数据库设计的典型案例

要点

- 学生选课管理系统的数据库设计

学习目标

- 学生选课管理系统的需求分析
- 学生选课管理系统的 ER 图
- 学生选课管理系统的关系数据库模式
- 学生选课管理系统数据库的建立

在第 7 章里我们已经学习了有关数据库设计的基本理论和方法。本章通过学生选课管理系统数据库设计案例，实际讲授数据库的设计方法，加深对第七章的理解，提高我们的综合设计的能力。

8.1 案例的系统需求简介

8.1.1 总体需求简单介绍

需求分析阶段是数据库应用系统开发的最重要阶段。需求分析要求应用系统的开发人员按照系统的思想，根据收集的资料，对系统目标进行分析，对业务的信息需求、功能需求以及管理中存在的问题等进行分析，抽取本质的、整体的需求，为设计一个结构良好的数据库应用系统的逻辑模型奠定坚实的基础。

高等学校的学生选课管理系统，在不同的学校会有不同的特点，因为作为教务工作部分它和学校本身的行政制度有关。本章的目的在于，作为数据库设计和应用开发的运用对象，对业务进行适度的简化，突出比较核心的成分，如院系算作一个级别的概念而且直接管理班(跳过专业一级的设置)，学生的免修重修等情况处理、教师的管理没有细化等。

8.1.2 用户总体业务构造

学生选课管理业务，包括 4 个主要部分：学生的学籍及成绩管理、制定教学计划、学生选课管理以及教学调度。各部分具体的内容：

（1） 学籍及成绩管理包括：各院系的教务员完成学生学籍注册、毕业、转学等处理，各授课教师完成所讲授课成绩的录入，然后教务员进行学生成绩的审核认可。

（2） 制定教学计划包括：由教务部门完成指导性教学计划、培养方案的确定，开设课程的注册和调整。

（3） 学生选课包括：学生根据开设课程和培养计划(和自己的状况)选择自己本学期所选修课程，教务员对学生所选修课程的确认处理。(注意：一般的必修课程是由教务员统一处理，只有辅修的课程才经过学生的选择过程)

（4） 执行教学调度包括：教务员根据本学期所开设的课程、教师上课的情况以及学生选课情况完成排课、调课等。

8.1.3 其它要求

如安全性，系统环境要求(根据现有的设备情况进行系统运行)等，这些不是本章的核心内容，所以就不再进一步叙述。

8.1.4 系统功能设想

这里的功能划分，是根据第一阶段需求调查基础上进行的初步划分。随着需求调查的深入，功能模块随着对需求了解的明确得到调整。

教务管理业务的 4 个主要部分，可以将系统应用程序划分为对应得 4 个子模块：包括学籍及成绩管理子系统、教学计划管理子系统、学生选课管理子系统以及教学调度子系统。根据各业务子系统所包括业务内容,还可以将各个子系统继续细化划分为更小的功能模块。划分的准则主要遵循模块的内聚性要求和模块间的低聚合性。如图所示表示一个教务管理系统功能模块结构图。

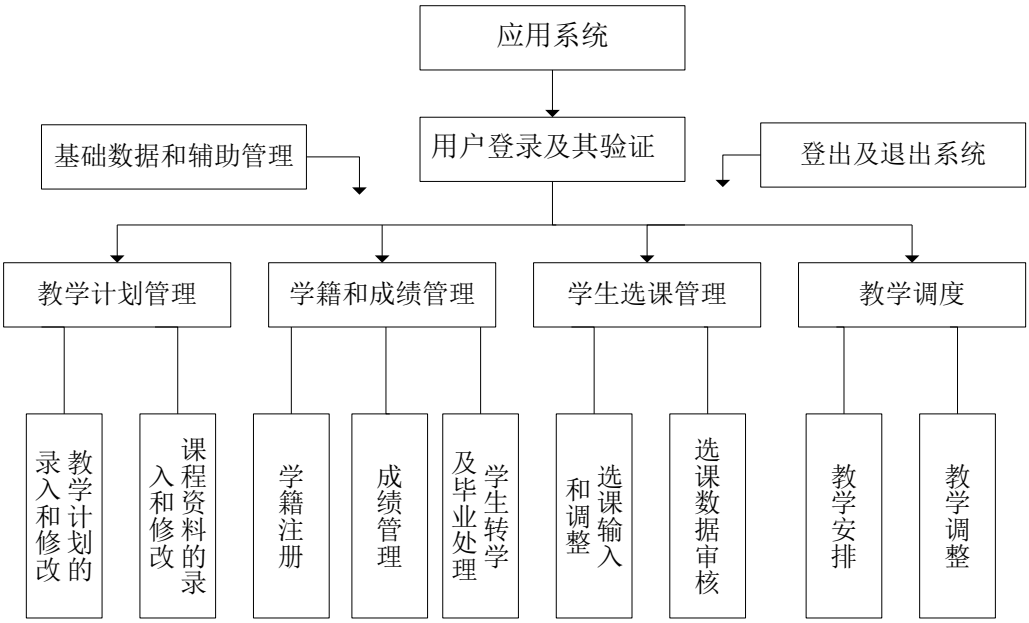

图 8.1 选课管理系统功能结构图

8.1.5 业务流程分析

一个简化的选课系统业务流程如图 8.2 所示：

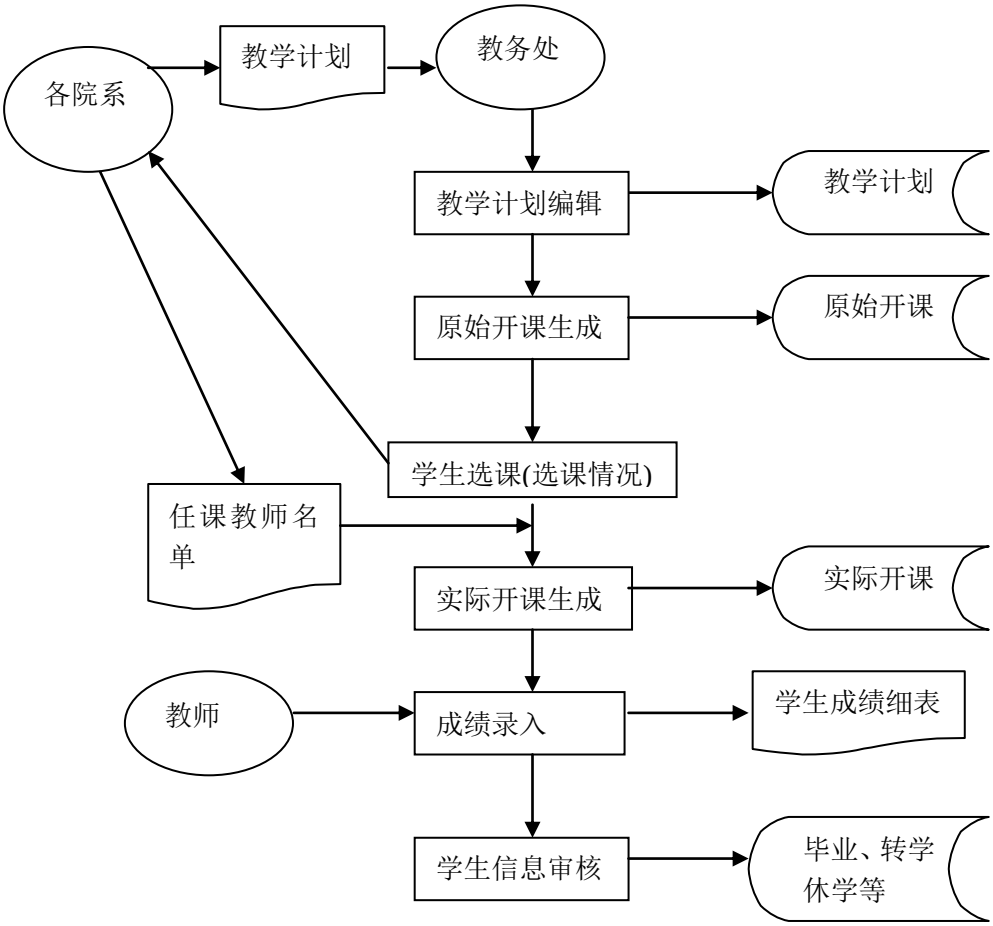

图 8.2 选课管理系统业务流程

8.2 需求描述

本阶段的成果的内容形式主要包括数据流图(Data Flow Diagram)和数据字典(Data Dictionary)。数据流图和数据字典是描述用户需求的重要工具以及阶段成果表达形式。它作为需求分析的成果和用户交流的主要手段和依据，是后续数据库设计的前提。设计人员从数据流图中可以比较充分地了解软件的结构，所以也是软件设计的重要依据。

调查了解用户的需求后，需要进一步表达用户的需求，分析和表达用户需求的方法很多，目前最常用的还是结构化分析法。该方法是基于数据流的需求分析方法，它

利用了图形的方式进行表达，容易学习和运用。

结构化分析法采用的是自顶向下、逐层分解的方式分析系统，即将系统的功能从宏观层面逐渐细化，达到最终的结构化分析方法主要使用以下几个工具：数据流图(Data Flow Diagram 简称 DFD)、数据字典(Data Dictionary 简称 DD)、判定表和判定树等。

数据流图描述了数据的来源和去向，以及所经过的处理；而数据字典是对数据流图中的数据流、数据存储和处理的明细描述。判定树和判定表用来描述据加工的逻辑构造。

不同的应用环境，对数据描述的细化程度会有所不同，常常应实际情况而定。下面就使用这两种工具来描述本例的用户需求，体现他们在实际中的应用方法。

8.2.1 数据流图

数据流图是通过系列符号及其组合来描述系统功能的输入、输出、处理或加工构造。

数据流图中使用的符号在各种书籍和资料上表达不尽相同，目前许多常用的一些流行的数据库辅助设计工具如 Microsoft Visio、Sybase PowerDesigner、Oracle Designer、Rational Rose、Erwin 等符号都不统一，我们这里以比较容易上手的 Visio 工具为例，针对 Gane-Sarson 模板中的符号作为参考：

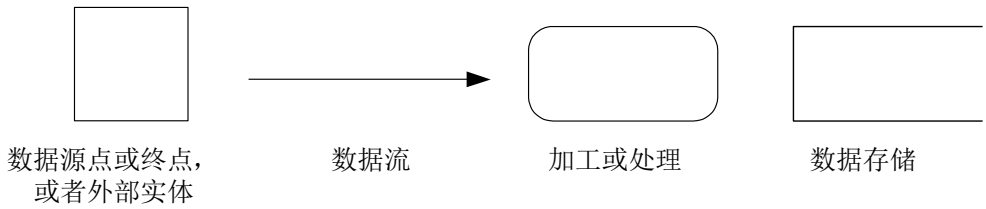

图 8.3 Gane-Sarson 模板中数据流图的基本元素

注意：DFD 表示数据被加工或处理的过程，箭头只是表示数据流动的方向，不能有分支、循环的情况。

数据流图命名规则之一：数据流图的中加工、处理过程一般采用动词及其短语；数据源点或终点、数据存储(数据文件或表单形式)、数据流(一项或多项数据)等一般为名词或名词短语。

数据流图命名规则之二：流图中的命令所使用的语言要基本上反映实际的情况，在整个 DFD 中必须要唯一，尽量避免含有像加工、处理、存储这样的元名称。

1。系统的全局数据流图

系统的全局数据流图，在具体的设计工具中往往也称为第 0 层或顶层数据流图，

主要是从整体上描述系统的数据流，反映系统中数据的整体流向，是设计者针对用户和开发者表达出来的一个总体描述。

我们经过对教学管理业务的调查、数据的收集和信息流程分析处理，明确了该系统的主要功能，分别为：制定学校各专业各年级的教学计划以及课程的设置；学生根据学校对所学专业的培养计划以及自己的兴趣，选择自己本学期所要学习的课程；学校的教务部门对新入学的学生进行学籍注册，对毕业生办理学籍档案的归档工作，任课教师在期末时登记学生的考试成绩；学校教务部门根据教学计划进行课程安排、期末考试时间地点的安排等，如图所示。

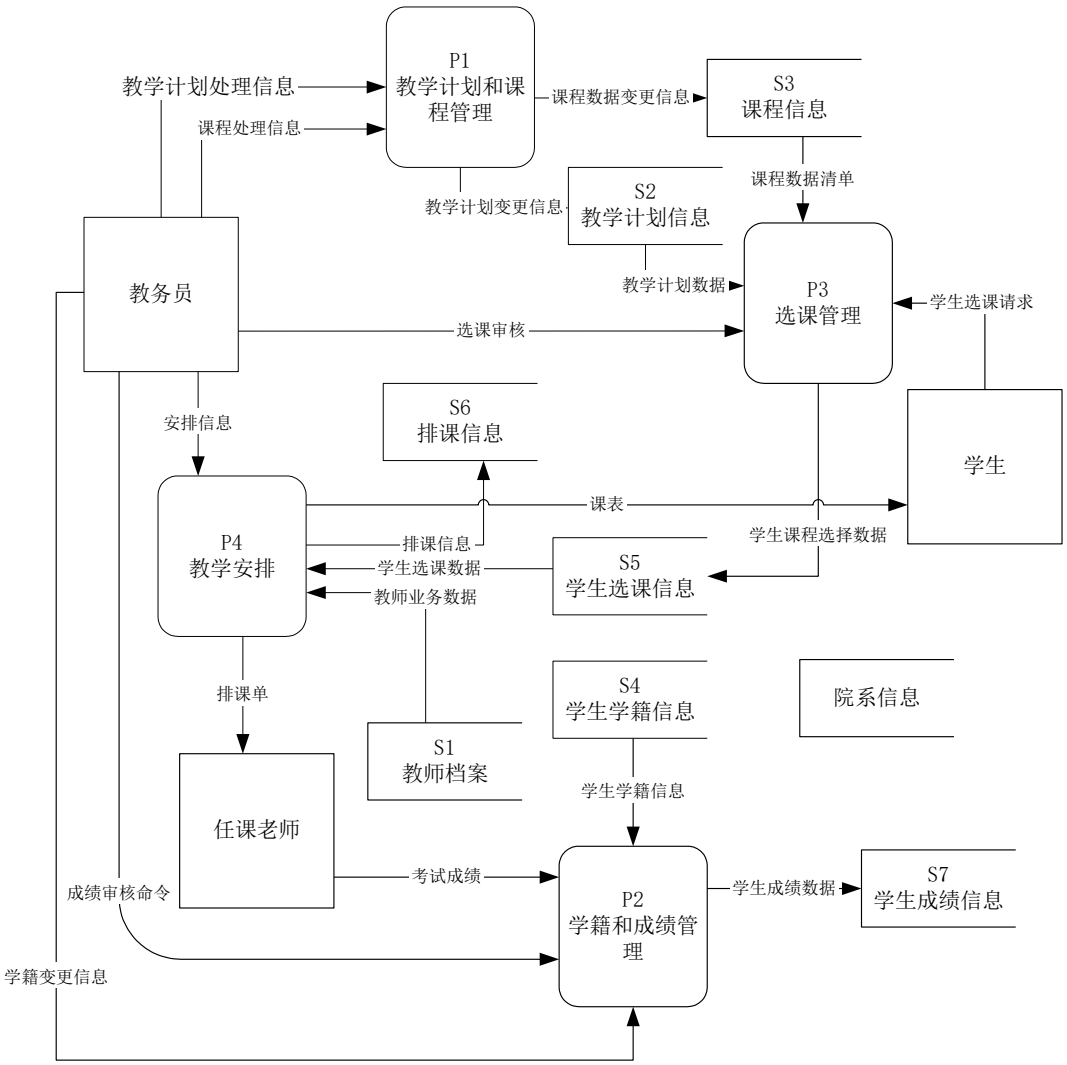

图 8.4 简化的选课管理系统 0 层数据流图

2。系统局部数据流图

全局数据流图，从整体上描述了数据流向和加工处理过程。但是一个较为复杂的系统来讲，要清楚地描述系统数据的流向和加工处理的每一个细节，仅用全局数据流图难以完成。因此需要在全局数据流图的基础上，对全局数据流图的某些局部单独放大，进一步细化，细化可以采用多级方式进行，便是所谓的分级数据流图来描述。这里以制定教学计划/学籍及成绩管理和选课等处理功能作细化的分析对象。

制定教学计划处理，主要分为 4 个子处理过程：教务员根据自己已有的课程信息，增补新开设的课程信息；调整课程信息；查询本学期的教学计划；制定新学期的教学计划。任课教师可以查询自己的教学计划。其处理过程如图 8.5 所示。

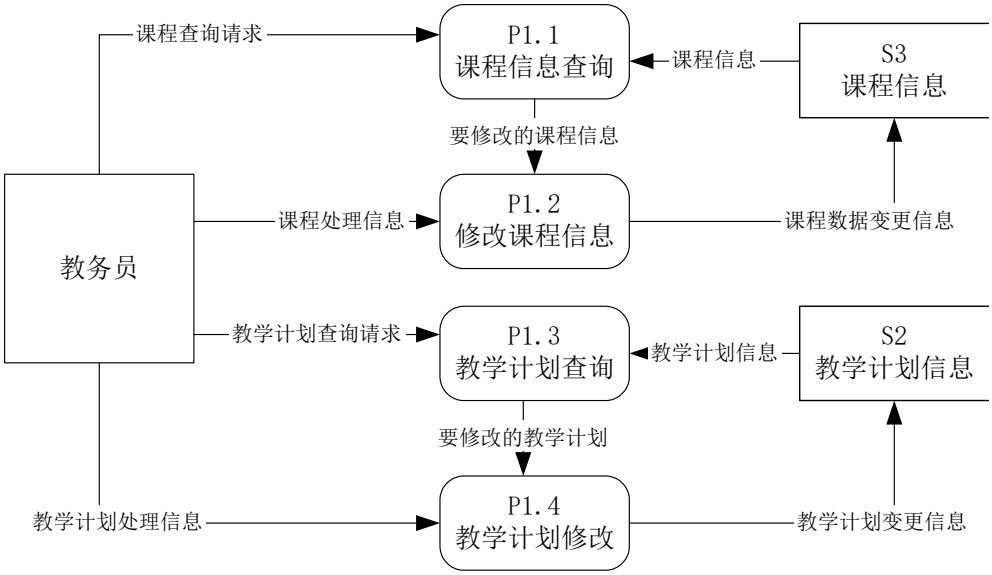

图 8.5　0 层 P1 的 1 层数据流图：制定教学计划

学籍及成绩管理相对比较复杂，教务员需要新生的学籍注册，毕业生的学籍和成绩的归档管理，任课教师输入学生的考试成绩后，需教务员审核并作认可处理，经确认的学生成绩不允许他人修改。其处理过程如图 8.6 所示。

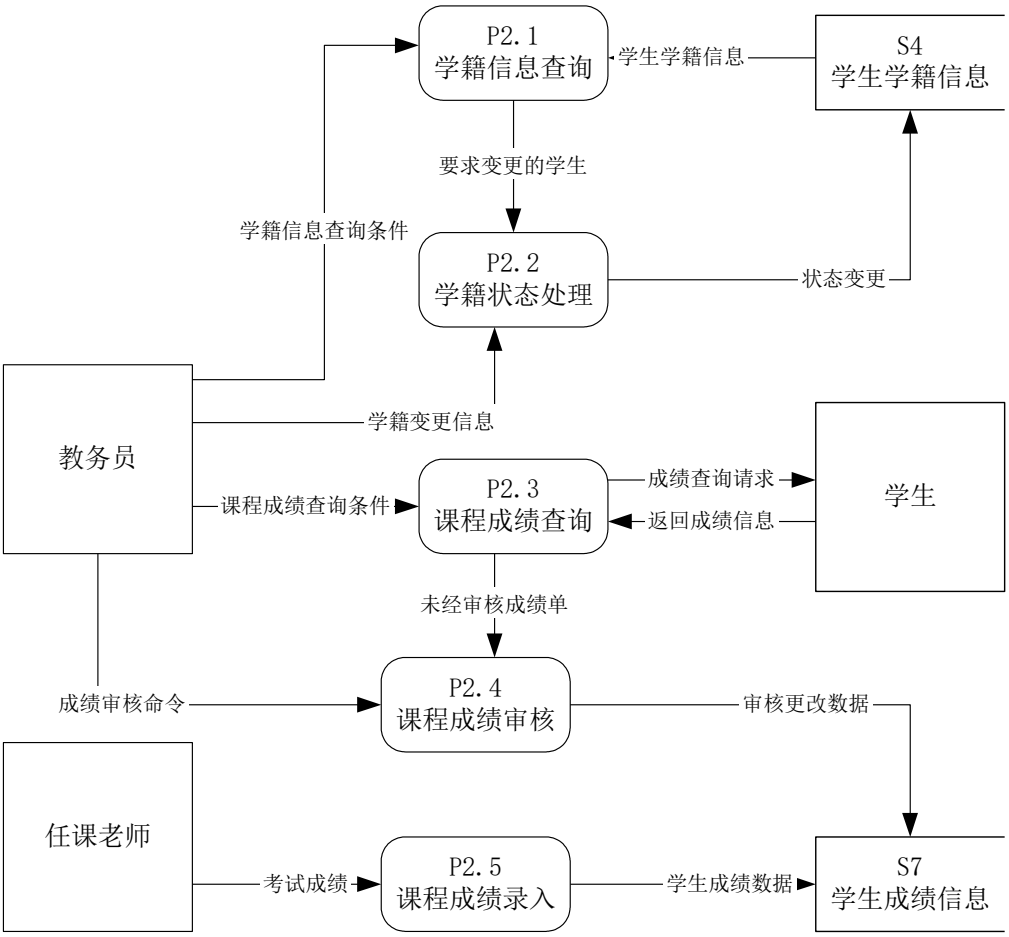

图 8.6　0 层 P2 的 1 层数据流图：学籍和成绩管理

选课管理中，学生根据学校对其专业制定的教学计划，录入本学期所选课程，教务员对学生选课记录进行审核，经审核得到的选课就为本学期的选课。其处理过程如图 8.7 所示。

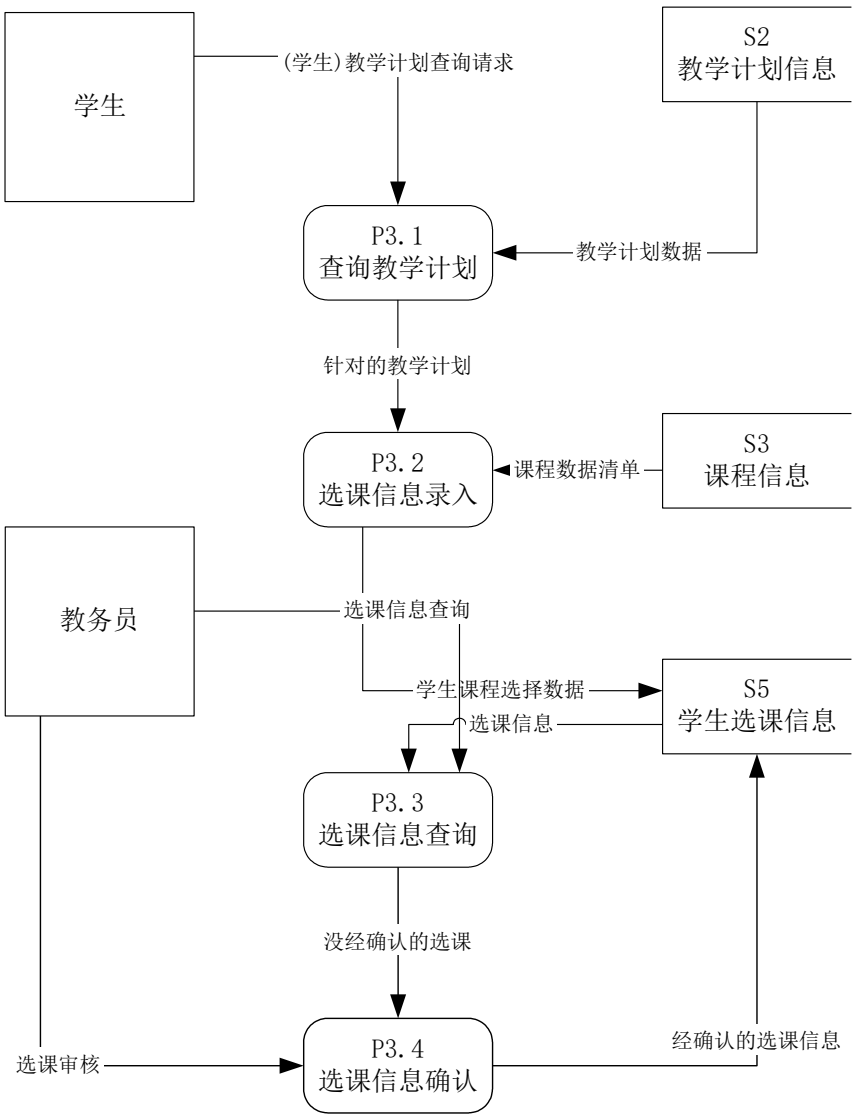

图 8.7 0 层 P3 的 1 层数据流图：选课管理

0 层 P4 的 1 层数据流图请读者自行描述。

我们可以使用许多的设计工具完成数据流图的创建，这些工具不但可以实现常用的数据流图的绘制，而且可以对多层的数据流图中的元素及其关系的正确性实现有效的检验，能帮助我们学习和理解数据流图的实现技术。本章有关的数据流图均使用 Microsoft Visio 工具进行绘制，相关的工具还有 Sybase 公司的 Power Designer 以及 Oracle 的 Designer 等，有兴趣的可以参考相关的资料或者下载试用版。

8.2.2 数据字典

数据流图表达了数据与处理的关系，数据流图作为直观的了解系统运行机理的手段，并没有具体描述各类数据的细节，只有通过数据字典进一步细化才能对系统的需求得到具体而确切的了解。数据字典用来说明数据流图中出现的所有元素的详细的定义和描述，包括数据流、加工处理、数据存储、数据的起点和终点或外部实体等。

数据字典包括的项目有：数据项、数据结构、数据流、数据存储、加工逻辑和外部实体。可使用一些符号来表示数据结构、数据流和数据存储的组成。

由于本实例涉及的数据字典项目较多，此处列举**"P3 选课管理"**处理功能中包含的几个对象加以描述。

1。数据流

表 8. 1 P3 中数据流的描述

序号	数据流名	来源	流向	组成	说明
1	(学生)教学计划查询请求	需要选课的学生	P3.1	班级号或学号	注意查询类别的区别
2	教学计划数据	S2 教学计划信息	P3.1	班级号+课程编号+开课学年+开课学期	
3	学生课程选择数据	P3.2	S5 学生选课信息	课程编号+年号+学期号	
4	选课信息查询	教务员	P3.3	班级号+课程号+学年+学期	

2。数据存储

表 8. 2 P3 中数据存储的描述

序号	数据文件	文件组成	关键标识	组织
1	S2 教学计划信息	班级号+课程编号+开课学年+开课学期	全部	按开课学年,学期,班级降序
2	S3 学生选课信息	学号+课程编号+开课学年+开课学期	全部	按开课学年,学期,班级降序
3	S5 课程数据清单	课程编号+课程名称+课程说明	课程编号	课程编号排序

3。处理过程逻辑

表 8. 3 P3 中处理过程逻辑的描述

序号	处理过程	编号	输入	输出	处理逻辑

1	查询教学计划	P3.1	学生选课查询请求+教学计划数据	针对的教学计划	针对选课请求进行查询
2	选课信息录入	P3.2	针对的教学计划	学生课程选择数据	根据学生对应的教学计划选择课程
3	选课信息查询	P3.3	选课信息查询+选课数据	没经确认的选课	根据班级和课程号检查对应的未确认的选课清单清单
4	选课信息确认	P3.4	选课审核+没经确认的选课	经确认的选课信息	选择选课清单进行确认

4。数据项

表 8. 4 P3 中数据项的说明

序号	数据项	数据对象说明	数据构成
1	学号	1{英文|数字}10	入学年号+班级序号+顺序号
2	选课时间	4{数字}-2{数字}-2{数字}	年+月+日
3	课程名称	1{汉字|英文|数字}20	
4	班级号	1{英文|数字}6	
5	教师编号	1{英文|数字}10	
6	开课学年	4{数字}	
7	开课学期	{1|2}	
8	课程说明	0{汉字|英文|数字}100	
英文=[‘a’…’z’|’A’…’Z’]			
数字=[‘0’…’9’]			

8.3 概念设计

上述的数据流图和数据字典共同构成了对用户需求的表达，它们是系统分析员(数据库管理员)在需求调查过程中和用户反复交互得到的。建设系统实际要处理的数据基本上已经在数据流图中得到体现，整个设计过程的后续步骤提供基础和依据。

概念设计就是通过对需求分析阶段所得到的信息需求进行综合、归纳与抽象，形成一个独立于具体数据库管理系统的概念模型，主要的手段为 ER 图。

在概念设计阶段，主要采用的设计手段目前还是实体联系模型(E-R Model)。绘制E-R 图的关键是确定 E-R 图的各种结构，包括实体、属性和联系。大部分的流行建模工具(**Power Designer、Oracle Designer、ERwin 等**)也都包含了对 E-R 设计手段的支持。

8.3.1 实体

要建立系统的 E-R 模型的描述，需进一步从数据流图和数据字典中提取系统所有的实体及其属性。这种提出实体的指导原则如下：

① 属性必须是不可分的数据项，即属性中不能包含其它的属性或实体

② E-R 图中的关联必须是实体之间的关联，属性不能和其它实体之间有关联

由前面分析得到的数据流图和数据字典，可以抽象得到实体主要有 5 个：学生、教师、课程、院系、班级。

(1) 学生实体属性有：学号、姓名、出生年月、性别、电话、系编号。

(2) 教师实体属性有：教师编号、教师姓名、性别、职称、出生年月、电话、电子邮件。

(3) 课程实体属性有：课程编号、课程名称、课程学时、课程学分。

(4) 院系实体属性有：系编号、系名称、负责人。

(5) 班级实体属性有：班级编号、班级名称。

8.3.2 系统局部 E-R 图

在需求分析阶段我们采用的是自上而下的分析方法，那么要在其基础上进一步作概念设计我们面临的是细化的分析数据流图以及数据字典，分析得到实体及其属性后，进一步可分析各实体之间的联系。

学生实体和课程实体存在选修的联系，一个学生可以选修多门课程，而每门课也可以被多个学生选修，所以它们之间是多对多的联系(n:m),如图 8.8。

教师实体和课程实体存在讲授的联系，一名教师可以讲授多门课程，而每门课也可以被多个教师讲授，所以它们之间是多对多的联系(n:m),如图 8.9。

学生实体和班级实体存在归属的联系，一个学生只能属于一个班级，而每个班级可以包含多个学生，所以班级和学生之间是一对多的联系(1:n),如图 8.10。

班级实体和系之间存在归属的联系，一个班级只能属于一个系，而每个系可以包含多个班级，所以班级和系之间是一对多的联系(1:n),如图 8.11。

教师实体和系实体之间存在归属的联系，一个教师只能属于一个系，而每个系可以拥有多名教师，所以教师和系之间是一对多的联系(1:n),如图 8.12，但是教师中会有一位充当该系的主任(正)，可见教师和系之间也存在一种一对一的领导关系(1:1),如图 8.12。

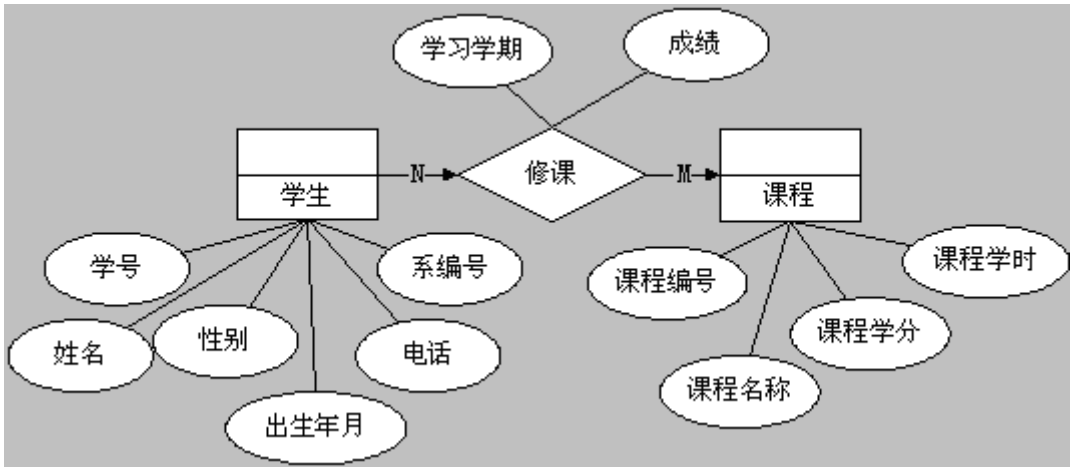

图 8.8 “学生-课程” 选课关系

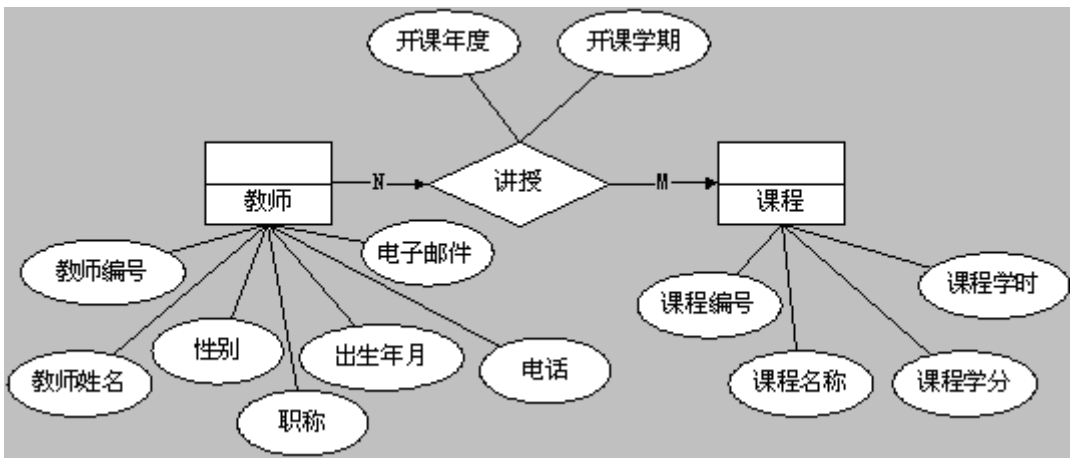

图 8.9 “教师-课程”实体间的关系

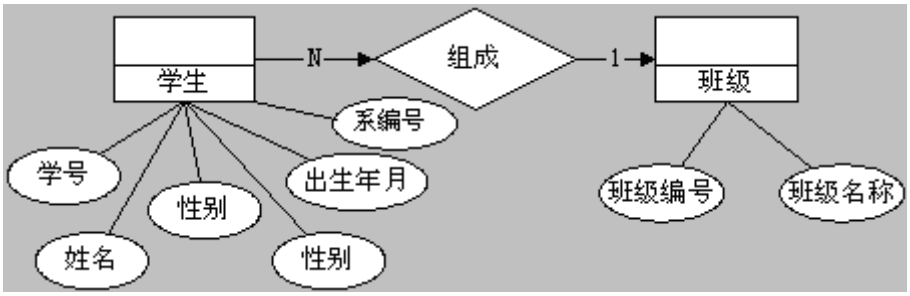

图 8.10 “学生-班级”的组成关系

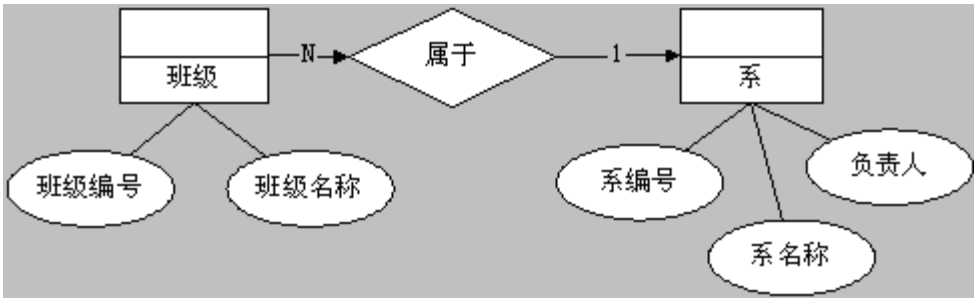

图 8.11 “班级-系”的属于关系

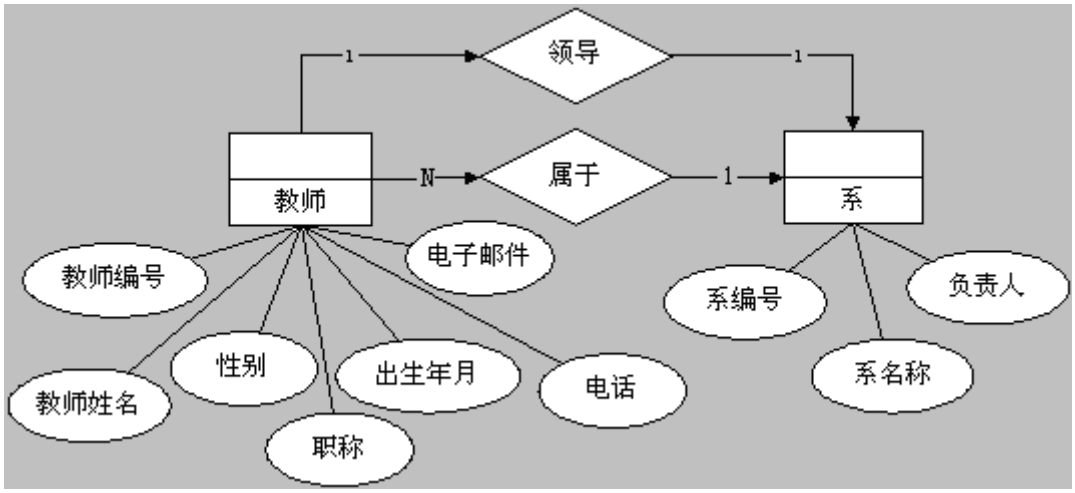

图 8.12 “教师-系”实体间的关系

8.3.3 系统全局 E-R 图

系统的局部 E-R 图，仅反映系统局部实体之间的联系，但无法反映系统在整体上实体间的相互联系。而对于一个比较复杂的应用系统来说，这些局部的 E-R 图往往有多人各自分析完成的，只反映局部的独立应用的状况，在系统整体的运作需要时，他们之间有可能存在重复的部分或冲突的情况，如实体的划分、实体或属性的命名不一致等，属性的具体含义(包括数据类型以及取值范围等不一致)问题，都可能造成上述提到的现象。

为解决这些问题，必须理清系统在应用环境中的具体语义，进行综合统一，通过调整消除那些问题，得到系统的全局 E-R 图。

从实际的情况以及上述的局部 E-R 图我们可以得知，学生实际修学某门课时必须只能对应一位老师的该门课。因此，可以使用一个聚集来表达学生参加实际授课课程的学习关系，会更加切合实际。各局部 E-R 存在不少的重复的实体，经过上述聚集分析和合并得到系统全局的 E-R 图如图 8-13 所示。该全局 E-R 图基本上不存在关系的冗余状况，因此它已经是一个优化的。

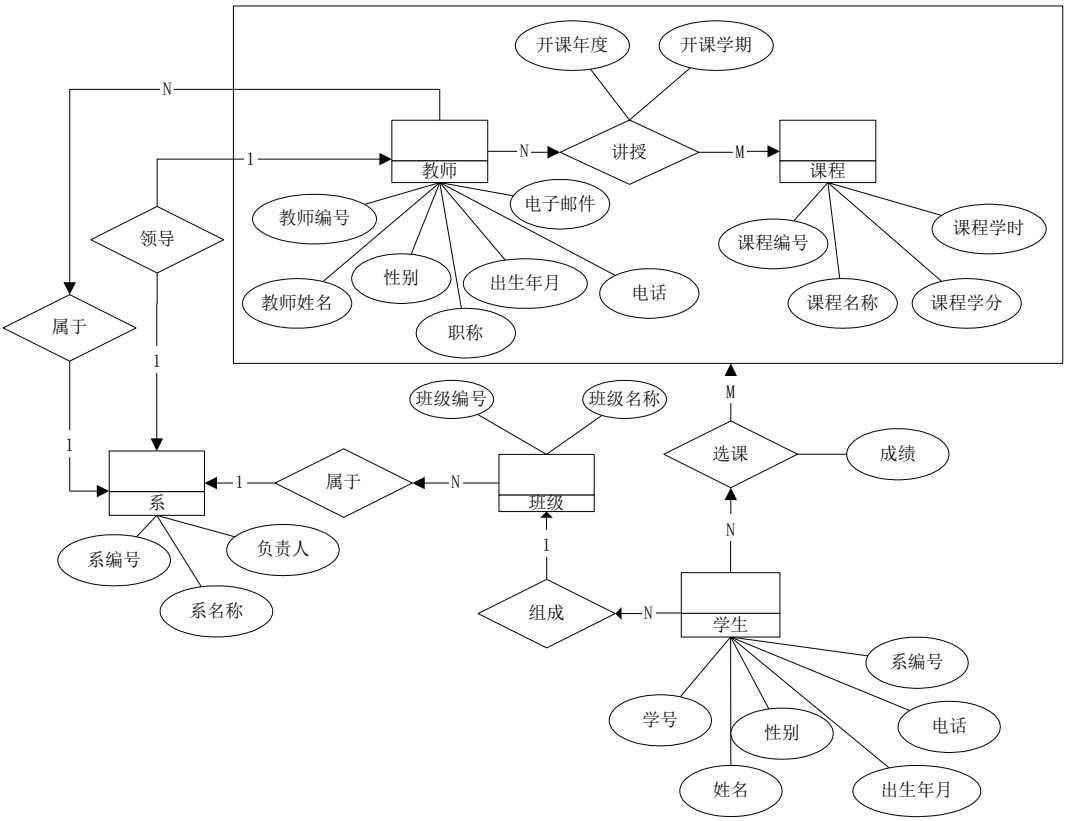

图 8. 13　选课管理系统的全局 ER 图

8.4　逻辑设计

逻辑设计就是把 E-R 图转换成关系模式，并对其进行优化。

8.4.1　ER 图到关系模式的转换

在概念设计阶段得到的数据模型，是独立于具体 DBMS 产品的信息模型。在逻辑设计阶段就是将这种模型进一步转化为某一种（某些类）DBMS 产品支持的数据模型。目前大部分的流行的数据库管理系统(SQL Server、Sybase 、Oracle、DB2 等)基本上都是基于关系的数据模型，包括该系统将采用的 SQL Server2000 数据库系统，因此，应将概念设计阶段的 E-R 图模型转化为关系数据模型。

首先，课程实体以及他们的联系。任课教师与课程之间的是多对多的联系类型，因此，将任课教师、课程以及讲授联系分别设计成如下的关系模式：

教师(教师编号,教师姓名,性别,职称,电话,系编号)

课程(课程编号,课程名称,课程学分,课时)

讲授(教师编号，课程编号，课程编号，开课年度，开课学期)

院系实体和班级之间是一对多的联系类型，所以只要两个关系模式就可表示，其中联系可以放到班级的实体中：

系(系编号、系名称、系主任)

班级(班级编号，班级名称，系编号)

班级实体和学生实体之间是一对多的联系类型，所以也可以只使用两个关系模式来表示。由于“班级”关系模式在上面已经给出，因此，只要再给出一个学生的关系模式，它们间的联系则被放在该关系模式中：

学生(学号，姓名，性别，出生年月，电话，班级编号)

学生实体与讲授是聚集方式的联系类型，它们之间的关系是多对多的关系，可以使用如下关系模式来表示：

学生选课(课程编号，学号，教师编号，开课年度，开课学期，成绩)

8.4.2　关系模式的规范及调整

在提出关系模式后，我们必须在规范化和实际要求进行优化，这实际上是一个权衡的过程。如果设计没有完全规范化，如可能用于决策支持(与需要大量更新的事务处

理相对)的数据库(如数据仓库)则可能没有冗余更新，而且可能对查询更易于理解和更高效。不过，在数据库应用程序内，未规范化的数据在设计过程更需要注意。一般的策略是以规范化设计为出发点，然后出于特定因素有条件地非规范化某些表,以达到系统总体的优化目的。

首先，需要我们确定上面建立的关系模式中的函数依赖，一般在作需求分析时就了解到一些数据项的依赖关系，如教师的编号决定了教师的姓名和其它的数据项信息，而实体间的联系本身也是反映了一种函数依赖关系，但是这不是研究的对象，我们针对的是在一个关系模式中的函数依赖对象。

其次，对上一步确立的所有函数依赖进行检查，判别是否存在部分函数依赖以及传递函数依赖，针对有的依赖通过投影分解，消除在一个关系模式中存在的部分函数依赖和传递函数依赖。

大部分数据库系统只要满足第三关系范式就可以，这也是我们这里规范化的基本要求。由于需求分析阶段的方法得当，经过简单的分析可以看出，上述所有关系中每个数据项都是基本的，任何非主属性都不存在对主码的部分依赖，也不存在非主属性存在着对主码的传递依赖。可见，以上所有的关系模式都属于 3NF。

在实际的应用中，关系模式的规范化程度并不是越高越好，因为在关系模式的规范化提升过程中，必须进行着将一个关系模式分解成为多个关系模式的过程。这样，在以后执行查询时，如果需要相关的信息，就必须作多个表的连接方能达到查询的目的，这无疑给系统增加一定的开销，特别存在很多用户同时访问或者关系中存在许多元组等因素其负担会越加明显。为了兼顾性能的需要，在适当的时候可能需要对相关程度比较高的一些关系模式进行合并处理，或者在关系模式中增加相关程度比较高的属性等。这是有可能选择第二范式甚至第一范式。

如果系统存在很多的元组数(记录数)，特别当记录达到百万甚至千万条记录时，系统的查询效率可能会受到明显的影响，分析关系模式的特点，可以根据某些关键属性不同将关系模式分解为多个关系模式，模式之间通过共同的主键一一对应起来。

前面设计出的教师、课程、班级、院系和学生等关系模式基本上是实际应用中事物的直接对应，一般不需要优化。

由于实际讲授安排都是后于学生选课行为之后进行的，因此“讲授”关系模式尽管是单独地放映了某一门课在某一个学期由某一个老师讲授，而一门课在一个学期也可以有多个老师主讲，关键还在于老师的讲课针对哪个(些)班级进行。“授课”关系的有关信息其实完全由课程学习关系模式中得到反映。所以可以合并“授课”到“课程学习”中去，实际上去除“授课” 关系模式便可。

对于“课程学习”关系模式，在上述需求分析阶段已经指出，选课后和成绩记录后都需经过教务员的审核确认方生效，对应需要增加审核信息属性。因此，对于“课程学习”关系模式修改为：课程学习(课程编号，学号，教师编号，开课年度，开课学期，成绩，课程审核，成绩审核)。

对于分析中提及的学生状态处理(转学、退学、休学或者毕业等的情况)，为了简单

化，不妨给关系模式增加一个“状态”的属性，成为：学生(学号，姓名，性别，出生年月，电话，班级编号，状态)。

为了满足实际应用对系统的系统要求，必须对使用系统的用户如教师和学生增加登录的验证口令，因此需要在“教师”和“学生”的关系模式中增加口令属性。自然地，如果根据其它的安全应用要求，还可以设置学生的登录地点如通过增加 IP 属性来达到目的等。

8.4.3 各个数据表的表结构设计

在上述经由 E-R 模型得到关系模式并且得到适当的调整后，我们可以结合在需求表述中数据字典包含的数据项信息，得到数据库的表结构。我们应该根据 8.4.1 节的内容，具体设计各个数据表的表结构，包括表名，表中各列的字段名、数据类型、数据长度和表的主键和外键；还要考虑应该建立哪些索引以及索引的类型。

需要指出的是，考虑到系统的统一兼顾如对数据库管理员和后续软件开发中对数据库管理以及编程引用的便利，表名和字段名的命名应该由表名的英文含义的词语为主或以其缩写字母构成；同时要为各个表名和字段名作出完整的中文文档说明。

表 8.1 至表 8.8 给出了数据库中各个数据表的表结构。

表 8.5　数据库中表清单

数据库表名	关系模式名称	备注
Teacher	教师	教师信息表
Student	学生	学生学籍信息表
Course	课程	课程基本信息表
Class	班级	班级基本对照表
StuCourse	学生选课	选课-授课合成信息表
Department	系	院系基本信息表
Schedule	教学计划	教学计划安排表

表 8.6　学生信息表 Student 字段信息列表

字段名称	含义属性	类型	长度	备注
Snum	学号	char	10	主键,也可以作为登录标识
Sname	学生姓名	nvarchar	6	Not null
Ssex	性别	nchar	2	男、女(M/F)
Sbirth	出生年月	datetime		
Clnum	班级号	varchar	6	所在班级编号,外键→Classes.Clnum

Email	电子邮件	nvarchar	40	支持中文邮箱
Passwd	密码	varchar	20	密码，可以是数字英文和符号等
Status	状态	nvarchar	8	表示在校或毕业或转学等

表 8.7　教师基本信息表 Teacher 字段信息列表

字段名称	含义属性	类型	长度	备注
Tnum	教师编号	char	10	主键,也可以作为登录标识
Tname	教师姓名	nvarchar	6	Not null
Tsex	性别	nchar	2	男、女(M/F)
Title	职称	nvarchar	8	教授、副教授…
Tphone	联系电话	char	15	
Email	电子邮件	nvarchar	40	支持中文邮箱
Tbirth	出生年月	datetime		
Passwd	密码	varchar	20	密码，可以是数字英文和符号等
Dnum	系编号	varchar	6	外键→Depart.Dnum

表 8.8　系基本信息表 Depart 字段信息列表

字段名称	含义	类型	长度	备注
Dnum	系编号	varchar	6	主键
Dname	系名称	nvarchar	10	Not null
Director	系主任	varchar	10	外键→Teacher.Tnum

表 8.9　班级信息表 Classes 字段信息列表

字段名称	含义	类型	长度	备注
Clnum	班级编号	varchar	6	主键
Cname	班级名称	nvarchar	10	Not null
Desscription	班级说明	nvarchar	100	如专业,本专科
Dnum	系编号	varchar	6	外键→ Depart.Dnum

表 8.10　课程基本信息 Course 字段信息列表

字段名称	含义	类型	长度	备注

Cnum	课程编号	varchar	10	主键
Cname	课程名称	varchar	20	Not null
Credit	学分	numeric	3,1	
Period	课时	int	3	

表 8. 11 学生选课信息表 StuCourse 字段信息列表

字段名称	含义	类型	长度	备注
Snum	学号	varchar	10	外建→Student.Snum
Cnum	课程编号	varchar	10	外建→Course.Cnum
Tnum	教师编号	varchar	10	外建→Teacher.Tnum
Ynum	开课年度	int	4	例如：2008
Term	开课学期	int	1	1|2
Grade	成绩	numeric	4,1	{0…100}注意考查课的数字化
CAuditor	选课审核者	nvarchar	6	直接取其姓名
Gauditor	成绩审核者	nvarchar	6	直接取其姓名

表 8. 12 教学计划信息表 Schedule 字段信息列表

字段名称	含义	类型	长度	备注
Cnum	课程编号	varchar	10	外建→Course.Cnum
Clnum	班级编号	varchar	6	外建→Classes.Clnum
Ynum	开课年度	int	4	例如：2008
Term	开课学期	int	1	如 1|2 针对一个学年只有两个学期情形

8.5 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：

① 使用哪种类型的磁盘硬件，如 RAID（独立磁盘冗余阵列）设备；

② 数据在磁盘上如何放置即数据的分配策略；

③ 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；

④ 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。

8.5.1 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 Server 2000 中实现该系统的数据库关系模式(SQL DDL)。

8.5.2 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：

（1） **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。

（2） **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。

（3） **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：

```
--创建学生选课管理系统的数据库“学生选课”
CREATE DATABASE 学生选课
```

```
ON
Primary
(NAME=css_Data1, FILENAME= 'C:\css\data\csmain.mdf '),
(NAME=css_Data2, FILENAME= 'D:\css\data\cssecd1.ndf '),
(NAME=css_Data3, FILENAME= 'E:\css\data\cssecd2.ndf ')
LOG ON
(NAME=css_Log, FILENAME= 'F:\css\data\cslog.ldf ')
```

8.5.3 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：

```
--创建系基本信息表 Depart
CREATE TABLE Depart
(
   Dnum varchar(6) PRIMARY KEY,
   Dname nvarchar(10)not null,
   Director varchar(10)
)
--创建班级基本信息表 Classes
CREATE TABLE Classes
(
   Clnum varchar(6) PRIMARY KEY,
   Clname nvarchar(10) not null,
   Dnum varchar(6),
   Bdate datetime,
   Description nvarchar(100),
Constraint ClDnumFK foreign key(Dnum) References Depart(Dnum)
)

--创建学生基本信息表 Student
CREATE TABLE Student
(
   Snum varchar(10) PRIMARY KEY,
   Sname nvarchar(6) not null,
   Ssex nchar(2),
   Sbirth datetime,
```

```
    Email nvarchar(40),
    Passwd varchar(20),
    Clnum varchar(6),
    Status nvarchar(6),     --0:正常,1:毕业,2:休学,3:退学,4:转学,5:其它
    Constraint ClnumFK foreign key(Clnum) References Classes(Clnum),
    Constraint SSexchk Check(Ssex='男'or Ssex='女')
)

--创建教师基本信息表 Student
CREATE TABLE Teacher
(
Tnum char(10) PRIMARY KEY,
    Tname nvarchar(6) not null,
    Tsex nchar(2),
    Tbirthdate datetime,
    Title nvarchar(8),
    Dnum varchar(6),
Tphone char(15),
Email nvarchar(40),
Passwd nvarchar(20),
Constraint TDnumFK foreign key(Dnum) References Depart(Dnum),
    Constraint TSexchk Check(Tsex='男' or Tsex='女')

)

--创建课程基本信息表 Course
CREATE TABLE Course
(
    Cnum varchar(10) PRIMARY KEY,
    Cname nvarchar(20) not null,
    Period int(3),
    Credit numeric(3,1)
)

--创建教学计划基本信息表 Schedule
CREATE TABLE Schedule
(
```

```
   Cnum varchar(10)    FOREIGN KEY REFERENCES Course(Cnum),
   Clnum varchar(6) FOREIGN KEY REFERENCES Classes(Clnum),
   Ynum int(4),
   Term int(1),
)

--创建学生选课基本信息表 StuCourse
CREATE TABLE StuCourse
(
   Snum varchar(10) FOREIGN KEY REFERENCES Student(Snum),
   Cnum varchar(10)    FOREIGN KEY REFERENCES Course(Cnum),
   Clnum varchar(6)    FOREIGN KEY REFERENCES Classes(Clnum),
   Tnum char(10) FOREIGN KEY REFERENCES Teacher(Tnum),
   Ynum int(4),
   Term int(1),
   Grade numeric(4,1),
   Cauditor nvarchar(6),
   Gauditor nvarchar(6)
)
--增加院系主任的外键
ALTER TABLE Depart
ADD FOREIGN KEY(Director) REFERENCES Teacher(Tnum)
```

8.5.4 索引设计

考虑到本系统主要的业务目的是学生的选课，此过程访问最频繁的功能是集中在教学计划的查询和选课信息表的查询上以及学生的学籍信息查询上。教学计划的制定基本上是一次为主，基本查询功能是基于学年、学期和班级联合条件进行的，所以可以考虑在 Schedule 表上建立聚簇索引：

```
Create Clustered Index StuCourseIndex on StuCourse(Clnum,Ynum,Term)
```

同样地，学生学籍信息的查询基本上集中在学号上，因此可以建立聚簇索引：

```
Create Clustered Index SnumIndex on Student(Snum)
```

而对于选课信息由于聚集关系以及整合的结果，该表上的变更也是比较频繁的，所以可以考虑建立普通的索引：

```
Create Index ScheduleIndex on Schedule(Cnum,Clnum,Ynum,Term)
```

当然，索引的设计不但需要仔细考虑查询功能的特点以及数据变更的权衡因素，而且也需要结合整体数据量的大小和增长的可能等（注意：本章需求部分没有对数据

量大小作分析），但是关键还是实际的性能需求。

8.5.5 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。

本章小结

在本章中，我们首先复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详细讲解，更为实际地说明了数据库设计的具体操作流程。

通过对本章的学习，读者可以在具体的实践活动中贯彻前面各个章节的理论思想，为以后的工作打下坚实的基础。