**存档编号**

**华北水利水电大学**

**North China University of Water Resources and Electric Power**

**毕 业 设 计**

**题目 教务系统的设计与实现（教师模块）**

**学 院 信息工程**

**专 业 计算机科学与技术**

**姓 名 潘靓妮**

**学 号 201215327**

**指导教师 郑作勇**

**完成时间 2016.05.25**

教务处制

独立完成与诚信声明

本人郑重声明：所提交的毕业设计（论文）是本人在指导教师的指导下，独立工作所取得的成果并撰写完成的，郑重确认没有剽窃、抄袭等违反学术道德、学术规范的侵权行为。文中除已经标注引用的内容外，不包含其他人或集体已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中作了明确的说明并表示了谢意。本人完全意识到本声明的法律后果由本人承担。

毕业设计（论文）作者签名： 指导导师签名：

签字日期： 签字日期

毕业设计（论文）版权使用授权书

本人完全了解华北水利水电大学有关保管、使用毕业设计（论文）的规定。特授权华北水利水电大学可以将毕业设计（论文）的全部或部分内容公开和编入有关数据库提供检索，并采用影印、缩印或扫描等复制手段复制、保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交毕业设计（论文）原件或复印件和电子文档（涉密的成果在解密后应遵守此规定）。

毕业设计（论文）作者签名： 导师签名：

签字日期： 签字日期：

目 录

[摘 要 I](#_Toc17032)

[Abstract II](#_Toc29462)

[第1章 绪论 1](#_Toc30506)

[1.1研究背景 1](#_Toc20004)

[1.2 目的与意义 1](#_Toc29255)

[1.3 国内外研究现状 1](#_Toc21147)

[1.3.1国外高校教务系统的研究现状  2](#_Toc13208)

[1.4 课题主要内容 4](#_Toc4640)

[1.5 论文组织结构 5](#_Toc7102)

[第2 章 系统实现的相关技术 6](#_Toc21048)

[2.1 .NET相关技术 6](#_Toc11260)

[2.1.1 NET Framework 6](#_Toc389)

[2.1.2 Visual Studio.2010 7](#_Toc2334)

[2.2 数据库编程 7](#_Toc1963)

[2.3系统开发和运行环境 12](#_Toc557)

[第3章 系统分析与概要 14](#_Toc19869)

[3.1 系统需求分析 14](#_Toc8730)

[3.1.1流程分析 15](#_Toc2345)

[3.1.2 功能 16](#_Toc19349)

[3.1.3 数据需求 16](#_Toc28148)

[3.2 开发设计原则 17](#_Toc6300)

[3.3 总体结构设计 17](#_Toc14152)

[第4章 教师模块的实现 19](#_Toc17291)

[4.1 主页面 19](#_Toc22880)

[4.2 教师模块 19](#_Toc19895)

[4.2.1 个人信息页面 20](#_Toc24092)

[4.2.2 成绩录入页面 20](#_Toc30685)

[4.2.3 教学任务查询页面 21](#_Toc20543)

[4.2.4 成绩的查看页面 22](#_Toc734)

[第5章 管理员模块 23](#_Toc29088)

[第6章 系统的测试与分析 24](#_Toc9131)

[6.1 系统的测试和意义 24](#_Toc7168)

[6.2 系统测试的结果 25](#_Toc11091)

[第7章 软件维护 26](#_Toc26968)

[第8章 总结 27](#_Toc11388)

[参考文献 30](#_Toc26597)

[致 谢 32](#_Toc29153)

[附 录 33](#_Toc32565)

[附录I 33](#_Toc30091)

[外文原文 33](#_Toc5894)

[中文译文 45](#_Toc18126)

[毕业设计任务书 55](#_Toc21945)

[开题报告 56](#_Toc32098)

[附录II 60](#_Toc15258)

[程序清单 60](#_Toc2572)

**教务系统管理——教师模块**

# **摘** **要**

教务系统是一个在学校中常常会用到的工具，方便学生选课，以及查询课程信息和课程成绩，同时，也为老师的工作带来便捷，如：为老师提供所上课程的信息，以及学生成绩的录入，查看老师的相关授课安排，方便工作，而且在相近计算机技术的飞速发展下，越来越多的人和企业都会使用相关技术软件来完成一些之前只能人工操作的工作，网络化的教务管理系统更是一个非常重要且具有现实意义的发展方向。同时可以让教职工能够更加高效的完成相关的工作，让学生更容易的查询到自己的成绩以及相关信息。实现资源信息共享和充分利用，较少人力资源的浪费，从而进一步实现工作效率的提升。

本教务系统（教师模块）有效的对相关工作进行分层次处理，方便了教职工工作，提高个人效率，有序的对教学信息进行采编和发放。本文中首先介绍关于教务系统管理的研究背景及其现实意义，其次介绍了制作过程中所采用的应用技术，本系统采用三次架构设计模型，主要开发工具是Visual Studio 2010,数据库SQL Server,使用Entity Framework对数据库进行相关操作。之后提出了毕设任务以及需求分析，进行系统的层次构建以及功能和数据库的设计，并进行相关的实践操作，实现功能。最后进行软件测试以及调试，提出自己的不足之处以及期望。实现信息共享和查询。

**关键字**：教务系统；数据库；教师模块

**中图分类号**：TP311.1

**Educational Management System--Teacher Module**

# **Abstract**

Senate system is a in school in the often will with to of tool, convenient students selected class, and query courses information and courses results, while, also for teacher of work brings convenient, as: for teacher provides by Shang courses of information, and students results of entry, view teacher of related taught arrangements, convenient work, and in similar computer of rapid development Xia, increasingly more of people and enterprise will using related technology software to completed some zhiqian only artificial operation of work, Networked educational administration system is a very important and significant development.Also allows more efficient completion of work related to the teaching and administrative staff, queries that make it easier for students to their own performance, and related information.Realization of information sharing and full utilization of resources, less waste of human resources, thus further efficiency improvements.The educational system (teacher module) valid for hierarchical processing related work, facilitating staff work, improve personal effectiveness, ordered the teaching information gathering and distribution.This article first introduces the background study on the educational administration management and its practical significance, then the technology used in the production process, this system based on MVC design model, the main development tool is a Visual Studio 2010, SQL Server databases, using the Entity Framework for database-related operations.After the set tasks and needs analysis, system construction and functional levels, and database design and related practices, to implement functionality.Finally, software testing and debugging, and put forward his own shortcomings as well as expected.Share information and queries.

**Keywords**: educational system；the database；the teacher module

# 第1章 绪论

## 1.1研究背景

教务系统是每个高校经常会用到的应用软件，不仅方便学生和老师的资源管理，而且对于成绩的查询以及录入，课程的查询以及录入等方面带来了便捷的操作，为学校的工作提供了有效率的可靠的服务，而这些是传统管理模式无法满足的。随着学分制和教学模式的改变，为了更高效的实现无纸化办公，教务系统的信息化以及网络化就显得相当重要了，而所做的就是在新的教学管理体制下，正确的利用计算机，利用校园网，以全局设计为基准进行相关内容的设计，考虑容错性以及扩展性，实现资源信息的共享，将学生、教师、课程等进行分析设计，方便工作和提高效率。

通过计算机软件的管理及应用，一方面带给我们更高效率的工作环境、一方面可以降低资源的不必要浪费，使得操作的时间更少，而且错误率更低，这些都会使得我们现在乃至将来的工作更加的方便快捷。

## 1.2 目的与意义

如何提高工作效率，如何行之有效的看到的增添，修改，查询想要看到的内容，如何实现无纸化信息传递以及共享，这都是教学系统的目的和意义所在，在传统的管理模式下，更加浪费人力、物力、财力等相关资源。而新模式下的教务系统重在对数据进行深度挖掘，而行为则是从大量的数据中发现规律，为我们所利用，综合经验知识和理论知识，解决我们所遇到的问题，对于想要了解的信息，可以利用查询联机分析处理或者是通过其他的相关工具去获取。21世纪的今天，更需要利用计算机来方便我们的日常包括工作，提高我们的工作效率。

## 1.3 国内外研究现状

从数据库以及.NET被提出到目前为止，以及有很多人将研究重点放在了从发现方法到系统应用这一部分，越来越多的人注意到了实践生活中计算机应用的必要性和快捷性，而教务系统又是高校管理的一部分，所以也越来越引起人们的关注。同时国内外也出现了许多基于SQL以及.NET的关于教务管理系统的设计。

### 1.3.1国外高校教务系统的研究现状

因为有较大规模而且技术稳定的队伍来提供可靠的服务，而且有属于自己的高带宽的专用网络用来调整教育目标制定专用的属于自己国家实际情况的改革方案，用于推进教育信息化的建设，所以世界上诸多发达国家都很重视信息技术对教育的影响和作用，对于这一块的经费投入也比其它一些国家要多出很多，但从现实看来，这是一项获利颇大的工程。国外许多大学杜宇教学系统的现代信息化管理也已经进行了很多年的探索和研究，甚至可以追溯到六七十年代之前，对相关技术的探寻以及研究到现今已比较成熟，形成了一套规范科学的管理模式。而据资料显示在国外，数字化校园的概念最早提出是由美国著名学院麻省理工学院在20世纪70年代提出的，在其后的四十多年的努力建设，最终构建出一个相对稳定和成熟的数字化校园平台。目前美国已经有85%的高校能够为学生提供在线课程目录服务，有72%的高校能够为学生提供在线课程注册服务。在欧洲，各个大学也相继建立了信息化教务管理平台。   
　　其管理信息系统模式一般都是以全局数据信息集中统一管理的中央数据库模型，而软件开发的系统架构多半也是以C/S，或者B/S模式进行设计的，它们将整个学校的教学资源以集中数据平台的方式从全局的角度统筹管理，以提高整个教学管理效率，加大不同系统和模块间的数据流通速度。由于多年的探索和改进，目前国外高校的这种管理模式规模已初具规模，其运行也处于一种比较完善和稳定的阶段。   
　　虽然国外的高校教务管理系统已逐渐成熟，但由于相应的教学机构和国情的不同，教学管理手段和方法的差异，我们还不能完全照搬他们的管理系统模式，只能是在借鉴的基础上开发出适合我国实际教育和管理行为的教务管理系统，所以也不在此详细叙述。   
1.3.2 国内教务管理系统的研究现状   
　　随着我国教育的发展和逐步产业化，我国的高校在信息化建设过程中也经过了很多年的努力，已经取得了一定成绩。从引入PC开始，到建设初级校园局域网、中型百兆校园网，乃至大型千兆校园网和互联网数据中心（IDC），高校信息化建设不断地跨上一个新的台阶。条件好的学校，依托校园网，围绕本校教务管理实际情况开发各管理模块；条件差一些的学校，就采用单机版的教务管理系统，仅实现其中一些相关的模块，并不是全部教务管理环节都采用计算机信息管理。现在，各大高校在经历了以硬件投资为主的校园网建设之后，在需求的拉动之下，开始了新一轮注重软件建设、以“强调应用”为主题的、以软件开发为主的应用系统改造的新的阶段。   
　　高校的信息化建设，主要是在传统校园建设的基础上，利用现代化的信息技术和手段，将分散的信息资源统一进行管理，形成一个虚拟的数字空间，使得现实的校园信息资源在时间和空间上得以延伸和扩展，从而实现提高教育教学管理水平和效率的目的。各高校的教务管理系统都有自身的特点，不尽相同。在该阶段，大部分高等院校的建设目标是在传统校园网系统的基础上，利用先进的开发技术，将各种分散的教学资源统一起来，搭建出一种异构的数据共享平台，并在该数据平台的基础之上完成各种老系统改造和新系统的开发，从而满足现代教育教学管理的需要。   
　　但，传统的信息系统架构基本上都是基于局域网、客户机/服务器模式。基于局域网方式开发的系统，相对来说比较简单，但系统稳定性和安全性较差。客户机/服务器模式是在局域网的基础上发展起来的，主要是以服务器的计算为主，对客户机的要求不高，对整个系统的安全性控制较强，目前这种架构多数用在对安全性要求较高的领域中，缺点是对于分布式应用的信息处理能力较差。   
　　浏览器/服务器（B/S）结构主要是针对分布式应用而开发的，是对客户机/服务器结构的一种改进和升级。通常情况下可以将该模式下的系统分为三层架构，也就是浏览器层、Web服务器层和数据库管理层。浏览器/服务器（B/S）模式是目前使用最多的一种教务管理信息系统架构，对于解决分布式应用有良好的效果，但是对于系统的安全和用户控制能力较弱。   
　　但随着信息技术的快速发展和教育改革制度的不断加强和深化，而教务教学工作是一所高校的工作重点内容，是整个学校日常工作的核心和基础，需要利用现代化高科技管理手段组织、管理、协调和指挥行政人员、教师和学生进行相关的活动，以便高质量、高效率的完成各项教学任务，同时需要能够灵活智能的应对各种突发事件，这就使得高校的日常教务工作越来越繁杂多变，而传统的信息系统架构，在技术层面和业务逻辑层面并不能直接的通信和交互，这就不能满足复杂多变的业务逻辑处理需求，而随着Internet的快速发展，新的基于Java EE的多层体系架构的信息管理系统快速地发展起来。基于Java EE的多层体系架构的信息管理系统主要是在传统软件的三层架构的基础上，增加了一层业务逻辑管理层，该层主要是集成整个大型系统中的业务功能模块，并将其抽象为业务组件。多层体系架构则根据具体的系统需求灵活的调用业务逻辑组件，完成不同商业应用之间的业务集成功能。多层体系架构为企业提供的威力和灵活性是强大的。如果一个组织将它的IT基础设施抽象出来，并以粗粒度的服务方式表示它的功能，那么服务的消费者就能够以独立于底层实现的方式访问。   
　　在功能上， 主要是包括信息管理功能、系统查询功能、学籍成绩管理功能、教学计划管理功能、课程管理功能、教材管理功能、教师管理功能、选课功能、排课管理功能、考级管理功能、考场编排功能、教务信息管理功能等等。如由苏州工艺美术职业技术学院所使用的教务管理系统主要包含数据维护、基本数据管理、教学计划管理、开课管理、学籍管理、教室管理、排课管理、毕业管理等。各模块的功能划分又体现了开发者对数据库的建模思路，模块的划分映射到相应表对信息的划分。在运行模式上，教务管理系统的基于网络使信息管理集中化，利用网络数据库存储信息，通过专用客户端界面，实现各院系与教务处的业务往来。采用文件共享的网络结构，利用桌面数据库存储信息，教务处内各模块管理人员通过专用客户端界面对各模块进行操作。   
　　目前国内一些高校在借鉴和吸收国内外先进的教务管理模式的同时，结合本校的具体实际情况，设计开发符合自己条件的教务管理系统。但是大多数的高校没有自己的教务管理系统技术研发能力，普遍热衷于以直接购买软件公司产品的方式来搭建本校综合教务管理平台，而且都十分期望能够选择一套技术成熟、功能完善的综合教务管理系统软件产品，如清华大学开发的“UEAS高校教务管理系统”、湖南青果软件有限公司的“高校教务网络管理系统”、杭州正方软件股份有限公司的“正方高校现代教学管理信息系统”、湖南强智科技发展有限公司的“强智综合教务管理系统”、西安康德信息科技发展有限公司的“高校综合教务管理系统”等。

## 1.4 课题主要内容

本课题主要是为了方便教师对学生成绩的管理以及课程的查询以及个人信息的修改等方面进行实现。同时要使管理员能够管理教师以及课程以及学生的相关操作。

本次课题中教师模块主要分为以下几个方面进行相关的编写，第一，教师登录界面的设计与应用；第二，教师的个人信息界面的设计与应用；第三，教师的教学任务，包括课程设计环节以及本科书本类课程的教学安排；第四，学生成绩的录入工作，从两个大的方面进行划分，一方面是以环节即课程设计（实践）方面进行划分，一方面是以所授课程的课程号来进行划分的；第五，可以查看所授课程对应的班级学生的成绩并且进行学生成绩的打印；最后，可以修改老师的个人信息，包括密码等相关内容。

而在管理员部分，则是重点强调了学生和教师以及课程的相关操作，诸如：增添、删除、修改、查询等功能，可以及时更新学生和老师以及课程的相关信息，同时可以看到课程的正选以及补选结果，实时的为老师的授课以及学生的学习安排提供修改意见。

## 1.5 论文组织结构

本篇论文的结构和安排如下：

第1章：介绍本论文的设计背景以及设计的必要性，设计的目的和任务，理清思路；

第2章：介绍系统实现所需要的相关技术和开发工具，开发及运行环境；

第3章：介绍系统的功能分析以及概要设计，对系统索要完成的功能进行整合概括，介绍系统的系统分析以及概念设计和逻辑分析；

第4章：介绍教师模块的具体实现以及界面设计，功能划分；

第5章：介绍管理员模块的工作范围以及所完成的工作；

第6章：介绍系统的测试并且分析测试结果；

第7章：对软件的维护进行论述；

第8章：总结经验反思操作过程中的不足之处，对课题未来的发展提出深度的展望。

# 第2 章 系统实现的相关技术

在进行系统实现的时候主要通过.net相关技术以及数据库的编程进行设计。同时在构建“从上到下”的分析环节时采用三层架构的方式对所需要处理的数据进行分层次管理和应用，通过C#中开发Windows程序所需要的组件的应用来完成界面的设计，通过数据库中表的建设来正确的处理系统中所对应的数据间的关系，使得软件在设计的时候处理的更加的快捷。

## 2.1 .NET相关技术

### 2.1.1 NET Framework

.NET Framework 用于构建具有视觉上引人注目的用户体验的应用程序，实现跨技术边界的无缝通信，并且能支持各种业务流程。而C#是.NET平台为应用开发而全新设计的一种现代化编程语言，并且C#在CLR层面统一了数据类型，使得.NET框架上的不同语言拥有了相同的类型系统：C#的对象模型建立在.NET框架虚拟对象系统之上，是基础架构的一部分，而不仅仅是编程语言的组成部分：C#开发的实质就是通过C#语言使用.NET库的组件，以实现应用业务逻辑。

最常见的C#项目有三类：1.控制台应用程序；2.Windows窗体应用程序；3.ASP.NET Web应用程序。这3类程序的创建步骤基本相同，都是通过选择主菜单【文件】->【新建】->【项目】，在“新建项目”对话框右侧的“模板”栏中选中某个模板类型，然后在“名称”栏中输入项目的名称，在“位置”栏中输入保存项目的路径，在“解决方案名称”栏中输入解决方案的名称，之后单击【确定】按钮就可以进入项目开发工作区。而开发Windows程序的步骤，分别是以下几点：

1. 建立模型，如上所述进行相关的操作；

2.界面设计：通过从工具箱向窗体设计区中拖放添加各种控件来设计应用程序的界面；

3.属性设计：在窗体中选中控件，然后在属性窗口设置各个控件相应的属性；

4.代码设计：双击控件或者经由属性窗口的事件列表，进入代码编辑器，编写相应控件的事件代码；

5.运行调试：单击工具栏上的相对应的按钮或者按F5快捷键。

在本次毕设中使用的界面控件有ComboBox控件，这个控件用在下拉组合框中显示数据以供用户从中选择；MenuStrip控件，这个控件是Windows文档类应用程序最基本的界面元素，也是最常使用的控件之一，提供应用程序的各项功能选择；而单选按钮则为用户提供了选项，当用户选择某个单选按钮的时候，同一组中的其他选项则不能同时选定，如个人简介处的性别。

C#具有一个统一的类型系统，所有C#类型（包括如int何string之类的基础数据类型）都继承于一个唯一的基类型：Object。因此，所有类型都共享一组通用操作，并且任何类型的值都能够以一直的方式进行存储、传递和操作，另外，Ｃ＃同时支持用户定义的引用类型和值类型，一方面允许对象的动态分析，一方面允许轻量结构的内联存储。为了确保Ｃ＃程序和库能够以兼容的方式逐步演进，Ｃ＃的设计中充分强调了版本控制，许多语言都不太重视这一点，导致采用哪些语言编写的程序，常常因为其所依赖的库的更新而无法正常工作。Ｃ＃的设计在某些方面直接考虑到了版本控制的需要，其中包括单独使用的virtual和override修改，方法重载决定规则以及对量式接口成员声明的支持。

### 2.1.2 Visual Studio.2010

Visual Studio 是微软.NET平台的集成开发环境（IDE），整合了多种开发语言，集合了诸如编辑代码，调试，测试以及打包，部署等功能，大大提升了软件开发的效率。而VS2010则是这个平台的最新版本，支持开发面向win7的应用程序，数据库方面出SQL之外还支持DB和Oracle。

## 2.2 数据库编程

数据库在建表之后，常常会使用一些sql语句进行相关内容的增删改查，如：

数据的**选择：**select \* from table1 where 范围  
**插入：**insert into table1(field1,field2) values(value1,value2)  
**删除：**delete from table1 where 范围 **更新**：update table1 set field1=value1 where 范围  
**查找**：select \* from table1 where field1 like ’%value1%’

通过Entity Framework可以将数据字段转换为数据的属性，同时也可以把数据间的关系转换为结合属性，这样可以让属于数据库的E/R模型转换成对象模型，方便操作。我们可以通过linq to entities来进行对数据库的相关连接，通过简洁明了的语言，及时的将数据库中的数据插入到想要的位置，及时更新改变数据库中的数据信息。Entity Framework 使用linq to entities 进行数据的查询，这种方法依赖于一个数据模型，不同于常用的连接数据库的语言，这种方式可以更加快捷和方便的找到数据库中所需要的内容，而不用找到数据库对应位置，造成不必要的操作浪费。同时通过这样的方法，表中的行可以被转换为C#对象的实例，而数据库中的表的列则可以转换为这些对象的属性。在这样的操作下，代码的设计也更加的简单。

数据库的架构和数据模型对象的映射是Entity Framwork的核心内容。为了能够生成模型，通过右击目录，添加新项，找到ADO.NET实体数据模型，继而设置创建的文件名称，之后点击“确定”，将一个已完成的并且存在的数据库生模型，即微软的 Northwind 示例数据库。配置数据库连接，并可以选择表、视图、和存储过程。还可以选择使用复数还是单数形式的对象名（例如，Products 表的行被命名为 Product ）、是否包含外键关系等。这里选择全部表并选中“所生成对象的单复数形式”。而这时的Visual Studio则可以帮助将已经选择的数据库元素创建模型图，它显示了已经创建的映射对象、对象拥有的字段以及对象之间的关系。Entity Framework将概念模型中定义的实体和关系映射到数据源，利用实体框架可以将数据源返回的数据具体化为对象；跟踪对象所做的更改；并发处理；将对象更改传播到数据源等。

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 是否允许为空 |
| ClassId | nvarchar(50) | 否 |
| ClassName | nvarchar(50) | 是 |
| SpeId | nvarchar(50) | 是 |
| Grade | int | 是 |

图 2.1 学生选课内容和成绩表

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 是否允许为空 |
| Sno | nvarchar(50) | 否 |
| Cno | nvarchar(50) | 否 |
| Terms | nvarchar(50) | 否 |
| Studienjahr | nvarchar(50) | 否 |
| ExamCharacter | nvarchar(50) | 否 |
| Score | nvarchar(50) | 是 |
| GetCredit | float | 是 |
| GradePoint | float | 是 |
| CouseType | nvarchar(50) | 是 |

图 2.2 管理员登陆表

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 是否允许为空 |
| AdminId | nvarchar(50) | 否 |
| AdminPasswords | nvarchar(50) | 是 |
| Autho | nvarchar(50) | 是 |

图 2.3 登录信息表

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 是否允许为空 |
| Sno | nvarchar(50) | 否 |
| GradeId | nvarchar(50) | 否 |
| EnrollStatus | nvarchar(50) | 是 |

图 2.4 等级考试报名表

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 是否允许为空 |
| GradeId | nvarchar(50) | 否 |
| GradeExamName | nvarchar(50) | 否 |
| GradeCategory | nvarchar(50) | 否 |
| ExamStyle | nvarchar(50) | 否 |
| ExamTime | nvarchar(50) | 否 |
| Cost | float | 否 |
| EnrollTimeStart | smalldatetime | 否 |
| EnrollTimeEnd | smalldatetime | 否 |

图 2.5 教材信息表

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 是否允许为空 |
| TId | nvarchar(50) | 否 |
| Tname | nvarchar(50) | 否 |
| Tgender | nvarchar(50) | 否 |
| Tpassword | nvarchar(50) | 否 |
| Tdegrees | nvarchar(50) | 否 |
| Ttitle | nvarchar(50) | 是 |
| SchoolYear | nvarchar(50) | 是 |
| Tel | nvarchar(50) | 是 |

图 2.6 教师信息表

老师课程表（通过职工号找到对应老师，通过学年学期以及课程号找到对应课程的相关信息）

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 是否允许为空 |
| id | int | 否 |
| TId | nvarchar(50) | 否 |
| Cno | nvarchar(50) | 否 |
| Terms | nvarchar(50) | 否 |
| Studienjahr | nvarchar(50) | 否 |
| ClassName | nvarchar(50) | 是 |

图 2.7 授课信息表

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 是否允许为空 |
| Cno | nvarchar(50) |  |
| Terms | nvarchar(50) |  |
| Studienjahr | nvarchar(50) |  |
| SpeId | nvarchar(50) |  |
| Cname | nvarchar(50) |  |
| Credit | float |  |
| TotalHours | float |  |
| TeachingHours | float |  |
| ComputerHours | float |  |
| CouseType | nvarchar |  |
| GroupNum | int |  |
| TeachStyle | nvarchar |  |
| ExamStyle | nvarchar |  |
| WeekStart | int |  |
| WeekEnd | int |  |

图 2.8 课程信息表

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 是否允许为空 |
| Sno | nvarchar(50) | 否 |
| Sname | nvarchar(50) | 否 |
| Spassword | nvarchar(50) | 否 |
| Sgender | nvarchar(50) | 否 |
| Sid | nvarchar(50) | 否 |
| Speoples | nvarchar(50) | 是 |
| Spolitics | nvarchar(50) | 是 |
| Province | nvarchar(50) | 是 |
| City | nvarchar(50) | 是 |
| ClassID | nvarchar(50) | 是 |
| tel | nvarchar(50) | 是 |

图 2.9 学生相关信息表

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 是否允许为空 |
| ColId | nvarchar(50) | 否 |
| Colname | nvarchar(50) | 否 |

图 2.10 院系信息表

|  |  |  |
| --- | --- | --- |
| 列名 | 数据类型 | 是否允许为空 |
| SpeId | nvarchar(50) | 否 |
| Spename | nvarchar(50) | 否 |
| ColId | nvarchar(50) | 否 |
| EduSysterm | nvarchar(50) | 否 |

图 2.11 专业信息表

## 2.3系统开发和运行环境

VS2010开发采用Windows窗体应用程序，通过项目和解决方案来管理应用程序的开发，如：创建应用程序所需要额引用以及数据链接等。一个解决方案可以包含多个项目，产生扩展名为.sln的解决方案文件。在分化层级的时候采用三级架构模式，和网页设计MVC模式不同的是，三层架构(3-tier application) 通常意义上的三层架构就是将整个业务应用划分为：表现层（UI）、业务逻辑层（BLL）、数据访问层（DAL）。区分层次的目的即为了“高内聚，低耦合”的思想。

1、表现层（UI）：通俗讲就是展现给用户的界面，即用户在使用一个系统的时候他的所见所得。   
 2、业务逻辑层（BLL）：针对具体问题的操作，也可以说是对数据层的操作，对数据业务逻辑处理。   
 3、数据访问层（DAL）：该层所做事务直接操作数据库，针对数据的增添、删除、修改、更新、查找等。 而MVC是一个设计模式，它强制性的使应用程序的输入、处理和输出分开。使用MVC应用程序被分成三个核心部件：模型、视图、控制器。它们各自处理自己的任务。缩写中的M表示的是模型，而模型表示企业数据和业务规则。在MVC的三个部件中，模型拥有最多的处理任务。被模型返回的数据是中立的，就是说模型与数据格式无关，这样一个模型能为多个视图提供数据。由于应用于模型的代码只需写一次就可以被多个视图重用，所以减少了代码的重复性。缩写中的V是视图的简称，是用户看到并与之交互的界面。对老式的Web应用程序来说，视图就是由HTML元素组成的界面，在新式的Web应用程序中，HTML依旧在视图中扮演着重要的角色，但一些新的技术已层出不穷，它们包括Macromedia Flash和像XHTML，XML/XSL，WML等一些标识语言和Web services.MVC一个大的好处是它能为你的应用程序处理很多不同的视图。在视图中其实没有真正的处理发生，不管这些数据是联机存储的还是一个雇员列表，作为视图来讲，它只是作为一种输出数据并允许用户操纵的方式。缩写中的C表示的是控制器，控制器接受用户的输入并调用模型和视图去完成用户的需求。所以当单击Web页面中的超链接和发送HTML表单时，控制器本身不输出任何东西和做任何处理。它只是接收请求并决定调用哪个模型构件去处理请求，然后再确定用哪个视图来显示返回的数据。。实现一个动态的程序设计，使得后续可以对程序的修改和扩展简化进行，并且可以是程序的某一部分重复利用。同时使用ADO.NET Entity Framework将每个数据库对象转化成应用程序对象。，MVC把三层架构中的UI层再度进行了分化，分成了控制器、视图、实体三个部分，控制器完成页面逻辑，通过实体来与界面层完成通话。三层架构的分层模式是典型的上下关系，上层依赖于下层。但MVC作为表现模式是不存在上下关系的，而是相互协作关系。

# 第3章 系统分析与概要

软件在开发的初始阶段就应该搞清楚用户“需要计算机解决什么样的问题”。如果不通过事先的了解就师徒去解决，那么最后做出来的势必不是最切合实际的，只会白白浪费时间和金钱，最终得出的结果很可能是毫无意义和价值的。所以确切地定义问题找到合适的对策在系统设计的初期是非常必要的一项工作;除此之外，系统的设计应具有可行性操作，不仅是技术可行性还涵盖着经济可行性，操作可行性，法律可行性。之后建立分析问题后的模型，对需求进行分析建模，结构化的将可能出现的问题逐一的分层，例如本文中所涉及的三级架构，采用自顶向下、逐步求精的技术对系统进行划分，分层次描述，知道找到满足所有功能要求的可以实现的软件为止。在这一阶段，数据流图以及实体关系图的创建是必不可少的。

## 3.1 系统需求分析

需求分析是软件开发过程中的一个重要的阶段，在这个阶段中，我们需要对目标系统提出精确的要求。需求分析是一个不断认识问题，发现问题的过程，我们应该逐步细化过程中的每一个细节，并良好的实现所需要的功能。

需求分析是整个系统开发的基础，之后的所有操作过程都是要根据它来进行实现以及逐步完善的，因此，是否能够做好一个优质的需求分析是整个项目成败的关键所在。

教务系统教师模块

教师信息管理

教师授课信息

教师成绩录入

学生成绩查询

图3.1 教师模块图

### 3.1.1流程分析

职工号

姓名

性别

学生

学号

姓名

性别

班级

课程

授课

选课

课程号

课程

学分

学时

教师

1

1

N

N

图 3.2 E-R图

### 3.1.2 功能

1.教师模块：

实现如下几方面的功能：

（1）教师的正常登陆；

（2）教师个人信息的查询；

（3）教师登录界面密码的修改；

（4）教师所授课程的信息，包括指导课程（实践课程）以及正常课程信息的查看；

（5）教师所授课程对应班级信息下的学生成绩的查看；

（6）教师所授课程的成绩的成绩的录入；

（7）教师所授课程对应学生的成绩的打印；

2.管理员模块：

主要实现以下几项功能：

（1）管理员的正常登陆；

（2）管理员对于学生信息的增添；

（3）管理员对于学生信息的修改；

（4）管理员对于学生信息的查看；

（5）管理员对于学生信息的删除；

（6）管理员对于教师信息的增添；

（7）管理员对于教师信息的修改；

（8）管理员对于教师信息的查看；

（9）管理员对于教师信息的增添；

（10）管理员对于课程信息的查看；

（11）管理员对于课程信息的增添；

（12）管理员对于课程信息的修改；

### 3.1.3 数据需求

通过对教务系统教师模块以及管理员的日常的工作进行可行性分析和详细调查，确定教务系统中对应的模块内容。通过调查研究、信息流程分析和数据收集，明确该系统子模块的主要功能是：对教师个人信息的管理，按照课程以及指导环节来划分教师所授内容，按照课程以及环节来录入成绩，按照课程和环节查看成绩以及登录界面的密码修改。所要的数据需求就是教师的工号，性别，入职时间等具体信息，课程的课程号以及对应课程名，学分，所授行政班级等相应信息，还有在成绩查询的时候通过对应的学年学期找到所教授的课程或者实践环节，通过所教授的行政班级来找到对应课程对应时间对应班级的学生的成绩信息。

## 3.2 开发设计原则

对应系统子模块的设计需要模块化，并且应该满足高内聚低耦合的设计准则，根据软件设计的准则来改进设计，提高软件设计的质量，同时设计应该具有可操作性，界面设计应该简洁明了方便用户使用，而且操作应该简单，这样才不会浪费资源去再次学习，设计时同样要考虑到可行性以及低经济高效率性，可以对软件进行测试，避免出现错误，对软件安全性要有一定的考虑，因为教务系统不仅管理的是学生的个人信息还有老师的个人信息，如果数据丢失，将会造成不可以避免的麻烦。同时，模块的作用范围即该模块内一个判定影响的所有模块的集合应该子啊控制范围之内；教务系统是一个大的工具软件，所需要的模块也不仅仅是教师模块，所以在设计的时候也要考虑和注意到可能存在的模块接口的问题，设计的时候应该主动降低模块接口的复杂性，这样可以避免软件在整合运行的时候发生一些不必要的错误，当然也要考虑模块间的相对独立性，模块的接口越简单，越便于理解，同时也方便实现、测试和维护。最后，在设计这个模块的时候应该可以预测到模块的功能，但是同时也要避免模块功能过分限制。

## 3.3 总体结构设计

三层架构:[三层架构](http://baike.baidu.com/view/687468.htm" \t "http://baike.baidu.com/_blank)(3.tier architecture) 通常意义上的三层架构就是将整个业务应用划分为：界面层（User Interface layer）、业务逻辑层（Business Logic Layer）、数据访问层（Data access layer）。区分层次的目的即为了“[高内聚低耦合](http://baike.baidu.com/view/3082578.htm" \t "http://baike.baidu.com/_blank)”的思想。在软件体系架构设计中，分层式结构是最常见，也是最重要的一种结构。微软推荐的分层式结构一般分为三层，从下至上分别为：数据访问层、业务逻辑层（又或称为领域层）、表示层。3个层次中，系统主要功能和业务逻辑都在业务逻辑层进行处理。所谓三层体系结构，是在客户端与数据库之间加入了一个“中间层”，也叫组件层。这里所说的三层体系，不是指物理上的三层，不是简单地放置三台机器就是三层体系结构，也不仅仅有B/S应用才是三层体系结构，三层是指逻辑上的三层，即把这三个层放置到一台机器上。三层体系的应用程序将业务规则、数据访问、合法性校验等工作放到了中间层进行处理。通常情况下，客户端不直接与数据库进行交互，而是通过COM/DCOM通讯与中间层建立连接，再经由中间层与数据库进行交互。

# 第4章 教师模块的实现

教师模块的实现，主要有以下几部分组成：教师登录主页面的设计，以及教师模块所能实现的一些功能：如查看教学任务，查看所授课程，按环节录入学生成绩，查看学生成绩以及修改个人密码等功能。

## 4.1 主页面

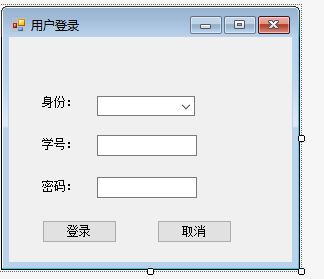


图 4.1 主页面图

打开窗体的时候显示登录界面，同时被学生、教师和管理员所共用。通过设置字段让不同身份的人登录进属于自己模块的界面

## 4.2 教师模块

教师模块主要分为以下几个方面，一个是个人信息页面，可以查看到教师自己的个人信息，诸如：职工编号、姓名、学历、性别以及入职时间等详细信息，在其他框里可以修改教师的个人信息，同时管理员也可以进行教师的增删。其次，在教学任务中通过学年学期找到所教授的课程，所教授的课程又分为按环节以及按课程两种，按环节就是课程设计（实践）一类的，教师通过学年学期找到自己所教授的课程号，找到课程名以及对应的学分，教授的班级等相关信息。按课程也采取同样的方法，通过设置字段来进行查找。成绩查找也分是以按环节或者是按课程来划分的，同样通过设置字段，按照学年学期查找对应课程，找到相应信息。

### 4.2.1 个人信息页面

打开窗体时，显示出所有教719347160215师记录的列表，输入内容可以更新教师的个人信息。

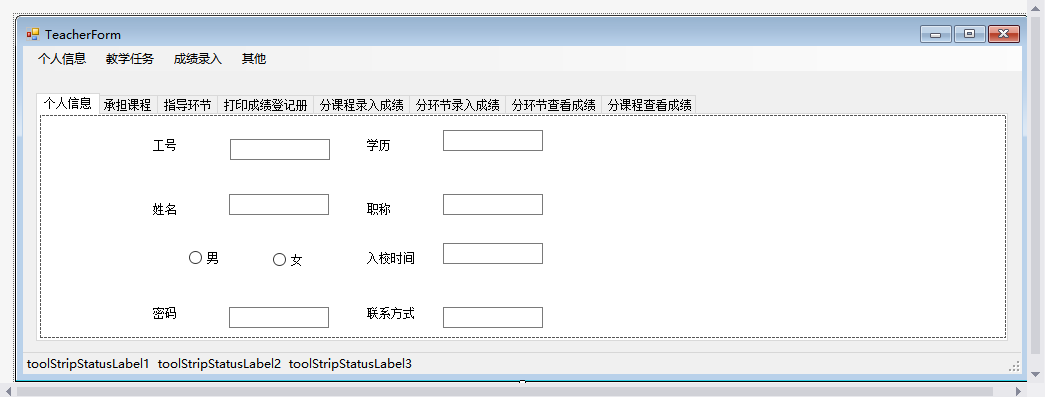


图 4.2 个人信息图

### 4.2.2 成绩录入页面

打开窗体时，显示分课程按行政班级录入成绩的列表，通过键入课程号，课程名找到对应教授课程的班级进行相关班级学生信息的查看，通过这个班级的名称以及学生的学号，为某一对应班级的对应学号的学生录入成绩，然后提交成绩更新学生的成绩。

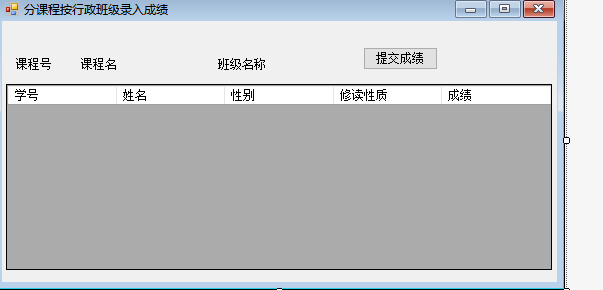


图 4.3 录入课程成绩图

课程实践的成绩录入

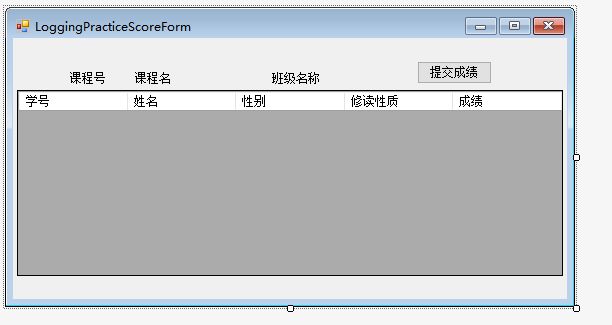


图4.4 显示成绩图

实践指导环节的课程的成绩录入，与前面窗体一致。

### 4.2.3 教学任务查询页面

教学任务页面分为两个小项，一个是老师所承担的上课的课程，一个是老师的课程设计的指导环节。

打开窗体，显示对应教学任务即教师所承担的课程的列表，在学年，学期输入想要查找的相对应的课程的信息，然后点击“查询”对数据库中对应课程进行直接查看，可以看到对应课程的课程号以及课程名等相关信息。



图 4.5 查询课程图



图 4.6 查询实践环节图

### 4.2.4 成绩的查看页面

分环节查看成绩课程实践，输入学年学期找到对应教学的课程环节的环节号和环节名等详细信息。



图 4.7 查看环节成绩图

分课程查看成绩，通过学年学期所授来查找所教学的课程，看到相关信息。

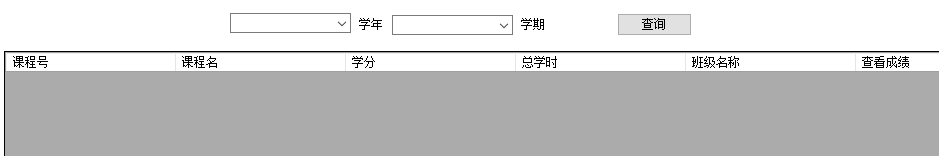


图 4.8 查看课程成绩图

# 第5章 管理员模块

对于学生、老师以及相关课程进行管理的任务是由管理员进行的，管理员可以对学生进行查看，修改，添加以及删除，在学生信息中连接数据库可以看到学生的学号，姓名，登录密码，以及身份证号等个人信息，同时也可以对老师进行相关的操作，只不过在表中看到的老师信息分别是老师的姓名，老师的工号，登录密码等相关信息；在课程信息的功能选项中可以看到课程号，课程名以及课程安排等信息；同时可以对班级以及专业进行管理，因为不同的学年学期会有不同的班级号对应不同的专业，对应不用的学院，很有可能今年这个学期的这个班级明年可能是另外一个专业，而且，有很大的可能会增添专业或者删除专业进行专业合并；在选课项目框中有两个不同的功能一个是正选，一个是补选，前者是学生选课后选课的内容，有相对应的课程的信息，而补选则是为了没有选到课程的学生提供选课的方式，同样会显示选择的课程的信息，比如：课程号，课程名等相关课程的信息。启动程序后，出现登录窗体，管理员登陆窗体，如果管理员输入的用户名和密码错误，就会弹出一个提示错误的对话框；如果管理员登陆成功，则会条状到添加员工的窗体。

管理员模块时教务系统很重要的模块，管理着教务系统学生模块以及老师模块能否真正的实现，在学生和老师发生变化的时候就要及时的更新为了有效的管理整个体系。

实际生活中，我校的管理员是按照院系分配的，每个院都配有一到两个对应的教学秘书进行对应院系的教师、学生以及课程的相对应的安排和调整，所以，管理员部分可以手动添加教师、学生以及课程的相关信息。

# 第6章 系统的测试与分析

软件是人类智力的结晶，全世界诸多国家都对计算机软件进行深入研究，但是随着软件的诞生，软件缺陷也带给人们许多损失，甚至于有些损失是无可挽回的。由于软件开发人员的主观认知能力的局限性和开发软件的复杂性，尽管采取了许多改进和保证软件质量的方法，但是在开发软件的各个过程中还是不可避免地会产生各种或大或小的错误，所以在软件正式投入使用之前必须对其做严格的测试，发现并纠正错误。由于软件程序的正确性证明在技术上还未得到根本解决，软件测试成为了发现软件错误和缺陷的主要手段。IEEE对软件测试的定义是：使用人工或自动的手段来运行或测定某个软件系统的过程，其目的在于检验它是否满足规定的需求或者弄清预期结果和实际结果之间的差别。

## 6.1 系统的测试和意义

系统的研发就是为了能够在实际生活中真实的应用，并且能够被不熟悉软件的用户简单快捷的使用，同时，作为用户也会更加的希望所获得的软件能够有更少的隐藏的错误和缺陷，造成工作中的不必要的麻烦，通过软件测试可以很好的检查在软件设计中可能出现的漏洞并且及时的改正这些漏洞，为后面的实际应用起到借鉴维护的作用，确保使用者对软件质量的信心，但同时也要认识到测试的结果决不能片面的代表程序的正确性。

而软件的测试原则则包含以下几点：1.应尽早地和不断地进行软件测试；2.所有测试都应该能追溯到用户需求；3.测试用例有输入数据和对应的与其输出数据两部分组成；4.测试用例应包括合力的输入条件和不合理的输入条件；5.对发现错误较多的程序段，应该进行更深入的测试；6.严格设计测试计划，排除测试的随意性；7.应当对每一个测试的结果做全面的检查；8.在对程序修改之后要进行回归测试；9.程序员应该避免测试自己的程序；10妥善保存测试计划、测试用例、出错统计和最终分析报告，为了维护提供方便。

软件测试可以先采用黑盒方法设计基本的测试方案，然后采用白盒方法补充一些必要的测试方案并验证其完整性。黑盒测试就是把测试对象当成一个黑盒子，测试人员完全不考虑程序内部的逻辑结构和处理过程，只是在软件的接口处进行测试并且根据程序所需要完成和实现的功能，检查程序的功能是否满足要求，及时发现是不是有遗漏或者不正确的功能，而白盒测试则是把程序当成一个透明的盒子，允许测试的时候利用程序内部的逻辑结构和相关信息，设计或者选择相关的测试。

## 6.2 系统测试的结果

基本能够完成本次毕业设计关于教务系统教师模块的功能实现，采用三级架构的方式将教师模块分层。各层之间既有独立的点，又有相似相互联系的地方，可以处理正常的工作要求，同时界面的设计也足够简洁明了能够让使用者正常的使用，而不会产生操作困难的问题。代码设计简单，使用linq to entities 连接数据库中的内容，对表内所需要查找的信息及时的更新。当然，软件设计的时候也会有个人的主观想法，所以有很大的可能性设计出来的并不是完美的，但是基本功能还是可以是实现的，存在的不完美的地方也还需要大家批评指正。

# 第7章 软件维护

软件在开发设计的时候就是以提高软件的可维护性，减少软件维护所需要的工作量，降低软件系统的总成本为目标，所以软件在交付使用之后，为了方便改正错误或者是满足用户所需要的新的需求都需要对软件进行及时的更新和维护，以防止测试阶段隐藏下来的软件错误造成不必要的资源浪费以及相应的系统故障，在修改错误之前必须理解待修改的对象，修改之后应该进行必要的测试，以保证所做的修改是正确的，但是，软件维护并非是一劳永逸的，在软件维护的过程中，首先软件组织建立维护的机构，再之后通过确立维护的过程，为维护的处理步骤进行安排，建立好的维护流程。同时要注意在维护过程中，会不会因为修改造成其它相关联的内容造成相对应的改变。

# 第8章 总结

本次毕业设计的任务是制作教务系统，通过在实现之前对相关资料分析的搜集、整理和查阅来找到可以借鉴学习的方法，在设计的过程中，找到相互对应的应用关系，并以一种交互的窗体展现出来。在设计实现的过程中使用了linq to entities连接数据库，找到对应的数据，在数据库中通过设置关键字找到对应不同表中的关键字，并且通过不同表间的联系进行设计，通过这次毕业设计的实践，使我对窗体模式的构建以及数据库的应用以及架构模式都产生了新的了解和认识，如何能够有效便捷的处理问题，是解决本次毕设的重点也是难点，同时理清楚模型中的各种属性的内在联系也是思考并且正确设计的一个关键性步骤。如何构建数据库中的表同样也是一个很需要思考的地方，比如说教师模块中的教师的教学任务，实际生活中我们可能每个学期都会有课程设计也会有教程课程，而其实在很大程度上我们现在所用的教务系统本身也是有可能有功能上的缺失和漏洞的，比如说以我们学校华北水利水电大学为例，学生和老师以及课程的增改都是需要下发到每个学院内各个教务主管来进行人工手动添加，课程的安排也是整合之后的结果而不是依据计算机软件，课程的临时改变也没有办法做到及时的更新，比如说由于假期或者补课等其它因素造成课程以及教师以及授课班级以及上课班级之间的相互联系只能依靠于人工手动的安排，而不能及时更新到教务系统上去显示；除此之外，学生模块的四六级报名是需要以班级为单位逐级人工上报给管理员的，老师的授课安排的调动以及授课班级的临时调动都是需要根据实际情况来完成的，所以说计算机发展还需要有很长的一段路要走，但是，现如今的软件设计已经能够初步完成所需要完成的任务，基本实现人机交互，基本符合我们所使用的习惯，对于现在的我们来说要做的更多的是去学习和掌握新的技术稳固已有的知识累积在实践生活中的经验，以更高的姿态去应对可能出现的困难。

在本次毕业设计中，开始的时候常常会有不知所措的感觉，因为觉得这个系统实在是太大了，关系太复杂了，数据太多了，需要实现的功能也大大小小形态各异，软件的开发研制以及测试包括运行维护都需要很多的心血，通过查阅相关的数据、报刊、资料、视频，通过询问老师相关的信息，通过同学的鼓励和帮助，这些难题也都在一步一步的克服然后实现。不得不说的是用窗体去实现教务系统是很简陋而且不实用的，就像我们学校的教务系统都是以web页面设计的，这样设计的好处就是方便多台主机共同的使用，而且不需要配备我们所用的软件就能够轻松的启动，而且界面美观，方便查找功能，但是通过上网查找相关的资料的同时，我也看到了网上有很多的使用ASP.NET实现的教务系统，而没有出现很多用windows窗体来实现的教务系统，这也是我们组决定使用窗体的原因之一。在之前的课程设计中，我们也对高校教务系统的数据库以及教务系统的模块设计做过相应的研究学习，但是使用的都是固定的连接数据库的语言，这种方式既复杂而且又容易出错，因而这次我们决定使用linq entities去连接数据库中的内容，采用这种方便且易操作的方式去进行代码的编写。

随着对计算机软件的开发深入，未来肯定会有更多的人才投入到计算机软件开发中，如何开发出高效率的使用工具一直是非常关键的地方，就像教务系统对高校工作管理一样，使用工具去开发和提高人的工作效率，让人工作的时间放在更需要使用的地方，这也是很重要的一点。教务系统的管理是非常复杂的，教师、学生以及课程之间的联系往往是一对多或是多对多，如何理清楚数据间的内在联系是软件程序顺利开发的重中之重，恰当的需求分析以及恰当的解决问题的对应策略是这项管理系统能够顺利实施并且开发出来的关键所在。

现如今，我们拥有越来越多的软件工具帮助我们实现创造，好的构思是必不可少的基础，其次是选用恰当的工具，选用好的方法，选择具有代表性的数据进行问题的研究和分析，遇到难解的困难可以在选择查阅书籍的同时和同学或者老师交流探讨及时沟通，为问题的成功解决打下基础，同时也可以为自己之后的学习生活累积经验，知识是无尽的，随着时代的步步转变，每个人都在进步，计算机领域更是日新月异，学习和掌握新的技术和知识是必不可少的，也是生活工作的重点，如何在自己的领域内发现细小的问题并且改进都是很重要的一项技能，这次毕业设计论文通过认真分析我校实际教务管理系统的基础上,利用基于.NET技术开发三层架构模式所构建的教务管理系统中教师模块以及管理员模块所涉及到的关键技术进行了深度展开,给出相应问题的解决方案,运用软件工程的方法设计并实现具有现实价值的教务系统教师模块和管理员模块,这个系统是以微软.NET为平台,采用三层结构构建结构模式,并以SQL Server作为后台数据库,使用流行编程语言C＃开发。系统实现了高校教务管理系统的主要功能,具有教师个人信息的管理、教师授课信息的查看、学生成绩管理、授课班级专业信息查询、授课行政班级信息管理、信息查询、系统管理等功能。  
　　 这篇文章对需求分析,系统设计及实现过程中涉及到的相关理论与实践问题进行了比较全面的阐述,着重分析了.NET技术在系统开发过程中的具体操作。通过软件测试,判定该系统基本能够运行稳定、安全可靠、维护方便、操作简单、支持网络办公模式,达到了预期效果。同时还有不足之处需要修改。

# 参考文献

* [1] 陈毓祯.[高校教务管理系统的分析与设计](http://www.cnki.net/kcms/detail/detail.aspx?filename=1013306374.nh&dbcode=CMFD&dbname=CMFD2014&v=" \t "http://www.cnki.net/KCMS/detail/frame/_blank)[D]. 云南大学 2013
* [2] 黄晓薇.[高职院校教务管理系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=1013317334.nh&dbcode=CMFD&dbname=CMFD2014&v=" \t "http://www.cnki.net/KCMS/detail/frame/_blank)[D]. 华南理工大学 2013
* [3] 赵成松.[基于.NET平台的教务管理系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=1013330141.nh&dbcode=CMFD&dbname=CMFD2014&v=" \t "http://www.cnki.net/KCMS/detail/frame/_blank)[D]. 电子科技大学 2013
* [4] 雷丽娟.[基于数字化校园教务管理系统的研究与设计](http://www.cnki.net/kcms/detail/detail.aspx?filename=1013330783.nh&dbcode=CMFD&dbname=CMFD2014&v=" \t "http://www.cnki.net/KCMS/detail/frame/_blank)[D]. 电子科技大学 2013
* [5] 王罗乐.[西华大学教务管理系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=1013334525.nh&dbcode=CMFD&dbname=CMFD2014&v=" \t "http://www.cnki.net/KCMS/detail/frame/_blank)[D]. 电子科技大学 2013
* [6] 朱其现.[基于JSP的教务管理系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=1014142169.nh&dbcode=CMFD&dbname=CMFD2014&v=" \t "http://www.cnki.net/KCMS/detail/frame/_blank)[D]. 厦门大学 2013
* [7] 栾柏宇.[高校教务管理系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=1013194366.nh&dbcode=CMFD&dbname=CMFD2013&v=" \t "http://www.cnki.net/KCMS/detail/frame/_blank)[D]. 吉林大学 2013
* [8] 潘芳芳.[高校教务管理系统设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=1013135586.nh&dbcode=CMFD&dbname=CMFD2013&v=" \t "http://www.cnki.net/KCMS/detail/frame/_blank)[D]. 南昌大学 2012
* [9] 杨志荣.[基于ASP.NET+SQL Server 2005技术的云师大住房公积金管理系统设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=1013149250.nh&dbcode=CMFD&dbname=CMFD2013&v=" \t "http://www.cnki.net/KCMS/detail/frame/_blank)[D]. 电子科技大学 2012
* [10] 徐鹏.[基于Web数据库的教务管理系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=JZGC201103167&dbcode=CJFQ&dbname=CJFD2011&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J].价值工程.2011(03)
* [11] 刘建花.[在线教务管理系统的研究与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=DNZS201026107&dbcode=CJFQ&dbname=CJFD2010&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J]. 电脑知识与技术.2010(26)
* [12] 韩利娟.[基于Structs框架的教务管理系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=FJDN201008061&dbcode=CJFQ&dbname=CJFD2010&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J].福建电脑.2010(08)
* [13] 陈戍.[基于WEB的成教教务管理系统设计探讨](http://www.cnki.net/kcms/detail/detail.aspx?filename=XXJK201011144&dbcode=CJFQ&dbname=CJFD2010&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J].中国科技信息.2010(11）
* [14] 朱璜润,唐建华.[教务管理信息化建设探讨](http://www.cnki.net/kcms/detail/detail.aspx?filename=ZDYX201202029&dbcode=CJFQ&dbname=CJFD2012&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J]. 卫生职业教育.2012(02)
* [15] 冯琰君,马霖,王琬华,樊晨辉,罗洋.[教务管理信息系统设计](http://www.cnki.net/kcms/detail/detail.aspx?filename=ZXLJ201136161&dbcode=CJFQ&dbname=CJFD2011&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J].科技资讯. 2011(36)
* [16] 方富贵.[基于web的教务管理系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=DNZS201118005&dbcode=CJFQ&dbname=CJFD2011&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J].电脑知识与技术.2011(18)
* [17] 刘强,吕晓娴,杨丽.[高校教务管理系统的设计与构建](http://www.cnki.net/kcms/detail/detail.aspx?filename=CYYT201107135&dbcode=CJFQ&dbname=CJFD2011&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J].产业与科技论坛. 2011(07)
* [18] 钱鸽.[基于Web的高校教务管理信息系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=HTSF201102117&dbcode=CJFQ&dbname=CJFD2011&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J].和田师范专科学校学报.2011(02)
* [19] 徐鹏.[基于Web数据库的教务管理系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=JZGC201103167&dbcode=CJFQ&dbname=CJFD2011&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J].价值工程.2011(03)
* [20] 刘建花.[在线教务管理系统的研究与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=DNZS201026107&dbcode=CJFQ&dbname=CJFD2010&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J].电脑知识与技术.2010(26)
* [21] 韩利娟.[基于Structs框架的教务管理系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=FJDN201008061&dbcode=CJFQ&dbname=CJFD2010&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J]. 福建电脑. 2010(08)
* [22] 陈戍.[基于WEB的成教教务管理系统设计探讨](http://www.cnki.net/kcms/detail/detail.aspx?filename=XXJK201011144&dbcode=CJFQ&dbname=CJFD2010&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J].中国科技信息.2010(11)

# **致 谢**

在本次论文设计过程中，感谢我的学校，给了我学习的机会；感谢我的同组成员古兆阳，以及多位同学给予的帮助，使得这次毕业设计能够顺利完成，同时还要感谢我的指导老师郑作勇老师，在学习和实践中给予的指导和帮助，在学习中，老师从题目指导、论文框架到细节修改，都给予了细致的指导，提出了很多宝贵的意见与建议，老师以其严谨求实的治学态度、高度的敬业精神、兢兢业业、孜孜以求的工作作风和善于倾听学生在学习中遇到的学习问题，并给予及时的辅导的工作方式，给我留下深刻的印象。他渊博的知识、开阔的视野和敏锐的思维给了我深深的启迪。这篇论文是在老师的精心指导和同学的帮助下才完成的。

感谢所有曾经或是现在教导我的老师，如果没有这些老师孜孜不倦的教导，是不可能有我现在所积累的知识的，更不可能有动力和信息来完成这篇论文。感谢之余，诚恳地请各位老师对我的论文多加批评指正，使我即使完善论文的不足之处。

谨以此致谢最后，我要向百忙之中抽时间对本文进行审阅的各位老师表示衷心的感谢。

# 附 录

## 附录I

### 外文原文

**The World According to LINQ**

Programmers building web- and cloud-based applications wire together data from many different sources such as sensors, social networks, user interfaces, spreadsheets, and stock tickers. Most of this data does not fit in the closed and clean world of traditional relational databases. It is too big, unstructured, denormalized, and streaming in real time. Presenting a unified programming model across all these disparate data models and query languages seems impossible at first. By focusing on the commonalities instead of the differences, however, most data sources will accept some form of computation to filter and transform collections of data.

Mathematicians long ago observed similarities between seemingly different mathematical structures and formalized this insight via category theory, specifically the notion of monads as a generalization of collections. Languages such as Haskell, Scala, Python, and even future versions of JavaScript have incorporated list and monad comprehensions to deal with side effects and computations over collections. The .NET languages of Visual Basic and C# adopted monads in the form of LINQ (Language-integrated Query) as a way to bridge the gap between the worlds of objects and data. This article describes monads and LINQ as a generalization of the relational algebra and SQL used with arbitrary collections of arbitrary types, and explains why this makes LINQ a compelling basis for big data.

LINQ was introduced in C# 3.0 and Visual Basic 9 as a set of APIs and accompanying language extensions that bridge the gap between the world of programming languages and the world of databases. Despite the continuing excitement about LINQ in the external developer community, the full potential of the technology has not yet been reached. Thanks to the foundational nature of LINQ, there is still enormous potential for its mapping scenarios outside object-relational (O/R), especially in the area of big data.

The advent of big data makes it more important than ever for programmers to have a single abstraction that allows them to process, transform, compose, query, analyze, and compute across at least three different dimensions: volume, big or small, ranging from billions of items to a handful of results; variety in models, structured or unstructured, flat or nested; and velocity, streaming or persisted, push or pull. As a result, we see a mind-blowing number of new data models, query languages, and execution fabrics. LINQ can virtualize all these aspects behind a single abstraction.

Take, for example, Apache's Hadoop ecosystem. It comes with at least eight external DSLs (domain-specific languages) or APIs: a set of low-level Java interfaces for MapReduce computations; Cascading, a "data-processing definition language, implemented as a simple Java API;" Flume, a "simple and flexible architecture based on streaming data flows;" Pig a "high-level language for expressing data analysis programs;" HiveQL, an "SQL-like language for easy data summarization, ad hoc queries, and the analysis of large data sets;" CQL, a "proposed language for data management in Cassandra;" Oozie, an XML-based "coordinator engine specialized in running workflows based on time and data triggers;" and Avro, a schema language for data serialization.

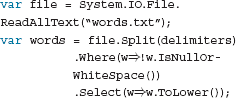
To create an end-to-end application, programmers need to use several of these external DSLs in addition to a general-purpose programming language such as Java to glue everything together. If data comes from an external RDBMS (relational database management system) or push-based source, then even more DSLs such as SQL or StreamBase are required. Using LINQ and C# or Visual Basic on the other hand, programmers can use internal DSLs to program against any shape or form of data inside a general-purpose OO (object-oriented) language that comes with tooling (Visual Studio or cross-platform solutions from Xamarin such as MonoDevelop, Mono Touch for iPhone, or Mono for Android) and an extensive collection of standard libraries (.NET Framework).

**Standard Query Operators and LINQ**

Assume that given a file of text—say, words.txt—you need to count the number of distinct words in that file, find the five most common ones, and visualize the result in a pie chart. If you think about this for a minute, it becomes clear that this is really an exercise in transforming collections. This is exactly the kind of task for which LINQ was designed. To keep things simple, we have implemented this example using LINQ to Objects to process the data in memory; however, with minimal modification the same code runs on LINQ to HPC (high-performance computing) over terabytes of data stored in commodity clusters.

The standard File.ReadAllText method provides the content of the file as a single giant string. You first need to chop up this string into individual words by breaking it at delimiter characters such as space, comma, period, etc. Once you have a list of words, you need to clean it up, removing all empty words. Finally, normalize all words to lowercase.

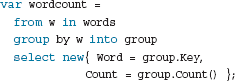
Using the LINQ sequence operators, you can transliterate the description from the previous paragraph directly into code:



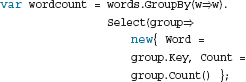
Instead of using the sequence operators directly, LINQ also provides a more "declarative" query comprehension syntax. Using comprehensions, you can rewrite the code as follows:

IMG_256

Once you have converted the file into a sequence of individual words, you can find the number of occurrences of each word by first grouping the collection by each word and then counting the number of elements in each group (which contains all occurrences of that word):



Without using query comprehension syntax, the code would look like this:



To find the top five most frequent words, you can order each record by Count and take the first five elements:

IMG_259

Now that you have a collection of the top five words in the file, you can visualize them in a pie chart, as in[Figure 1](http://cacm.acm.org/magazines/2011/10/131398-the-world-according-to-linq/fulltext" \l "F1). A pie chart is really nothing more than a collection of slices, where each slice consists of a number that represents the proportion of the total pie and a legend that describes what the slice represents. This means that by defining the charting API to be LINQ friendly, you can create charts by writing a query over the Google image charts API:

IMG_260

The await keyword is used in an unorthodox way to make the expensive coercion fromGoogle.Linq.Charts. Pie into an image that requires a network round-trip explicit.

This example just scratches the surface of LINQ. It provides a library of sequence operators such as Select, Where, GroupBy,... to transform collections, and it provides syntactic sugar in the form of query comprehensions that allows programmers to write transformations over collections at a higher level of abstraction.

To truly understand the power of LINQ, let's take a step back and investigate its origins and mathematical foundations. Don't worry, you need knowledge of only high school-level mathematics.

To truly understand the power of LINQ, let's take a step back and investigate its origins and mathematical foundations. Don't worry, you need knowledge of only high school-level mathematics.

**Datacentric Interpretation**

Relational algebra, which forms the formal basis for SQL, defines a number of constants and constructors for sets of values {Σ}, such as the empty set ØIMG_256{Σ}; injection of a value into a singleton collection {\_}IMG_257Σ→{Σ}; and the union of two sets into a new combined set IMG_258IMG_259{Σ}x{Σ}→{Σ}; There are also a number of relational operators such as projection, which applies a transformation to each element in a set πIMG_260(Σ→Λ) x{Σ}→{Λ};selection, which selects only those elements in a set that satisfy a given property IMG_261; Cartesian product, which pairs up all the elements of a pair of sets XIMG_262{Σ} x {Λ}→{ΣΛ}; and cross-apply, which generates a secondary set of values for each element in a first set @IMG_263(Σ→{Λ}) x{Σ}→{Λ}.

[Figure 2](http://cacm.acm.org/magazines/2011/10/131398-the-world-according-to-linq/fulltext" \l "F2) depicts the relational algebra operators using clouds to denote sets of values. An SQL compiler translates queries expressed in the familiar SELECT-FROM-WHERE syntax into relational algebra expressions; to optimize the query it applies algebraic laws such as distribution of selection: IMG_264 and then translates these logical expressions into a physical query plan that is executed by the RDBMS.

For example, the SQL query SELECT Name FROM Friend WHERE Likes(Friend, Sushi) is translated into the relational algebra expression π(fIMG_265f.Name, (σ (fIMG_266Likes(f,Sushi),Friend). To speed up the execution of the query, the RDBMS may use an index to quickly look up friends who like Sushi instead of doing a linear scan over the whole collection.

The cross-apply operator @ is particularly powerful since it allows for correlated subqueries where you generate a second collection for each value from a first collection and flatten the results into a single collection @ (f,{a,...,z})=f(a)IMG_267...IMG_268f(z). All other relational operators can be defined in terms of the cross-apply operator:

IMG_269

As a programmer you can easily imagine writing up a simple implementation of cross-apply: you would just iterate over the items in the input set, apply the given function, and accumulate the results into a result set. Such an implementation, however, wouldn't need its argument to be as set {Σ}; anything that we can iterate over such as a list, array, or hash table would suffice. Similarly, there is no reason at all that relational algebra operations should be restricted to sets of values {Σ}. They can be implemented based on other types of collections as well.

Perhaps surprisingly, there is also no reason that the operations passed into π, σ, and @ should be restricted to concrete functions Σ→Λ. In fact, you can use any representation of a function from which to determine which computation to perform. For example, in a language such as JavaScript you could simply pass a string and then use eval to turn it into executable code.

What you are searching for is the underlying interface that relational algebra implements. As long as there is a type constructor for collections M<Σ> that provides the operations that satisfy similar set-like algebraic properties as {Σ}, and a type constructor for computations IMG_270 that satisfies similar function-like properties as Σ→Λ, you can generalize relational algebra to the following set of operators and still be able to write SQL queries over these collections by desugaring query syntax:

IMG_271

For programmers this is just separating interface from implementation; mathematicians call the resulting structure monads, and instead of queries they speak of comprehensions.

An OO language such as C# uses the canonical interface for collections IEnumerable<T> as a specific instance of the abstract collection type M<T> and uses delegates Func<Σ,Λ> to represent computations IMG_272. By doing this, you recognize the operators from relational algebra as the LINQ standard query operators as defined in the Linq.Enumerable class, as shown in [Figure 3](http://cacm.acm.org/magazines/2011/10/131398-the-world-according-to-linq/fulltext" \l "F3).

Alternatively, you can use the IQueryable<T> interface to represent collections M<T> and expression treeExpression<Func<Σ,Λ>> to represent computations IMG_273. In that case you recognize the relational algebra operators as the LINQ standard query operators as defined in the Linq.Queryable class. The ability to treat code as data using morphisms—or in the C# case using the Expression type and lambda expressions for code literals—is a fundamental capability that allows the program itself to manipulate, optimize, and translate queries at runtime.

Instead of SQL syntax, the C# language defines XQuery-like comprehensions of the form from-where-select. The previous SQL query example looks like this:

IMG_274

Just as in SQL, comprehensions are translated by the compiler into the underlying LINQ query algebra:

IMG_275

Depending on the overloads of Where and Select, the lambda expressions will be interpreted as code or data. A simplified implementation of IQueryable is discussed later.

As already shown, monads and their incarnation in practical programming languages such as LINQ are simply a generalization of relational algebra by imagining the interface that relational algebra implements. The concepts and ideas behind LINQ should therefore be deeply familiar to both database people and programmers.

**Theory into Practice**

Unlike Haskell, which has incorporated monads and monad comprehensions in a principled way, the C# type system is not expressive enough for the mathematical signatures of the monad operators. Instead, the translation of query comprehensions is defined in a purely pattern-based way. In a first pass, the compiler blindly desugars comprehensions, using a set of fixed rules, into regular C# method calls and then relies on standard type-based overload resolution to bind query operators to their actual implementations.

For example, the method Foo Select(Bar source, Func<Baz, Qux> selector), which does not involve any collection types, will be bound as the result of translating the comprehension

IMG_256

into the desugared expression

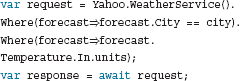
IMG_257

This technique is used extensively in the example presented next.

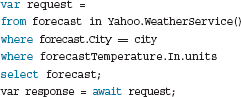
Another difference between LINQ and its monadic basis is a much larger class of query operators including grouping and aggregation, which is more SQL-like. Interestingly, the inclusion of comprehensions in C#, which was inspired by monad and list comprehensions in Haskell, has recursively inspired Haskell to add support for grouping and aggregation to its comprehensions.

**Custom Query Providers**

The Yahoo weather service (<http://developer.yahoo.com/weather/>) allows weather forecast queries for a given location, using either metric or imperial units for the temperature. This simple service is a good way to illustrate a non-standard implementation of the LINQ query operators that is completely specialized for this particular target and that will allow only strongly typed queries of the form



or equivalently using query comprehensions



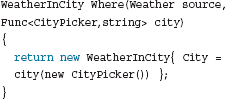
The implementation of the operators extracts the city and temperature unit from the query and uses them to create a REST call (<http://weather.yahooapis.com/forecastrss?w=woeid&u=unit>) to the Yahoo service as a result of using the await keyword to explicitly coerce the request into a response.

The technical trick in this style of custom LINQ provider is to project the capabilities of the target query language—in this case the Yahoo weather service that requires (a) a city and (b) a unit—into a type-level state machine that guides users in "fluent" style (and supported by IntelliSense) through the possible choices they can make ([Figure 4](http://cacm.acm.org/magazines/2011/10/131398-the-world-according-to-linq/fulltext" \l "F4)).

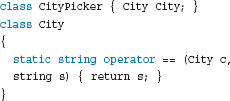
At each transition in the state machine we collect the various parts of interest of the query—in this case, the particular city and the temperature unit. In principle, the city doesn't really need to come first, but it might be more natural for the graph to allow either type of where clause to be specified first, but with the restriction that both where clauses are required. I leave the lifting of this restriction in the state machine as an exercise for the reader.

Note that none of the types Weather, WeatherInCity, or WeatherIn-CityInUnits implements any of the standard collection interfaces. Instead they represent the stages in the computation of a request that will be submitted to the Yahoo Web service, for which you do not need to define an explicit container type. What also surprises many people is that neither of the two Where methods actually computes a Boolean predicate. Even stranger is that each of the three occurrences of the range variable forecast in the query has a different type.

The Weather class defines a single method that picks the city specified in the query and passes it on toWeatherInCity, which is the next state in the type-based state machine:



The "predicate" in the Where method is a function that takes a value of type CityPicker, which has a single property that returns the phantom class City that exists only to facilitate IntelliSense and whose equality check immediately returns the string passed to the equality operator:



As a result of this, calling Yahoo. Weather().Where(forecastIMG_260 forecast=="Seattle") really is just a convoluted way of creating a new WeatherInCity{City = "Seattle"} instance using a Where method that does not take a Boolean predicate and an equality operator that returns a string.

You can use the same trickery in WeatherInCityInUnits Where(Func<UnitPicker,Unit> predicate), so that calling Where(forecastIMG_261forecast.Temperature.In.Celsius) on the result of the previous filter creates an instance of new WeatherInCityInUnits{City = "Seattle", Unit = Unit. Celsius}. The techniques used here are not only useful for defining custom implementations of the LINQ operators, but also can be leveraged for building fluent interfaces in general.

Since the Yahoo service requires the city as a WOEID (where on earth ID), we need to make two service calls under the hood in order to retrieve the weather forecast. The first service call retrieves the WOEID of a requested city via <http://where.yahooapis.com/v1/places.q(city)?appid=XXXX>. If that successfully returns, then a second call is made to retrieve the weather forecast for that location. The calls to the Web server are performed asynchronously and both return a Task<T> (in Java you would use java.util.concurrent. Future<T> to represent the result of an asynchronous operation). Since we can consider a Task<T> as a kind of collection that contains (at most) one element, it also supports the LINQ query operators, and we have turtles all the way down; the LINQ implementation for Weather is defined using the LINQ implementation ofTask<T> (see [Figure 5](http://cacm.acm.org/magazines/2011/10/131398-the-world-according-to-linq/fulltext" \l "F5)).

Though this is an extremely small and limited example, it clearly illustrates many of the techniques used to create real-world LINQ providers such as LINQ to Objects, LINQ to SharePoint, LINQ to Active Directory, LINQ to Twitter, LINQ to Netflix, and many more.

**Generic Query Providers**

The weather service query provider example is structured as an internal DSL. While this provides a great user experience with maximum static typing, it allows little room for reusing the actual implementation of the provider. It is custom built for the particular target top-to-bottom. At the other end of the spectrum we can create a completely generic query provider that records a complete query "as is," using a little bit of meta-programming magic.

In C# a lambda expression such as xIMG_256x>4711 can be converted into either a delegate—say, of typeFunc<int,int>—or into an expression tree of type Expression<Func<int,int>>, which treats the code of a lambda expression as data. In Lisp or Scheme one would use syntactic quoting to treat code as data. In C# lambda expressions in combination with the type expected by the context provide a type-based quotingmechanism.

The class Queryable implements LINQ standard query operators that take expression trees as arguments and return an Expression representation of their selves, very much like a macro recorder as shown in [Figure 6](http://cacm.acm.org/magazines/2011/10/131398-the-world-according-to-linq/fulltext" \l "F6).

For example, given a value xs of type Queryable<int>, the call xs.Select(xIMG_257x>4711) causes the lambda expression to be converted into an expression tree (shown in bold), and then returns an expression tree that represents the call itself xs.Select(xIMG_258x>4711). Now it is up to the specific query provider (such as LINQ to SQL, Entity Framework, LINQ to HPC) to translate the resulting expression tree and compile it into the target query language.

The IQueryable-based implementation that ships with the .NET Framework uses the same scheme as the simplified example code just shown, except that it is interface based, and it therefore relies on a second interface IQueryProvider to supply a factory for creating instances of IQueryable.

The advantage of a generic query provider is that you can offer general services such as query optimization, which implement rewrite rules such as xs.Union(ys).Where(p) = xs.Where(p). Union(ys.Where(p))that can be reused across many LINQ providers.

**LINQ-Friendly APIs**

All examples so far have dealt with implementing particular LINQ providers. An orthogonal aspect of LINQ is APIs that leverage particular LINQ implementations, often LINQ to Objects. For example, LINQ to XML is an API for manipulating XML documents that has been designed specifically with LINQ in mind, which eliminates the need for a DSL such as XQuery or XPath to query and transform XML.

The Google Chart API is a Web service that lets you dynamically create attractive-looking charts, using a simple URI (Uniform Resource identifier) scheme. The URI syntax for Google charts, however, is not very sequence friendly. For example, the URI for the earlier sample pie chart looks like this:



The problem is that the specification for the labels (chl=the|of|a|that|is) and the specification for the data set (chd=t:21,12,7,7,6) of the chart are given in two separate collections. On the other hand, to generate a pie chart using a query, you want a single collection of pairs that specify both the value and the label for each slice as in from w in top5 select new Slice(w. Count){Legend = r.Word}.

In other words, to make the Google Chart API sequence friendly, you must transpose a collection of pairsM<SxT> into a pair of collections M<S>xM<T>. Functional programmers immediately recognize this as an instance of the function UnzipIMG_257(R→S x R→T x M<R>)→M<S>xM<T>. Unzip can convert a chart that contains a sequence of slices into the URI format required by the Google Chart API by formatting the various collections using the separators prescribed by the chart service as shown in [Figure 7](http://cacm.acm.org/magazines/2011/10/131398-the-world-according-to-linq/fulltext" \l "F7).

**Conclusion**

Big data is not just about size. It is also about diversity of data, both in terms of data model (primary key/foreign key versus key/value), as well as consumption pattern (pull versus push), among many other dimensions. This article argues that LINQ is a promising basis for big data. LINQ is both a generalization of relational algebra and has deep roots in category theory—in particular, monads.

With LINQ, queries expressed in C#, Visual Basic, or JavaScript can be captured either as code or expression trees. Either representation can then be rewritten and optimized and subsequently compiled at runtime. We have also shown how to implement custom LINQ providers that can run in memory and over SQL and CoSQL databases, and we have presented LINQ-friendly APIs over Web services. It is also possible to expose streaming data so as to implement the LINQ standard query operators, resulting in a single abstraction that allows developers to query over all three dimensions of big data.

**Acknowledgments**

Many thanks to the Cloud Programmability team members Savas Parastatidis, Gert Drapers, Aaron Lahman, Bart de Smet, and Wes Dyer for all the hard work in building the infrastructure and prototypes for all flavors of LINQ and coSQL; to Rene Bouw, Brian Beckman, and Terry Coatta for helping to improve the readability of this article; and to Dave Campbell and Satya Nadella for providing the necessary push to actually write it.

### 中文译文

根据LINQ的世界

程序员从许多不同的来源，例如传感器，社交网络，用户界面，电子表格和股票行情构建Web和基于云的应用程序一起线数据。大多数这类数据不适合在传统的关系型数据库的封闭和清洁的世界。它太大了，非结构化，非规范化和流媒体的实时性。呈现在所有这些不同的数据模型和查询语言的统一编程模型起初似乎是不可能的。通过集中在共同点代替的差异，但是，大多数的数据源将接受某种形式的计算来过滤和转换数据的集合。

数学家看似不同的数学结构之间早就观察到的相似，正式通过范畴论，单子专门的概念作为收藏品的推广这种洞察力。语言如Haskell中，斯卡拉的Python和JavaScript，甚至未来的版本中已纳入清单，单子解析来处理副作用和计算过的集合。Visual Basic和C＃的.NET语言通过了LINQ（语言集成查询）的形式单子，以此来弥补对象和数据的世界之间的差距。本文介绍了单子和LINQ与任意类型的任意集合使用的关系代数和SQL的推广，并解释了为什么这使得LINQ大数据令人信服的依据。

LINQ介绍在C＃3.0和Visual Basic 9为一组API和相应的语言扩展，弥补编程语言的世界和数据库世界之间的差距。尽管在外部开发者社区有关LINQ的持续刺激，该技术的全部潜力尚未达到。由于LINQ的基本性质，仍然有巨大的潜力，其映射情况之外的对象关系（O / R），尤其是在大数据领域。

大数据的出现，使得它比以往任何时候都程序员有一个抽象，让他们来处理，转换，作曲，查询，分析，并计算横跨至少有三个不同的维度更重要的是：音量，或大或小，从数十亿;项目成果屈指可数的各种模型中，结构化或非结构化，扁平或嵌套; 和速度，流或持续，推或拉。其结果是，我们看到了新的数据模型，查询语言和执行面料令人兴奋的数字。LINQ可以虚拟一个抽象背后所有这些方面。

举个例子来说，Apache的Hadoop的生态系统。它配备了至少八个外部的DSL（领域特定语言）或API：一组低级别的Java接口为MapReduce的计算; 层叠，一个“数据处理定义语言，作为一个简单的Java API实现，” ，“基于流数据的简单和灵活的架构流程，” ，承载大量数据的“高层次的语言表达数据分析程序;” HiveQL，一个“类似于SQL的语言，便于数据的汇总，即席查询和大型数据集分析;“ CQL，一个”建议的语言对Cassandra的数据管理;“Oozie的，基于XML的”专业从事基于时间和数据运行的工作流协调引擎触发;“ 和Avro的，对于数据串行化的模式语言。

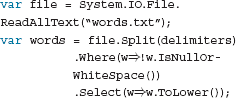
要创建一个终端到终端应用，程序员需要使用多个这些外部DSL的除了通用编程语言如Java胶水都在一起。如果数据来自外部RDBMS（关系数据库管理系统），或推基于源，然后更DSL的诸如SQL或StreamBase是必需的。另一方面使用LINQ和C＃或Visual Basic，程序员可以使用内部 DSL的编程针对附带工具（Visual Studio或跨平台解决方案的任何形状或通用的面向对象的内部（面向对象）语言数据的形式从Xamarin如MonoDevelop的，单触iPhone或单声道的Android）和标准库，广泛收集（.NET框架）。

**标准查询操作和LINQ**

假设给定的文本，比如说一个文件，words.txt -你需要计算的不同词的数量在该文件中，找到五个最常见的，而在饼图中显现的结果。如果你觉得这个一分钟，很明显，这是真的转化集合练习。这正是那种任务为其LINQ的设计。为了简单起见，我们已经实现了使用LINQ to对象来处理内存中的数据这个例子; 然而，稍加修改同样的代码在LINQ对存储在商品集群TB级的数据运行到HPC（高性能计算）。

标准File.ReadAllText方法提供的文件的内容作为单个巨型串。首先，您需要通过分隔符如空格，逗号，句号等，一旦你有一个单词列表打破它砍了这个字符串转换成个人的话，你需要把它清理干净，清除所有空话。最后，规范所有单词小写。

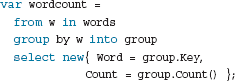
用LINQ顺序运算符，可以音译从前面的段落直接进入代码的描述：



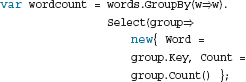
而不是直接使用序列运营商，LINQ还提供了更多的“声明”查询理解语法。使用解析，你可以重写代码如下：

IMG_257

一旦你已将该文件转换成单个字的序列，则可以通过首先由每个单词分组的集合，然后计数各组中的元素的数量（其包含该字的所有实例）找到的每个单词的出现次数：



如果不使用查询理解语法，代码是这样的：



要查找的五大最频繁的话，你可以命令每个记录计数，并采取先五个要素：

IMG_260

现在，你有文件中的前五个词的集合，你可以想像他们在饼图中，如在[图1中](http://cacm.acm.org/magazines/2011/10/131398-the-world-according-to-linq/fulltext" \l "F1)。饼图实在没有什么比片的集合，其中每个片由若干更多的代表总馅饼和介绍了片代表一个传奇的比例。这意味着，通过定义图表API是LINQ友好，你可以写在谷歌图片图表API查询创建图表：

IMG_261

该的await关键字以非正统的方式用于制作从昂贵的胁迫Google.Linq.Charts。馅饼切成需要一个网络往返明确的图像。

这个例子仅仅触及LINQ的表面。它提供了顺序运算符，如库中选择，其中，GROUPBY，...改造收藏，并在查询解析的形式，允许程序员在更高的抽象水平写对集合转换提供了语法糖。

要真正了解LINQ的力量，让我们退后一步，并探讨其起源和数学基础。别担心，你只需要高中层次的数学知识。

要真正了解LINQ的力量，让我们退后一步，并探讨其起源和数学基础。别担心，你只需要高中层次的数学知识。

**数据为中心的解读**

关系代数，形成正式的基础SQL，定义了数套值{Σ}，常数和构造，如空集 Ø IMG_256{Σ}; 注入的值到一个单集合{\_} IMG_257Σ→{ Σ}; 与工会的两套进入一个新的组合集{}ΣX {}Σ{→Σ}; 也有一些关系运算符如投影，它适用于转型的每个元素在一组π（Σ→Λ）X {Σ}→{Λ}; 选择，只选择在一组满足这些元素鉴于财产; 笛卡儿积，这对一对的集合X中的所有元素{Σ}×{Λ}→{ΣΛ}; 和跨应用，其产生在第一组@每个元素的次级组值（Σ→{Λ}）X {Σ}→{Λ}。IMG_258IMG_259IMG_260IMG_261IMG_262IMG_263

[图2](http://cacm.acm.org/magazines/2011/10/131398-the-world-according-to-linq/fulltext" \l "F2)描述了使用云来表示组值的关系代数运算符。一个SQL编译器将在表达的查询熟悉的SELECT-FROM-WHERE语法为关系代数表达式; 优化查询它适用代数的法律，如选择的分布：IMG_264然后会将这些逻辑表达式成由RDBMS执行的物理查询计划。

例如，SQL查询SELECT名字从朋友WHERE喜欢（朋友，寿司）被翻译成关系代数表达式π（F IMG_265f.Name，（σ（F IMG_266喜欢（F，寿司），朋友）。为了加快执行查询时，可以RDBMS使用索引以便快速查找朋友谁喜欢寿司，而不是在整个集合做线性扫描。

操作符@的申请十字架是特别强大，因为它允许您在哪里产生从第一集合中的每个值的第二收集和扁平化的结果到一个单一的集合@相关子查询（F，{A，...，Z}） = F（一）IMG_267... IMG_268。F（z）的所有其它关系运算符可以在交叉应用算子来定义：

IMG_269

作为一个程序员，你可以很容易地想象写了一个简单的实现跨应用：你只是遍历输入集中的项目，运用给定的函数，结果积聚成一个结果集。这样的实施，但是，不会需要它的参数是作为集合 {Σ}; 任何我们可以遍历如列表，数组或哈希表就足够了。同样，没有理由在所有的关系代数操作应限制值集{}Σ的。它们可以基于其它类型的集合，以及来实现。

令人惊讶的是，还有没有理由的操作传递到π，σ，并@应仅限于具体的功能 Σ→Λ。事实上，你可以使用一个函数，从中确定要执行哪些计算任何声明。例如，在语言如JavaScript，你可以简单地传递一个字符串，然后使用eval把它变成可执行代码。

你所寻找的是底层接口的关系代数工具。只要存在用于集合类型构造中号<Σ>提供满足组类似状代数性质为{Σ}的操作，和用于计算一个类型构造IMG_270满足相似功能样特性如Σ→Λ，则可以概括关系代数以下集合运算符，并仍然可以通过脱糖查询语法来写了这些集合的SQL查询：

IMG_271

对于程序员来说这仅仅是接口和实现分离; 数学家调用生成的结构单子，而不是查询他们讲的内涵。

一个面向对象的语言，如C＃使用规范的接口集合的IEnumerable <T>的抽象集合M型<T>和使用委托Func键<Σ，Λ>代表计算的特定实例IMG_272。通过这样做，从关系代数为如在所限定的LINQ标准查询操作符识别运营Linq.Enumerable类，如图[图3](http://cacm.acm.org/magazines/2011/10/131398-the-world-according-to-linq/fulltext" \l "F3)。

另外，您也可以使用的IQueryable <T>接口表示集合中号<T>和表达式树表达<Func键<Σ，Λ>>代表计算IMG_273。在这种情况下，你承认作为定义的关系代数运营商为LINQ标准查询运算符Linq.Queryable类。对待代码使用数据的能力态射，或者使用C＃的情况下表达代码类型和lambda表达式文字，是一个基本的功能，它允许程序本身来操作，优化，并在运行时转换查询。

相反，SQL语法，C＃语言定义表单中的XQuery般的推导从-那里选。前面的SQL查询的例子如下：

IMG_274

正如在SQL中，解析由编译器进入下面LINQ查询代数翻译：

IMG_275

根据的重载凡和选择的lambda表达式将被解释为代码或数据。的简化实施的IQueryable将在后面讨论。

正如已经显示的，单子和他们在实际编程语言如LINQ化身只是通过想象关系代数实现接口关系代数的推广。因此LINQ背后的理念和思路应该是深深熟悉的两个人数据库和程序员。

**理论到实践**

不像哈斯克尔，这已纳入单子和单子内涵有原则的方式，C＃的类型系统是不是单子操作符的数学签名足够的表现力。相反，查询内涵的翻译纯粹基于模式的方式进行定义。在第一遍时，编译器一味desugars解析，使用一组固定的规则，进入正规C＃的方法调用，然后依赖于标准型为主的重载解析到查询操作结合自己的实际实现。

例如，该方法美孚选择（酒吧来源，Func键<巴兹，Qux>选择器），这不涉及任何集合类型，将被绑定为翻译理解的结果

IMG_256

进入脱糖表达

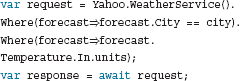
IMG_257

这种技术在未来呈现的示例广泛使用。

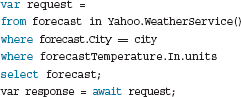
LINQ和其一元基础之间的另一个区别是一个更大的类查询操作符，包括分组和聚集，这是更类似SQL的。有趣的是，内涵在C＃中的包容，这是在Haskell灵感单子和list解析，递归地激发了哈斯克尔增加对分组和聚合到它的内涵支持。

**自定义查询供应商**

雅虎气象服务（<http://developer.yahoo.com/weather/>）允许天气预报查询，对于一个给定的位置，使用公制或英制单位的温度。这个简单的服务是要说明一个非标准实施LINQ查询运算符是完全专门为这个特定目标，这将允许一个很好的方式只强类型形式的查询



或等价使用查询解析



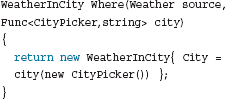
运营商的实施从查询中提取城市和温度单位，并用它们来创建一个REST调用（<http://weather.yahooapis.com/forecastrss?w=woeid&u=unit>）到雅虎服务作为使用的结果在的await关键字来明确强制要求到响应。

在这种风格的自定义LINQ提供的技术诀窍是项目目标的查询语言，在这种情况下，雅虎的天气要求（一）城市和服务的能力（B）的单位，变成一种层次状态机引导中的“流畅”的风格用户（以及由智能感知支持）通过可能的选择，他们可以作出（[图4](http://cacm.acm.org/magazines/2011/10/131398-the-world-according-to-linq/fulltext" \l "F4)）。

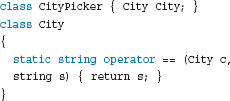
在该状态机中的每个过渡我们收集的查询-在这种情况下，特别是城市和温度单位感兴趣的各个部分。原则上，这个城市并不真的需要是第一位的，但它可能是更自然的曲线图，让任一类型的，其中首先被指定的条款，但其限制是双方在这里的必备条款。我离开这个限制在状态机的提升作为一个练习留给读者。

请注意，没有任何类型的天气，WeatherInCity或WeatherIn-CityInUnits实现任何标准集合接口。相反，他们代表了将提交给雅虎的Web服务，为您不需要定义一个明确的容器类型要求的计算阶段。什么也很多人惊奇的是，无论是两凡方法实际计算布尔谓词。更奇怪的是，这三个事件的范围变量的预测在查询中有不同的类型。

在天气类定义挑选查询指定的城市，将其传递到一个方法WeatherInCity，这是在基于类型的状态机的下一个状态：



在“上游”，在其中的方法是一个函数，类型的值CityPicker，其中有一个返回幻象类的一个属性城只存在，以促进智能感知，其平等检查立即返回传递给相等操作字符串：



由于这个结果，主叫雅虎。天气（）。如果（预测IMG_260预报==“西雅图”），真的只是创建一个新的令人费解的方式WeatherInCity {市=“西雅图”}例如使用其中的方法，这并不需要一个布尔谓词，并返回一个相等运算符字符串。

可以使用在相同的诡计WeatherInCityInUnits凡（Func键<UnitPicker，单元>谓词），以便在调用凡（预测IMG_261forecast.Temperature.In.Celsius）上先前的滤波器的结果创建的新的实例WeatherInCityInUnits {市=“西雅图“，单位=单位。摄氏}。此处所使用的技术不仅用于限定LINQ运算符的自定义实现是有用的，但也可以利用在一般建筑物流利接口。

由于雅虎服务需要城市作为一个WOEID（在地球上的ID），我们需要做的引擎盖下两个服务电话以获取天气预报。第一个服务调用通过检索请求的城市的WOEID <http://where.yahooapis.com/v1/places.q(city)?appid=XXXX>。如果成功返回，那么第二个调用来检索该位置的天气预报。对Web服务器的调用是异步执行，都返回一个任务<T> （在Java中你会使用的java.util.concurrent。未来<T>表示异步操作的结果）。因为我们可以考虑一个任务<T>作为一种集合，它包含（最多）一个元素，它也支持LINQ查询运营商，我们有海龟一路下滑; 为实现LINQ 天气使用LINQ执行规定任务<T> （参见[图5](http://cacm.acm.org/magazines/2011/10/131398-the-world-according-to-linq/fulltext" \l "F5)）。

尽管这是一个非常小的和有限的例子，它清楚地说明了许多用于创建真实世界的LINQ提供程序，如LINQ到对象，LINQ到SharePoint，LINQ到Active Directory，LINQ到Twitter，LINQ到Netflix和许多技巧更多。

**通用查询供应商**

天气服务查询提供的例子的结构为内部DSL。虽然这提供了最大的静态类型的用户体验，它可以让小空间复用实际执行的供应商。它是定制的自顶至底的特定目标。在光谱的另一端，我们可以创建一个记录的完整查询完全通用的查询提供“按原样”使用的元编程魔术一点点。

在C＃中的lambda表达式，如点¯x IMG_256X> 4711可以转换成任何一个代表说，类型的函数功能<INT，INT> -或成类型的表达式树表达<Func键<INT，INT >>，其中对待Lambda表达式作为数据的代码。在Lisp或计划，应当使用语法引用对待代码的数据。在与由上下文期望的类型组合的C＃lambda表达式提供一种基于类型的引用机制。

类可查询实现LINQ标准查询运算符采取表达式树作为参数，并返回一个表达如图他们自我的表现，非常像宏录制[图6](http://cacm.acm.org/magazines/2011/10/131398-the-world-according-to-linq/fulltext" \l "F6)。

例如，给定值XS型可查询<int>的，呼叫xs.Select（点¯x IMG_257X> 4711）引起lambda表达式转换成表达式树（以粗体显示），然后返回表示表达式树调用本身xs.Select（点¯x IMG_258X> 4711）。现在，它是由特定的查询提供程序（如LINQ to SQL中，实体框架，LINQ到HPC）翻译结果表达式树，并把它编译成目标查询语言。

该IQueryable的基础的附带的.NET框架使用上面所示相同的方案简化示例代码，但它是基于接口的实现，因此它依赖于第二接口IQueryProvider用于创建实例提供一个工厂的IQueryable。

一个通用的查询提供的好处是，你可以提供一般的服务，如查询优化，它们实现重写规则，如xs.Union（YS）。凡（P）= xs.Where（P）。联盟（ys.Where（P）） ，可以在许多LINQ提供重复使用。

**LINQ友好的API**

所有的例子都涉及执行特定的LINQ提供程序。LINQ的正交方面是利用特定的LINQ的实现，往往LINQ到对象的API。例如，LINQ到XML是用于处理已专门LINQ的想法，这样就无需对DSL等的XQuery或XPath查询和转换XML设计的XML文档的API。

该谷歌图表API是一个Web服务，它可以让你动态地创建寻找有吸引力，图表，使用简单的URI（统一资源标识符）方案。对于谷歌图表URI语法，但是，是不是非常友好的序列。例如，URI的早期样品饼图看起来是这样的：

问题是，本说明书中为标签（叶绿素=在|的| A |即|是）和用于所述数据集的规范（CHD = T：21,12,7,7,6）的图表中给出两个独立的集合。另一方面，以使用查询生成饼图，想要对的单个集合，同时指定的值，并为每个切片作为标签从w的TOP5选择新切片（瓦特数）{图例= R 。字}。

换句话说，使谷歌图表API序列友好，你必须转 ​​对集合中号<S×T对>成一对藏品中号<S> XM <T> 。功能的程序员立刻认识到这一点作为函数的一个实例解压IMG_257（R→S个R→T的×M的<R>）→M <S> XM <T>。解压缩可以转换包含如图切片成通过格式化使用由图表服务规定的隔板的各种集合由谷歌图表API所需URI的格式的序列的图。

**结论**

大数据不只是大小。它也是对数据的多样性，无论是在数据模型而言（主键/外键与键/值），以及消费模式（拉与推），许多其它尺寸之间。本文认为，LINQ是大数据有前途的基础。LINQ既是关系代数的概括和范畴论，特别是单子有着很深的渊源。

随着LINQ，查询在C＃，Visual Basic中表示，或JavaScript可以被捕获或者作为代码或表达式树。然后或者表示可以改写和优化，并随​​后在运行时编译。我们还展示了如何实现可在存储器中，并且SQL和数据库CoSQL运行自定义LINQ提供程序，我们已经在Web服务提出了LINQ友好的API。另外，也可以以露出流数据，以便实现LINQ标准查询操作符，产生一个单一的抽象，它允许开发人员查询过大数据的所有三个特点。

**致 谢**

非常感谢云可编程队员萨瓦斯Parastatidis，格特·布商行，亚伦Lahman，巴特迪斯和韦斯·戴尔在建设LINQ和coSQL的所有不同的基础设施和原型，所有的辛勤工作; 勒内BOUW，布赖恩·贝克曼和特里Coatta帮助改善这一文章的可读性; 戴夫·坎贝尔和萨蒂亚·纳德拉提供必要的帮助写出来。

### 毕业设计任务书

教务系统的设计与实现（教师模块）

**一、毕业设计目的**

随着科技的进步，用电脑软件来完成相关工作的操作，简化人力，通过毕业设计对学生所学的专业知识以及专业能力进一步的充分利用和考核，同时提高学生进行应用系统设计开发的能力，培养自己独立动手解决问题和困难的能力，以及学习新的知识，广泛查阅资料的能力，使得自己以后工作生活更加的有经验和活力。

**二、主要内容**

通过设计教务系统帮助学校管理教师和学生的相关信息，使得教师工作更加的高效且快捷的实现，对教师去查询和选择培养学生方案打下基础，教师可以通过查看所授班级对应的学生的成绩及时的调整自己的教学工作安排，及时的改进教学方式，同时也可以比较客观的评估课程。教师在登陆教务系统的时候也可以看到自己的个人简历以及入学时间，同时可以进行相关的信息修改，如登陆密码。

1. **重点解决的问题**

教师所授课程内容的查询以及教师对授课的行政班级的学生的成绩的查询以及录入，教师个人信息以及登录密码的修改。如何针对分析的问题，提出相对应的解决方法来实现预期功能。

1. **主要技术指标或主要参数**

SQL Server 数据库、VisualStudio2010.三级架构，linq to entities.Entities Framework

**五、基本要求**

系统所需要实现的功能都能有所实现，测试结果无异常

1. **其它（包括选题来源）**

选题来源：实践

指导教师： 年 月 日

### 开题报告

**华北水利水电大学本科生毕业设计（论文）开题报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 学生姓名 | 潘靓妮 | 学号 | 201215327 | 专业 | 计算机科学与技术 |
| 题目名称 | 教务系统的设计与实现（教师模块） | | | | |
| 研究或设计概述  （500字左右） | 教务系统的设计与实现（教师模块）主要是为了教师的工作更加的便捷而展开的软件设计，教师可以通过自己的账号和密码进入自己的操作页面，教师可以看到自己的个人简历，并进行个人密码的修改等相关操作，可以查看到自己不同学年不同学期所教授的课程以及授课的班级信息。通过班级信息可以查看到班级内的同学信息。通过所教课程给每个同学进行分数的评估，而没有选到相关课程以及相关老师的学生的成绩则不由相关老师来评定成绩。成绩录入中可以打印成绩登记册，录入学生成绩并且查看学生成绩。录入成绩中通过学年学期可以看到对应课程的学分，学时，上课班级以及录入人数。通过学年学期可以看到教学安排，看到课程号，课程名，学分，学时，上课班级等相关信息。学年学期看到指导环节的相关信息，如：学分，班级等。  在设计的过程中，需要梳理清楚各个实体及属性的对应信息。并且构建好一个结构良好的概念模型，通过数据库以及C#来完成相关内容的实施。 | | | | |
| 主要内容 | 通过设计教务系统帮助学校管理教师和学生的相关信息，使得教师工作更加的高效且快捷的实现，对教师去查询和选择培养学生方案打下基础，教师可以通过查看所授班级对应的学生的成绩及时的调整自己的教学工作安排，及时的改进教学方式，同时也可以比较客观的评估课程。教师在登陆教务系统的时候也可以看到自己的个人简历以及入学时间，同时可以进行相关的信息修改，如登陆密码。 | | | | |
| 主要参考文献（不少于10篇） | [1] 陈毓祯.[高校教务管理系统的分析与设计](http://www.cnki.net/kcms/detail/detail.aspx?filename=1013306374.nh&dbcode=CMFD&dbname=CMFD2014&v=" \t "http://www.cnki.net/KCMS/detail/frame/_blank)[D]. 云南大学 2013  [2] 黄晓薇.[高职院校教务管理系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=1013317334.nh&dbcode=CMFD&dbname=CMFD2014&v=" \t "http://www.cnki.net/KCMS/detail/frame/_blank)[D]. 华南理工大学 2013  [3] 赵成松.[基于.NET平台的教务管理系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=1013330141.nh&dbcode=CMFD&dbname=CMFD2014&v=" \t "http://www.cnki.net/KCMS/detail/frame/_blank)[D]. 电子科技大学 2013  [4] 雷丽娟.[基于数字化校园教务管理系统的研究与设计](http://www.cnki.net/kcms/detail/detail.aspx?filename=1013330783.nh&dbcode=CMFD&dbname=CMFD2014&v=" \t "http://www.cnki.net/KCMS/detail/frame/_blank)[D]. 电子科技大学 2013  [5] 王罗乐.[西华大学教务管理系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=1013334525.nh&dbcode=CMFD&dbname=CMFD2014&v=" \t "http://www.cnki.net/KCMS/detail/frame/_blank)[D]. 电子科技大学 2013  [6] 朱其现.[基于JSP的教务管理系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=1014142169.nh&dbcode=CMFD&dbname=CMFD2014&v=" \t "http://www.cnki.net/KCMS/detail/frame/_blank)[D]. 厦门大学 2013[7] 栾柏宇.[高校教务管理系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=1013194366.nh&dbcode=CMFD&dbname=CMFD2013&v=" \t "http://www.cnki.net/KCMS/detail/frame/_blank)[D]. 吉林大学 2013  [8] 潘芳芳.[高校教务管理系统设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=1013135586.nh&dbcode=CMFD&dbname=CMFD2013&v=" \t "http://www.cnki.net/KCMS/detail/frame/_blank)[D]. 南昌大学 2012  [9] 杨志荣.[基于ASP.NET+SQL Server 2005技术的云师大住房公积金管理系统设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=1013149250.nh&dbcode=CMFD&dbname=CMFD2013&v=" \t "http://www.cnki.net/KCMS/detail/frame/_blank)[D]. 电子科技大学 2012  [10] 徐鹏.[基于Web数据库的教务管理系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=JZGC201103167&dbcode=CJFQ&dbname=CJFD2011&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J].价值工程.2011(03)  [11] 刘建花.[在线教务管理系统的研究与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=DNZS201026107&dbcode=CJFQ&dbname=CJFD2010&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J]. 电脑知识与技术.2010(26)  [12] 韩利娟.[基于Structs框架的教务管理系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=FJDN201008061&dbcode=CJFQ&dbname=CJFD2010&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J].福建电脑.2010(08)  [13] 陈戍.[基于WEB的成教教务管理系统设计探讨](http://www.cnki.net/kcms/detail/detail.aspx?filename=XXJK201011144&dbcode=CJFQ&dbname=CJFD2010&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J].中国科技信息.2010(11）  [14] 叶建国.[高校教务管理问题浅析](http://www.cnki.net/kcms/detail/detail.aspx?filename=WJZZ201203091&dbcode=CJFQ&dbname=CJFD2012&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J]. 文教资料.2012(03)  [15] 朱璜润,唐建华.[教务管理信息化建设探讨](http://www.cnki.net/kcms/detail/detail.aspx?filename=ZDYX201202029&dbcode=CJFQ&dbname=CJFD2012&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J]. 卫生职业教育.2012(02)  [16] 冯琰君,马霖,王琬华,樊晨辉,罗洋.[教务管理信息系统设计](http://www.cnki.net/kcms/detail/detail.aspx?filename=ZXLJ201136161&dbcode=CJFQ&dbname=CJFD2011&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J].科技资讯. 2011(36)  [17] 方富贵.[基于web的教务管理系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=DNZS201118005&dbcode=CJFQ&dbname=CJFD2011&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J].电脑知识与技术.2011(18)  [18] 刘强,吕晓娴,杨丽.[高校教务管理系统的设计与构建](http://www.cnki.net/kcms/detail/detail.aspx?filename=CYYT201107135&dbcode=CJFQ&dbname=CJFD2011&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J].产业与科技论坛. 2011(07)  [18] 钱鸽.[基于Web的高校教务管理信息系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=HTSF201102117&dbcode=CJFQ&dbname=CJFD2011&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J].和田师范专科学校学报.2011(02)  [19] 徐鹏.[基于Web数据库的教务管理系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=JZGC201103167&dbcode=CJFQ&dbname=CJFD2011&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J].价值工程.2011(03)  [20] 刘建花.[在线教务管理系统的研究与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=DNZS201026107&dbcode=CJFQ&dbname=CJFD2010&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J].电脑知识与技术.2010(26)  [21] 韩利娟.[基于Structs框架的教务管理系统的设计与实现](http://www.cnki.net/kcms/detail/detail.aspx?filename=FJDN201008061&dbcode=CJFQ&dbname=CJFD2010&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J]. 福建电脑. 2010(08)  [22] 陈戍.[基于WEB的成教教务管理系统设计探讨](http://www.cnki.net/kcms/detail/detail.aspx?filename=XXJK201011144&dbcode=CJFQ&dbname=CJFD2010&v=" \t "http://www.cnki.net/kcms/detail/frame/_blank)[J].中国科技信息.2010(11) | | | | |
| 采取的主要技术路线或方法 | SQL Server 数据库、VisualStudio2010.三级架构，linq to entities.Entities Framework | | | | |
| 时间安排 | 1、第1-3周，搜集资料并拟定毕设题目。  2、第3-4周，提交毕设开题报告。  3、第4-7周，开始做设计并写毕设报告初稿。  4、第7-8周，做毕设并提交设计初稿。  5、第9-10周，修改设计，提交正式设计报告。  6、第11-13周写论文，论文提交与修改。  7、第13-14周准备答辩及正式答辩 | | | | |
| 指导教师意见 | 签 名：  年 月 日 | | | | |
| 备注 |  | | | | |

## 附录II

### 程序清单

登录界面

private void LoginBtn\_Click(object sender, EventArgs e)

{

//获取用户的信息

string id= "000000";

string pwd = "111111";

string ID = IDcmb.Text;

//判断用户名和密码是否为空,并提示

UserSkinlabel.Text = log.loginUserOrEmpty(id);

PasswordSkinlable.Text = log.loginPwdOrEmpty(pwd);

if (UserSkinlabel.Text == "" && PasswordSkinlable.Text == "")

{

if (ID == "学生")

{

//通过给定的学号返回学生登录对象

st = log.StudentGetBySno\_Password(id, pwd);

if (st == null)

{

MessageBox.Show("请检查用户名或密码");

return;

}

StudentForm stf = new StudentForm(st);

stf.Show();

}

else

{

if(ID == "教师")

{

//通过给定的工号返回教师登录对象

tea = log.TeacherGetByTid\_Password(id, pwd);

if (tea == null)

{

MessageBox.Show("请检查用户名或密码");

return;

}

TeacherForm tfm = new TeacherForm(tea);

tfm.Show();

}

else

{

if (ID == "管理员")

{

//通过给定的帐号返回管理员登录对象

admin = log.GetByAdminId\_Password(id, pwd);

if (admin == null)

{

MessageBox.Show("请检查用户名或密码");

return;

}

AdminForm afm = new AdminForm(admin);

afm.Show();

}

}

}

}

}

成绩录入

public partial class SelectCourseScoreForm : Form

{

public string studienjahr = null;//学年

public string term = null;//学期

public string classname = null;//班级名称

public string cno = null;//学号

//创建业务逻辑层对象

CourseBll couBll = new CourseBll();

ClassBll claBll = new ClassBll();

StudentBll stBll = new StudentBll();

TeacherSCBll teascBll = new TeacherSCBll();

SCBll scBll = new SCBll();

public Course course = null;//课程对象

public Class cla = null;//班级对象

public SelectCourseScoreForm()

{

InitializeComponent();

}

public SelectCourseScoreForm(string Studien, string Terms, string Classname, string Cno)

{

InitializeComponent();

this.studienjahr = Studien;

this.term = Terms;

this.classname = Classname;

this.cno = Cno;

this.course = couBll.GetCourseByCno(Cno);

this.cla = claBll.GetClassByClassname(classname);

}

//加载数据

public void load()

{

CnoLabel.Text = course.Cno;

CnameLabel.Text = course.Cname;

ClassNameLabel.Text = cla.ClassName;

//得到当前学年，学期，学生对象学习某门课的成绩

List<TeacherSC> teacherscs = teascBll.GetTeacherSCsByClass\_Course\_studienjahr\_Term(cla, course, studienjahr, term);

SelectCourseScore\_Tnamedgv.AutoGenerateColumns = false;//不自动创建列

SelectCourseScore\_Tnamedgv.DataSource = teacherscs;

}