摘 要

随着我国市场经济的建立和不断发展完善，高校财务管理的主体已经从单一国家财政拨款逐步转向多种渠道筹措资金为主，财务管理的职能也转向适应市场经济需要型的财务管理。为了能提供充足、准确、及时、综合的财务信息资源支持高校财务与决策活动，提高高校资金使用的效益与效果，利用现代信息技术对高校财务管理工作模式进行重组，将高校财务活动与以计算机软硬件系统、网络与通信为主的信息技术进行整合实现高校财务信息化，已成为高校财务管理的必然要求。而会计电算化是财务信息化的核心组成部分，担负着财务核算和管理功能，并为整个财务信息化系统提供主要的财务数据。因此，一个优秀的会计电算化系统可以大大提高财务人员的工作水平和效率、优化管理资源，从而实现规范、科学的管理。

本文在对电算化有关理论和开发技术认真学习研究的基础上，结合自身在高校会计电算化工作中的实际经验，对现行的高校财务处理业务进行了详细的分析，设计出一个可行的应用于高校的会计电算化系统总体设计方案。并详细分析了数据库结构，在功能模块方面，以 VS2008 为开发工具，结合 ActiveX Data Objects（ADO）数据库访问技术，实现系统管理 、系统初始化、帐务处理、辅助帐管理、报表管理等功能。同时，还对设计中的一些关键算法进行了研究。

关键词：会计电算化，设计与实现，VS2008，ADO.NET

ABSTRACT

With the establishment of China's market economy and continue to evolve, the main university financial management has gradually shifted from a single national financial allocation multiple channels to raise funds mainly, financial management functions are also turning to adapt to the market economy need-based financial management. In order to provide adequate, accurate, timely and comprehensive financial information resources to support university financial decision-making activities to enhance the effectiveness and effects of the use of university funds, the use of modern IT to restructure the university financial management model of university financial activities and computer hardware and software systems, network-based information and communication technology integration that the inevitable requirement of the university financial information, has become a university financial management. Computerized accounting is a core component of the financial information responsible for financial accounting and management functions, and provide key financial data for the entire financial information system.

On the computerization of the relevant theory and development of technology to seriously study the basis of practical experience in colleges and universities in the computerized accounting, carried out a detailed analysis of the current college financial processing business, design a viable computerized accounting system design program applied to the university. And a detailed analysis of the database structure, functional modules in VS2008 development tools, database access technology with ActiveX Data Objects (ADO), system management, system initialization, accounting treatment, the secondary account management, reporting and management functions .

Keywords: Computerized accounting, design and implementation， VS2008， ADO.NET

目　录

第一章 引言 1
1.1 课题的背景 1
1.2 我国会计电算化信息系统发展概况 1
1.3 本文的研究工作介绍 3
第二章 会计信息系统基本理论 5
2.1 会计信息系统基本知识 5
2.1.1 会计信息系统 5
2.1.2 会计信息系统的主要作用 5
2.1.3 会计信息系统功能 6
2.2 主流的会计信息系统 6
2.3 会计软件的发展趋势 7
2.3.1 集成化 7
2.3.2 面向服务 7
2.3.3 OLAP 8
2.4 高校会计电算化的特点 8
2.5 开发工具介绍 9
2.6 三层架构介绍 9
2.7 本章小结 10
第三章 会计电算化系统需求分析 11
3.1 会计核算 11
3.2 会计管理 13
3.3 权限管理 15
3.4 数据管理 16
3.5 日志管理 17
3.6 系统功能结构 18

3.7 其他需求 19
3.8 本章小结 19
第四章 会计电算化系统方案及设计 20
4.1 系统编码设计方案 20
4.2 系统架构 22
4.2.1 本系统中的数据访问层 22
4.2.2 本系统中的业务逻辑层 22
4.3 数据库设计 24
4.3.1 数据库及访问方式 24
4.3.2 实体关系 24
4.3.3 E-R 图 25
4.3.4 数据表结构设计 26
4.3.5 视图设计 30
4.4 设计中设计模式的应用 32
4.4.1 单例模式的应用 33
4.4.2 桥接模式的应用 34
4.5 使用反射技术的条件下系统的设计 35
4.5.1 数据访问的矛盾 35
4.5.2 反射技术的应用 36
4.6 会计电算化系统对校园一卡通的支持 36
4.6.1 校园一卡通对会计电算化系统的影响 37
4.6.2 基于中间件的会计电算化系统与校园一卡通系统衔接 38
4.7 本章小结 41
第五章 系统的实现 42
5.1 公共模块中配置文件读取功能的实现 42
5.2 数据访问层的实现 44
5.3 凭证管理模块的实现 46
5.3.1 凭证管理模块实现的前提 47
5.3.2 凭证管理模块实现 48
5.4 本章小结 53
第六章 高校会计电算化系统的实施与测试 54
6.1 系统的初始设置 54

6.1.1 管理员设置 54
6.1.2 会计期间设置 54
6.1.3 科目设置 55
6.1.4 部门项目 56
6.2 系统的使用示例 56
6.2.1 系统初始化 56
6.2.2 经费指标初始化 57
6.2.3 凭证管理 58
6.2.4 出纳复核 58
6.2.5 银行对账 59
6.3 实施结果 60
6.4 系统测试 60
6.4.1 功能测试示例 60
6.4.2 安全测试示例 61
6.5 本章小结 62
第七章 全文总结与展望 63
7.1 总结 63
7.2 展望 63
致 谢 64
参考文献 65

第一章 引言

1.1 课题的背景

在信息技术高速发展的今天，各种技术手段不断被应用于日常的业务活动，随着高等学校资金活动的日益频繁，高校的会计电算化系统已经成为高校管理资金的一个不可缺少的一环。自上世纪 80 年代开始，高校的会计电算化系统经历了一个从无到有，由简单到复杂的一个过程，并在此过程中日渐成熟起来。由于会计电算化系统掌握并控制着高校运作赖以存在的财务平台，因此具有举足轻重的作用。

目前市场上有许多通用的财务会计软件，这些软件由于客户群体的原因，往往功能过多，实用性不强，无法针对高校特殊的业务进行扩展与维护，不能满足高校财务处理上的需要。因此，开发一个针对高校业务，在功能、界面等诸多方面有针对性的财务处理软件是很有必要的。

通过该系统能学校的各项会计活动进行实时的处理和有效记录；使用者可以通过网络直接访问本系统，并根据需要处理与提取相关的数据；另外还可以随时获取历史的财务报告。会计电算化系统可以保证会计信息的准确、完整、真实和可靠，由于系统的内置处理过程和方法符合现有的法规，减少因人为因素而造成的损失。

1.2 我国会计电算化信息系统发展概况

随着信息技术的飞速发展，会计电算化在我国也经历了从空白到完善的过程。从发展趋势、社会应用及功能结构上看，我国的会计电算化主要经历了以下的过程。

从系统的发展趋势以及功能结构来看，会计电算化经历以下三个主要阶段：

1、单项业务电算化阶段。这一阶段从 1979 年开始到 1991 年，是会计电算化的起步阶段。这一阶段由于计算机只存在于少数的重点单位，因此主要利用计算

机处理计算量较大的任务。

2、会计核算电算化阶段。这一阶段从 1991 年到 1995 年，在此期间会计电算化实现了成本、固定资产、往来和销售核算，本质上发生了根本性的变化。另外对于会计报表编制等工作已经能够利用计算机进行处理，在这一阶段形成了较为完善的电算化会计核算系统。

3、会计管理电算化阶段。这一阶段从 1996 年开始至今。这一阶段的特点是在会计核算电算化的基础上，进行各种会计信息的集成，建立起会计电算化核算系统，此外还能为企业管理信息系统提供财务信息，对于会计工作的事前、事中、事后的三个阶段进行全面管理。这一阶段是会计电算化发展的最新阶段。

4、财务与业务一体化。随着信息技术的全面应用，在企业内部的人事、销售、采购、财务、生产等诸多领域均引入了计算机进行辅助决策、辅助管理，此时会计电算化不再是一个单纯的信息孤岛，需要与其他系统集成。目前的这一阶段正在起步。

从社会运用以及发展趋势分析，我国会计电算化可分为以下四个阶段：

第一阶段是研制试点阶段（1979 ——1983 年）：这一阶段的特点是各种软硬件依赖于进口，仅在小量重点单位使用，资金来源于财政，人才奇缺。只用于解决少量的专门性的问题。操作不便。

第二阶段是自发发展阶段（1983——1987 年）：这一阶段，信息技术已经有了一定的发展，硬件技术逐步成熟，微型计算机已经进入一些单位。这一阶段的软件功能单一，各单位之间又存在大量重复开发的现象。从现在的角度看这一阶段的会计电算化系统过于简单，但这一阶段为以后的会计财务系统的开发、推广及实施积累了大量经验。

第三阶段是有计划发展阶段（1987——1998 年）：这一阶段，政府相关部门出台了大量的政策法规，对会计电算化工作的开展进行了规范。对于许多企业而言，在实现会计核算电算化后，也逐步对应用领域进行了扩展。从软件开发行业来看，出现了一批专门从事会计电算化软件开发的企业，这标志着会计电算化发展进入了一个新的阶段。

第四阶段是竞争提高阶段（1999 年至今）：这一阶段的特点是会计软件公司的迅速成熟和发展养大。同时国外的一些优秀的软件如 Oracle、SAP 等也在国内大量应用。

1.3 本文的研究工作介绍

本文针对高校会计工作的需求和特点，进行高校会计电算化系统的设计、开发和实现。首先对高校会计的工作流程进行系统调查和分析，给出相应的软件逻辑模型，在此基础上进行研究设计，着重对系统的预算控制和内部控制功能的研究，最后上机调试并予以实现。考虑到高校会计信息系统安全性能要求较高，而C/S 模式结构网络不需要依赖外部，一般面对相对固定的用户群，网络系统稳定，可以对权限多层次校验,对信息安全的控制能力很强， 对系统运行速度可以较少考虑，网络管理规范化，比较适合高校会计电算化软件的开发与应用。 所以本软件拟采用 C/S 模式，其中客户端采用 VS2008 开发用户界面。服务器端采用 Sql_server2005 作为数据库。服务器端采用 Windows2000 作为操作系统，而客户端采用 WindowsXP/Windows7 为操作系统。

本文从七个部分对系统的开发与设计过程进行描述。

第一部分引言

主要介绍课题的背景与会计电算化的发展。

第二部分会计信息系统基本理论

主要介绍会计信息系统的基本知识，如会计系统的功能、作用及发展趋势，并对主流的会计系统进行介绍。

第三部分会计电算化系统方案及设计

主要对会计电算化系统进行规划和分析，首先对系统的需求进行分析，列出所需要实现的功能。接着进行数据库的设计，包括实体关系、E－R 图、表结构及视图设计，最后进行编码工具与开发方案的介绍。

第四部分为系统实现所用的关键技术

在本系统开发的整体结构上采用了三层架构的方式进行设计开发。为了提高模板的复用率及降低模板之间的耦合性，在设计时采用了单例模式与桥接模式分别对不同的模块进行了设计。在数据的访问上为了避免更换数据库及提高代码的开发效率，采用反射方式生成了数据库的访问模块。文中重点对外部程序与本系统的协同工作进行了研究，针对不同情况提出了采用 Web Service 方式和 COM+中间件的方式进行不同系统之间的代理。特别是在不同系统之间协作时出现的异常处理与事务处理，本文进行了研究与解决，这也是该研究的创新之处。

第五部分系统实现

针对第四部分的设计，进行了一些基础模块的设计，并给出关键的编码与相关说明。

第六部分为实施部分

在该部分列出系统运行过程的关键节点，并针对运行的界面进行适当的介绍。

第七部分全文总结与展望

主要对全文的研究工作进行总结，并对研究工作的今后方向做出展望。

第二章 会计信息系统基本理论

2.1 会计信息系统基本知识

2.1.1 会计信息系统

会计信息系统（Accounting Information System，简称 AIS）是一个组织处理会计业务，并为管理者和决策者提供信息的信息系统。它通过收集、加工、存储、传送和利用会计信息，对经济活动进行反映和控制。[1]

电算化会计信息系统简称会计信息系统（CAIS），其特点是采用计算机技术来解决会计活动中的问题。

电算化会计信息系统根据会计信息可以划分为以下三类：

1、电算化会计决策支持系统

2、电算化会计管理系统

3、电算化会计核算系统

目前我国的会计电算化系统主要以电算化会计核算系统应用为主。[2-4]

2.1.2 会计信息系统的主要作用

会计信息系统主要的作用有会计核算、会计管理、会计决策等：

（1）会计核算[5]

会计核算是会计信息系统的基本功能，是指应用会计核算软件来完成包括账户、会计科目的设置，会计凭证的填写，会计账簿、会计报表的生成等。除了管理货币资金外，还核算固定资产和库存等实物资产。

（2）会计管理

会计管理以资金运动为管理对象，其的任务是保证会计工作合法、合理、有效、平稳地进行。管理信息功能的模块包括预算管理、资金管理、项目管理和成本管理。

（3）会计决策

会计电算化系统中的决策功能是指应用计算机软硬件资源，在会计核算的基础上完成预决算、监督、控制及分析功能，这一阶段的功能是管理会计中的高级功能。

2.1.3 会计信息系统功能

为了实现会计信息系统的主要作用。会计信息系统中需要实现以下的功能：

（1）账务处理。

（2）会计报表。

（3）工资核算。

（4）固定资产核算。

（5）材料或库存产品核算。

（6）成本核算。

（7）产品销售。

2.2 主流的会计信息系统

目前国内主流的会计信息系统用友、金碟、易飞等，而国外知名的有如甲骨文等。

用友 U8 的一个出发点是以企业的财务管理为最核心内容，在此基础上以行为管理为主线。这种主次关系与其他软件的平行结构有所不同。这种模式可适应处于不同发展阶段的企业对于不同业务、不同目标及不同管理手段的需求。这些模式包括了产业型和投资型的企业集团模式、连锁经营模式、普通单一企业模式、具有分支机构的单一企业模式等。用友 U8 采用泰勒的管理思路，注重于各个环节的精确管理，这一点与中国的国情相符，而在整体的结构上变动较少，适合整体结构不会发生变化的企业。

金蝶 K/3 系统开发设计的核心指导思想是企业绩效管理（BPM），它使用各种 KPI 指标和平衡积分卡（BSC）方法进行任务的设立、监控和分解，通过自上而下的规范化管理，最终实现企业的战略目标。该系统对企业的决策支持、资金管理、预算管理、报表合并、财务集中等需求均较好的时行了解决。主要包括：报表、总账、应付款管理、应收款管理、工资管理、固定资产管理、现金流量表、财务分析、现金管理等。

易飞软件由神州数码与台湾最大的 ERP 管理软件商鼎新电脑合资成立的神州数码管理系统有限公司（DCMS）开发。其特点为操作界面友好，易于使用。该系统内嵌在线屏幕提示系统（on line monitor）与自动化引擎，可以进行线上事件的自动监控。远程客户可通过互联网方便地对系统进行远程数据的获取与操作。该系统可以在掌上电脑上使用。

2.3 会计软件的发展趋势

2.3.1 集成化

随着信息化建设的不断深入，目前会计软件呈现出集成化的趋势。企事业内部的各软件之间不再是各自毫无关联的信息孤岛，彼此之间的联系也越来越紧密。会计软件的集成化体现了管理一体化的思想。在软件功能上包括财务、固定资产、采购、销售理、存货、成本控制以及人事等诸多方面的管理。其衍生或关联的领域包括：供应链管理(SCM)、决策支持系统(DSS)、办公自动化系统(OA)、客户关系管理(CRM)、企业资源计划(ERP)、企业流程管理(BPM)、人力资源管理、知识管理等诸多领域。例如金蝶 EAS 采用了 ERPⅡ管理思想和一体化设计，有超过 50 个应用模块高度集成，涵盖企业内部资源管理、供应链管理、客户关系管理、知识管理、商业智能等，并能实现企业间的商务协作和电子商务的应用集成。

2.3.2 面向服务

面向服务的体系结构，也叫面向服务架构，简称 SOA（Service-Oriented Architecture）。SOA 是一种组件模型， 组成应用程序的各个组件需要按照一定的规则发布与调用功能。这种规则采用中立的方式，不依赖于具体的硬件与系统平台。[6]

目前的会计电算化软件不仅仅使用于企业内部，为了实现与上下游企业及如税务、工商、银行等相关单位的协作。需要一种开放式的体系结构与外部联通。面向服务的体系结构在会计电算化企业中的应用是种新的发展趋势。

2.3.3 OLAP

联机事务处理即 OLTP，On-line Transaction Processing。目前，大量的企事业单位采用关系型数据库来进行数据的存储与管理业务数据。并在此基础上开发大量的应用软件来支持日常业务的运行。这种以支持业务处理为主要目的的应用，被称为联系事务处理应用。

随着社会的发展，竞争越来越激烈，企业决策的及时性与准确性越来越重要，因此能支持决策管理分析的应用迅速发展，这类应用被称为联机分析处理（OLAP）。[7]

一些企业发展到一定程度，其活动范围也不再限定于某一区域、某一国家。而是在全球范围内配置资源。这些企业即为跨国公司。这些企业的信息系统产生了海量的数据，如何从这些海量数据中提取对企业决策分析有用的信息成为企业决策管理人员所面临的重要难题。对于处于信息核心的会计电算化软件应用 OLAP 为企业提供决策就成了一种趋势。当然数据挖掘、商业智能等相关技术与被应用于这一领域。

2.4 高校会计电算化的特点

高校会计电算化软件的有其自身的特点，高校财务管理与一般财务管理有较大的不同，具体表现如下：[8]

一是涉及面广。在高校的教学科研、日常业务、后勤生活等各方面均需要使用会计电算化软件。

二是综合性强。与企业的财务管理软件相比，高校的会计电算化软件需要在不同部门、不同活动中开展会计工作，资金来源类别的广泛也使得高校会计电 算化更具综合性。

三是敏感性高。高校的业务活动最终需要以经济效益与社会效益进行评价，因此数据上具有一定的敏感性，例如科研成果等级与效果的评定、学生培养的经费成本、社会服务收益的大小等在需要在会计电算化系统上进行体现。

2.5 开发工具介绍

1）VS2008

Visual Studio 2008 是面向 Windows Vista、Office 2007、Web 2.0 的下一代开发工具,代号“Orcas”，经历了大约 18 个月的开发，是对 Visual Studio 2005 一次及时、全面的升级。VS2008 引入了 250 多个新特性，整合了对象、关系型数据、XML 的访问方式,语言更加简洁。 使用 Visual Studio 2008 可以高效开发 Windows 应用。设计器中可以实时反映变更,XAML 中智能感知功能可以提高开发效率。同时 Visual Studio 2008 支持项目模板、调试器和部署程序。Visual Studio 2008 可以高效开发 Web 应用，集成了 ASP.NET AJAX 1.0，包含 ASP.NET AJAX 项目模板，它还可以高效开发 Office 应用和 Mobile 应用。

2）SQL Server 2005

SQL Server 2005 是一个全面的数据库平台，使用集成的商业智能 (BI) 工具提供了企业级的数据管理。SQL Server 2005 数据库引擎为关系型数据和结构化数据提供了更安全可靠的存储功能。可以用于构建和管理用于业务的高可用和高性能的数据应用程序。

SQL Server 2005 数据引擎是本企业数据管理解决方案的核心。此外 SQL Server 2005 结合了分析、报表、集成和通知功能。可以构建和部署经济有效的 BI 解决方案，通过记分卡、Dashboard、Web services 和移动设备将数据应用推向业务的各个领域。

2.6 三层架构介绍

为应对业务的变化对软件结构的影响，在设计时需要采用良好的结构。三层架构是目前通用的一种软件体系架构，三层架构由上而下可以分为 UI 层、业务逻辑层、数据访问层。

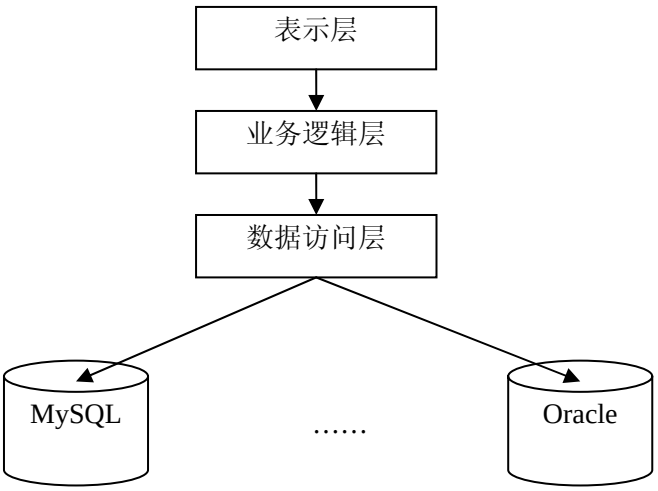


图 2-1 三层架构

2.7 本章小结

本章主要对会计电算化的概念、类型以及目前市场上的主流会计电算化系统进行了介绍。除了功能与应用的介绍外，还对其发展的新特点进行了描述。

第三章 会计电算化系统需求分析

3.1 会计核算

作为一个会计电算化系统，总先需要满足会计核算功能。会计核算常用内容如下：

1、账户、会计科目的设置

2、会计凭证的填写

3、会计账簿、会计报表的生成

4、核算固定资产和库存

其用例如下：

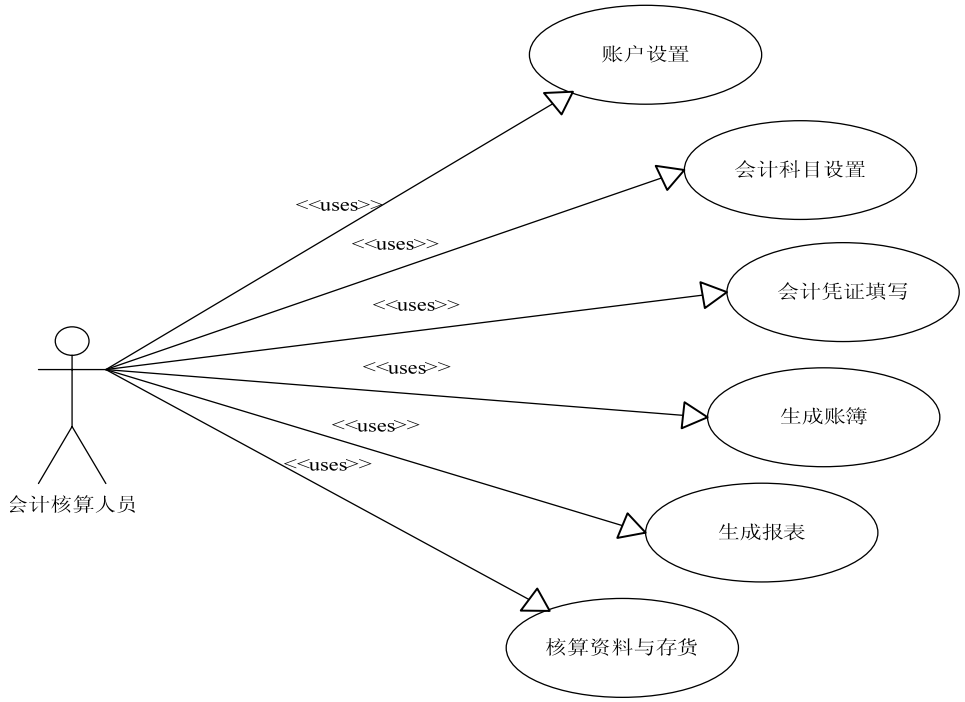


图 3-1 会计核算用例

在上述内容中，账户、会计科目的设置及会计凭证的填写属于基本信息，此处就不进行介绍，此处对报表生成与固定资产的情况进行描述。

报表功能需要完成报表的操作及相关的审核工作。生成的报表与具体格式

有一定的关联，在分析需求阶段可以不用考虑。而报表的审核与项目的设计有密切的关联，具体流程如下所示：

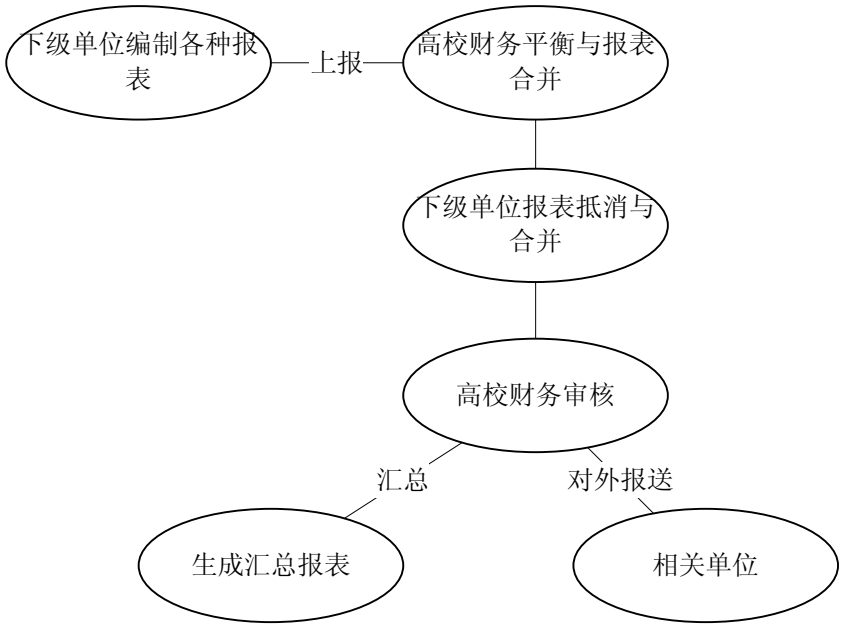


图 3-2 报表功能管理

目前一些高校已经建立起了报表的下发、上报体系。总部财务机关将需要统计的各类报表及核算关系下发到二级单位，二级单位填写完毕上报总部，由总部根据管理需要进行汇总及合并。总部财务机关首先汇总二级单位及结算中心的报表，抵销其中的内部收支。然后再合并二级单位的会计报表，并抵消其中的内部收支，向外报送整个学校的合并报表。具体报送流程因学校不同而有所差异。

固定资产的管理包括实物管理及财务处理两部分。总部机关及各二级单位都设有资产管理部门。实物管理由财务资产部下设资产管理处及各二级单位资产管理部门负责。财务处理工作由总部财务处及各二级单位财务处负责。

总部的实物管理工作有：

1、负责总部机关及所属二级单位购置固定资产的审批工作。

2、负责固定资产的登记工作

3、负责总部机关及所属二级单位固定资产的变动审批工作，包括：二级单位之间互相调拨、盘盈、盘亏、报废等。

4、负责固定资产的购置工作，

5、建设公司负责大型建设项目。

6、各单位的固定资产折旧存在上缴、下拨流程

各二级单位的财务资产管理部门负责本单位的固定资产的登记、定期盘点、内部调拨等工作。

对于财务工作：所有固定资产的购置及变动处理经资产管理部门审批后，在财务中进行处理。

因此，确定固定资产的业务流程如下：

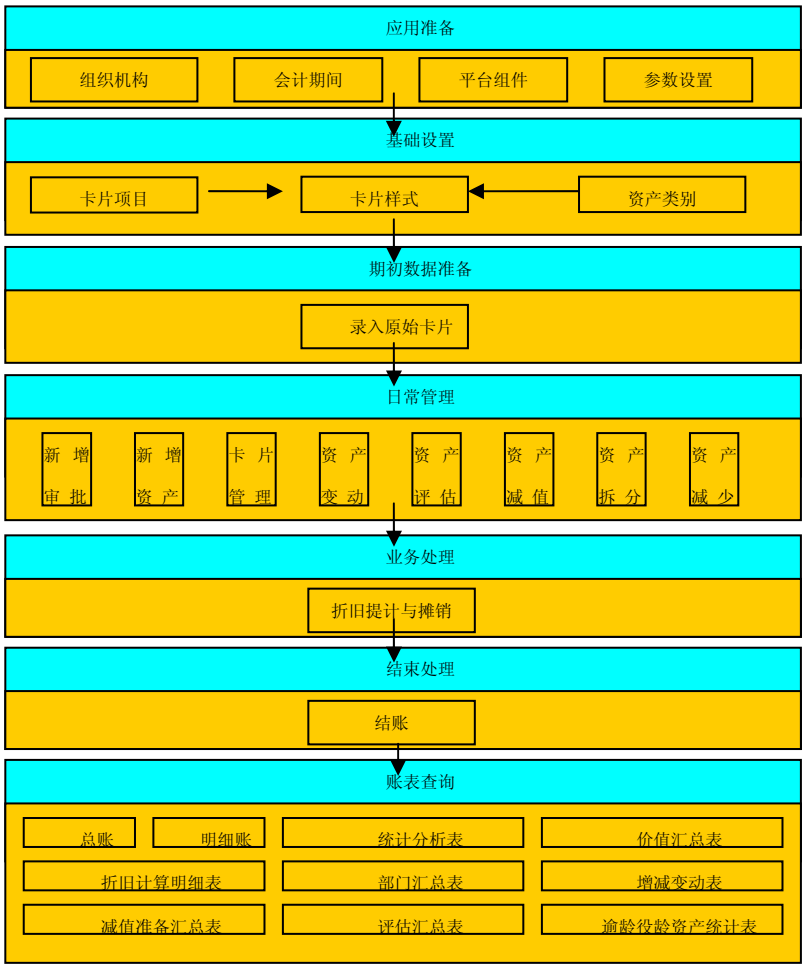


图 3-3 固定资产流程管理

3.2 会计管理

会计管理这一层次为企业提供会计信息化的更高层次管理功能，管理信息功能的模块包括成本管理、项目管理、资金管理和预算管理。

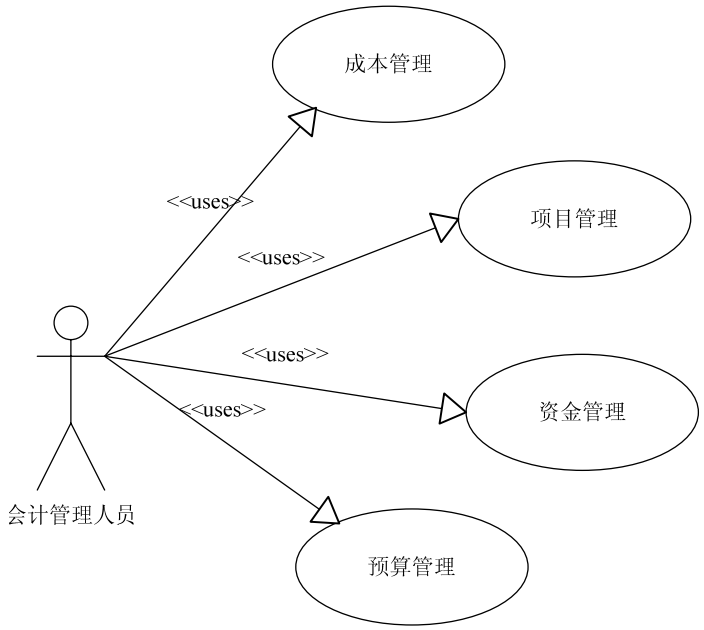


图 3-4 会计管理用例

在高校中成本管理、项目管理及资金管理的需求不如企业明显，而预算管理则较为重视。此处以全面预算为例进行会计预算需求的描述。

与高校不同的是，通常，企业在编制计划预算时，根据行业特色或者编制内容的不同，有如下几种常见的计划体系：通常，企业在编制计划预算时，根据行业特色或者编制内容的不同，有如下几种常见的计划体系：

销售起点型：销售起点型计划/预算模式适合于可以以销售情况调整生产能力的企业，比如家电、电子、服装等行业。以利润要求/销售额为企业全面预算的起点，分解生成其他相关计划预算。

生产起点型：生产起点型适合于以生产能力为企业发展的基础，生产能力不便于以外部情况作重大调整的企业。比如冶金、化工、电信等行业。要求以生产计划为全面计划/预算的起点，分解生成其他相关计划预算。

作为高校的运行体制与企业存在很大的区别，但若将学生与科研成果视为产品，则似乎与生产起点型企业有较大的共通之处。实际上学生培养与科研及其他活动视为具体项目，而不是产品。因此全面预算需求确认如下：

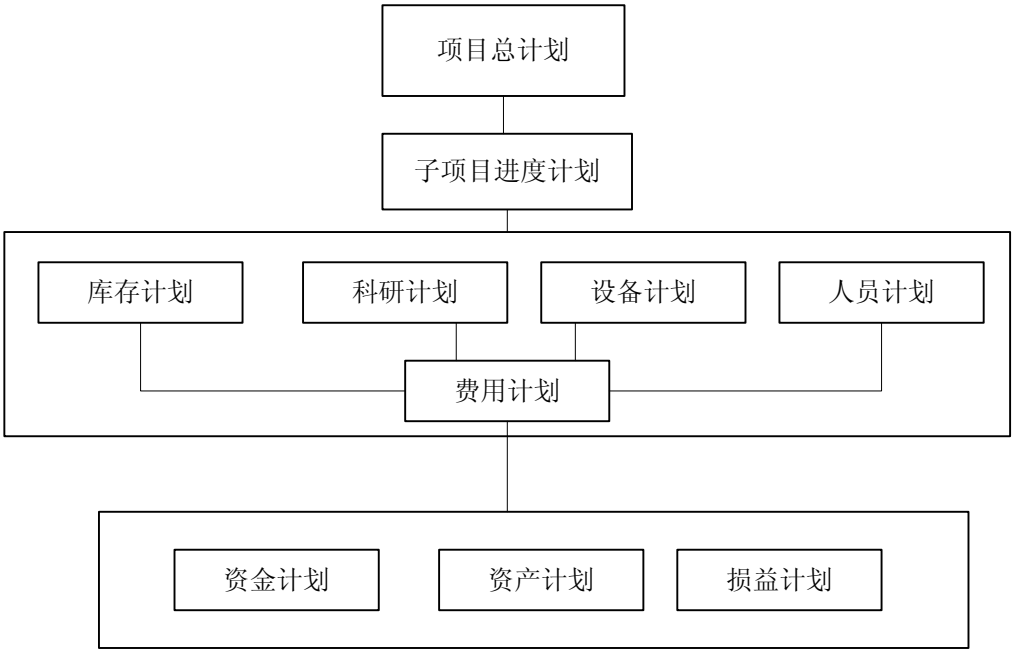


图 3-5 预算需求

上图为专项起点型预算的结构图，专项起点型适合于承接项目，比如建筑施工项目、大修理项目等。要求以项目总计划为起点，分解生成子项目进度计划，再根据子项目进度计划分解生成其他相关计划预算。

3.3 权限管理

为了保证系统及数据的安全与保密,本系统提供“操作员设置”功能,以便在计算机系统上进行操作分工及权限控制。系统管理员通过对系统操作的分工和权限的管理，一方面可避免与业务无关人员对系统的操作，另一方面可以对系统所含的各个子系统的操作进行协调，以保证系统的安全与保密性。

权限管理的操作有：权限分派、权限撤消。由于权限种类在系统初始就是已经确定，因此不需要进行更新、删除与修改工作。

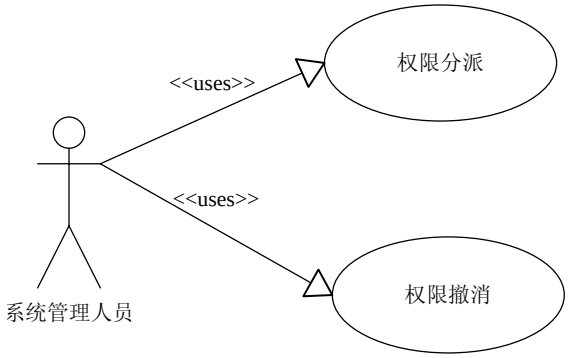


图 3-6 权限管理用例

3.4 数据管理

数据管理是系统管理员的重要工作内容，主要是对财务工作中一些常用数据进行管理。其操作有以下内容：

a) 数据库管理

对整个数据库进行恢复

b) 数据表管理

用于数据库中某些数据表的备份与恢复

c) 数据管理

“数据管理”维护后台数据的重要工具。它能够直接对后台数据进行修改、删除等操作。

d）历史数据备份

上一年年终冲转完成之后，总帐数据就会自动结转到下一年，因此上一年的数据就变为历史数据。将历史数据单独可以使当前库数据存贮量变小，会提高查询、存贮速度。

另外如果将历史数据从当前库中备份出来，对当前数据的操作不影响历史数据，避免误操作出现问题。

e）历史数据删除

历史数据备份完毕后，可以将历史数据从当前数据库中删除，以减轻当前数据库的负担，使软件查询速度更快。

数据管理用例如下：

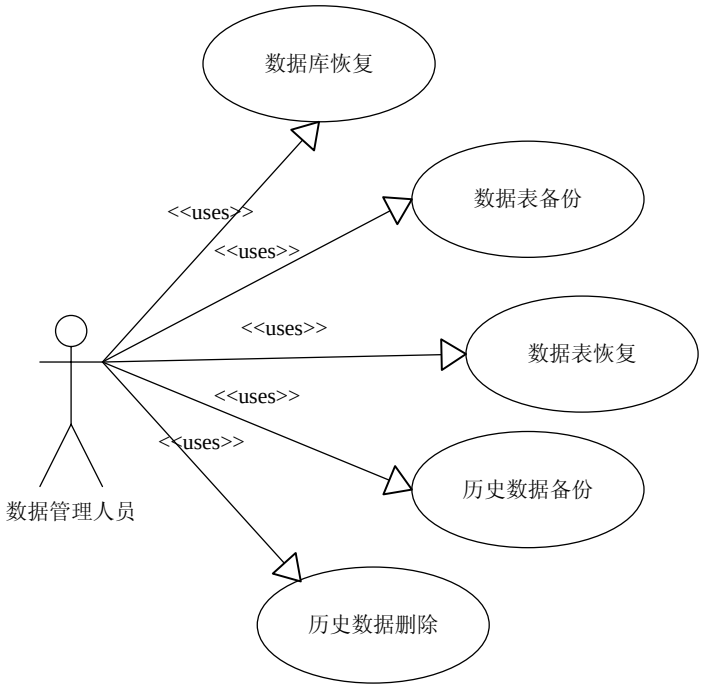


图 3-7 数据管理用例

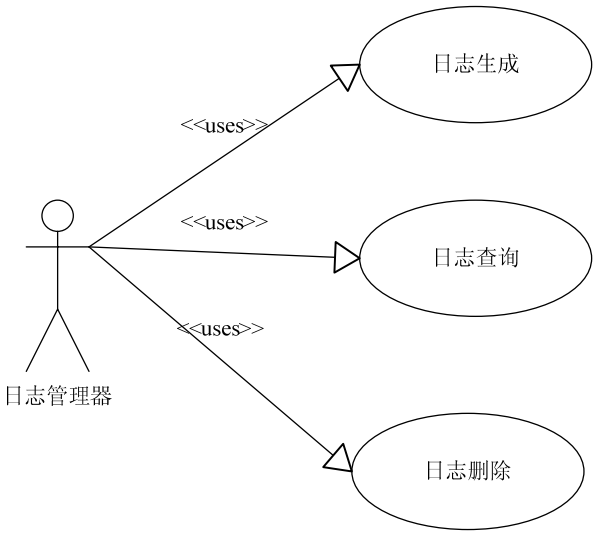


图 3-8 日志管理用例

3.5 日志管理

会计电算化软件的使用相当严格，为了对系统的使用进行更好的监督与管理，需要设置日志管理功能。日志管理需要记录的信息包括进入、离开系统的时间、

具体作的工作等等。日志管理如图 3-8 所示。

3.6 系统功能结构

根据会计电算化系统的需求，给出一级的系统功能结构图如 3-9 所示，具体的每个模块还可以根据业务具体细分。

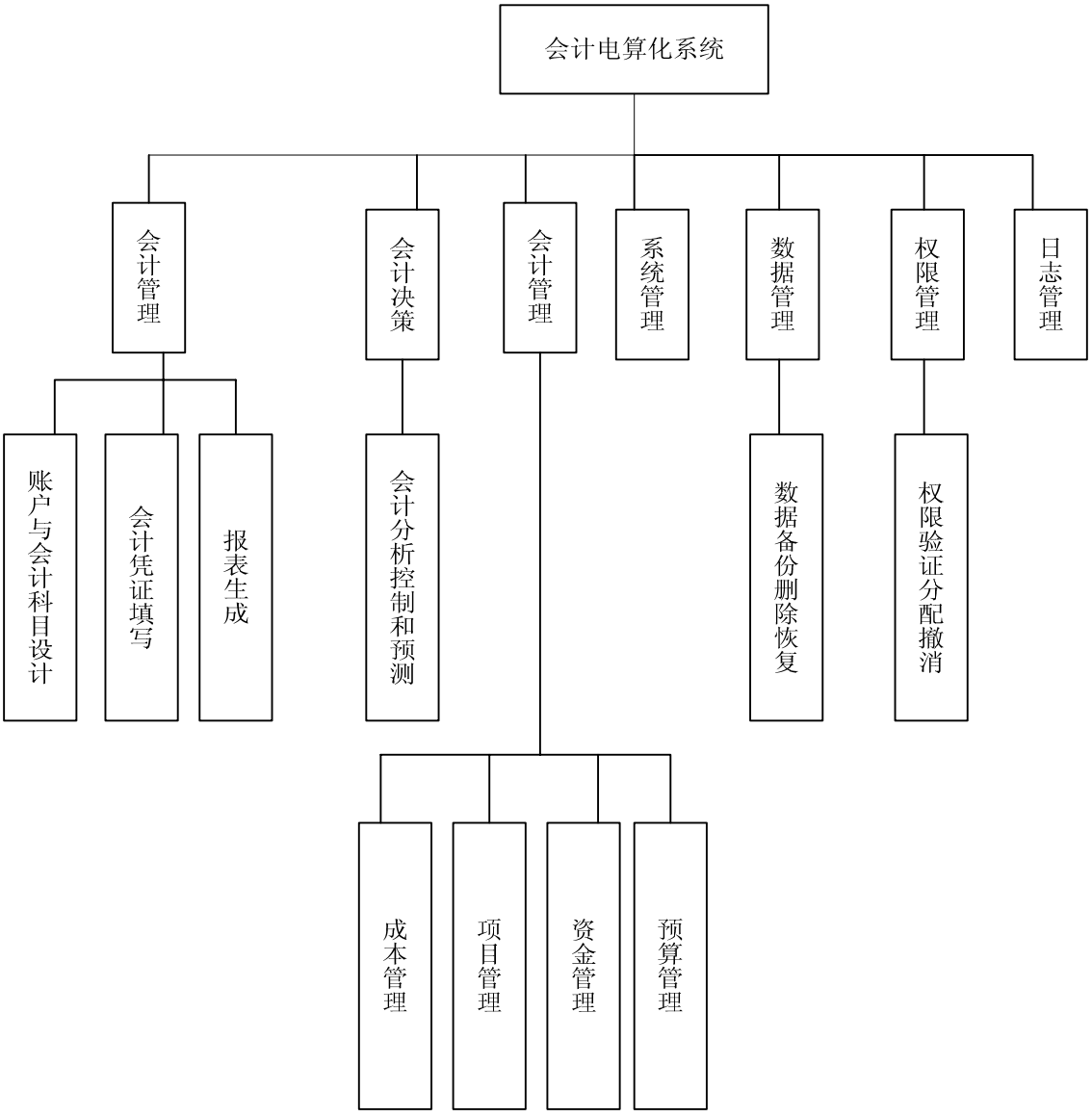


图 3-9 系统功能结构图

3.7 其他需求

除了功能上的需求外，客户还有性能上的需求及业务变化方面的需求，这一部份的内容较多，最为核心的有以下两个方面:

1、目前数据库采用的为 Sql Server，但客户有合并数据、迁移数据的需求，即今后可能需要对数据库进行变更。

2、会计电算化系统今后存在与其他系统协同合作的可能性，需要预先留出接口以便与其他系统交互。

从开发的角度，本系统还有以下隐含需求:

1、即代码的开发需要尽量有序，编码规则需要尽可能的简单。

2、软件的开发需要有较好的体系统结构，以应对业务需求的不断变化。

3.8 本章小结

本章重点对系统的需求进行了分析，对系统的会计核算、会计管理、权限管理、数据管理以及日志管理等功能进行了整体的介绍，以及对每个功能所需要实现的具体细节进行了扩展。

第四章 会计电算化系统方案及设计

4.1 系统编码设计方案

考虑到本系统各功能的不同应用场景及对灵活性安全性等方面因素的考虑，确定系统采用混合模式进行开发。

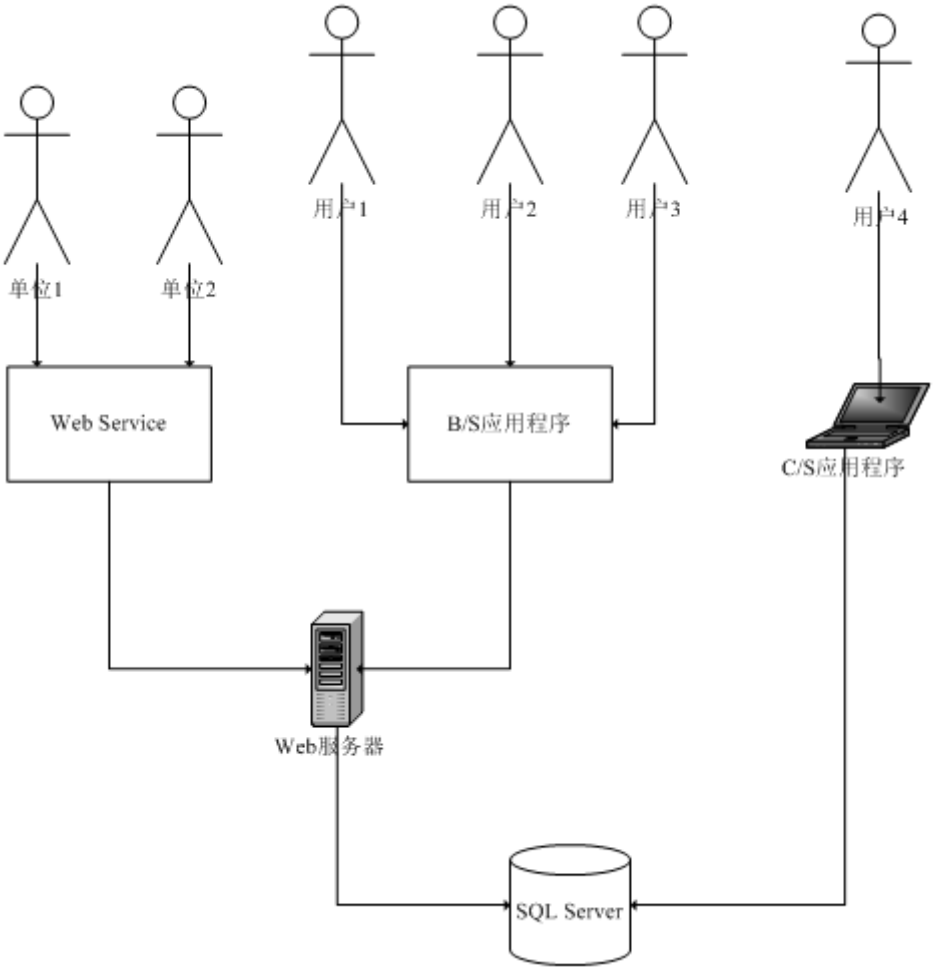


图 4-1 混合架构

对于数据安全及响应速度要求较高的功能，采用 C/S 方式进行设计。这些功能包括：账户与会计科目设计、会计凭证填写、成本管理、项目管理、资金管理、

预算管理及数据管理等绝大部分功能。但为了使 C/S 模式的应用程序能不依赖于其他部分运行，实际上所有的功能均在 C/S 中实现。

出于方便使用的角度考虑，在 C/S 已经提供相关功能的情况下，还需要为报表的生成、会计分析、预测、权限管理等功能可以设计为 B/S 模式为客户使用。

为了与外部接口进行交互，一些外围功能需要设计成 Web 服务的方式提供给外界访问。例如报表模块与数据接口部分。

此外，还有一些功能需要不依赖于具体的架构能独立运行。例如日志功能，这一部分功能需要写在数据库中以触发器的方式进行操作。

以上内容如图 4-1 所示。

对于这种混合模式的开发，在 VS2008 中使用 VSS 进行统一的源代码管理，并创建解决方案结构如下：

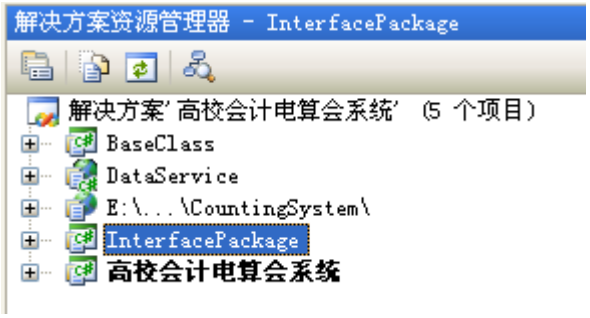


图 4-2 解决方案结构

在上述解决方案中 InterfacePackage 为所有数据基类的接口，在这一接口中由项目经理角色构建了一组的抽象类与接口。其他的项目均针对这一些接口进行编程。

而 BaseClass 类库项目则完成了对接口的实现工作。在这一类库中实现所有的业务逻辑与数据访问，该类库项目由其他子项目组成。

DataService 为 Web Service 在本项目中的实现，该模块为外部的调用提供了一个公有的接入方式。在该项目中通过 InterfacePackage 实现了对 BaseClass 的调用。

CountingSystem 为 Web 应用程序，而“高校会计电算化系统”则为 Windows 应用程序。与 DataService 一样，也通过 InterfacePackage 实现了对 BaseClass 的调用。

4.2 系统架构

4.2.1 本系统中的数据访问层

为了让系统有更好的体系结构以应对需求的变化，本系统中对三层结构的实现进行了扩展，数据访问层可以修改成以下结构：

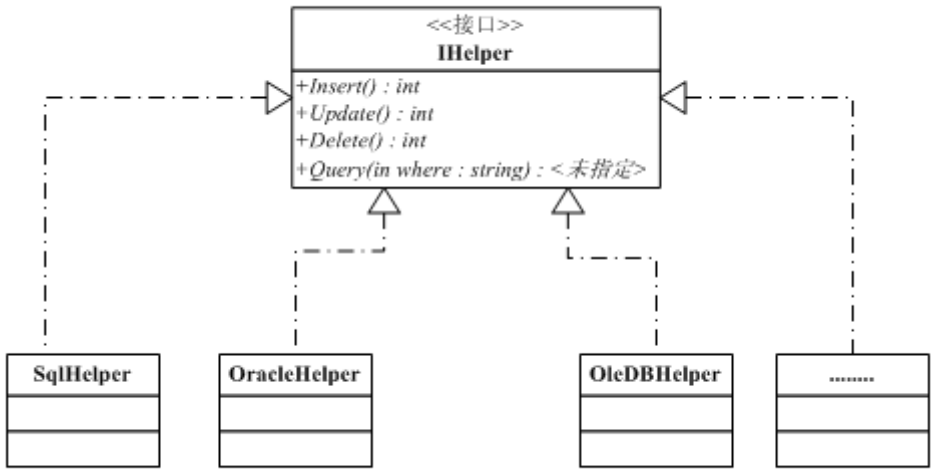


图 4-3 数据访问层

上述结构的数据访问层从接口 IHelper 派生出针对不同数据库如对 Sql Server、Oracle 等的访问类。这是为应对客户数据库服务器发生变化的情况。这种应对在后续代码中会有进一步的说明。

4.2.2 本系统中的业务逻辑层

中间的业务逻辑层设计成以下格式：

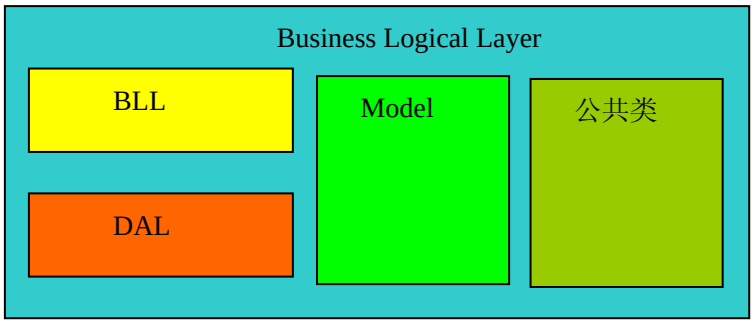


图 4-4 业务逻辑层设计

在开发中 Model 表示数据库体，其结构往往与数据库的表结构类似，有什么样的表名，就有什么样对应的 Model，表中的行、列、约束与数据库中的应用相当。如图 4-5，左侧为数据库表，而右边则为实体(Model，两者存在映射关系)

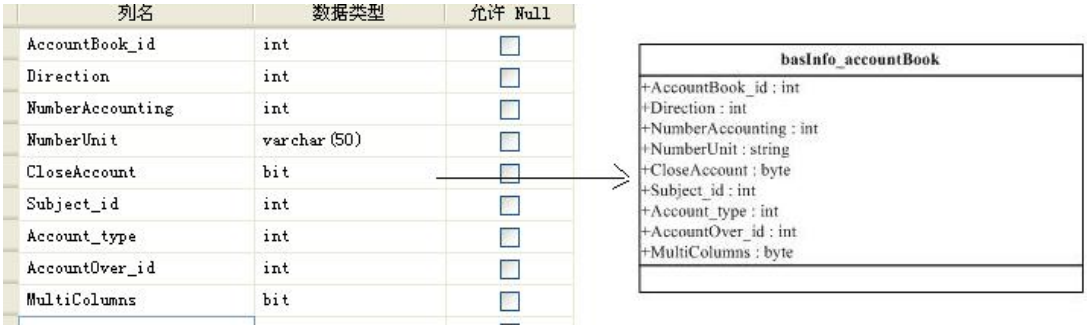


图 4-5 数据库表与类的映射

BLL 则真正应用于对业务逻辑的操作。

在数据库中有对应的 Insert、Update、Delete、Select 操作，出于面向对象的考虑，basInfo_accountBook 类实现了对数据的封装后，还需要实现对操作的封装。但在实际中，数据库的操作并非只针对单表进行。例如，在转账中就需要涉及到一个账户资金的增加及另一账户资金的资少，同时也要在日志文件中有相关记录。如果仅在 basInfo_accountBook 中封装相关方法，从代码的角度看系统的耦合性将会有很大的上升，不利于今后的维护。为解决这一目的，就需要将业务逻辑与数据实体分离。如图 4-6 所示。

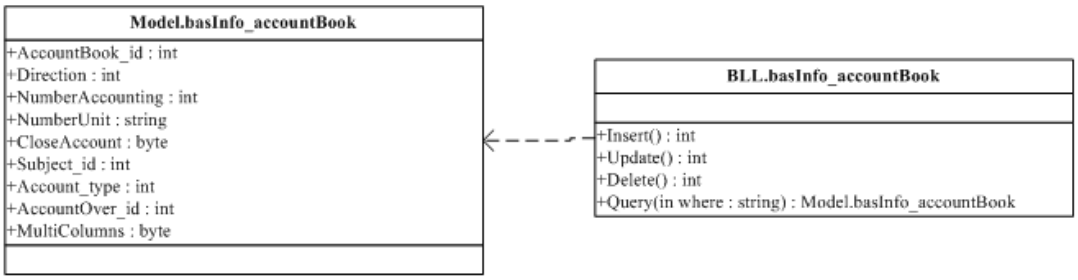


图 4-6 BLL 与 Model 调用

DAL 则根据 Model 与 BLL 中的数据实现对数据库的访问。

4.3 数据库设计

在明确需求的情况下需要着手进行数据库的设计。数据库设计的内容包括数据库及访问方式的选择、数据表设计及视图与存储过程等可编程性元素的设计。

4.3.1 数据库及访问方式

从规模上来说，高校的会计电算化系统在数据量上与银行、移动、电信等部门的记录相比并不大。而其数据量又远超过了一些简单的桌面应用软件。因此从数据量的角度考虑，该系统需要选择中等规模的数据库。

而会计电算化系统本身对安全要求较高，考虑到在软件集成化的背景，尽可能发挥数据的作用。因此在数据库的选择上采用微软的 SQL Server 2005 数据库。

为了发挥 SQL Server 2005 的功能，在数据访问接口的选择上采用 ADO.NET 进行数据操作。ADO.NET 是微软提供的一种数据库访问技术。通过 ADO.NET 技术可方便的访问各种类型的数据库，尤其对于 SQL Server 类型的数据库，ADO.NET 更是有着速度、安全、性能上的优势。

4.3.2 实体关系

在本系统中存在以下的实体关系：

1、科目信息(科目代码、科目名称、助忆码、科目类别、科目账户类型、余额方向、数量核算、数量单位、结算)

2、账户类型信息(属性、账户名称、账户名称值

3、科目类别（属性、科目类别、科目类别值）

4、帐套信息（会计期间、会计科目、记账凭证、账簿、帐套启用日期、帐套名称、帐套当前期间）

5、会计期间信息（属性、年度、期间、起始日期、结束日期）

6、用户信息（姓名、工号、口令、职位、权限、联系电话、Email）

7、权限信息（权限节点名称、父权限、权限代码）

8、凭证类别（编码、名称、借方必有、贷方必有、借或贷必有、借方必无、贷方必无、借或贷必无）

9、凭证信息（凭证类别、凭证号、附件张数、单位、凭证日期、摘要、科目、借方金额、贷方金额、合计、主管、复核、记账、制证、出纳、结算方式、结算号、数量、单价）

10、结算方式信息（结算方式）

11、账簿基本信息(账户、账户类型、分录日期、摘要、凭证、本期合计、本年累计、上年结转)

12、三栏式账簿信息（借或贷、余额、借方、贷方、对方科目）

13、多栏式账簿信息（借方项目、贷方项目、合计、余额、借或贷、对方科目）

14、数量三栏式账簿信息（借方数量、借方单价、借方金额、贷方数量、贷方数量、贷方单价、贷方金额、余额、借或贷、对方科目）

15、现金盘点信息（科目、、实存金额、盘点人、备注）

16、实物盘点信息（科目、计量单位、存放数量、单价、金额、盘点人、备注）

17、日记账信息（账户、摘要、凭证、本期合计、本年累计、收入、发出、结存、对方科目、账簿创建日期、分录日期、结算方式、结算号、借或贷）

18、银行对账单（日期、结算方式、结算号、借方、贷方、勾兑）

19、企业银行帐（日期、结算方式、结算号、借方、贷方、勾兑、摘要）

4.3.3 E-R 图

在实体的基础上，可构建 E-R 图，以下给出了系统中部分的 E-R 图。

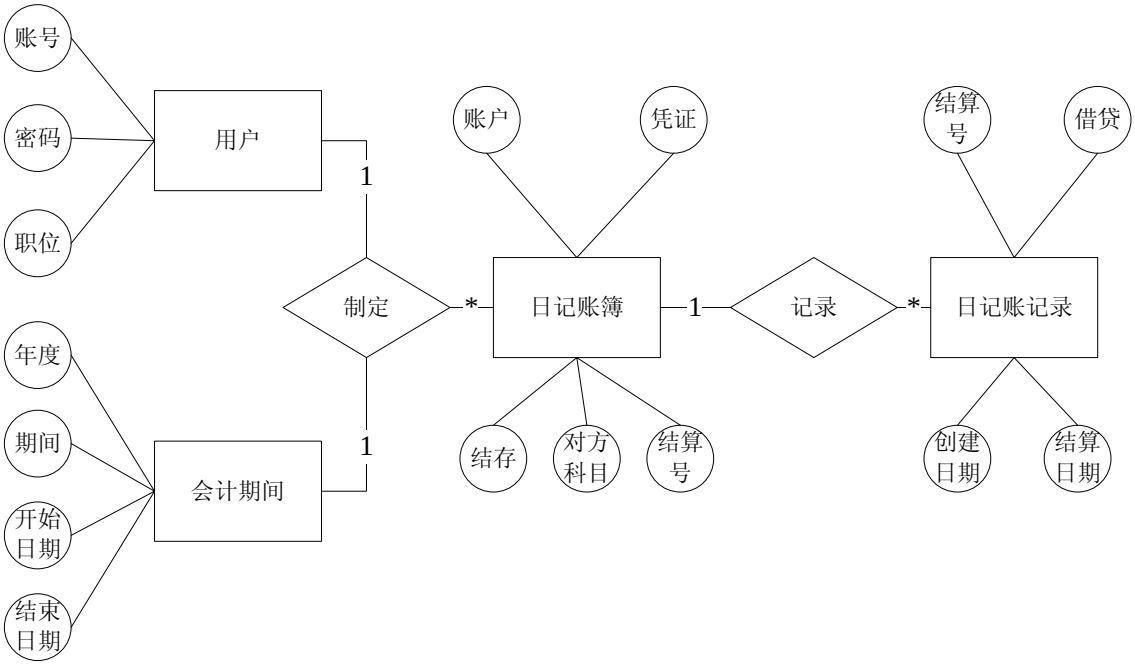


图 4-7 E-R 图(a)

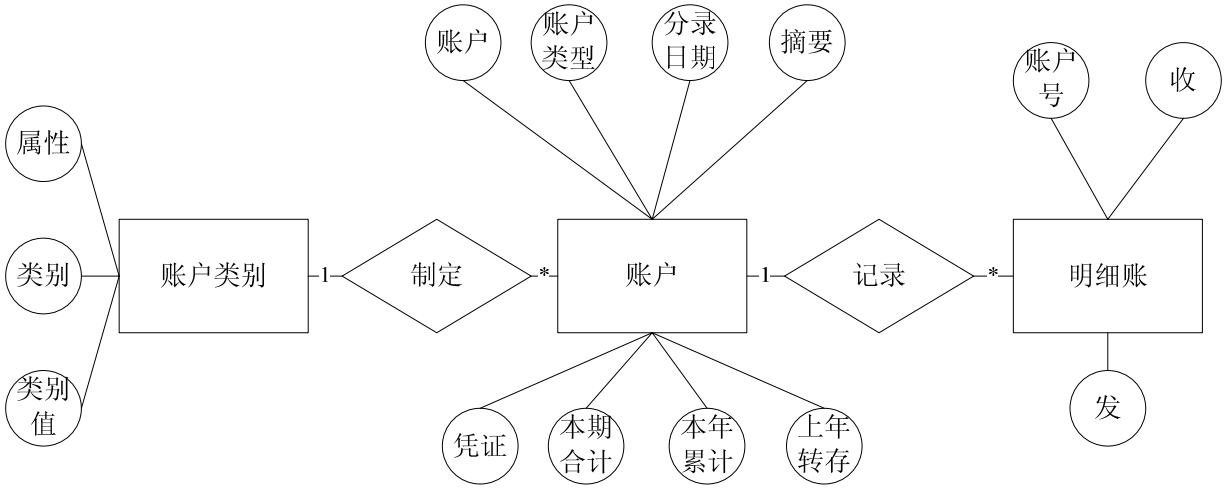


图 4-8 E-R 图(b)

4.3.4 数据表结构设计

在本系统中，数据表与实体一一对应，但在具体实现的过程中需要在自动增加列、数据类型、初值及主外键上进一步的处理。由于数据库表的数量较多，以下就列举少量数据表进行示例：

表 4-1 (帐套表)basInfo_accountOver

代码	数据类型	初值	主键	外键	注释
Account_id	Int	Identity(1,1)	Y	N	自动增长
CreateTime	Datetime		N	N	启用日期
Name	Varchar(255)		N	N	帐套名称
CurrPeriod_id	Int		N	Y	当前会计期间，引自会计期间表

表 4-2 (会计期间表)basInfo_period

代码	数据类型	初值	主键	外键	注释
Period_id	Int	Identity(1,1)	Y	N	自动增长
Year	Int		N	N	2009 等年度数字
Period	Int		N	N	1~12 期间
BeginTime	Datetime		N	N	当前期间开始日期
EndTime	Datetime		N	N	当前期间结束日期
Account Over_id	Int		N	Y	引自帐套表，期间对应的帐套

表 4-3 (科目表)basInfo_subject

代码	数据类型	初值	主键	外键	注释
Subject_id	Int	Identity(1,1)	Y	N	自动增长
Parent	Int		N	N	引自自身父节点
Code	Varchar(50)		N	N	科目代码
Name	Varchar(200)		N	N	科目名称
Mnemocode	Varchar(200)		N	N	助记码
Subject_type	Int		N	Y	引自科目类别表
Account Over_id	Int		N	Y	引自帐套表，科目对应的帐套

表 4-4　(科目类别表)basInfo_subjectType

代码	数据类型	初值	主键	外键	注释
Type_id	Int	Identity(1,1)	Y	N	自动增长
Name	Varchar(200)		N	N	资产、负债等类别名称
Value	Varchar(200)		N	N	类别值，对应类别名称
Account Over_id	Int		N	Y	引自帐套表，科目类别对应的帐套

表 4-5　(账簿表)basInfo_accountBook

代码	数据类型	初值	主键	外键	注释
AccountBook _id	Int	Identity(1,1)	Y	N	自动增长
Direction	Int		N	N	余额方向 1.借　2.贷
Number Accounting	Bit		N	N	是否数量核算　0.否 1.是
NumberUnit	Varchar(20)		N	N	数量单位
CloseAccount	Bit		N	N	结算 0.否 1.是
Subject_id	Int		N	Y	账户科目，引自科目表(唯一约束)
Account_type	Int		N	Y	引自账户类别表
Account Over_id	Int		N	Y	引自帐套表，科目对应的帐套
multiColumns	Bit		N	N	是否多栏账 0.是 1.否

表 4-6　(账簿类别表)basInfo_accountBookType

代码	数据类型	初值	主键	外键	注释
Type_id	Int	Identity(1,1)	Y	N	自动增长
Name	Varchar(200)		N	N	三栏式、多栏式等类别名称
Value	Varchar(200)		N	N	类别值，对应类别名称
Account Over_id	Int		N	Y	引自帐套表，科目对应的帐套

表 4-7 (用户表)basInfo_user

代码	数据类型	初值	主键	外键	注释
User_id	Int	Identity(1,1)	Y	N	自动增长
Name	Varchar(255)		N	N	用户名
Employee Number	Varchar(50)		N	N	001、002
Position	Varchar(255)		N	N	用户职位
Phone	Varhcar(200)		N	N	联系电话
Email	Varchar(200)		N	N	电子邮件
Account Over_id	Int		N	Y	引自帐套表，对应的帐套

表 4-8 (权限表)basInfo_right

代码	数据类型	初值	主键	外键	注释
Right_id	Int	Identity(1,1)	Y	N	自动增长
Name	Varchar(200)		N	N	权限名称
Code	Varchar(200)		N	N	权限代码
Parent	Int		N	N	自引用，引自自身父权限 ID

表 4-9 (用户权限关联表)basInfo_userRightRelation

代码	数据类型	初值	主键	外键	注释
Urr_id	Int	Identity(1,1)	Y	N	自动增长
User_id	Int		N	Y	引自用户表
Right_id	Int		N	Y	引自权限表

表 4-10 (凭证类别表)basInfo_voucherType

代码	数据类型	初值	主键	外键	注释
Type_id	Int	Identity(1,1)	Y	N	自动增长
Name	Varchar(200)		N	N	类别名称
DebitHave	Varchar(255)		N	N	借方必有科目

LenderHave	Varchar(255)		N	N	贷方必有科目
DebitOr LenderHave	Varchar(255)		N	N	借或贷必有科目
DebitNotHave	Varchar(255)		N	N	借方必无的科目
LenderNotHave	Varchar(255)		N	N	贷方必无的科目
DebitOrLenderNotHave	Varchar(255)		N	N	借或贷必无的科目
AccountOver_id	Int		N	Y	引自帐套表，对应的帐套

表 4-11　(凭证表)vouMng_voucher

代码	数据类型	初值	主键	外键	注释
Voucher_id	Int	Identity(1,1)	Y	N	自动增长
Type	Int		N	Y	引自凭证类别表
Number	Int		N	N	凭证号
Enclosure	Int		N	N	附件张数
CreateTime	dateTime		N	N	填制日期
Director	Int		N	Y	主管，引自用户表
Check	Int		N	Y	复核人，引自用户表
Keep Account	Int		N	Y	记账人，引自用户表
Create	Int		N	Y	制证人，引用户表
Cashier	Int		N	Y	出纳，引用户表
Account Over_id	Int		N	Y	引自帐套表，对应的帐套
CurrPeriod_id	Int		N	Y	当前会计期间，引自会计期间表

4.3.5 视图设计

数据表的设计考虑到了数据冗余等原因，其设计相对简洁，而访问相对繁琐。但出于业务逻辑的考虑，有时需要简化操作而可以忽视其存储方面的开销。在这

情况下可以使用数据库视图进行设计。

视图的设计可以采用可视化工具进行设计，也可以使用以下的 SQL 脚本进行操作：

```
CREATE VIEW ViewName
As
SELECT (COLUMN1,COLUMN2…..) FROM TABLE1,TABLE2…..
    WHERE condition1 AND condition2 AND…..
```

在本系统中为了简单业务的执行，先设计了大量的视图。以下仅对各种视图及其来源进行简单描述。

1)(科目账户视图) Vw_subjectAndAccountBook

由(科目表)basInfo_subject 、(科目类别表)basInfo_subjectType 、(账簿表)basInfo_accountBook 、(账簿类别表)basInfo_accountBookType 合并而成

2)(科目视图)Vw_subjects

由(科目表)basInfo_subject 、(科目类别表)basInfo_subjectType 合并而成

3)(帐套期间视图)Vw_accountBookAndPeriod

由(帐套表)basInfo_accountOver、(会计期间表)basInfo_period 合并而成

4)(用户权限视图)Vw_userAndRights

由(用户表)basInfo_user 、 (权限表)basInfo_right 、(用户权限关联表)basInfo_userRightRelation 合并而成

5)(凭证信息视图)Vw_voucherInfo

由(凭证表)vouMng_voucher 、(凭证类别表)basInfo_voucherType 、(用户表)basInfo_user、(帐套表)basInfo_accountOver 合并而成

6)(凭证分录视图)Vw_voucherEntry

由(分录表)vouMng_entry 、(科目表)basInfo_subject 、(结算方式表)vouMng_closeAccount 合并而成

7)(账簿信息视图) Vw_accountBookInfo

由(科目表)basInfo_subject 、(账簿表)basInfo_accountBook 、(账簿类别表)basInfo_accountBookType、(会计期间表)basInfo_period 合并而成

8)(账簿记录视图)Vw_accountBookAndEntry

由(分录表)vouMng_entry、(凭证表)vouMng_voucher 合并而成

9)(实物盘点视图)Vw_checkArticle

由(实物盘点) chkProp_checkArticle、(科目表)basInfo_subject、(用户表)basInfo_user 合并而成

10)(现金盘点视图)Vw_checkCash

由(现金盘点)chkProp_checkCash、(科目表)basInfo_subject、(用户表)basInfo_user 合并而成

11)(日记账簿信息视图)Vw_dayBookInfo

由(会计期间表)basInfo_period、(用户表)basInfo_user、(日记账)cashierMng_dayBook 合并而成

12)(日记账记录视图)Vw_dayBookdRecord

由(日记账记录) cashierMng_record、(凭证表)vouMng_voucher 合并而成

13)(企业银行对账视图)Vw_ enterBankAccount

由(企业银行帐)cashierMng_enterBankAccount 、(结算方式表)vouMng_closeAccount 合并而成

14)(银行对账视图)Vw_bankStatement

由(银行对账单)cashierMng_bankStatement、(结算方式表)vouMng_closeAccount 合并而成

4.4 设计中设计模式的应用

设计模式[9]（Design pattern）是一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结。使用设计模式是为了可重用代码、让代码更容易被他人理解、保证代码可靠性。 毫无疑问，设计模式于己于他人于系统都是多赢的，设计模式使代码编制真正工程化，设计模式是软件工程的基石，如同大厦的一块块砖石一样。

在本系统开发设计过程中也大量应用设计模式，以下以单例模式与桥接模式为例进行研究。

4.4.1 单例模式的应用

在本系统中需要大量地对数据库的访问。通过 ADO.NET 能很方便地实现数据库的操作。ADO.NET 中对数据库的操作需要有正确的连接字符串 ConnectionString。如果 ConnectionString 硬编码于代码中，则系统存在很大的耦合性，今后对数据库服务器的变更均需要修改源代码。

为了降低这种耦合性，可以考虑将连接字符串放置于外部文件中。若需要变更数据库服务器，则只需要修改外部文件中的内容即可，无需对代码进行重编译。

上述的方法实现了系统的解耦，但又带来一个性能的问题。对于文件的访问相对耗时，在压力面前将带来极大的性能延迟，影响用户体验。

可以在系统第一次被访问时进行连接字符串的读取。然后将其缓存于内存中，后续的访问将不再对文件进行访问，而是直接从内存中返回该值，从而提升了系统的效率。

为实现这一目的，在 COM 中常用设置计数器的方式来判断对象是否创建。而在面向对象的操作中可以使用单例模式来实现对文件的唯一读取。

单例模式[10]，保证一个类仅有一个实例，并提供一个访问它的全局访问点。其类图如下：

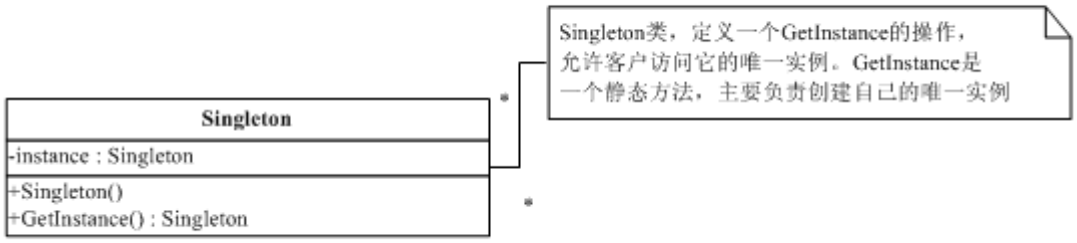


图 4-9 单例模式

在标准单例模式中，需要将构造函数设置成私有，而提供实例化对象的方式 GetInstance 方法来保证该对象的创建只能通过该函数进行，从而在 GetInstance 方法中实现对构造函数调用的限制来保证最多只创建一个对象。这种操作适合于对象较为复杂，创建过程相对耗时的情况。

在本例中，由于获取的是获取的是配置文件上的字符串对象，其构建过程也相对简单，因此可以对单例模式进行适当的改动。即不需要提供私有构造函数，只通过一个获取连接字符串的方法来返回相应的字符串即可。

以上的操作在单线程情况下不会出错，但在多线程的情况下，仍然会发生多次读取文件的现象，这了解决这一问题，需要对读取文件的代码进行同步，保证对于文件的访问同一时刻只能有一个线程进行文件访问。即：

```
//线程同步
同步代码区
{
 如果 connStr 为空值
        则读取连接字符串并赋给 connStr;
}
```

在该例中，具体的实现请见 5.1

4.4.2 桥接模式的应用

日志模板需要完成各种日志信息的记录，以便后续进行审查及判断问题的根源。在本系统中日志有不同的分类，例如按格式分类有操作日志、借贷日志、异常日志;按照事件发生的来源，有本地日志、服务器日志等。同一类型的日志在处理方法上有相同的操作。

考虑到日志操作的不同分类及今后的扩展，将发现不同的日志组合可能会使业务复杂化。如下表所示，在系统中将要定义 6 个类。

表 4-12 日志类表

	操作	借贷	异常
本地	本地操作	本地借贷	本地异常
服务器	服务器操作	服务器借贷	服务器异常

假设 mi 为某一种分类标准，在该标准下有 i 个子类，则总类别为各子类数量的乘积。如上图所示，子类数量为 2*3=6 个。当分类标准越多，且子类数量较多时，最后总类的数量将会多到一个难于维护的级别。

为了解决这一问题，可以使用桥接模式减少类的数量。

桥接模式[11-12]指在一个软件系统的抽象化和实现化之间使用组合/ 聚合关系而不是继承关系,从而使两者可以相对独立地变化。其 UML 图如图 4-6 所示：

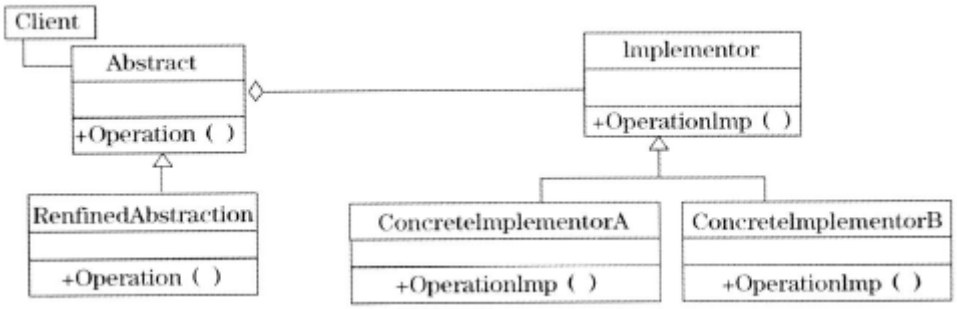


图 4-10 桥接模式结构图

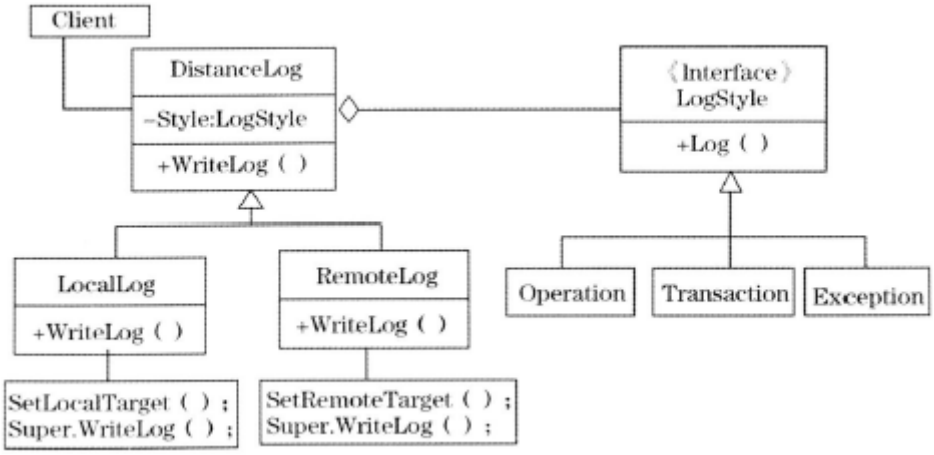


图 4-11 日志模块的桥接模式实现

在本系统中，可以根据分类标准提取接口，并最终实现对不同类别的组合。

具体 UML 图实现如图 4-11 所示。

4.5 使用反射技术的条件下系统的设计

4.5.1 数据访问的矛盾

3.4 中要求尽量简单代码的开发。该系统的大部分工作为数据访问操作。在数据访问中存储过程的使用可以提高地的效率，但由于代码不易于维护，在业务变化时难于维护。SQL 语句可以写在代码中，然而这种硬编码方式仍然难以应付需求的变化。因此，.NET 的数据访问与需求变化构成了一个矛盾。如何解决这一矛盾成了系统开发过程中的一个重要内容。

参考文献[13]提出了一种使用 T-SQL 来生成 T-SQL 的 SQL 动态生成技术。该文献从业务的角度出发列举了一个修改雇员字段的例子，然而现实中该字段分布于多个表中。如果逐一修改，不仅费时费力，也存在遗漏。文献中提出的解决方法基于 SQL 的对象。

在本系统中，情况又有所不同，客户存在变更数据库服务器的可能。当变更需求确立后，如是新数据库非 SQL 系统，上述的操作将不能复用。这势必给后期的维护带来很大的困难，为应对这一困难，可以使用.NET 的反射技术进行解决。

4.5.2 反射技术的应用

在参考文献[14]中对反射进行了定义，指出：反射是在运行时检查程序集清单中的元数据的功能。在.NET 中，EXE 与 DLL 文件都称为程序集。程序集的程序集清单是提供程序集和程序集中所有类型（包括泛型类型）的相关信息。使用元数据，可以运行时加载程序集并创建和调用在程序集中定义的类型实例。通俗的描述，程序集清单就是装箱单，通过装箱单，外界可以清楚的知道箱中有什么东西。

在.NET 中可以使用 System.Reflaction 命名空间下的 Assembly 类及 Type 类来实现反射。

为实现对数据库的访问，需要生成相关的 SQL 语句，编写 SQL 语句是繁重的工作，而写出的 SQL 语句也因数据库的变更需要进行适当的变化。因此 SQL 语句

的自动生成在开发时可以节省大量的成本。复杂的 SQL 语句难以自动生成，但复杂的 SQL 语句可以转化为简单的 SQL 语句，而将相应的逻辑在 BLL 中实现。具体的实现请见 5.3

4.6 会计电算化系统对校园一卡通的支持

3.3 中提到会计电算化系统要与其他系统协作的需求，校园一卡通系统就是其中一个。所谓校园一卡通系统是指基于校园网，采用成熟、先进的非接触式 IC 卡实现数据采集而建成的校园个人数据管理应用平台。它集收费管理、证件管理、教务管理、师生考勤、食堂管理、机房管理等多种功能将学校各个系统连成一体，动态掌握每一持卡人情况，不但满足了学校不同管理层次的需要，而且解决了各种规格管理卡的兼容问题，为校园师生的日常生活和学习带来极大方便和快捷，同时也为校园电子化建设[23]。

4.6.1 校园一卡通对会计电算化系统的影响

校园一卡通系统基于学校的局域网，以非接触式的 IC 卡或 RFID 标签卡为载体应用于教学生活的各种活动中。这些活动的数据以信息化的方式进行管理，为高校规范收费管理、信息化建设提供了必要的条件，这也为高校的会计电算化管理带来了新的变化。

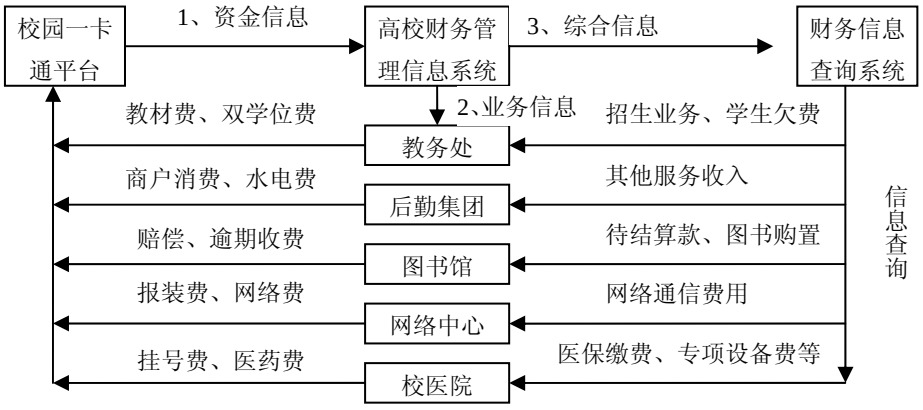


图 4-12 一卡通系统与高校财务系统

在上图中，说明了高校财务系统在三个方面的应用：1、高校的收费模式。该

模式应用一卡能的电子支持功能，减少现金的使用量，并能将所有活动的信息进行统一分析处理。2、高校的财务信息管理模式，该部分与校园中的其他系统如MIS，OA 等进行异构系统的衔接，完成信息的共享与利用。3、高校的财务信息查询模式。指利用校园卡为一种可靠的身份识别方式建立校园网内高效的财务信息管理系统。

校园一卡通系统对实现高校财务的现代化、提升学校的决策水平、降低管理成本、提高工作效率起到了很重要的作用。

4.6.2 基于中间件的会计电算化系统与校园一卡通系统衔接

从结构上看，校园一卡通系统与会计电算化系统是两个不同的异构项目。其所实现的平台、体系结构、设备等可能存在着较大的差异。在完成的时间上也不可能同时完成开发与部署，而是必定先有会计电算化系统，再有校园一卡通工程。

在 4.4 中的研究中，已经在会计电算化系统中所外部提供了基于 Web Service 的访问接口，通过该接口系统其他系统可以完成对会计电算化相关功能的调用，从而实现异构系统的相互协作。

但是在具体的不同系统中，方法的调用不一定会按照预期的方式进行。例如校园的一卡通系统可能出现用餐时间访问量激增的情况。在短时间内出现大量的并发事件将会加大这一时段服务器的负载，但 Web Service 接口能否正常访问不仅与接口的编码质量有关，还与网络环境、服务器的软硬件环境有关。因此在出现大量的并发事件时将可能出现运行速度过慢的现象。Web 服务可以应用于远程的调用，也适应于低压力的工作环境，但在高负荷的情况下，不适于作为主要的接口。

4.6.2.1 模板的划分与设计

由于 Web 服务不适用于近距离高负荷的接口，那么就得提供其他接口以实现会计电算化系统与其他系统的衔接。

并发的问题可以采用多线程的方式进行解决。线程是 CPU 调度的基本单位。由于每个线程所需的资源不同，因此引入多线程可以从整体上提高资源的利用率，提升总体业务的吞吐量。

由于异构系统的未知性与复杂性，当今后增加一种其他系统时可能会对系统造成不确定的影响，而不是仅限于并发。增加异构系统后为了解决其不利影响而将整个会计电算化系统进行修改并重新编译部署是不明智的。因此新接口要求与

会计电算化系统相对独立。

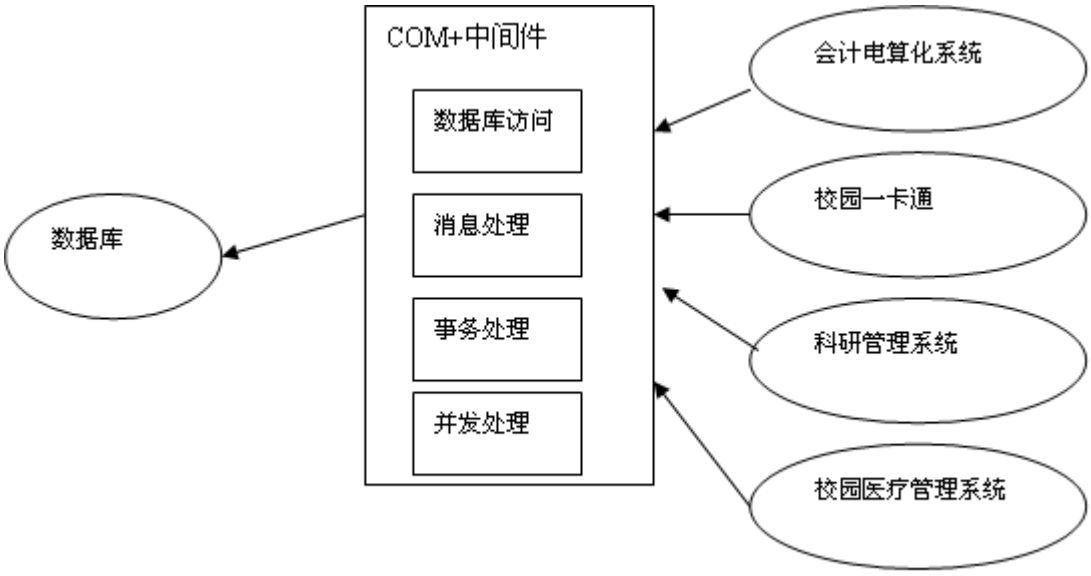


图 4-13 基于 COM+中间件的系统模块

为达到上述的要求，首先需要对系统的模块进行重新设计，在保留主体业务不变的情况下，将易变化的部分分离。易变化的部分即为会计电算化系统与校园一卡通(异构系统)的接口模块，该接口模块可以部署为 COM+中间件的格式。在保持接口不变的情况下，当需求有变化时，会计电算化系统与一卡通系统无需变动，只要更新接口模块的业务逻辑即可。如图 4-11 所示，在开发会计电算化系统时提供一个 COM+的中间件用于为数据库的访问提供消息处理、事务处理、并发处理等服务。后续的其他项目需要与会计电算化系统集成时，只需实现 COM+中所提供的接口即可。

4.6.2.2 使用 COM+中间件实现接口模块

从 4.4.2.1 中得到需要采用多线程来处理并发问题，那么就需要对多线程进行同步与管理[24-25]。由于一个具体的操作涉及到不同的系统，往往要涉及到应用程序级别的事务处理[26-29]。例如，使用一卡通进行购物后需要在一卡通系统与会计电算化系统中同时记录相关的操作，这种记录要么都成功，要么都失败，不许出一个成功一个失败的情况[29-30]。而这两个操作分属两个应用程序，因此需要进行应用程序集别的事务处理[31]。

此外，该模块还具有远程调用与分布式计算的局部特征。因此该模块的实现可以采用中间件的方式进行开发实现。.NET 中提供了 COM+类型的模板，在用户开发中间件时让相关的类继承自 ServicedComponet 基类即可完成中 COM+中间件

的模板制作。

4.6.2.3 COM+中间件中多线程的实现

多线程技术在计算机中的应用很广泛[32]。在.NET 中可以很方便的实现多线程的调用[33-36]。.NET 平台提供了名为 Thread 的线程类，该类的实例需要接受一个委托（函数的指针），从而开辟一个独立于主线程的分支线程进行线程的处理。

为了避免操作过程中多线程对独占资源的争抢，从而造成出现无效数据、脏读、幻读等各种情况，需要对线程进行同步。

对业务进行分析后总结出以下的同步级别。对于一些全局性的配置数据，由于在许多函数中都要进行读写。为避免争抢过程中对全局配置进行争抢，这一类型的数据需要在专门的类中对其进行封装。同时设置该类从 ContextBoundObject 类继承从而实现上下文的绑定。当该类的一个实例被调用时，该类的其他实例就无法被调用。从而避免临界资源争抢访问中的出错。

有一些临界资源仅在某个函数中访问，那么针对该临界资源的访问方法可以将其添加[MethodImpl(MethodImplOptions.Synchronized)]标记，从而避免了整个对象级别的同步。与上下文绑定相比，方法级别的同步更能提高资源的利用率。

在对一些只读资源如配置文件、会计类别、部门类别等大多情况下需要读取而非修改的资源数据进行读取时 ,在这种情况下获取资源的独占权无疑会影响运行效率,因此.Net 提供了一种机制,使用 ReaderWriterLock 进行资源访问时,如果在某一时刻资源并没有获取写的独占权,那么可以获得多个读的访问权。如果某一时刻已经获取了写入的独占权,那么其它读取的访问权必须进行等待。

最后对于较小代码需要同步的区域,如会计电算会系统中的大量临时变量等，可以使用 lock 关键字进行同步。lock 是一种比较好用的简单的线程同步方式，它是通过为给定对象获取互斥锁来实现同步的。它可以保证当一个线程在关键代码段的时候，另一个线程不会进来，它只能等待，等到那个线程对象被释放，也就是说线程出了临界区。

4.6.2.4 COM+中间件中事务的实现

事务具有 ACID（原子性、一致性、执久性、隔离性）的特点。如果要完全实现 ACID 四个特性那么将面临着巨大的工作量。在 COM+中提供了很好的方式来实现事物的 ACID 属性。在微软 Windows2000 及以上版本的操作系统中提供了分布式事务协调器（Distribute Transaction Coordinator）服务以实现事务处理管理器[37-40]。

在.NET 中可以使用[Transaction]标记在 COM+中使用事务。该全中具体的实现请见 5.4 。

需要注意的是会计电算化处理中的事务并非一定是数据库操作的集合体。也有可能是文件的操作。例如有两个操作组成了一个事务：1、使用校园卡消费(超过一定量)，2、给用户短信通知。如果两操作执行完后需要回滚，此时已经发送的短信是无法回滚的。那么可以使用 C#语言完成自定义的回滚操作，例如再发一次短信申明前一次短信无效等。

4.7 本章小结

本章在需求分析的基础上对系统进行了设计。设计内容包括系统的架构设计、数据库设计等。针对具体技术的引入或特定的功能，本章也分别对这些设计技术与功能进行了深入的设计。这些设计有：针对设计模式使用的设计、针对数据访问的设计、针对与其他系统协作的设计。

第五章 系统的实现

系统实现部分的内容从底层基础类的设计到界面的功能涉及面较多，以下就自底向上对各层面分别举例说明系统的实现。

5.1 公共模块中配置文件读取功能的实现

在需求分析的 3.7 部分，提到客户的数据库可能存在变更的可能。因此为了应对这种需求，需要减少数据库与应用程序的耦合。为了减少这种联系，相关信息需要记录在配置文件中。

配置文件是系统中最核心的部分，记录了系统运行所依赖的各项参数。连接字符串就记录在配置文件中用于为数据库的访问提供统一方式。配置文件的读取在公共模块中实现。

在 4.3.1 中提到如果要避免将数据库的访问字行串硬编码到系统中，而是采取注入依赖的方式，将提升本系统的可扩展性。4.3.1 中对使用单例模式进行了设计。以下是其实现：

首先在项目文件夹下找到 Web.Config 文件，该文件是 XML 格式，可以用记事本打开进行编辑。在 Web.Config 中增加节点 ConnectionString。设置如下：

```
<connectionStrings>
    <!--<remove name="LocalSqlServer"/>-->
    <add name="connStr" connectionString="data source=LENOVO-5EB7C3FA\SQLEXPRESS;Integrated Security=SSPI;AttachDBFilename=|DataDirectory|KSH1_Sys2.mdf;User Instance=true" providerName="System.Data.SqlClient"/>
</connectionStrings>
```

在上述的文本中增加了一个名为 connStr 的元素，该元素记录的 connectionString 记录了连接字符串的内容，包括服务器地址及访问方式，数据库的实例等。后面的 providerName 属性指明了代码对该数据库访问的驱动方式。

对于上述的文本可以直接使用文件流进行读取，但这种操作将对文件进行反复的读取，降低性能。因此单例模式代码如下：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Configuration;

namespace CommonBase
{
    /// <summary>
    ///Config 的摘要说明
    /// </summary>
    public class Config
    {
        //获得连拉字符串
        private static string connStr = null;
        public static string GetConnString()
        {
            //单例模式
            if (connStr == null)
            {
            connStr = configurationManager.ConnectionStrings["connStr"].ConnectionString;
            }
            return connStr;
        }
    }
}
```

上述代码定义了一个私有静态的字符串变量 connStr，在后面的 GetConnString() 函数中首先判断该字符串是否为空值，如果为空，则调用 configurationManager 中的 ConnectionString 属性，来获取连接字符串的值。ConfigurationManager 是微软为访问配置文件 Web.Config 而封装的类。如果字符串不为空，则直接返回该字符串的值。由于 static 类型的特点，当第一次赋完值后，该值被保留在内存。等到后续出现新的请求时，将保存的值直接发给相关请求。从而避免了对文件的反复操

作。

5.2 数据访问层的实现

同样在需求分析中的 3.7 部分，提到了开发时的隐含需求，即代码的开发需要尽量有序，编码规则需要尽可能的简单；需要有良好的体系结构以应对业务需求的不断变化。

在本文第四部分完成设计的情况下，此处使用.NET 中的反射机制来对数据访问层进行实现。为减少冗余，此处仅以 INSERT 操作为例进行说明。

在 4.1 中实现了 BLL 与 Model 分离后，简单的 SQL 语句具有很强的规律性。例如，对于一个 Users(Name,Age)类，其生成的 Insert 语句为：INSERT INTO Users([Name],[Age]) VALUES(@Name,@Age)，其中@Name 与@Age 为执行该语句所需要的参数，由外界(代码)提供。从规律中得知若有一对象 Demo，其中拥有 P1-Pn 个属性，则需要动态获取对象名与属性名，生成 Insert 语句如下: “INSERT INTO Demo([P1],[P2],[P3],….[Pn]) VALUES(@P1,@P2,@P3,…..@Pn)”，其流程图如下：因此可以使用反射自动生成 SQL 语句。

但在生成 SQL 语句时需要注意的是有些值不希望写入数据库，例如某个类中有一属性，对应于数据库的自动生成列，很显然是不能写入到数据库中的。另外也有些列需要标记，如主键列。在删除操作时就需要识别主键列。因此使用反射定义标记类如下：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace CommonBase
{
    [AttributeUsage(AttributeTargets.Property)]  //声明该类 AutoInc 只用于修饰属性
    public class AutoInc:Attribute   //该类为自动万的标记，无需定义属性
    {
    }
    [AttributeUsage(AttributeTargets.Property)] //声明该类 AutoInc 只用于修饰属性
```

```
    public class PrimaryKey : Attribute   //该类为主键列的标记，无需定义属性
    {
    }
    [AttributeUsage(AttributeTargets.Property)]
    public class NonColumn : Attribute
    {
      //该类为非数据库列的标记，无需定义属性。该标记示表当前属性不用写入数据库
    }
}
```

由于相关代码较多，此处只提供 Insert 语句的生成具体代码如下：

```
static string CreateInsertSQL(object obj)
{
    //根据属性拼接出一个带参数的 SQL 语句
    //INSERT INTO Users([Name],[Age]) VALUES(@Name,@Age)
    //定义字符串构建器，该字符串初始为 SQL 语句中的“INSERT INTO”
    StringBuilder sb = new StringBuilder("INSERT INTO ");
    Type t = obj.GetType(); //得到当前类型
    sb.Append(t.Name);     //将类型名加入到字符串中(根据之有的规则，类名就是表名)
    sb.Append("(");          //补上左边括号
    PropertyInfo[] pinfo = t.GetProperties(); //得到类名中的属性名（属性名为表中列名）
    foreach (PropertyInfo p in pinfo) //遍历属性，生成 SQL 语句
    {
        sb.Append("[");
        sb.Append(p.Name);
        sb.Append("],");
    }
    sb.Remove(sb.Length-1,1); //移除多余的","
    sb.Append("VALUES (");       //以下生成 SQL 语句中 VALUES 后的内容
    foreach (PropertyInfo p in pinfo)
    {
        sb.AppendFormat("@{0},",p.Name);
    }
```

```
        sb.Remove(sb.Length - 1, 1);//移除多余的","
        sb.Append(")");
        return sb.ToString();
    }
```

上述代码生成的语句为带参数的 SQL，在执行时还需要赋予相关的参数。相关参数直接从对象中获取。代码如下：

```
public static List<SqlParameter> CreateInsertParameter(object obj)
{
    if (obj == null) throw new Exception("参数不能为 null");
    List<SqlParameter> list = new List<SqlParameter>();
    Type t = obj.GetType();
    PropertyInfo[] info = t.GetProperties();
    if (info.Length == 0) throw new Exception(t.Name + "可用属性数量为 0");
    foreach (PropertyInfo p in info)
    {
        //判断自动增加
        if (IsAutoInc(p) == true) continue;
        //判断非数据库列
        if (IsNonColumn(p) == true) continue;
        object o = p.GetValue(obj,null);   //关键语句，从对象中获得相关属性的值
        list.Add(new SqlParameter("@"+p.Name,o)); //将属性加到集合中
    }
    return list;
}
```

5.3 凭证管理模块的实现

在本系统中，具体的功能模块数量较多。此处以凭证管理模块为例说明系统相关功能模块的实现过程。凭证管理模块是需求中会计管理模块下的一个子模块。

5.3.1 凭证管理模块实现的前提

银行对账模块是具体应用的一个模块，它直接应用于前台。但是该模块的代码极其简单，原因就在于良好的系统结构。为了清晰的说明这一部分的内容，此处先对一些基础类进行介绍。

5.1 中提到实现数据库连接字符串读取的类 Config，该类用于从配置文件中获取连接字符串。

5.2 中提到了 AutoInc、PrimaryKey、NonColumn 类，这些类的作用是用于标记实体的某字段的特征，在使用反射自动生成 SQL 语句及所需参数时需要根据这些参数来采取相应的行为，如将某个字符放到 Where 语句或有意在语句中忽略这一字段。另外在 5.2 中还提到了 CreateInsertSQL 及 CreateInsertParameter 两方法，分别用于生成 SQL 语句与 SQL 语句所需的参数。这两函数存在于 SqlCreator 类下，该类结构如下：

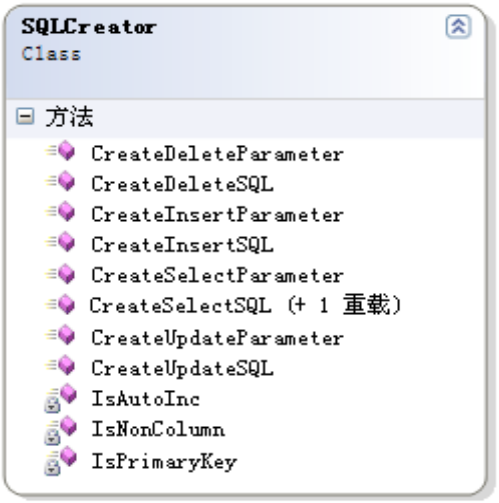


图 5-1 SQLCreator 结构

从上图中可以看到除了生成 Insert 语句及参数外，删除、修改、查询也均提供了类似的写法。此外还有 IsAutoInc、IsNonColumn 及 IsPrimary 三个私有方法，分别用于判断某属性是否具有自动增加、非数据库列及主键列的特征。

除了上述的类之外，还需要提供一个核心的与数据库交互类 SqlHelper。该类代码较多，仅提供格式如下：

图 5-2 SqlHelper 结构

上述的代码中 SqlHelper 的构造函数完成对连接字符串的初始化，而 ExecuteScalar、ExecuteSQL 与 Query 分别于用执行单值查询、非查询及查询操作。以上的类构成了数据访问层的基础，该类是 4.1.2 三层结构中的一个子集，结构如下：

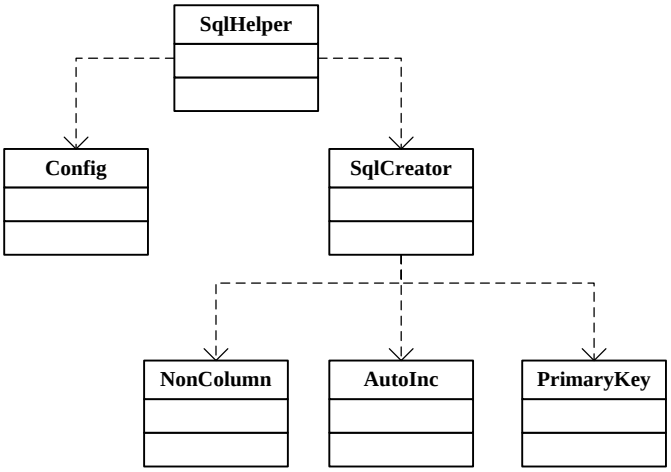


图 5-3 数据访问层结构

5.3.2 凭证管理模块实现

不管凭证的实际业务如何，其本质上仍然是对数据库的增加、删除、修改与查询。根据数据库表的定义，设计实体类如下：

```
class vouMng_voucher:ModelBase
{
 [AutoInc]
```

```
        public int Voucher_id     {get;set;}
    public int Int              {get;set;}
    public int Number       {get;set;}
    public int Enclosure    {get;set;}
    public int CreateTime   {get;set;}
    public int Director     {get;set;}
    public int Check  {get;set;}
    public int Account      {get;set;}
    public int Create {get;set;}
    public int Cashier      {get;set;}
    public int Account      {get;set;}
    public int Over_id      {get;set;}
    public int CurrPeriod_id{get;set;}
    }
```

上述的类的类名与数据库的表名一致，而属性名称与表的字段名称一致，只有该规范起作用时才生正确生成数据库要执行的 SQL 语句及参数。该类继承自 ModelBase 类，即实体类的基类。该类提供 Insert、Delete、Update 及 Query 方法。每种方法均能根据自身对象的属性生成所需的 Sql 语句及所需参数，并调用 SqlHelper 类执行。ModelBase 类的格式如下：

```
public class ModelBase
    {
        //增加
        public virtual int Insert()
        {
            try
            {
                //生成 SQL 语句
                string cmdText = SQLCreator.CreateInsertSQL(this);
                //生成 SQL 所需的参数
                List<SqlParameter> list = SQLCreator.CreateInsertParameter(this);
                //执行 SQL 语句
                SqlHelper helper = new SqlHelper();
```

```
            return helper.ExecuteSQL(cmdText, list.ToArray());
        }
        catch (Exception ex)
        {
            throw ex;
        }
    }
    public virtual int Delete()
    {
        try
        {
            //生成 SQL 语句
            string cmdText = SQLCreator.CreateDeleteSQL(this);
            //生成 SQL 所需的参数
            List<SqlParameter> list = SQLCreator.CreateDeleteParameter(this);
            //执行 SQL 语句
            SqlHelper helper = new SqlHelper();
            return helper.ExecuteSQL(cmdText, list.ToArray());
        }
        catch (Exception ex)
        {
            throw ex;
        }
    }
    public DataTable Query()
    {
        string cmdText = SQLCreator.CreateSelectSQL(this);
        List<SqlParameter> list = SQLCreator.CreateSelectParameter(this);
        SqlHelper helper = new SqlHelper();
        return helper.Query(cmdText, list.ToArray());
    }
    public virtual int Update()
```

```
        {
            try
            {
                //生成 SQL 语句
                string cmdText = SQLCreator.CreateUpdateSQL(this);
                //生成 SQL 所需的参数
                List<SqlParameter> list = SQLCreator.CreateUpdateParameter(this);
                //执行 SQL 语句
                SqlHelper helper = new SqlHelper();
                return helper.ExecuteSQL(cmdText, list.ToArray());
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }
    }
```

在完成上述工作后，界面上凭证管理模块实现的代码极其简单。当增加一条记录时，只须从界面上收集数据并调用 Insert 即可。

```
vouMng_voucher     mv = new vouMng_voucher ();
mv. Int = int.Parse(tInt.Text);
mv. Number = int.Parse(tNumber.Text);
mv. Account = int.Parse(tAccount.Text);
//…….对其他属性的赋值
try
{
  int i=mv.Insert();   //执行增加工作
  if(i>0) WebMsg.Show(“增加成功”);   //信息呈现
  else WebMsg.Show(“增加不成功”);
}
catch(Exception ex)   //异常处理
{
```

```
 Response.Write(ex.Message);
}
```

上述的代码当收集完数据后只需调用 Insert 即可完成操作，这种操作使得前台的工作人员掌握数据收集与数据呈现的少量代码即可完成开发工作。降低了软件开发的成本。

删除凭证记录的代码，删除只需要从界面上获取要删除的主键即可。

```
vouMng_voucher     mv = new vouMng_voucher ();
mv. Voucher_id = int.Parse(dropVoucher.SelectValue.ToString());//从下拉列表中获取要删除记录的 ID
try
{
  int i=mv.Delete();   //执行删除工作
  if(i>0) WebMsg.Show(“删除成功”);   //信息呈现
  else WebMsg.Show(“删除不成功”);
}
catch(Exception ex)   //异常处理
{
 Response.Write(ex.Message);
}
```

修改凭证信息的代码与增加的代码类似，只是需要在删除前指定主键值

```
vouMng_voucher     mv = new vouMng_voucher ();
mv. Voucher_id = int.Parse(dropVoucher.SelectValue.ToString()); //主键
mv. Int = int.Parse(tInt.Text);
mv. Number = int.Parse(tNumber.Text);
mv. Account = int.Parse(tAccount.Text);
//…….对其他属性的赋值
try
{
  int i=mv.Insert();   //执行修改工作
  if(i>0) WebMsg.Show(“修改成功”);   //信息呈现
  else WebMsg.Show(“修改不成功”);
}
catch(Exception ex)   //异常处理
```

```
{
 Response.Write(ex.Message);
}
```

查询的代码是根据主键得到其他列的值，该值只接保存在对象的属性中。

```
vouMng_voucher      mv = new vouMng_voucher ();
mv. Voucher_id = int.Parse(dropVoucher.SelectValue.ToString()); //主键
try
{
  mv.Query();
  tInt.Text =mv. Int; //显示工作
  tNumber.Text = mv. Number;
  tAccount.Text = mv. Account;
  //……..其他显示
}
catch(Exception ex)    //异常处理
{
 Response.Write(ex.Message);
}
```

同理，其他模块的开发在已经有上述基础类的情况下只需根据数据库表结构定义一个实体类，并从 ModelBase 继承，则可直接在界面上调用。有一些复杂业务涉及到多表操作的，则需要定义业务逻辑业 BLL，这一点在设计时已经给出，受于篇幅限制，此处就不进行介绍。

5.4 本章小结

本章主要从底向上介绍了公共模块中连接字符串读取操作、数据访问层中 INSERT 功能的开发及会计管理中凭证管理子模块的实现等内容进行了介绍。基础类开发的情况下，其他功能的实现与凭证管理子模块的实现基本一致。

第六章 高校会计电算化系统的实施与测试

6.1 系统的初始设置

6.1.1 管理员设置

在系统运行前需要建立管理员账号，该管理员拥有所有的功能权限，在设计管理员时需要输入用户名与口令。

口令的输入有长度的限制，另外还需要使用字母、数字及符号的组合，如果连续两次输入出错，则需要重新进行设置。

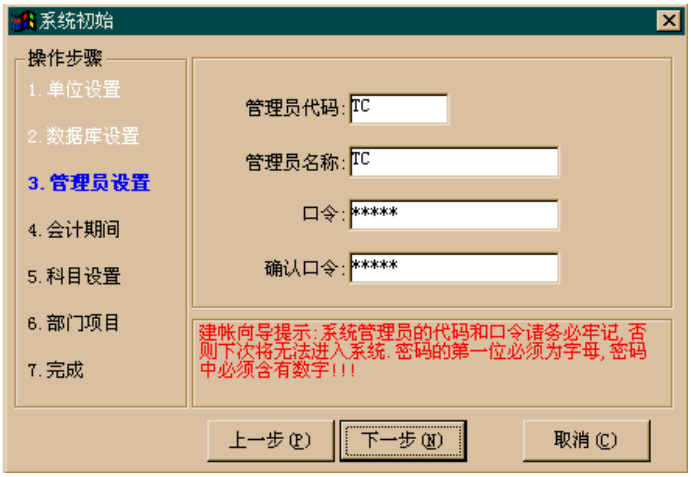


图 6-1 管理员设置

6.1.2 会计期间设置

在系统初始化时还需要进行会计期间的设计，会计期间与自然年月并不相同，因此需要进行起始时间的设置。系统缺省按照自然月设置，用户可以自行修改。在会计期间以前的时间段需要设置为不可更改的蓝色标记。而之后的时间需要将背景色设置为白色，标识为可以修改的部分。对于一些特殊的日期，如，第 13 会计期间的日期范围设置为“12.31-12.31”，此日期是专门用作年终转账的。

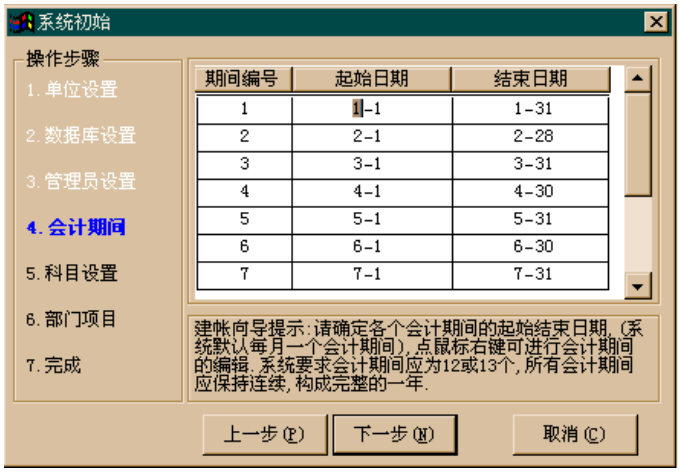


图 6-2 系统初始化

6.1.3 科目设置

该界面可以设置修改最大科目级次以及各级科目长度。

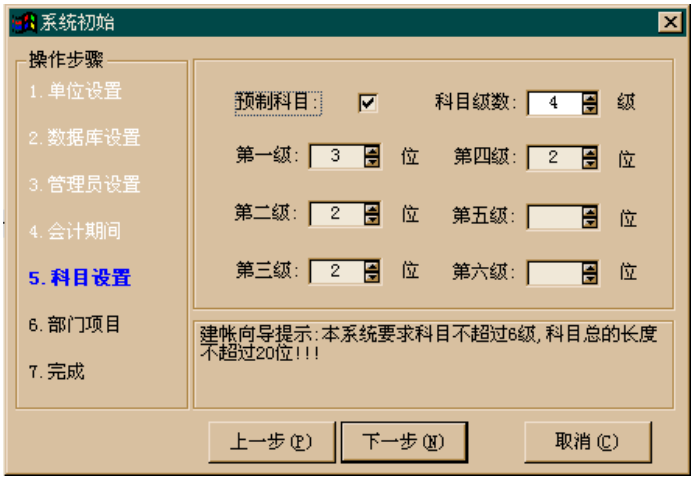


图 6-3 科目设置

在本系统中，最多可以设置 6 级的科目级别，还可以在此设置各科目相应代码的长度。科目总长度有 20 位的上限。

如果“预置科目”的选择控件被打勾，则可以许可用户预置所属教育行业的标准总账科目。否则不进行处理。

6.1.4 部门项目

本系统部门、项目都可分级核算，最多为四级。在此修改最大部门级次、项目级次以及各级部门编码长度和项目编码长度。

最大部门编码级次和项目编码级次为 4 级，总长度不能超过 12 位。

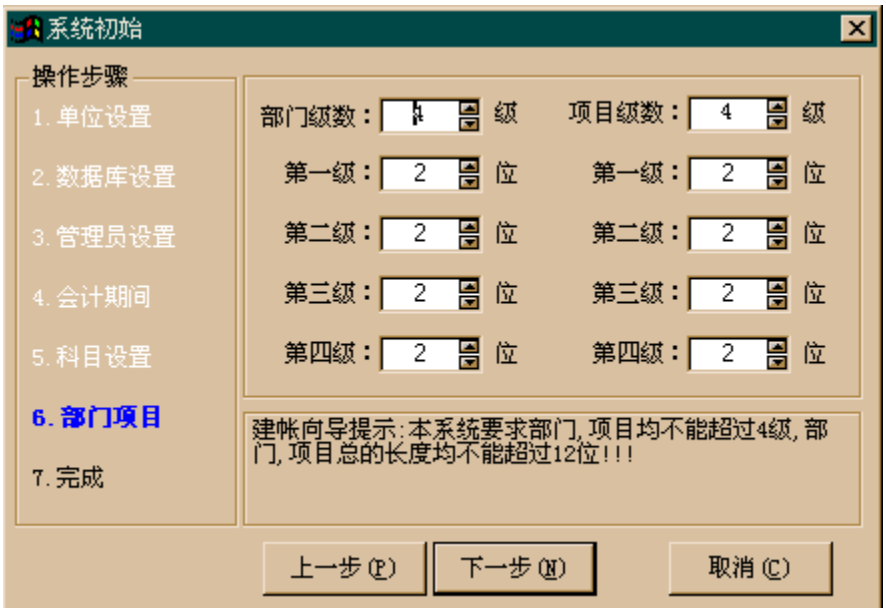


图 6-4 系统初始

6.2 系统的使用示例

系统功能较多，且操作复杂，因此以下仅对核心模板的使用进行简单的描述。

6.2.1 系统初始化

如果是首次使用本系统，用户会进行大量初始化工作，只有在初始化后，系统才可以正式使用。与 6.1 不同的是 6.1 针对软件环境的外围功能进行初始化，此处针对核心功能的应用初始化。

此处初始化的内容包括：

定义各类核算基本编码，包括凭证类型、科目、部门、项目、个人、银行账号、凭证打印格式等

录入科目余额初始化（各明细科目年初余额）

录入项目余额初始化（各经费项目年初余额）

录入往来款初始化（未还清暂付款、暂存款明细账）

银行对账初始化（录入单位、银行未达账项，使调节后余额相等）

系统参数设置及其他准备工作，建账成功后开启系统。

系统初始化是使用系统的基础，为今后系统的使用开辟了工作环境，所以要经过多方面的准备筹划，

把基础工作做好、做细、做准确。在满足当前会计核算、管理的同时，还要有更高度的规划能力，要考虑前瞻性、规范性及可扩展性，为今后业务的不断发展打下良好基础。

6.2.2 经费指标初始化

在开始操作前，需要进行“经费指标初始”工作。它与辅助账管理中的暂付、暂存款初始化，“账务管理”中的本币科目余额初始化并称为初始化的三大任务。该任务主要是将截止到期的余额输入到计算机，形成各项目的期初余额。然后就可以在这一基础上进行项目的记账工作。具体记账过程由计算机自动进行。

图 6-6 经费指标初始化

6.2.3 凭证管理

登记账簿前需要进行记账凭证的登记。在电算化系统中，数据是否准确与完整有赖于记账凭证录入的准确完整。

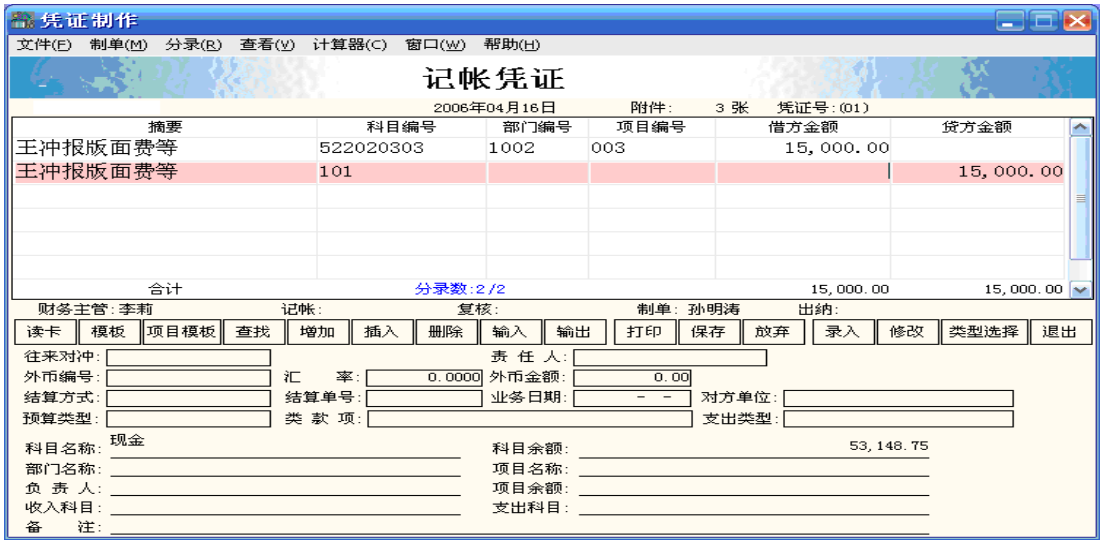


图 6-7 凭证管理

6.2.4 出纳复核

出纳复核主要是对会计凭证涉及现金银行的收入与支出进行核对的功能。只有在完成复核工作后才可进行出纳工作。该功能由出纳人员根据原始凭证与系统中的账目一一核对。主要核对的对象为出纳凭证的出纳科目的金额是否正确，复核不成功的数据需要由记账人员重新录入。

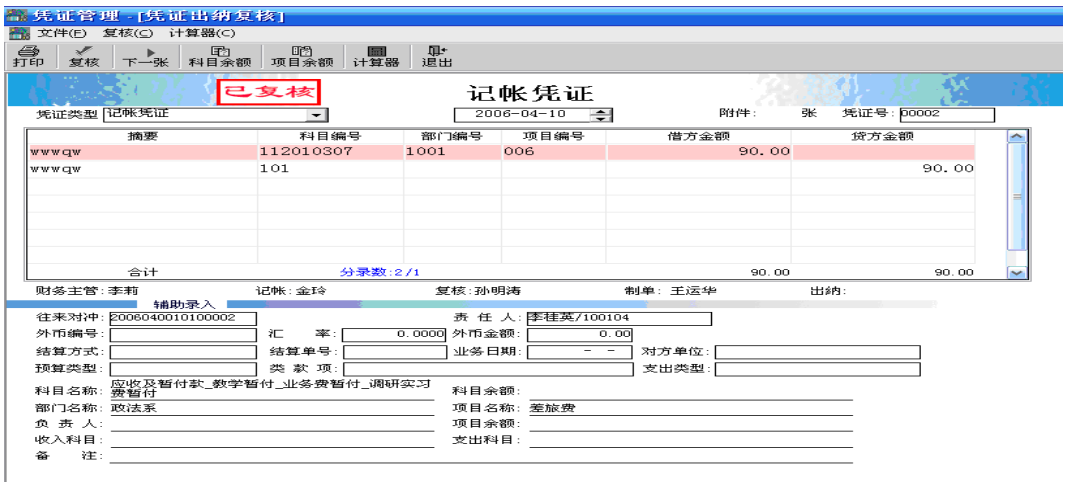


图 6-8 出纳复核

6.2.5 银行对账

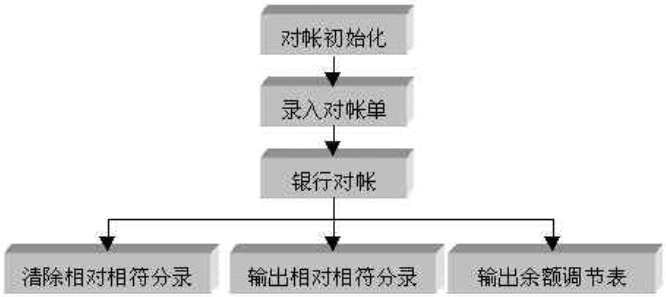
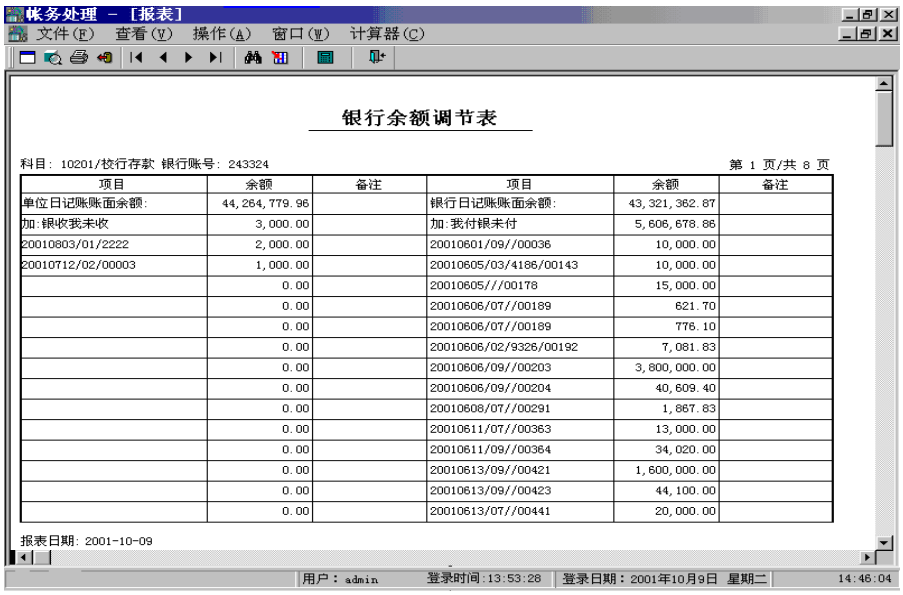


图 6-9 银行对账顺序

银行对帐是指定期将银行存款日记账的记录与银行对账单逐笔核对，发现其中的未达帐项，编制“银行存款余额调整表”，将学校银行存款日记账余额与银行对账单余额调整一致。对帐时学校帐与银行帐可以进行多次自动试对，也可以在自动对帐的基础上进行手工调整，手工调整的帐也能够自动进行对帐正确性检查。使用顺序图示:



项目	余额	备注	项目	余额	备注
单位日记账账面余额:	44,264,779.96		银行日记账账面余额:	43,321,362.87	
加:银收我未收	3,000.00		加:我付银未付	5,606,678.86	
20010803/01/2222	2,000.00		20010601/09//00036	10,000.00	
20010712/02/00003	1,000.00		20010605/03/4186/00143	10,000.00	
	0.00		20010605///00178	15,000.00	
	0.00		20010606/07//00189	621.70	
	0.00		20010606/07//00189	776.10	
	0.00		20010606/02/9326/00192	7,081.83	
	0.00		20010606/09//00203	3,800,000.00	
	0.00		20010606/09//00204	40,609.40	
	0.00		20010608/07//00291	1,867.83	
	0.00		20010611/07//00363	13,000.00	
	0.00		20010611/09//00364	34,020.00	
	0.00		20010613/09//00421	1,600,000.00	
	0.00		20010613/09//00423	44,100.00	
	0.00		20010613/07//00441	20,000.00	

图 6-10 银行对账

银行对账需要以采用手工对账与自动对账同时进行的方式。本系统中的自动对账是根据对账依据自动进行核对、勾销。要进行自动对账时，系统根据操作人员指定的会计期间选择方向、金额及其他自定义的条件来核对账目，已经自动的账目进行自动标记，而未核对的账目则需要列出以提示用户。为了保证对账更彻

底正确，用户可用手工对账来进行调整。具体账务处理如图 6-10 所示。

6.3 实施结果

在实施了本项目后，通过该系统可以很好的完成会计工作中的会计核算、会计管理、会计监督三大模块的功能。此外对于高校会计信息化工作中会计活动涉及面广、综合性强、敏感性高等对会计信息化系统有特别要求的特点，本项目也很好的进行了满足。

由于系统设计到位，为后续的升级与扩展预留了接口，因此在与校园一卡通系统、图书馆管理系统等其他系统进行协作时没有对会计电算化系统进行改动。据反馈得知，该系统的易用性得到了客户的认可。

6.4 系统测试

系统实现阶段完成之后，系统已经基本成型，但是还存在的诸多不稳定之处，为了能够及时发现和修正系统中存在的错误和漏洞，项目组对系统进行了测试工作，主要是使用功能测试与压力测试的方法进行。

6.4.1 功能测试示例

功能测试也叫黑盒子测试或数据驱动测试,只需考虑各个功能，不需要考虑整个软件的内部结构及代码.一般从软件产品的界面、架构出发，按照需求编写出来的测试用例，输入数据在预期结果和实际结果之间进行评测,进而提出更加使产品达到用户使用的要求。

该部分的测试用例很多，基本每个操作就至少需要一个测试用例，因此该部分内容较为复杂，以下就给出一个测试用例。

对于图 6-1 中的管理员设置设计测试用例如下：

表 6-1 管理员设置测试用例

用例标识		Init-001	项目名称	会计电算化			
开发人员			模块名称	管理员设置			
用例作者			参考信息	项目详细设计说明书			
测试类型		功能测试	设计日期	2010-8-4	设计人员		
测试方法		黑盒	测试日期	2010-9-4	测试人员		
用例描述		添加试题					
前置条件		数据库已预设可供测试使用的相关信息					
编号	优先级	测试项	描述/输入/操作	期望结果		实际结果	备注
001	低	用户名能否正常显示	输入用户名	显示用户名			
002	低	密码能否用*代替	输入密码	密码显示为*			
003	低	该窗口是否任意身份登录就能显示	使用普通账号登录	不能显示			
004	中	设置的密码是否没有限制	长度不足的字符串含非字母数字的字符的字符串	提示设置成功			

6.4.2 安全测试示例

功能测试与压力测试是本系统最核心的测试，但系统也使用其他技术方法进行了相关测试，例如安全性测试。

安全测试的一项重要内容是测试系统是否存在注入式攻击的漏洞。注入式攻击往往发生在对 SQL 语言本身的拼接上。

因此设计用例如下：

表 6-3 安全测试用例

测试名称	SQL 注入测试		
前提条件	使用 WINDOWS XP 系统 Pentium® 4 CPU 3.00GHz 1.5G 内存 SQL SERVER 2005		
输入/动作	输出/响应	期望	是否能正常运行
在密码框输入注入的 SQL 片段	提示安全出错	提示安全出错	能

在密码框输入以下字符

abc’ && ‘1=1

当运行结果如下时，说明不支持注入式攻击。

图 6-11 安全测试用例的期望结果

6.5 本章小结

本章主要对系统的实施过程与系统的测试进行了介绍。在系统实施过程中，重点介绍各模拟操作的注意事项与相关界面。在系统测试中对功能测试与安全测试分别进行了测试用例的设计。

第七章 全文总结与展望

7.1 总结

该研究在对会计电算化系统原理与技术进行研究的基础上，设计了适用于高校的会计电算化系统。该系统在可维护性及与外部平台交互的情况下具有适用性。总体来说本文的研究工作具体体现在以下几个方面：

1、完成了会计电算化系统的需求分析与设计。对普通的三层架构进行了扩展，将业务与数据实体分离，模板之间的耦合性降低。

2、对数据库进行访问采用了.NET 的反射机制，通过反射的方式自动生成 SQL 语句与 SQL 语句执行时所需要的参数。该方式的实施减少了应用程序与数据库之间的耦合。

3、在系统的多个地方采用设计模式进行操作进行设计，提高了系统的灵活性与可维护性。

4、系统对外通过 Web 服务进行接口的发布，使本系统与外部程序能很好的协作。

7.2 展望

会计电算化系统。在后续的工作中，将针对这些较弱的地方做进一步的强加。在技术的实现上并没有太大的困难，因此本文的重点放在可系统维护性、灵活性的提升上。实际上目前会计电算化系统的发展已经有新的变化，更多地强调用与其他系统如决策支持系统、专家系统、知识管理系统等的整合上。这是本文研究较弱的地方。

致　谢

时光荏苒，岁月如歌，一晃三年的研究生学习生涯即将结束，在毕业之际，我感慨良多，首先我要感谢我的导师，他高尚的人格和对学术一丝不苟的精神深深地影响着我，他在百忙之中抽空耐心地指导我的学位论文，在整个论文写作过程中给我诸多良好的建议，在此我对导师的无私付出表示深深地感谢。同时也要感谢电子科技大学的老师们，你们在百忙之中的授课让我接触到了许多计算机技术和经典理论，同时也丰富了我的实践经验，提高了动手能力，为我未来的工作和学习打下良好的基础。此外，我还要感谢我的同学们，和你们相处的日子是快乐的、充实的，希望我们的友谊长存！

参考文献

[1] 唐巧巧,会计电算化的现状及发展趋势[J]. 企业科技与发展，2010(20), 290-291

[2] 胡毅,论知识经济下的会计创新及发展趋势[J].科技信息， 2010(12)， 17

[3] 王淑兰,会计信息化对企业财务管理的影响及对策[J].科技信息， 2010(16) ， 574-577

[4] 胡荣香,会计信息化及其在企业管理中的应用[J].科学中国人， 2000(3) ，51-52

[5] 张春梅,论成本会计的基础--成本核算[J]. 知识经济， 2012(01) ,138

[6] 王赟. 高校财务管理存在的问题及对策[J]. 现代营销(学苑版), 2012,(02) :237

[7] 占小忆 数据仓库和 OLAP 技术在高校教学管理系统中的应用研究[J]. 中国科技信息,2010(22):122-123

[8] 罗革新, 吴建平, 丁闫, 刘光来, 宋力巍. 面向服务体系架构软件平台及其应用[J]. 信息技术, 2012,(02) :109-113

[9]Patterns: Service-oriented architecture and Web service.[2009-11-02].http: //www.redbooks.ibm.com.

[10] http://baike.baidu.com/view/66964.htm

[11] 程杰,大话设计模式[M]. 北京:清华大学出版社, 2010:214-215

[12] 赵春霞,宫明明. 桥接模式在日志系统中的应用[J].青岛职业技术学院学报，2010,1:58-59

[13] 王小鉴．使用 T-SQL 生成 T-SQL． 程序员, 2009(11)：90~93

[14] 微软公司．.NET Framework2.0 程序设计．人民邮电出版社, 2010. 321~322

[15] Lai R. J2EE 平台 Web Services[M] . 周 斌, 刘亚萍, 冯艳玲, 译. 北京: 电子工业出版社, 2005.

[16] 许 峰, 林果园, 黄 皓. Web Services 的访问控制研究综述[J] . 计算机科学, 2005, 32(2) : 1- 4.

[17] 佘名高, 贾秀峰, 林坤江, 等. 基于 Web 服务的企业应用集成[J] . 计算机技术与发展, 2007, 17(5) : 55- 58.

[18] Kreger H. Web Services Conceptual Ar chitecture 1. 0, IBMSoftware Group[EB/ OL] . 2001. http: / / www- 3. ibm. com/software/ solution/ webservices/ pdf/ WSCA. pdf.

[19] Bellwood T , Clement L, Ehnebuske D, et al. OASIS Specification, UDDI v3. 0[EB/ OL] . 2002. http: / / uddi. org/ pubs/uddi- v3. htm.

[20] Basiura R, Batongbsealetal M. ASP. NET 万维网服务高级编程[M] . 北京: 清华大学出版社, 2002: 88- 105.

[21] 邓 言, 曾文华. SOAP 的原理与实现[J] . 杭州电子工业学院学报, 2002, 6(3) : 19- 23.

[22] 施明辉, 孙荣胜. 用基于 XML 的 SOAP 机制构建应用系统[J] . 计算机应用研究,2002, 22(4) : 80- 83.

[23]孙慧玲 陈伟晓 李华军 校园一卡通平台下高校财务管理模式探讨[J] 财会通迅 2010(9) 89-90

[24] 刘娜, 田巍. 多线程通讯技术原理分析及应用[J]. 信息通信, 2012,(01) :64-65

[21] 陈平, 高春庚. 基于 WIN 杀死进程的设计技术研究[J]. 数字技术与应用, 2012,(01) :77

[25] 高水娟. 软件开发中调试死锁问题的研究[J]. 软件导刊, 2012,(02) :23-24

[26] 叶小莺. 基于回调机制的异步日志服务的开发[J]. 电子世界, 2012,(04) :42-43

[27] 何新贵，刘云生．特种数据库技术[M] ．北京：科学技术出版社，2007，203-230.

[28] Pan Yi，Lu Yan-sheng. Nested transaction concurrency control in parallel real-time databases [J]. J of Donghua Univ，22(2):2005，114.

[29] 刘去生，丁力. 嵌入式实时数据库管理系统的设计[J]. 计算机应用研究，2006，8 ：230-231.

[30] Abhott R K， Garcir-Molina H.Scheduling real-time transactions ： a performance evaluation[J].ACM Trans on Database Syst,17(3):1992,513.

[31] 夏家莉．基于替代/ 补偿的实时事务模型[J]. 计算机工程与应用，2003，39（34）：25-27.

[32] 夏家莉，刘云生．满足嵌入式实时数据库系统的可预见能力[J].小型微型计算机系统，2003.24（2）：234-237.

[33] Braoudakis,Timeliness Via Speculation for Real-timeDatabase,In Pro.of IEEE Real-time System Symposium,1994.

[34] 张捷 使用事务处理解决数据操作不一致的问题[J] 电子商务，2010,11： 58-61

[35] 张文梅, 廖福保 Web 应用异步任务处理的实现研究[J] .微型机与应用 2012,(31)：14-16

[36] 李菁菁，卢冠华，王春红 基于.Net 多线程技术的排队叫号系统的设计与实现[J]. 中国医疗设备, 2012(27):44-45

[37] 隋新, 朱云龙, 南琳, 晏晓辉. 基于 SOA 的供应链管理平台设计与实现[J]. 计算机工程与设计, 2012,(01) :147-152

[38] 万年红. 面向服务的自适应云资源信息集成软件架构[J]. 计算机应用, 2012,(01) :170-174

[39] 蹇崇军, 洪欣. 一种支持外部 Web 服务集成的过程模型[J]. 计算机工程, 2012,(02) :66-68

[40] 王莉莉.基于分布对象的异步消息模型研究[J]. 软件导刊,2010,(7):31-32

电子科技大学
UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

专业学位硕士学位论文

MASTER THESIS FOR PROFESSIONAL DEGREE