

**软件学院大作业任务书**

题 目： 学生选课信息管理系统

专 业： 软件工程

班 级： 软工1809班

学 号： 8002118240

学生姓名： 杨孟衡

起讫日期： 2020-6-15~2020-8-30

任课教师： 职称：

部分管主任：

完成时间： 2020-8-30

**说 明**

1. 本任务书由任课教师填写后，下达到学生。
2. 任务完成后，任课教师需填写小结表。
3. 任务书、学生成绩与学生完成后的大作业（纸质和电子两种）一并报送各教学研究部审核后转教务办。
4. 另附一份全班大作业总结

|  |
| --- |
| **大作业的要求和内容：（包括题目选择范围、技术要求、递交时间、考核方法等）**   1. 内容要求   每组（包括1-5名同学）完成一个数据库系统项目，项目工作量要求适中，选题不宜太简单，实训内容包括需求分析报告（项目概述、功能模块图、数据流图、输出样表）、数据库建模报告（CDM、PDM、SQL语句）、系统实现报告（功能设计、流程设计、复杂功能的PL/SQL编程代码、JavaEE代码实现）。   1. 技术要求 2. 使用建业工具（如Visio）建立功能模块图; 3. 使用PowerDesigner建立数据流图; 4. 使用PowerDesigner创建CDM、PDM； 5. 使用PL/SQL Developer编写Oracle脚本； 6. 软件功能模块图（思维导图）、复杂功能的流程图、算法描述等 7. 软件功能的运行测试（用例、截图、关键代码） 8. 利用Word编制实训报告，要求文档规范（封面、作者、目录、正文）。 9. 质量要求   数据库最少不能少于5张表格，一定要有1个触发器、1个以上存贮过程。   1. 设计说明书   要求提交3份实训报告，即需求分析报告、数据库建模报告、系统实现报告。 |
| 教师小结：  成绩：  教 师 签 名：  教研部负责人：  学生姓名： 杨孟衡 |

**本科学生综合性实验报告**

**课程名称：数据库系统原理**

学生选课数据库设计

项目组长 廖严浩 学号 8002118225

专 业： 软件工程

班 级： 软工1809班

学 号： 8002118225

学生姓名： 杨孟衡

起讫日期： 2020-6-15~2020-8-30

任课教师： 职称：

部分管主任：

完成时间： 2020-8-30

**目 录**

[1、需求分析 1](#_Toc50964659)

[1.1 编写目的 1](#_Toc50964660)

[1.2功能描述 1](#_Toc50964661)

[1.3系统总体功能图 3](#_Toc50964662)

[1.4系统流程图 3](#_Toc50964663)

[1.5数据流图 4](#_Toc50964664)

[1.6 数据实体分析 8](#_Toc50964665)

[1.6.1实体数据 8](#_Toc50964666)

[1.6.2数据与联系 9](#_Toc50964667)

[1.6.3数据字典 9](#_Toc50964668)

[2概念结构设计 12](#_Toc50964669)

[2.1实体图 12](#_Toc50964670)

[2.2总体ER图 13](#_Toc50964671)

[3逻辑结构设计 14](#_Toc50964672)

[3.1关系设计 14](#_Toc50964673)

[3.3约束的说明 14](#_Toc50964674)

[3.4实体转换后的基本表 14](#_Toc50964675)

[4数据库的物理设计 16](#_Toc50964676)

[4.1存储介质类型的选择 16](#_Toc50964677)

[5.2 数据库“学生选课”的建立 17](#_Toc50964678)

[5.3 各个数据表(视图)的建立 18](#_Toc50964679)

[5.4数据库服务器性能优化 19](#_Toc50964680)

[5. 个人小结 20](#_Toc50964681)

## 1、需求分析

### 1.1 编写目的

我们都知道信息的管理和操作是非常精细的工作，人工处理过多信息的时候易出现失误导致信息的不匹配，而交由软件来进行处理的话可以减少人工录入信息的失误，只要操作正确就可以处理大部分的情况；尤其是在重视教育的现在，学生的信息管理要求对数据操作有一定的精度，不然就会对学生的权益造成损害。现阶段大多数学校都有关于自己的学生管理系统，而有一些学校的管理系统的效率、功能甚至是服务效果均不是很理想。所以有关学生信息管理系统的开发在实际应用中也是很有市场的，因此，为了解决这一需求，同时相应国家素质教育全面发展的号召，本小组决定针对学生、教师的信息管理以及选课功能开发一款学生选课管理系统，可以对学生、教师、相关课程的信息进行添加、删除以及修改的操作，并且允许学生选择自己中意的课程，可以让学校的管理老师及时迅速的了解学生、教师以及课程的相关信息及相关信息的变化情况。

### 1.2功能描述

学生选课信息管理系统的用户分为三类：

（1）学校管理员：该用户为学校的具体或者高层管理人员，可以控制该系统中除管理员信息之外的所有数据；其权限是该系统中最高的。

（2）学校教师：该用户为学校的授课老师，可以决定自己所教授的课程以及该课程的各种信息数据（如上课时间、最大选修人数等），同时可以改写和查阅自己所授课班级中学生的各种信息（如是否选择、课程成绩等）；其系统权限在管理员之下

（3）学生：该用户的主体为学校的学生，在校生应该待学校管理员将学生信息全部录入之后进行身份的核验，之后才能进行选课和修改个人信息等操作。

作为学生选课信息管理系统，其主要功能还是在于学生选课上面。该系统针对三种用户分别提供以下功能：

（1）针对学生用户：

①登录功能：学生用户必须等待后台将信息录入数据库后才能登陆该系统，且在登录时需要输入自己的学号和初始密码，同时需要输入验证码。有关密码的操作将在后续说明。

②选课功能：学生在登录了该系统之后，可以查阅学校该学期开展的所有课程并获取其相关信息，再根据个人的相关情况来决定是否选择该课程；同时学生也可以查看自己已经选择的课程，并且根据实际情况决定是都退选；同时还可在该个人选课表上看到自己所选课程的成绩。

③修改个人信息功能：学生在登录后可以根据实际情况更改自己的个人信息以及账号密码。

④客户注册成功以后，其注册信息将自动被加入客户表中。登录系统后，客户可以查询或修改个人信息。

（2）针对教师用户：

①登录功能：教师通过自己的工号进行系统的登录，密码与验证码的设置与学生用户一致，修改个人信息功能也与学生用户一致。

②课程信息修改功能：老师可以在学校的授课数据库中添加自己所教授课程的相关信息，以供学生进行选择，同时可以修改、删除自己添加过的课程信息（不过在删除课程时需要注意该课程应处于无学生选择的状态）。

③学生选课信息的查阅与修改功能：教师用户可以在该系统上看到自己授课班级的学生选修情况，同时可以在后台对学生的选课情况进行操控；此外，教师还可以对学生在该课程上的学习情况进行评分，即各处成绩。

④学生信息添加功能：教师可以添加学生的信息进入到数据库中，用以防止有的学生未被后台和管理员添加进数据库的情况（当然也可以删除）。

（3）针对管理员：

由于学校管理员在该系统中拥有最顶级的权限，故其在功能上是覆盖学生和教师的；为管理员提供的功能是为学生用户提供的功能与为教师用户提供的功能的集合，此外管理员还可以添加、修改、删除教师用户的信息，不过需注意在删除教师用户的时候该教师不应该教授任何课程。

还需注意的是，管理员不能添加其他的管理员。

### 1.3系统总体功能图

根据上节分析的系统功能需求，我们可以得到系统的功能模块，如图1.1所示。

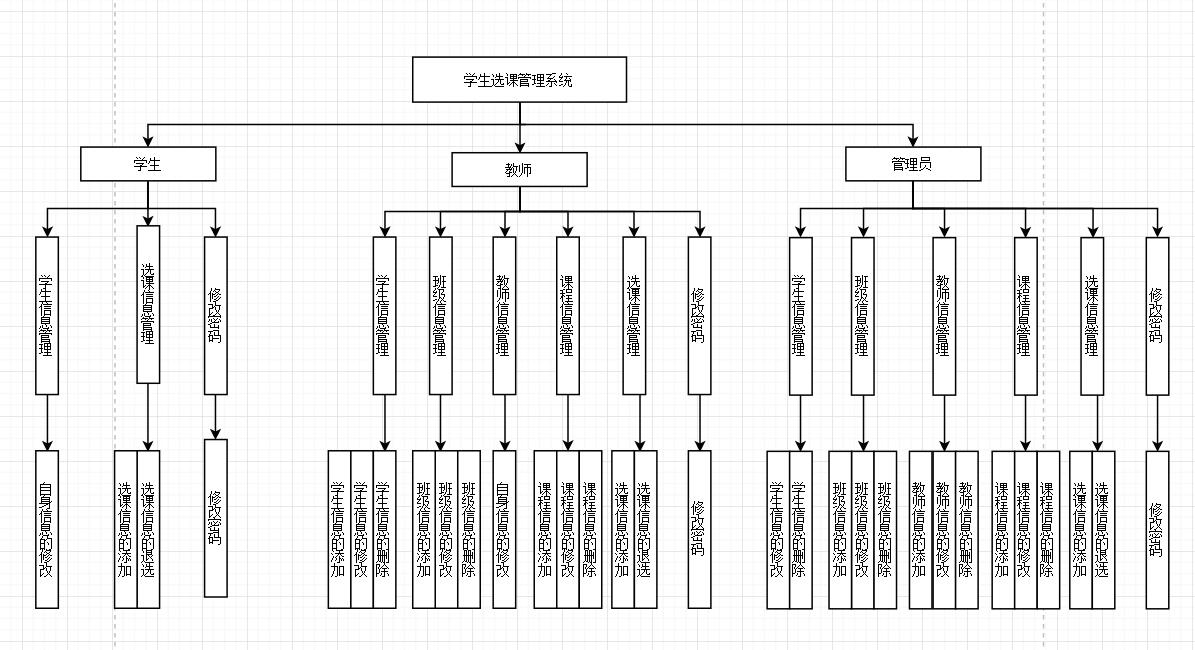


图1.1 系统功能图

### 1.4系统流程图

本商城客户购买商品的系统流程图，如图1.2所示。

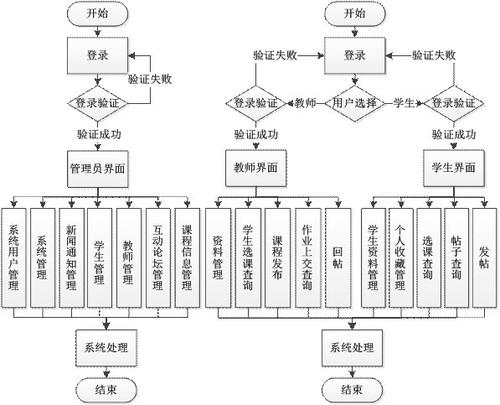


图 1.2 系统流程图

### 1.5数据流图

本次的数据流图部分由本人绘制，其中第1层和第2层DFD图包含了本人负责的部分-教师管理。

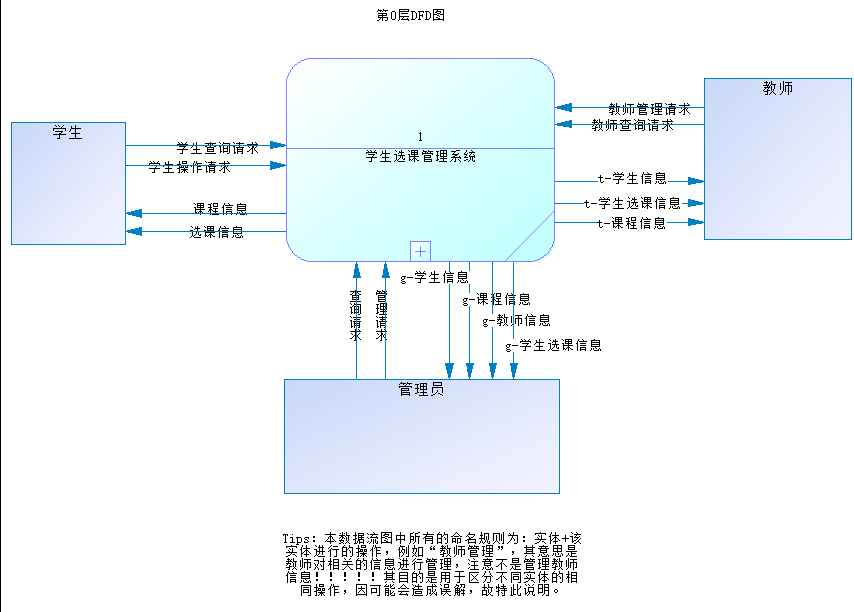


图 1.3 第0层DFD图

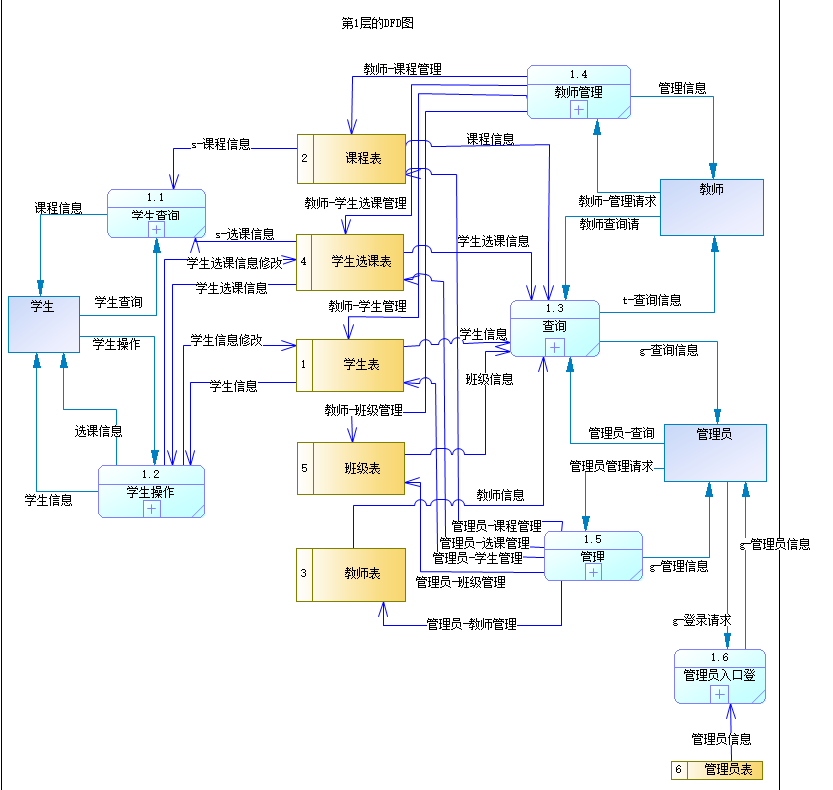


图1.4 第1层DFD图

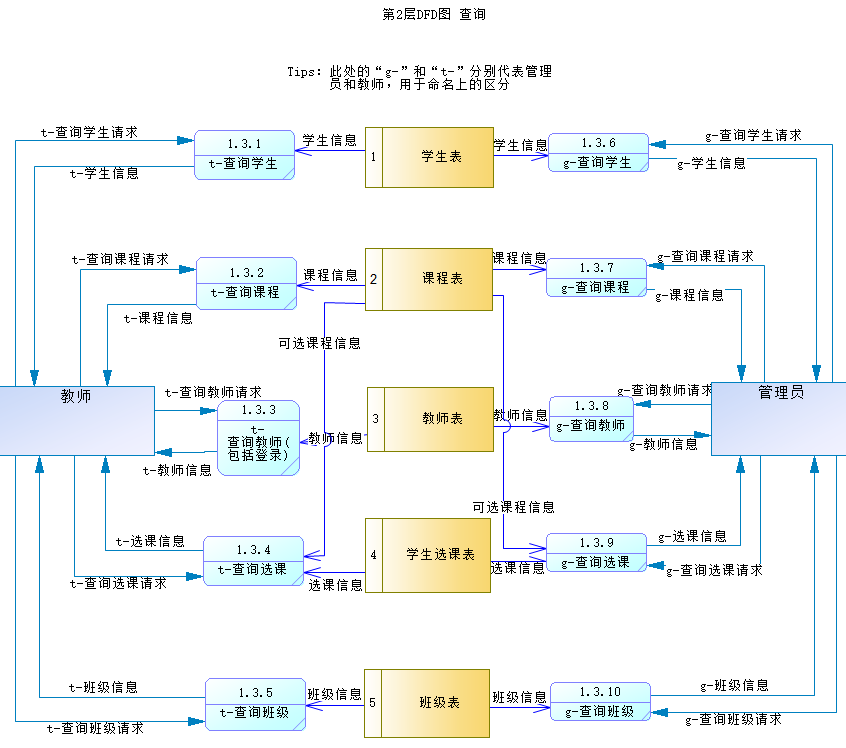


图1.5 第2层教师查询DFD图

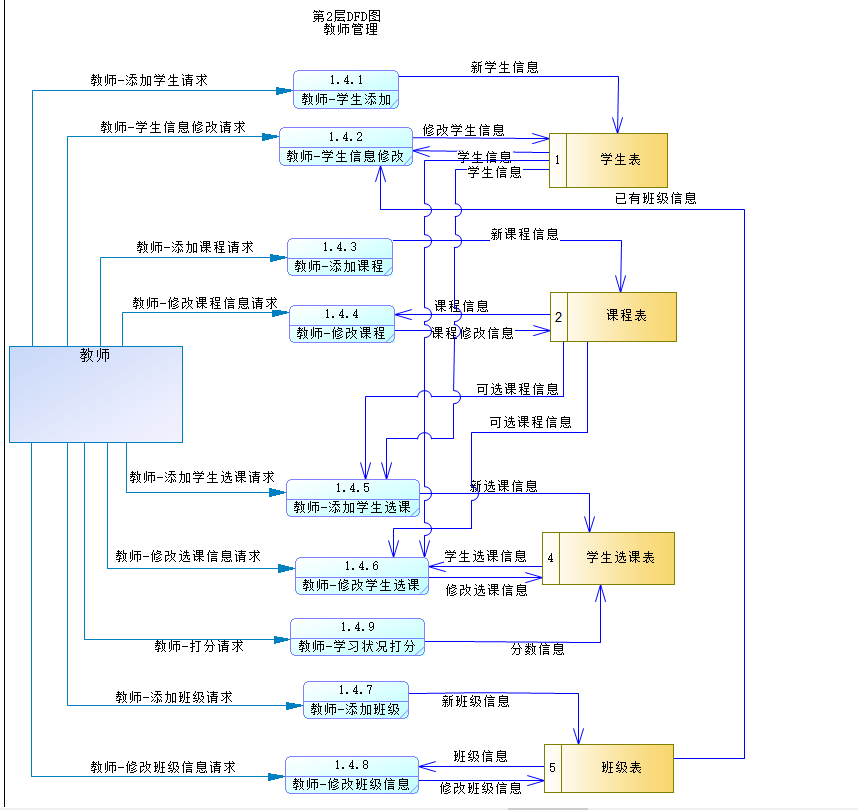


图1.6 第二层教师管理DFD图

### 1.6 数据实体分析（教师部分）

### 1.6.1实体数据

通过对选课系统各方面分析，我们可以知道该选课系统中的实体包括：管理员、教师、学生、班级、课程、学生选课表这五个实体。为了便于后续报告内容讲解，在此列出全部实体，之后再单独展示教师部分实体数据：

（1）学生：编号，学号、姓名、密码、性别、电话、班级编号、QQ号、照片。

（2）课程：编号、课程名称、教师编号、上课时间、选课人数、最大人数、课程学分、信息

（3）班级：班级编号、班级名称、班级介绍

（4）管理员： 编号、姓名、密码

（5）教师：编号、工号、姓名、密码、班级编号、性别、电话、QQ号、照片

### 1.6.2数据与联系

此外，我们作出如下规定：

（1）一个班级可以有多个学生，一名学生只属于一个班级；

（2）一名教师可以教授多门课程，一门课程只能由一名教师教授；

（3）一个班级可以有多个教师带领，一名教师只能带领一个班级；

（4）一名学生可以选择多门课程，一门课程可以被多名学生选择；

通过以上对实体与数据的分析，我们可以得到如下实体间的联系：

（1）学生与班级：班级编号

（2）课程与教师：教师编号

（3）班级和教师：教师编号

（4）学生与课程：需要增加一张选课表，其数据项包括：选课编号、学号、课程编号、成绩；此部分详见报告的第二部分概念设计章节的实体E-R图。

### 1.6.3数据字典

数据字典包括数据项、数据结构、数据流、数据处理4个部分。其中数据项是数据的最小组成单位，若干个数据项可以组成一个数据结构，数据字典通过对数据项和数据结构的定义来描述数据流、数据存储的逻辑内容。本人主要负责教师管理功能的实现，故在此只写入与教师、课程相关的数据字典。

**（1）数据项**

如表1.1所示。

表 1.1 数据项表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 数据项 | 数据类型 | 别名 | 含义说明 | 取值范围 | 取值含义 |
| id | Int | Id号 | 唯一标识每一教师 | 1-n | N代表无穷大，每次有新增教师就加一标识 |
| tn | Varchar(32) | 教师号 | 教师表的主键，唯一标识每一教师 | 6-32位字符串 | 尽可能的将教师号简化，字符串最方便处理 |
| name | Varchar(32) | 教师姓名 | 教师姓名 | 不得超过16字 | 设置的偏大了一点，不是很影响 |
| password | Varchar(32) | 账号密码 | 教师账号的密码 | 6-32位字符串 | 账号密码是多元化的，用户可自行设置密码难易程度 |
| clazz\_id | Int | 班级id | 教师教导的班级 | 1-n | N代表无穷大，每次有新增班级就加一 |
| sex | Varchar(5) | 性别 | 教师性别 | 5位字符串 | 一个中文字占用2个字符，5位正好储存性别字符串 |
| mobile | Varchar(12) | 手机 | 教师手机 | 12位字符串 | 将手机号设定为12位，便于我们调试代码 |
| qq | Varchar(18) | qq | 教师qq | 18位字符串 | 设置的偏大一点，但不影响qq输入 |
| course\_id | int | 课程编号 |  |  |  |
| course\_name | varchar(32) | 课程名称 |  |  |  |
| teacher\_id | int | 教师编号 | 授课教师的编号 |  | 注意不是工号，用编号是为了简化外键以及选课表 |
| course\_date | varchar(32) | 上课时间 |  |  |  |
| select\_num | int | 选择人数 |  |  | 已选择该课程的学生人数 |
| max\_num | int | 最大人数 |  |  | 该课程最大允许学生选择的人数，注意select\_num<\_=max\_num |
| point | int | 课程学分 |  |  |  |
| course\_info | varchar(32) | 课程信息 |  |  |  |
| select\_id | int | 记录号 | 选课编号 |  | 用于标识选课表中的记录 |

**（2）数据结构**

1.数据结构：教师

含义说明：是教师管理子系统的主体数据结构，定义了一个教师的有关信息

组成：id，教师号，教师姓名，密码，班级id，性别，电话，图片

**（3）数据流**

1.数据流： **教师的个人信息**

说明： 教师在修改时所登记的个人信息

数据流来源： 修改

数据流去向： 保留在教师表中

组成： id，教师号，教师姓名，密码，班级id，性别，电话，图片

**（4）数据处理**

数据处理过程如表所示。

表 1.2 数据处理表

|  |  |  |
| --- | --- | --- |
| **数据存储名** | **功能** | **处理说明** |
| insert\_teacher | 插入新的教师信息 | 当教师成功注册后，自动在教师表中添加该教师的注册信息 |
| select\_ teacher | 查询教师个人信息 | 登录系统后，教师可以查询个人信息 |
| update\_ teacher | 修改教师个人信息 | 登录系统后，教师可以修改个人信息 |
| delete\_ teacher | 删除教师信息 | 若教师离最后一次登录的时间超过一年，则管理员删除该教师信息 |
| insert\_course | 插入新的课程信息 | 由教师或者管理员上传本学期开设的相关课程信息 |
| update\_course | 修改课程信息 | 教师或者管理员可以修改课程的相关信息 |
| delete\_course | 删除课程信息 | 管理员或者教师可以删除某一门课程 |
| 触发器名 | 功能 | 处理说明 |
| limit\_dele\_course | 限制课程信息的删除 | 仅当该课程的选择人数为0是允许教师或者管理员删除该课程信息信息 |
| limit\_dele\_teacher | 限制教师信息的删除 | 仅当该教师没有教授课程时管理员可以删除该教师信息 |
| teacher \_tri | 仅允许管理员用户删除教师信息 | 当删除教师信息时,如果为学生,他没有删除的权限,若为管理员,则可以删除 |

# 

# 2概念结构设计

通过对用户需求进行综合、归纳与抽象，我们将另其形成一个独立于具体DBMS的概念模型，并采用自底向上的方法．用从局部到整体的方法将其通过E-R图来表示各实体之间的联系。不过由于本人主要负责的是教师管理功能，故在此只展示教师、课程实体的单独E-R图、教师管理的E-R图以及学生选课系统的整体E-R图

## 2.1实体图

以下是上述中教师、课程实体单独的E-R图：

（1）学生实体图，如图2.1所示。

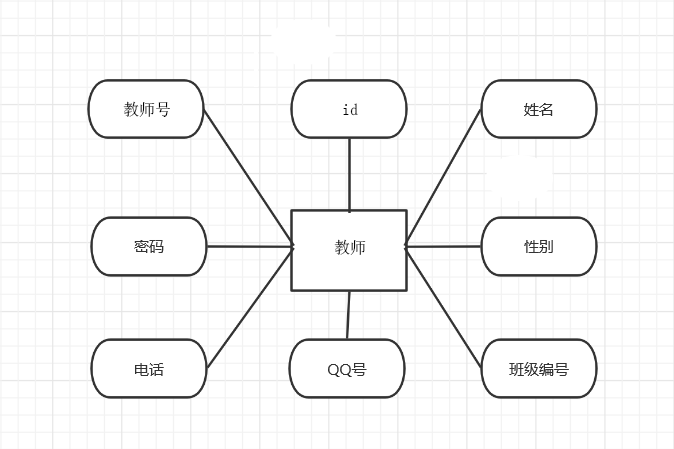


图 2.1 学生实体图

（2）课程实体图，如图2.3所示。

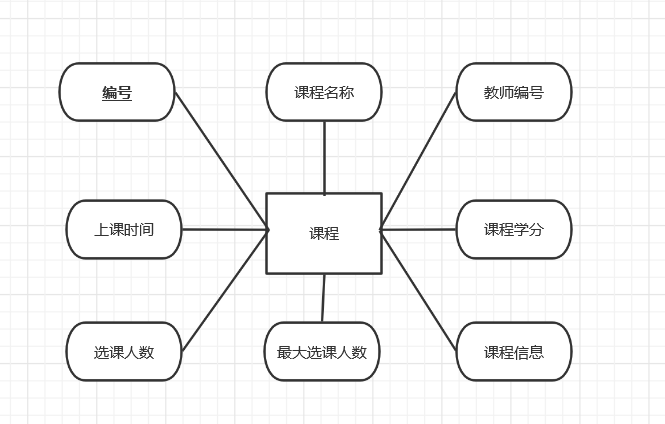


图 2.2 课程表实体图

## 2.2总体ER图

在此直接展示总体的E-R图。

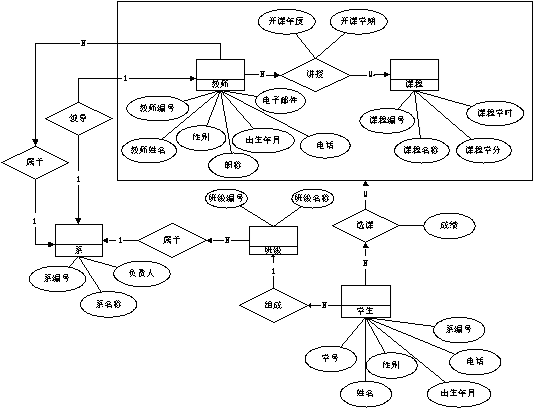


图 2.3 总ER图

# 3逻辑结构设计

本次的学生选课系统数据库在SQL Server2000上进行实现，依托于在Power Designer上实现准备好的数据流图、物理模型来进行辅助和整理，以将概念结构设计中的E-R图转换成SQL Server2000支持的关系数据模型。

## 3.1关系设计

将上述分析中需要进行转换的实体和联系转换为表格。

将每个单独的实体以及额外增加的联系之间转换为一张单独的表，具体转换如下：

课程：编号、课程名称、教师编号、上课时间、选课人数、课程学分、信息

班级：班级编号、班级名称、班级介绍

教师：编号、工号、姓名、密码、班级编号、性别、电话、QQ号、照片

选课表：学号、课程编号、成绩

## 3.3约束的说明

根据参照完整性，表与表之间有主键、外键、用户自定义约束。本人主要负责的是教师管理功能的实现，所以在此主要说明与教师表、课程表和选课表相关的约束。

1. 教师表的主键是id和教师号，即(‘id’,’tn’)。
2. 教师表的外键是班级编号(‘class\_id’)。
3. 课程表的主键是课程编号(‘course\_id’)。
4. 课程表的外键是教师编号(‘teacher\_id’)。
5. 选课表的主键是学生编号和课程编号(‘student\_id’,’course\_id’)。
6. 选课表的外键是学生编号以及课程编号，这是分别两个外键，即(‘student\_id’)和(‘course\_id’)。
7. 我们规定，在教师删除课程的时候，如果选课表中显示该课程有学生选择，则无法将其进行删除。

## 3.4实体转换后的基本表

通过上面的关系转换与关系优化,我们最终得到6张基本表，在此主要展示教师表、课程表以及选课表；

（1）教师表 如表 3.1所示。

表 3.1 教师表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 含义 | 是否为主键 |
| id | int | not null | 编号 | primary key |
| tn | varchar(32) | not null | 教师号 | primary key |
| name | varchar(32) | not null | 姓名 |  |
| password | varchar(32) | not null | 密码 |  |
| class\_id | int | not null | 班级编号 |  |
| sex | varchar(5) | not null | 性别 |  |
| mobile | varchar(12) |  | 电话 |  |
| qq | varchar(18) |  | QQ号 |  |
| photo | mediumblob |  | 头像、照片 |  |

（2）课程表 如表 3.3所示。

表 3.2 课程表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 含义 | 是否为主键 |
| id | int | not null | 课程编号 | primary key |
| name | varchar(32) | not null | 课程名称 |  |
| teacher\_id | int | not null | 教师编号 |  |
| course\_date | varchar(32) |  | 上课时间 |  |
| select\_num | int | not null | 选择人数 |  |
| max\_num | int | not null | 最大选择人数 |  |
| info | varchar(32) |  | 课程信息 |  |

（3）选课表 如表 3.5所示。

表 3.3 选课表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 数据类型 | 是否为空 | 含义 | 是否为主键 |
| id | int | not null | 选择编号 |  |
| student\_id | int | not null | 学生编号 | primary key |
| course\_id | int | not null | 课程编号 |

# 4数据库的物理设计

数据库的物理设计任务，主要是将逻辑设计映射到存储介质上，利用可用的硬件和软件条件能可靠地、高效地对数据进行物理访问和维护。存储介质及其存储模式是任何关系数据库的关键组件。数据库的成功执行通常需要在工程的前期阶段精心设计。关系数据库的存储设计在此数据库设计过程中占了很大份量，其中主要考虑的内容：

使用哪种类型的磁盘硬件，如 RAID（独立磁盘冗余阵列）设备；

数据在磁盘上如何放置即数据的分配策略；

从访问性能的角度采用适当的索引技术和设计具体的索引项；

以及基于特定数据库有关的参数配置以使数据库很好地运行。

## 4.1存储介质类型的选择

RAID（独立磁盘冗余阵列）是由多个磁盘驱动器（一个阵列）组成的磁盘系统，可提供更高的性能、可靠性、存储容量和更低的成本。容错阵列分为从 0 到 5 共 6 个 RAID 等级。每个等级使用不同的算法实现容错。

SQL Server 一般使用 RAID 等级 0、1 和 5(注:RAID仅在Windows NT 4.0 、Windows 2000 及Windows 2003等系统上配合使用)。

RAID0由于该等级使用数据分割技术(称为条带集)的磁盘文件系统，数据分成块并在阵列内的所有磁盘中按固定顺序展开。RAID 0 通过在多个磁盘内的独立而同时地数据的读/写操作,具备非常高的读写性能。但是这种组合方式和普通计算机上硬盘使用的模式一样没有任何容错机制，如果一个磁盘发生故障，则需要的所有数据将不可访问。因此关键数据不能安放在RAID0中。常用的关系数据库管理系统安装技术是在 RAID 0 驱动器上配置数据库，然后将事务日志放置在镜像驱动器上 (RAID 1)。通过镜像事务日志，可以为数据库获取最佳的磁盘 I/O 性能并维护数据可恢复性（假定执行定期数据库备份）。

RAID1也称为镜像集的磁盘文件系统或称磁盘镜象系统。磁盘镜像提供选定磁盘的冗余的、完全一样的复本。所有写入主磁盘的数据均写入镜像磁盘。RAID 1提供容错能力，如一个磁盘数据的损坏总是可以从另一个磁盘得到恢复。这种级别的RAID基本上能保证数据读取的性能，但是由于在写数据时需要将相同的数据同时写到两个硬盘上，因而RAID1会降低数据的写性能。

RAID5等级，也称带奇偶校验的数据分割技术，是目前设计中常用的策略。该等级在陈列内的磁盘中，将数据分成大块，并在所有的磁盘中写入奇偶校验信息，数据冗余有这些奇偶信息提供。数据和奇偶信息排列在磁盘阵列上，而且两者始终错开存放在不同的磁盘上，所以RAID 5提供阵列上的所有数据冗余，在大多数情况下允许单个磁盘发生故障并被替换，而不会中断系统运行(指所谓的热插拔)。RAID 5提供的性能比 RAID 0 或 RAID 1 要低一些，但提供更高的可靠性和更快的恢复能力。相对RAID1，RAID5在同样保证数据可靠性前提下，实现更高的性能和存储量。

RAID具体运行控制机制主要分两种,其一:磁盘的输入/输出操作在RAID内置的电路中得到高效地处理，这种所谓硬件的RAID可以显著地提高 I/O 性能；其二，基于操作系统的 RAID 使成本较低但要占用较长的处理器周期。当成本是考虑因素之一而且需要冗余和高性能时，我们可以选择Windows 2000 RAID-5 卷的解决方案，该系统启用4个物理磁盘，为后面的数据分配中涉及的多文件存放提供条件。

在经过前面几节的分析和设计之后，就可以得到数据库的关系模式。创建学生选课管理系统的数据库。这一节给出了在SQL Server2000中实现该系统的数据库关系模式。

## 5.2 数据库“学生选课”的建立

SQL Server2000 使用一组操作系统文件映射数据库。数据库中的所有数据和对象（如表、存储过程、触发器和视图）都存储在下列三种文件类型的操作系统文件中：

**主文件** 这些文件包含数据库的启动信息。主文件还用于存储数据。每个数据库都包含一个主文件。

**次要文件** 这些文件含有不能置于主要数据文件中的所有数据。如果主文件足够大，能够容纳数据库中的所有数据，则该数据库不需要次要数据文件。有些数据库可能非常大，因此需要多个次要数据文件，或可能在各自的磁盘驱动器上使用次要文件，以便在多个磁盘上存储数据。其扩展名一般为ndf。

**事务日志** 这些文件包含用于恢复数据库的日志信息。每个数据库必须至少有一个事务日志文件（但是可以有多个）。日志文件最小为 512 KB, 其扩展名一般为ldf。

创建简单的一个数据库demo时，可以只使用一个包含所有数据和对象的主文件和一个包含事务日志信息的日志文件。另一种情况是，创建更复杂的数据库 mis时，可以使用一个主文件和五个辅助文件，数据库内的数据和对象扩展到所有的六个文件中，另外有四个日志文件包含事务日志信息。

文件组允许对文件进行分组，以便于管理和数据的分配／放置。例如，可以分别在三个硬盘驱动器上创建三个文件（Data1.ndf、Data2.ndf 和 Data3.ndf），并将这三个文件指派到文件组 **fgroup1** 中。然后，可以明确地在文件组 **fgroup1** 上创建一个表。对表中数据的查询将分散到三个磁盘上，因而性能得以提高。在 RAID（独立磁盘冗余阵列）上创建单个文件也可以获得相同的性能改善。然而，文件和文件组使得我们可以在新磁盘上轻易地添加新文件。

假定系统的逻辑盘C、D、E、F是对应于RAID上的四个物理硬盘。我们将数据文件分成三个，主数据文件C:\css\data\csmain.mdf，两个次数据文件分别为：主数据文件D:\css\data\cssecnd1.ndf和主数据文件E:\css\data\cssecnd2.nmdf；日志文件F: \css\data\cslog.ldf。

首先，我们为系统开发的需要分别在C:、D、E、F上建立四个文件夹css\data\分别用于存放上述三个数据文件和一个日志文件。这样系统由于可以对四个磁盘实现并行访问而达到高效的读写效率。创建数据库的语句如下：

## 5.3 各个数据表(视图)的建立

建立数据库“学生选课”中各个数据表的SQL语句如下：

--创建系管理员表

CREATE TABLE `s\_admin` (

  `id` int(5) NOT NULL AUTO\_INCREMENT,

  `name` varchar(32) NOT NULL,

  `password` varchar(32) NOT NULL,

  `status` tinyint(1) NOT NULL DEFAULT '1',

  PRIMARY KEY (`id`,`name`)

) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=utf8;

--创建班级表

CREATE TABLE `s\_clazz` (

  `id` int(5) NOT NULL AUTO\_INCREMENT,

  `name` varchar(32) NOT NULL,

  `info` varchar(128) DEFAULT NULL,

  PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=6 DEFAULT CHARSET=utf8;

--创建学生基本信息表

CREATE TABLE `s\_student` (

  `id` int(5) NOT NULL AUTO\_INCREMENT,

  `sn` varchar(32) NOT NULL,

  `name` varchar(32) NOT NULL,

  `password` varchar(32) NOT NULL,

  `clazz\_id` int(5) NOT NULL,

  `sex` varchar(5) NOT NULL DEFAULT '男',

  `mobile` varchar(12) DEFAULT NULL,

  `qq` varchar(18) DEFAULT NULL,

  `photo` mediumblob,

  PRIMARY KEY (`id`,`sn`),

  KEY `student\_clazz\_id\_foreign` (`clazz\_id`),

  KEY `id` (`id`),

**CONSTRAINT** `student\_clazz\_id\_foreign` **FOREIGN KEY** (`clazz\_id`) **REFERENCES** `s\_clazz` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=10 DEFAULT CHARSET=utf8;

--创建教师基本信息表

CREATE TABLE `s\_teacher` (

  `id` int(5) NOT NULL AUTO\_INCREMENT,

  `tn` varchar(32) NOT NULL,

  `name` varchar(32) NOT NULL,

  `password` varchar(32) NOT NULL,

  `clazz\_id` int(5) NOT NULL,

  `sex` varchar(5) NOT NULL DEFAULT '男',

  `mobile` varchar(12) DEFAULT NULL,

  `qq` varchar(18) DEFAULT NULL,

  `photo` mediumblob,

  PRIMARY KEY (`id`,`tn`),

  KEY `student\_clazz\_id\_foreign` (`clazz\_id`),

  KEY `id` (`id`),

**CONSTRAINT** `s\_teacher\_ibfk\_1` **FOREIGN KEY** (`clazz\_id`) **REFERENCES** `s\_clazz` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=12 DEFAULT CHARSET=utf8;

--创建课程基本信息表

CREATE TABLE `s\_course` (

  `id` int(5) NOT NULL AUTO\_INCREMENT,

  `name` varchar(32) NOT NULL,

  `teacher\_id` int(5) NOT NULL,

  `course\_date` varchar(32) DEFAULT NULL,

  `selected\_num` int(5) NOT NULL DEFAULT '0',

  `max\_num` int(5) NOT NULL DEFAULT '50',

  `info` varchar(128) DEFAULT NULL,

  PRIMARY KEY (`id`),

  KEY `course\_teacher\_foreign` (`teacher\_id`),

**CONSTRAINT** `course\_teacher\_foreign` **FOREIGN KEY** (`teacher\_id`) **REFERENCES** `s\_teacher` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=15 DEFAULT CHARSET=utf8;

--创建学生选课基本信息表StuCourse

CREATE TABLE `s\_selected\_course` (

  `id` int(5) NOT NULL AUTO\_INCREMENT,

  `student\_id` int(5) NOT NULL,

  `course\_id` int(5) NOT NULL,

  PRIMARY KEY (`id`),

  KEY `selected\_course\_student\_fk` (`student\_id`),

  KEY `selected\_course\_course\_fk` (`course\_id`),

**CONSTRAINT** `selected\_course\_course\_fk` **FOREIGN KEY** (`course\_id`) **REFERENCES** `s\_course` (`id`),

**CONSTRAINT** `selected\_course\_student\_fk` **FOREIGN KEY** (`student\_id`) **REFERENCES** `s\_student` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=15 DEFAULT CHARSET=utf8;

## 5.4数据库服务器性能优化

服务器的性能优化是通过调节的目的是通过将网络流通、磁盘 I/O 和 CPU 时间减到最小，使每个查询的响应时间最短并最大限度地提高整个数据库服务器的吞吐量。为达到此目的，需要了解应用程序的需求和数据的逻辑和物理结构，并在相互冲突的数据库使用之间（如联机事务处理 (OLTP) 与决策支持）权衡。

对性能问题的考虑应贯穿于开发阶段的全过程，不应只在最后实现系统时才考虑性能问题。许多使性能得到显著提高的性能事宜可通过开始时仔细设计得以实现。为最有效地优化SQL Server2000 的性能，必须在极为多样化的情形中识别出会使性能提升最多的区域，并对这些区域集中分析。

虽然其它系统级性能问题（如内存、硬件等）也是研究对象，但经验表明从这些方面获得的性能收益通常会增长，因为机器的配置根据需要很容易实现更新甚至更换。通常情况下，SQL Server 自动管理可用的硬件资源，从而减少对大量的系统级手动调节任务的需求（以及从中所得的收益）。

总的来说，数据库服务器在安装过程中做到基本优化的配置，通常不需要作太大的调整，根据实际运行(维护)阶段的实际运行效果再作必要的调整。

# 个人小结

在本次实训中，我们小组首先学习了数据库设计各个阶段的目标和步骤，然后通过对一个实例—学生选课管理系统的数据库设计过程的实际训练，较好的掌握了数据库设计过程。

通过对数据库设计的学习，我个人在具体的实践活动中收获颇丰，虽然因为疫情的特殊原因，我个人未能到校上课，不能和老师面对面交流，但是与小组的交流沟通也让我受益无穷，团队合作能力非常重要，这是一次很好的实训体验，感谢我的小组，感谢老师。