个教学班，类似我们的计算机导论或者通识选修的生命科学课，就是说这门课的不同时间段对应不同的老师教）



teacher\_group（老师与课程组的关系，一个老师可能属于多个课程组，一个课程组包含多个老师）



除了以上几个表外，我们还建立了多个视图方便我们的查询操作

为什么采用视图，

视图的好处：

1. 视图能够简化我们的操作

对于后端来说，返回给前端数据只需要从视图查询就好了，不需要子查询或者连表查询

1. 视图使我们能以多种角度看待同一数据

视图相当灵活，可以让我们以不同的方式看待同一数据

1. 视图对重构数据库提供了一定程度的逻辑独立性

增加新的关系或对原有关系增加新的字段等，我们的应用程序不会受到影响

1. 视图能够对机密数据提供安全保护

有了视图机制，就可以在设计数据库应用系统时对不同的用户定义不同的视图，使机密数据不出现在不应看到这些数据的用户视图上。这样视图机制就自动提供了对机密数据的安全保护功能。例如，student表设计全校15个院系的学生数据，可以在其上定义15个视图，每个视图只包含一个院系的学生数据，并只允许每个院系的主任查询和修改本院系的学生视图。

1. 适当的使用视图可以更清晰地表达查询

例如当我们想显示选课信息给学生的时候，我们需要返回课程名字，课程类型，学分信息，课程安排时间段，教师信息等等，而通过建立视图我们可以很方便的获取上述数据。

运行sql语句

CREATE VIEW C\_C(courseName,courseCno,type,credit,curriculaVariable,timePeriod,teacherName,tno)

AS

SELECT course.name,course.cno,course.type,course.credit,course\_class.curricula\_variable,course\_class.time\_period,teacher\_class.tno,teacher.name

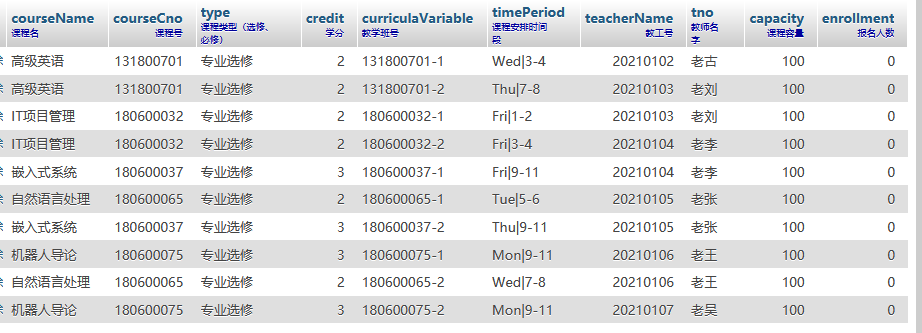
FROM course,course\_class,teacher\_class,teacher

WHERE course.cno = course\_class.cno AND course\_class.curricula\_variable=teacher\_class.curricula\_variable AND

teacher\_class.tno=teacher.tno AND course.type="专业选修"

可以建立如下视图c\_c





1.耦合性低

视图层和业务层分离，这样就允许更改视图层代码而不用重新编译模型和控制器代码，同样，一个应用的业务流程或者业务规则的改变只需要改动MVC的模型层即可。因为模型与控制器和视图相分离，所以很容易改变应用程序的数据层和业务规则。

2.重用性高

MVC模式允许使用各种不同样式的视图来访问同一个服务器端的代码，因为多个视图能共享一个模型，它包括任何WEB（HTTP）浏览器或者无线浏览器（wap），比如，用户可以通过电脑也可通过手机来订购某样产品，虽然订购的方式不一样，但处理订购产品的方式是一样的。由于模型返回的数据没有进行格式化，所以同样的构件能被不同的界面使用。

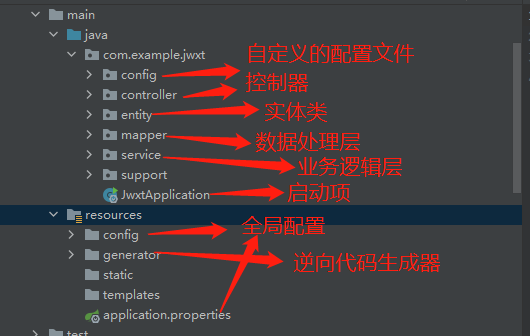
3.部署快，生命周期成本低

MVC使开发和维护用户接口的技术含量降低。使用MVC模式使开发时间得到相当大的缩减，它使程序员（Java开发人员）集中精力于业务逻辑，界面程序员（HTML和JSP开发人员）集中精力于表现形式上。

4.可维护性高

分离视图层和业务逻辑层也使得WEB应用更易于维护和修改。

**项目框架**



**1、dao层**

dao层主要做数据持久层的工作， 负责与数据库进行联络的一些任务都封装在此 ，dao层的设计 首先 是设计dao层的接口，然后在Spring的配置文件中定义此接口的实现类，然后就可以再模块中调用此接口来进行数据业务的处理，而不用关心此接口的具体实现类是哪个类，显得结构非常清晰，dao层的数据源配置，以及有关数据库连接参数都在Spring配置文件中进行配置。

**2、service层**

service层主要负责业务模块的应用逻辑应用设计。同样是 首先设计接口 ， 再设计其实现类 ，接着再Spring的配置文件中配置其实现的关联。这样我们就可以在应用中调用service接口来进行业务处理。service层的业务实， 具体要调用已经定义的dao层接口 ，封装service层业务逻辑有利于通用的业务逻辑的独立性和重复利用性。程序显得非常简洁。

**3、controller层**

controller层 负责具体的业务模块流程的控制 ，在此层要 调用service层的接口来控制业务流程 ，控制的配置也同样是在Spring的配置文件里进行，针对具体的业务流程，会有不同的控制器。我们具体的设计过程可以将流程进行抽象归纳，设计出可以重复利用的子单元流程模块。 这样不仅使程序结构变得清晰，也大大减少了代码量。

排课算法的设计：

设计排课算法的时候有查过百度参考，大家都说排课算法是个坑（NP难），不要乱碰，然后相关资料也不是很多。现成的方法也有，如遗传算法、动态规划等等，但由于时间有限，自己复现这些算法会有点难度，所以我采用了一个简单的思路-回溯算法

首先我们先来分析问题：

前面说到我们建立了一个教学班实体，建立教学班实体有一个目的就是为了方便排课，

因为一个教学班对应唯一一个时间段（4学分除外），确定一个教学班的时间段后，学生和教师的课程安排就出来了，所以我们只需要对教学班进行排课就好了。首先考虑的问题就是教学班的个数问题，我们如何确认教学班的个数呢？想到一般来说都是一个老师对应一个教学班，一门课对应几个教学班，从而得到

1、老师与排课的关系

课程组->一个老师能上多少门课->统计出一门课有多少个老师上->

这门课开多少个教学班

假设老师对课程时间地点没有要求，那么老师与排课的关系也就仅限于此了，对于必修的话可能要另算

假设每周的课程安排都一样->简化到只考虑一周的情况

假设每门课周学时只有2|3|4三种（分别是32/16，48/16，64/16）

那么就是说32学时(2学分)的课是在白天上两小节课，48学时只能安排在晚上三小节，

而64学时就是每周上两天，每天是白天两小节课

一天有11小节，其实也就是五大节课，最后一节是三小节

再深入分析，可以发现32学时的课选择范围就是每天的前四大节课，48学时只能是每天的第五大节，

64学时范围跟32学时的一样，也是每天的前四大节课，只不过要上两天。

应该是根据选修课的学分来生成一个时间段对象，比如2学分生成一个两小节时间段，3学分生成一个三小节时间段，4学分生成两个两小节时间段

每个班对象结构是：（如果是4学分的还要额外生成一个对象

1、教学班号

2、教工号

3、课程号

4、时间段类型 只有两种

简单算法就是逐步选择位置插入

如果出现两个相同的教学班号则把时间段用|合并

考虑这个时间段安排这个教学班是否合理应考虑如下：

1、这个班在这个时间段是否已经有其它课程的班上课，如有，则不应该选择该时间段

2、这个班的任教老师是否已经有另外的课或者教学班在这个时间段上课 只要查看这个时间段的tno就好了

3、时间段是否对应（3学分对应三小节

根据上述建一个课表对象，25格

每格结构：

1、tnoList

2、cnoList不相等则说明有其它选修课在上课，不应该选，相等的话，判断教学班号是否一致，一致则另选时间（4学分情况，其实上面第一步已经排除了），不一致则安排

3、时间类型可以通过下标判断（5\*i+j,i=4就是第五大节）或者加多字段time

25个棋格，有五个大格（三小节），二十个小格（两小节）

第一步先从25个位置随机选择

每一步都随机选择

选修课：

1、直接按课程组老师数量分配教学班，开班过程是：生成班号：课程号+班序号->插入course\_class和teacher\_class表->设置时间段

2、不同课程的教学班不能在同一时间段重叠，不然会出现学生不能所有选修课都选

3、一个老师任教的教学班不能在同一时间段上

必修课：

1、一个班同一时间点不能上多门课

2、不用学生选课，直接由系统安排或老师后面人工分配

3、一门课程学生人数多而课程组老师少，那么一个老师可能要多上几门课

3、必修的话操作可能有点不一样，可能是根据学生人数和课程组老师数来分配教学班，而不是只根据课程组老师数

4、简化操作，每个教学班人数都一样