摘 要

软件技术的发展已经经历了面向过程、面向对象和面向组件几个阶段，近几年，又提出了面向服务的体系结构 SOA(Service Oriented Architechture)。SOA 具有松藕合、易集成、可重用和业务驱动等特点，现已成为企业应用开发的热点。在基于 SOA 的分布式企业应用开发，其中最基础的工作就是建立一个可靠、稳定、有利于整合的数据模型，进行数据的集成和重构，以解决庞大而复杂的业务数据在各个系统之间的流动和异构系统之间的数据共享。

本论文从中央广播电视大学教务管理系统出发，将面向服务架构的思想应用到数据模型建模过程中。将数据模型、业务策略和服务定义结合起来，以建立统一数据模型为中心，在系统的分层体系结构中抽象一个数据服务层，专门用来解决 SOA 的数据服务要求，从而隔离应用与底层数据源。

论文首先讨论了数据模型建模的相关理论和技术，详细地阐述了 SOA 的相关概念、特点、优势、及主要实现技术，并分析了基于 SOA 的分布式应用系统的参考架构。其次，论文借鉴借鉴数据仓库主题数据存储的思想和面向服务分析和设计的方法，提出了分主题域 E-R 实体数据模型和服务数据模型两阶段来建立基于 SOA 的数据模型，以及采用分层描述的方法从数据实体层、数据聚合层和跨组织服务聚合层的三个层次来描述数据模型。最后，在中央电大教务管理系统的实际开发中应用这种方法，分析建立系统的数据模型；同时根据中央电大教务管理系统的特点，提出了数据服务的实现框架，并针对不同的实际问题和技术细节，分析设计了相应的解决方案，完成理论与实践的结合。

关键词 SOA；分布式应用系统；数据服务；数据模型

Abstract

The software technology has already experienced the Process-oriented, the Object-oriented as well as the Component-oriented stage, and recently, the SOA (Service Oriented Architecture), which is not only coupled loosely, easy to integrate and reuse, but also driven by business, has obtained the point. As far as the distributional enterprise development based on the SOA, the most primary and elementary job is to establish a reliable, stable and being easily integrated data model for data flowing of huge and complex service data between subsystems and data sharing in heterogeneous system.

This thesis, having basis of the Central Radio and TV University education administration system, combined data model, operation strategies and service definition to build up a special data service layer in data model. This policy is tending to separate data application from data source.

In the first place, the thesis discussed related theory and technology of data model, and described the concepts, characteristics, advantages and methods of operation of SOA in addition to analyze the architecture of distributional system based on SOA. In the second place, the writer, learning lessons from the theory of topic data saving in data warehouse and the service oriented method of analyzing and designing, raised the issue that we can not only build up a data model based on the SOA through two steps: set up E-R entity data model for diverse subject and establish service data model, but also profiled a data model having three layers: data entity layer, data integration layer and crossing organization service integration layer. Finally, the writer used the method which we discussed above to analyze and build up a data model in the Educational Management System for Central Radio and TV University; in the meantime, according to the characteristics of the system, the author put forward the operation architecture of data service, while analyzing and designing a resolution aiming at different actual problem and technique detail.

Keywords Service Oriented Architectures; Distributed application system; Data Model; Data services

独创性声明

本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京工业大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

签名：王亮 日期：2007.6.13

关于论文使用授权的说明

本人完全了解北京工业大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

（保密的论文在解密后应遵守此规定）

签名：王亮 导师签名：[illegible] 日期：2007.6.13

第 1 章 绪论

1.1 本研究课题的学术背景及其理论与实际意义

随着网络技术和企业信息化的发展，分布式应用系统得到了越来越广泛的应用。人们可以分散在不同的地理位置上，通过网络跨时空的共享信息、协同工作，极大地提高了工作效率。但是，这一方面导致分布式应用系统的功能、性能、规模和复杂性的极大增长，同时也要求各个系统之间能相互交互，使得这些分布在不同地域的系统提出了集成和整合的需求。

另一方面，软件的发展已经经历了面向过程、面向对象和面向组件等几个阶段，近两年来，为了适应新的软件发展，又提出了面向服务的体系结构(Service Oriented Architechture)的软件设计方法，因其松耦合、易集成、可重用和业务驱动等特点，已成为现在企业应用开发的热点。SOA 既是一种软件体系结构，也是一种新的软件开发方法和 IT 策略[1]。目前各大厂商都分别提出了各自的相关技术和解决方案。但不论采用哪种软件体系结构或设计方法，建立一个可靠、稳定、有利于整合的数据模型都是构建一个分布式应用系统的关键，而对基于 SOA 的分布式应用系统开发，其中最基础的工作就是基于 SOA 企业信息系统参考架构进行数据的集成和重构。

各个企业应用系统中异构数据的集成有其自己的特征，主要表现为分布性、自治性和异构性。分布性是指异构数据存放在分散的、彼此可相互通信的多个场地；自治性是指集成系统不能影响各局部应用系统，被集成的每个局部数据源之上都可能运行着自己原来的应用程序，这就要求局部数据源在被集成之后仍然保持一定程度的独立性；异构性主要有运行环境的异构和数据模型的异构[2]。

本论文研究基于 SOA 解决分布式异构数据集成问题，在企业分层体系结构中抽象出一个数据服务层，建立一个统一的综合数据模型，专门用来解决 SOA 的数据服务要求；通过数据转换、数据移植、数据共享等方式捕获与数据访问和数据更新相关的逻辑，完成数据服务的建模和元数据管理，创建和维护数据服务；最后给出了一个实现分布式应用系统数据服务的逻辑框架。

1.2 国内外研究现状和发展动态

分布式系统的特点是运作的敏捷性、信息需求的多样性、信息系统的异构性以及良好的扩展性。因此，建立分布式系统的统一数据模型和实现的数据服

务应具备开发和兼容性、适应和柔性、协作和灵活性。而传统数据建模和数据共享主要是针对同一企业内部或相同组织结构内部的紧耦合性的数据共享和交换，对松散的异构数据无能为力。

目前，最新的面向服务的体系结构的提出，突破传统的像"烟囱"一样的数据模型，把它们扁平化，在设计数据模型的时候，直接从所在领域出发，建立面向主题的域模型，从而建立对应的数据模型，这种数据模型可以在某个领域内通用，而并不依赖某个具体业务。

SOA 是思考构建软件模型的一种优秀方式，因为 SOA 的松耦合的、抽象的。它的优势在于可以构造组件、企业域、服务和规范数据模型。要快速搭建满足 SOA 结构的数据模型要求一个关注复用、避免垂直式应用程序和 IT 资源"孤岛"的体系结构。把 SOA 应用于数据集成领域是采用 SOA 进行软件架构设计和实现的一个热点，也是分布式异构数据集成的重大机遇。

IBM 和 BEA：IBM 和 BEA 是 SOA 的先行者。其中 IBM 公司先后发布了应用于 SOA 架构的 31 种产品，覆盖了他们提供的五大产品线 Websphere、Workplace、Tivoli、DB2 及其 Rational。BEA 公司在 2007 年发布业界第一个能把业务和 IT 置于同一工作环境中的 SOA 工具 SOA360°产品系列，它涵盖 BEA 的三个产品系列：Tuxedo、WebLogic 和 AquaLogic。而针对数据建模和数据集成，IBM 和 BEA 等还联合提出了服务数据对象 (Service Data Objects，SDO) 体系结构。SDO 是一种应用程序编程接口，可以简化和统一对异构数据的访问，同时也可用于数据处理的其他方面。其中 BEA 的 AquaLogic Data Services Platform 就是专门用来为 SOA 构建的企业级数据服务平台的。

Microsoft：使用基于 XML Web 服务的方式解决异构数据集成是微软推出的主要解决方案，并提供了相关的技术和产品，如 ADO.net。近来，又提出了面向 SOA 的架构 Indigo 和集成工具 Microsoft BizTalk。相对于其他厂商而言，微软更加"明智"的选择了从开发人员入手，引导开发人员进入 SOA，从 MBF(Microsoft Business Framework)来看，就是提供给开发人员的参考架构。

Oracle：Oracle 的服务导向架构套件（Oracle SOA Suite）已全面上市。这是一组全面的、基于标准的中间件产品，融合了易构建、易配置、易管理且服务导向架构的多个特性。该套件具备的"插座式"能力使客户在利用现有中间件技术的同时可充分享受到服务导向架构的种种益处。

Red Hat：早在 2005 年，JBOSS 通过一系列的收购(收购 Hibernate , Drools，Arjuna)和整合推出 JEMS（JBoss Enterprise Middleware Suite）产品结构，以支持 SOA。Red Hat 在收购 JBOSS 后，立即宣布将与 2007 年九月推出 JBoss Enterprise SOA Platform 的产品组合包，包含基本的 JBoss AS 与 Seam、Hibernate 等，以实现将操作系统与全套的 SOA 解决方案相整合，为企业级计算提供一个

更灵活、成本更低的基础平台。

与国外 IT 业相比，中国企业的 IT 系统积累时间短，复杂度不是特别高，很多情况下并不一定用 SOA 来解决问题。国内虽然也已开始研究 SOA 并进行应用，但大多数经验都来自欧美，国内还缺乏 SOA 的成功样例。

SOA 本身只是如何将软件组织在一起的抽象概念，仅定义了服务如何相互理解以及如何交互，并没有确切地定义服务具体如何实现和交互，其依赖于用 XML 和 Web 服务等技术实现，并与具体应用相结合以软件的形式存在的更加具体的概念和技术[3]。

1.3 课题来源与主要研究内容

本课题来源于远程开放教育信息系统和中国贸易业绩评价体系。远程开放教育信息系统由很多信息系统组成，如电大在线，电大教务管理系统，多媒体教学平台，这些系统是按照业务发展的需要独立开发的，从而形成了多个信息孤岛。随着国家现代远程教育工程的实施和电大系统开放教育的深入开展，对远程开放教育提出了新的更高的要求。不仅要求每个电大单位都能独自完成信息化建设，更要从整个远程开放教育体系来整理和整合业务。为了满足远程开放教育发展的需要，必须依据 SOA 策略重新对远程开放教育信息系统进行规划。

中国贸易业绩评价体系将数据仓库技术应用于外贸业务分析和决策，对世界贸易组织及中国海关的海量外贸数据进行分析和处理[4]。原始外贸数据以多种文件格式存放，国内交易数据存放于 FoxPro 数据库，世界各国交易数据存放于其特定的 PC—TAS 系统数据库，人口数据、国内生产总值数据存放于 EXCEL 文件中。在构建数据仓库前，需要根据数据仓库中按主题组织的数据模型将这些异构数据源中的非标准数据进行转换。

本论文把面向服务的思想引入数据模型的建立，借鉴数据仓库按主题组织数据的思想，把数据模型分为数据实体服务层、服务聚合层、跨组织聚合层三个层次描述，通过数据实体、数据服务对象等对数据模型进行逐层抽象和逐层组合，建立从组织内数据实体视图到虚拟化的、面向用户的、统一的跨组织数据视图，研究提出一个解决企业级数据集成和共享的方案。

在项目和论文中要完成的主要内容：

- 建立统一的综合数据模型，为所有信息源的数据提供一个统一视图。
- 把数据模型进行转换和映射，创建数据服务。
- 应用前面所述的方法建立中央电大教务管理系统的数据模型。
- 提出中央电大教务管理系统数据服务的实现框架，并对关键技术和解

决方案进行分析。

1.4 论文的结构与内容

本文将从以下几个部分来阐述：

第一部分：（第 1 章）介绍与本课题相关的背景知识，对国内外相关领域的研究和进展进行综述。介绍本课题的来源及主要研究内容。

第二部分：（第 2 章）介绍系统相关的关键技术。首先是对数据模型的概念、分类，以及几种主要的几种数据模型进行了详细介绍，然后重点对 SOA 的架构模型、支撑技术和相关的标准进行了研究和分析，讨论了 SOA 技术架构在企业应用集成中的作用、地位和优势。

第三部分：（第 3 章）本章详细介绍了基于 SOA 建立分布式应用系统数据模型的方法，把数据模型分为数据实体服务层、服务聚合层、跨组织聚合层三个层次描述，利用传统的 E-R 分析方法并借鉴数据仓库按主题存储数据模型的结构建立业务数据的主题域数据模型；使用面向对象、面向组件和面向服务的建模方法，进行逐层抽象、组合和映射，最终建立应用系统的服务数据模型。

第四部分：（第 4 章）本章讨论了基于 SOA 的分布式应用系统数据模型技术支撑平台的分析与实践，研究建立基于 SOA 的中央电大教务管理系统的数据模型，分析了数据服务模型的实现框架，阐述了如何利用现有技术来实现数据服务。最后，讨论了实现一个分布式应用系统数据服务的一些技术细节和实际应用实例。

第 2 章 关键技术和理论

自从开始构建分布式应用系统开始，应用程序设计者就开始关注数据模型的建立。最早的做法都是应用传统信息系统据模型的建模，加上一些分布式因素；或者采用工作流的方法去建立，针对某一个业务设计数据模型，最后把所有业务的数据模型设计完成，组成完整的系统的数据模型。目前，随着面向服务的体系结构(Service-oriented Architecture,SOA)的在分布式应用系统中的应用，需要突破传统的像“烟囱”一样的数据模型，从所在领域出发，建立扁平化的全局数据模型。

2.1 面向服务的体系架构

在 1996 年，Gartner Group 就提出了 SOA 的概念，但是那个时候软件技术还不足以从真正意义上支持企业实现 SOA。随着网络技术的不断发展，在 2002 年 12 月，Gartner 再次提出了 SOA 是“现代应用开发领域最重要的课题”。

2.1.1 SOA 概述

面向服务的体系结构(SOA)是一个组件模型，它将应用程序的不同功能单元(称为服务)通过这些服务之间定义良好的接口和契约联系起来。接口是采用中立的方式进行定义的，它应该独立于实现服务的硬件平台、操作系统和编程语言[4]。这使得构建在各种这样的系统中的服务可以以一种统一和通用的方式进行交互。

这种具有中立的接口定义(没有强制绑定到特定的实现上)的特征称为服务之间的松耦合。松耦合系统的好处有两点，一点是它的灵活性，另一点是，当组成整个应用程序的每个服务的内部结构和实现逐渐地发生改变时，它能够继续存在[5]。相反，紧耦合意味着应用程序的不同组件之间的接口与其功能是紧密相连的，因而当需要对部分或整个应用程序进行某种形式的更改时，它们就显得非常脆弱。

作为一种新型的软件开发体系结构，SOA 具有无可比拟的优势，符合现在软件开发的发展潮流，尤其适用于企业级的应用开发。随着互联网络的进一步发展和分布式应用的不断普及，SOA 的应用会更加的普遍并被人们所接受，成

为继面向对象、面向组件之后新的设计方式。

2.1.2 SOA 的基本架构和操作

SOA 的基本结构如图 2.1 所示。该结构描述了面向服务的体系结构的角色、操作和基本构件[6]。

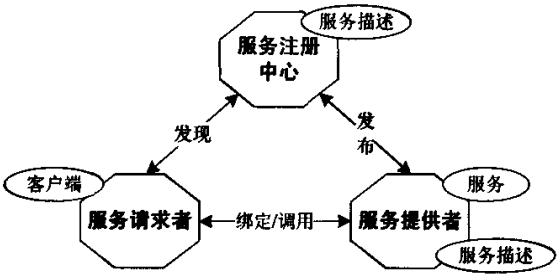

图 2-1 SOA 体系结构图

Figure 2-1 SOA Architecture

面向服务的体系结构中的角色包括：

- 服务使用者：服务使用者是一个应用程序、一个软件模块或需要一个服务的另一个服务。它发起对注册中心中的服务的查询，通过传输绑定服务，并且执行服务功能。服务使用者根据接口契约来执行服务。
- 服务提供者：服务提供者是一个可通过网络寻址的实体，它将自己的服务和接口契约发布到服务注册中心，以便服务使用者可以发现和访问该服务。
- 服务注册中心：服务注册中心是服务发现的支持者。它包含一个可用服务的存储库，并允许感兴趣的服务使用者查找服务提供者接口。

面向服务的体系结构中的每个实体都扮演着服务提供者、使用者和注册中心这三种角色中的某一种(或多种)。面向服务的体系结构中的操作包括：

- 发布：为了使服务可访问，需要发布服务描述以使服务使用者可以发现和调用它。
- 发现：服务请求者定位服务，方法是查询服务注册中心来找到满足其标准的服务。
- 绑定和调用：在检索完服务描述之后，服务使用者继续根据服务描述中的信息来调用服务。

面向服务的体系结构中的构件包括：

- 服务：可以通过已发布接口使用服务，并且允许服务使用者调用服务。
- 服务描述：服务描述指定服务使用者与服务提供者交互的方式。它指定来自服务的请求和响应的格式。服务描述可以指定一组前提条件、

后置条件或服务质量(Quality of Service, QoS)级别[7]。

采用面向服务的体系结构设计分布式应用系统，可以通过发布的可发现的接口为其他的应用程序提供服务，可以降低升发成木、持续集成和降低风险。当用 Web 服务技术来实现面向服务的体系结构，更是一种构建应用程序的强大而灵活的的方式。

想成功地创建 SOA，需要的不仅仅是好的工具和标准，其中非常重要的过程就是面向服务的数据建模，这需要一些规范的步骤来支持。对于任何对企业应用程序开发感兴趣的人来讲，理解面向服务的数据建模非常重要的。

2.2 数据模型

从建模的观点来看，SOA 带来的挑战是如何设计并描述良好的数据模型以及如何系统地实现它们。首先，这需要一些数据模型方面技术和理论的支持。

在传统软件系统的开发中，数据模型是描述数据内容和数据之间联系的方法，是一组描述数据库的概念。这些概念精确地描述数据、数据之间的关系、数据的语义和完整性约束，很多数据模型还包括一个操作集合。数据模型应满足三方面要求：一是能真实地模拟现实世界；二是容易为人们理解；三是便于在计算机上实现。企业应用系统设计的核心问题之一就是设计一个好的数据模型，SOA 也不例外。

2.2.1 数据模型概念

数据是现实世界中符号的抽象，而数据模型(data model）则是数据特征的抽象[8]。数据模型描述数据的结构、定义在其上的操作以及约束条件。它从抽象层次上描述了系统的静态特征、动态行为和约束条件，为系统的信息表示和操作提供一个抽象框架。

数据模型按应用层次分成三种类型，它们是：概念数据模型（Conceptual Data Model)、逻辑数据模型（Logic Data Model)、物理数据模型（Physical Data Model)。

概念数据模型又称概念模型，它是一种面向客观世界、面向用户的模型，它与具体的信息系统无关，与具体的计算机平台无关。概念模型着重于对客观世界复杂事物结构的描述及事物间的内在联系的刻画，而将与 DBMS 和计算机有关的物理的、细节的描述留给其它层次的模型。因此，概念模型是整个数据模型的基础[9]。目前，较为有名的概念模型有 E-R 模型、扩充的 E-R 模型、面向对象模型等。

逻辑数据模型又称逻辑模型，它是一种面向数据库系统的模型，该模型着

重于在数据库系统一级的实现。它是客观世界到计算机之间的中介模型，具有承上启下的功能。概念模型只有在转换成逻辑模型后才能在数据库中得以表示。

物理数据模型又称物理模型，它是一种面向计算机物理表示的模型，此模型给出了数据模型在计算机上物理结构的表示。

数据模型所描述的内容有三个部分，它们是数据结构、数据操作与数据约束[10]。

- 数据结构。数据模型中的数据结构主要描述数据类型、性质以及数据间的联系。数据结构是数据模型的基础，数据操作与约束均建立在数据结构之上。不同数据结构有不同的操作与约束。因此，一般数据模型均依据数据结构的不同而分类。
- 数据操作。数据模型中的数据操作主要描述在相应数据结构上的操作类型与操作方式。
- 数据约束。数据模型中的数据约束主要描述数据结构内数据间的语法、语义联系，它们间的制约与依存关系，以及数据动态变化的规则，以保证数据的正确、有效与相容。

数据模型可以将复杂的现实世界要求反映到计算机数据库中的物理世界，这种反映是一个逐步转化的过程，它分为四个阶段，被称为四个世界。由现实世界开始，经历概念世界、信息世界而至计算机世界，从而完成整个转化。

那么，数据模型和这四个世界是什么样的关系呢？现实世界就是具体的应用，与数据模型是没有关系的；概念世界对应了概念数据模型；逻辑世界对应了逻辑数据模型；计算机世界对应了物理数据模型。表 2-1 给出了概念数据模型和逻辑数据模型的分类以及它们的对应关系。

表 2-1 概念模型和逻辑模型对应关系图

Table 2-1 Concept Model and Logic Model Relationship

概念模型	E-R 模型	对象关系模型	面向对象模型
逻辑模型	关系模型	对象关系模型	面向对象模型

2.2.2 E-R 模型

E-R 模型（Entity-Relationship Model）又称实体关系模型，它于 1976 年由 Peter Chen 首先提出，这是一种概念化的模型，它将现实世界的要求转化成实体、关系、属性等几个基本概念以及它们间的两种基本关系，并且用一种较为简单的图即 E-R 图(Entity-Relationship Diagram）表示[11]。

E-R 模型有如下三个基本概念：

实体（Entity）：现实世界中的事物可以抽象成为实体，实体是概念世界中的基本单位，它们是客观存在的且又能相互区别的事物。凡是有共性的实体可组成一个集合称为实体集（Entity Set）。

属性（Attribute）：现实世界中事物均有一些特性，这些特性可以用属性这个概念表示。属性刻画了实体的特征。属性一般由属性名、属性型和属性值组成，其中属性名是属性标识，属性的型与值则给出了属性的类型与取值。一个属性有一定的取值范围，称为属性的值域（Value Domain）或称为值集（Value Set）。

关系（Relationship）：现实世界中事物间的关联称为关系。在概念世界中联系反映了实体集间的一定关系。就联系所包含的实体集的个数而言，关系可分为以下几种：两个实体集间的关系、多个实体集间的关系、一个实体集内部的关系。

在建立了实体集和实体集间的关系以后，就可以从概念模型出发建立逻辑模型了。与 E-R 模型对应的逻辑模型是关系模型。

关系模型的基本思想是用二维表形式表示实体及其联系。二维表中的每一列对应实体的一个属性，其中给出相应的属性值，每一行形成一个由多种属性组成的多元组，或称元组，与一特定实体相对应。

关系数据库结构的最大优点是它的结构特别灵活，可满足所有用布尔逻辑运算和数学运算规则形成的查询要求；在关系数据库中还能搜索、组合和比较不同类型的数据，加入和删除数据都非常方便。

2.2.3 面向对象模型

面向对象模型（Object-Oriented Model）是在吸收了以前各种概念模型优点的基础上并借鉴了面向对象的设计方法而建立的模型。这种模型具有更强的表示能力。

在面向对象模型中的最基本的概念是对象（Object），它是客观世界中概念化的基本实体，它可以是简单的实体也可以是复杂的实体；它可以是具体的实体，也可以是抽象的实体。总之，相互能区别的事物均可视为对象。

- 对象有如下特点：对象的封装（Encapsulate）：对象的属性与方法是封装在一起的，亦即是说，操作与属性是相互依存的。
- 对象标识符的独立性：每个对象均有一个独立于属性以外的名字，亦即是说，对象的属性值与对象名是有区别的。在 E-R 模型中每个实体的名是由一个属性值集表示的，这与面向对象模型截然不同。

- 对象属性值的多值性：一个对象的属性，可以是单值也可以是多值。如有学生修读课程这一属性可为多值，因为学生一般修读课程可超过一门，这就是对象属性值的多值性。

为便于概念上认识的方便，一般将具有相同属性、方法的对象集合在一起称为类（Class），而此时类中对象称为实例（Instance）。

类与类之间存在着某些联系，在面向对象模型中，这些联系一般被分为两种：子类(Sub-Class）与超类（Super-Class)，类的子集也可以是一个类，称为该类的子类，而原来的类，则称为子类的超类，子类继承（Inheritance）超类的属性与方法，并具有自己的属性与方法；类的聚合（Composition）与分解（Decomposition)，在现实世界中往往存在着由简单的对象组合成复杂的对象的现象，在面向对象模型中即是类的聚合。类的聚合即表示若干个简单类可以聚合成一个复杂类。

面向对象模型能描述复杂的现实世界，具有较强灵活性、可扩充性与可重用性。它的动态特性描述、对象标识符、类的泛化/特化、类的聚合/分解以及消息功能等都比前面所介绍的模型要好。此模型既是一种概念模型，同时还可以依据它直接构造逻辑模型，称为面向对象的逻辑模型。

2.3 XML 和 Web Services

SOA 很早就已经出现了，而在软件开发中真正应用 SOA 却是从 XML 和 Web Service 技术的成熟和广泛应用开始的。

2.3.1 XML

2.3.1.1 XML 概述

XML(Extensible Markup Language，可扩展标记语言)是一种新的标记语言。同HTML一样是特别为Web应用服务的标准通用标记语言(Standard Generalized Markup Language，SGML)的一个简化子集，用于支持 Internet 上结构文档的交换[12]。XML 因具有更多的语义、良好的可扩展性、简单易用、自描述等特点而特别适用于 Web 上的数据交换，自万维网联盟(W3C)于 1998 推出 XML1. 0 规范后，已经得到行业的广泛支持和应用，成为数据组织和交换的事实标准之一。

XML 具有如下重要特性。

- 扩展性。XML 是一种用于设计标记语言的元语言，而不是象 HTML 那样，是一种只有一个固定标记集的特定标记语言。XML 允许用户根据其需要创建自己的标记，这些标记可通过 XML DTD(Document Type

Definition)和 XML 模式加以定义。

- 灵活性。XML 提供了一种结构化的数据表现方式，从而使用户界面与结构化数据相分离。
- 自描述性。XML 文档通常包含一个文档类型(DTD)声明，从而便于机器理解数据的意义。XML 文档中的数据可被任何能够对 XML 数据进行解析的应用程序所提取、分析和处理，并以所需格式显示。

2.3.1.2 XML 相关技术

与 XML 相关的技术有很多，下面对主要的几种进行介绍。

(1) 文档类型定义(DTD)和 XML 模式

XML 文档可以事先定义其结构，在编制包含相似内容的文档时可以限定遵循的标准，这样编制出来的文档具有相同的结构，便于计算机统一处理。XML 文档结构定义有两种办法：DTD(Document Type Definition)和 XML 模式。

DTD 是一组能融合在 XML 数据里或者以单独的文档存在的声明，它定义一些规则来描述被允许的 XML 数据内容[13]。一个 DTD 只能和一个给定的 XML 文档或数据对象关联起来。DTD 的缺点是很难理解和数据类型相当有限，如在 DTD 中不提供如整数、浮点数、布尔数等数值数据类型，所有的文档内容都是字符数据。

XML 模式和 DTD 的功能一样，都是用来定义标记和标记的用法，但跟 DTD 相比，它至少有三个优点。首先，XML 模式文档是一种特殊的 XML 文档，它遵循 XML 的语法要求，避免读者再去学习一套语法的负担；其次，XML 模式的语法结构更简单，易于学习和使用，而且提供了丰富的数据类型，不但有整数、浮点数等常用的类型，还提供了自定义数据类型机制；最后，XML 模式在代码的重用性和可扩展性方面要远远优于 DTD。XML 模式则采用了名域空间的机制，使得一个 XML 文档可以调用多种模式文档。

XML 模式有很多种，其中 XML Schema 是 W3C 的推荐标准，于 2001 年 8 月正式发布，应用也最为广泛。

(2) DOM 与 SAX 编程接口

用于将 XML 解析绑定到需要它的应用程序，两个最常见引擎技术是 W3C 的文档对象模型(Document Object Model，DOM)和开放标准一用于 XML 的简单 API(Simple API for XML，SAX)o

DOM 是 XML 解析器的一种解析接口标准，是一种允许代码直接指向 XML 文档各部分属性的 API[14]。DOM API 是一个对象模型驱动的 API。实现了 DOM 的 XML 在内存中生成一个对象模型。实际上就是一个 XML 文档内容和结构的 DOM 树。DOM 一次将整个文档读入内存，然后向程序提供整个文档可操作的

DOM 树。DOM API 的核心是文档和节点界面。文档(Document)是表示 XML 文档的顶层对象。Document 用一颗节点树来保存数据，节点可以是代表元素、属性或者其他内容的基本类型。

SAX 同 DOM 一样也是一个访问 XML 文档的接口。SAX 的设计实现与 DOM 是完全不同，DOM 处理 XML 文档是基于将 XML 文档解析成树状模型，调入内存进行处理。而 SAX 则是采用基于事件驱动的处理模式，它将 XML 文档转化成一系列的事件，由单独的事件处理器来决定如何处理。SAX 是把 XML 文档解析成事件流，所有没有被过滤的事件都会保留下来。

(3) XPath、XSLT 和 Xquery

XPath 用于在 XML 文档的层次树形结构中便利的路径表示法，它的主要功能有：提供紧凑的非 XML 语法来引用 XML 文档内的一个或一组节点；为编程处理 XML 文档提供标准的字符串、数字和布尔表达式处理的函数库；作为一种有效的语言工具，可以引用被任何标记符号分割的 XML 文档内容；和其它 XML 编程接口或处理模型(如 XSLT 等)紧密结合。

XSLT (Extensible Stylesheet Language Transformations)是用于 XML 数据转换和显示的编程语言[15]。它支持一个小的灵活类型集合：布尔值、数字、字符串和外部对象。还包括对数据的操作集合如<xsl:template>, <xsl:apply-templates>, <xsl:sort>等和流程控制语句如<xsl:if>, <xsl:for-each>等。XSLT 集成了 XPath，通过 XPath 定位和引用 XML 文档中的节点和节点集，然后对这些节点或者节点集进行按照转换要求进行编程处理，得到需要的输出结果。

XQuery 是用于定位和过滤 XML 文档中的元素、属性和文本的查询语言，它把数据库查询处理带给了 XML。

2.3.2 Web Services

2.3.2.1 Web Services 概述

根据 W3C 关于 Web 服务的定义，Web 服务是用于支持网络上机器与机器间互操作的软件系统。包含机器可处理的格式描述的接口(特别的如 WSDL)。其他系统与 Web 服务的交互，采用 SOAP 消息描述，HTTP 以及与 XML 序列化相关的其他 Web 标准进行传输[16]。

Web 服务可以从多个角度来看。从技术方面，一个 Web 服务是可以被 URI 识别的应用软件，其发布和绑定由 XML 描述和发现，并可与其他基于 XML 消息的应用程序交互。从功能角度看，Web 服务是一种新型的 Web 应用程序，具有自包含、自描述以及模块化的特点，可以通过 Web 发布、查找和调用。其实现的功能可以是响应客户一个简单的请求，也可以是完成一个复杂的业务流程。

具体而言 Web 服务应具有如下特点：

- 可描述，可通过服务描述语言来描述；
- 可发现，可在注册中心注册描述信息并发布；
- 可查找，通过向注册服务器发送请求可以找到满足查询条件的服务，获取服务的绑定信息；
- 可绑定，通过服务的描述信息生成可调用的服务实例或服务代理；
- 可调用，使用服务描述信息中的绑定细节可以实现服务的远程调用；
- 可组合，可与其他服务组合形成新的服务[17]。

尽管对 Web 服务进行描述的出发点或应用类型不同，但它们均具有如下共同特征：

- 应用的分布式：为适应网络应用中分布式的数据源和服务提供者，分布式的服务响应、松散耦合是 Web 服务必须具备的特征。在应用中，服务请求者不必关心服务提供者的数据源格式是什么，某一服务请求需要调用哪些服务，服务请求在 Web 上怎样被执行等，即 Web 服务对用户具有分布透明性。
- 应用到应用的交互：在分布式环境中，若采用集中控制方式，服务器有较大的负荷，并且系统不具有健壮性，因此应用到应用的交互使得 Web 服务更具有可伸缩性。
- 平台无关性：Web 服务的界面、跨 Web 服务的事务、工作流、消息认证、安全机制均采用规范的协议和约定；由于 Web 服务采用简单、易理解的标准，可屏蔽不同软件平台的差异，因此具有可集成能力。

2.3.2.2 Web Services 的相关技术标准

为了完成在松散耦合环境下的对象访问，以及在基本对象访问之上的事务、工作流、安全机制等，需要有一系列的协议规范来支撑。

(1) 服务通信协议 SOAP

SOAP (Simple Object Access Protocol)，简单对象访问协议)是一种利用 XML 技术在分散型、分布式环境中交换结构化信息的轻量级协议。它提供了一种可通过多种底层协议在对象间进行信息交换的消息结构[18]。简单地说，SOAP 是一种类似于 CORBA 的 llOP，DCOM 的 ORPC 或 Java 远程方法调用的 Java 远程方法协议(JavaRemote Method Protocol } JRMP)

SOAP 是在 XML 基础上定义的，它完全继承了 XML 的开放性和描述可扩展性。SOAP 使用现有基于 TCP/IP 的应用层协议 HTTP、 SMTP、 POP3 等，可以获得与现有通信技术最大程度地兼容。SOAP 的消息路径机制和可扩充的 Header 和 Body 机制又为分布式计算提供了很好的支持。

SOAP 协议主要由三部分组成：

- SOAP 信封（envelop），它构造定义了一个整体的 SOAP 消息表示框架，可用于表示消息中的内容是什么，是谁发送的，谁应当接受并处理它，以及这些处理操作是可选的还是必须的等。
- SOAP 编码规则(encoding rules)，定义了一个数据的编码机制，通过这样一个编码机制来定义应用程序中需要使用的数据类型，并可用于交换由这些应用程序定义的数据类型所衍生的实例。例如可能应订单服务的需要，使用 SOAP 编码规则定义了订单的数据类型，并可以在订单生成的客户端与订单服务之间交换订单实例。
- SOAP RPC 表示(RPC representation)，定义了一个用于表示远端过程调用和响应的约定，例如如何使用 HTTP 或 SMTP 协议与 SOAP 绑定，如何传输过程调用，在具体传输协议的哪个部分传输过程响应[19]。

(2) WSDL

WSDL (Web Services Description Language)通过定义一套基于 XML 的语法，将 Web 服务描述为能够进行消息交换的通讯端点集合(communication collection)，或者说是端口的集合[20]。WSDL 对操作和消息的描述是抽象性的，并在定义端点时，将消息和操作绑定到具体的网络协议和消息格式上。WSDL 是可扩展的，它允许对端点和端点间的消息进行描述，同时不去考虑具体的消息格式或者双方用于通讯的网络协议。

WSDL 服务描述是一个 XML 文档，它与 WSDL 模式(schema)的定义一致。WSDL 保持协议中立，但它确实内建了绑定 SOAP 的支持，从而同 SOAP 建立了不可分割的联系。本质上，WSDL 描述说明的是 Web 服务的以下三个基本属性：

- 服务做些什么—服务所提供的操作(方法)。
- 如何访问服务—数据格式详情以及访问服务操作的必要协议。
- 服务位于何处—由特定协议决定的网络地址，如 URL[21]。

(3) UDDI

UDDI 是指通用描述、发现和集成(Universal Description, Discovery and Integration, UDDI)规范。它的核心是 UDDI 企业注册，是一个全球性的公共在线目录，为企业提供了一种统一的方式来描述他们的服务、查找来自其它公司的服务以及详细了解如何实现服务的软件连接和交互[22]。服从 UDDI 的注册表为描述 Web 实体或企业服务提供了信息框架等。服务提供者通过 UDDI 协议注册其已有服务。潜在的用户依照 UDDI 规范像 UDDI 注册表搜索合适的 Web Service，然后得到相应的 WSDL 文档并进行调用。

UDDI 定义了一套统一的 XML 格式，以描述企业与其提供的 Web Service 的信息。这些信息数据使用标准的分类法进行分类，因此可以按分类来查询信

息[23]。最重要的是，UDDI 包含了有关网络服务的技术接口的信息。这些信息分为三个部分：

- White Pages：包括了地址，联系方法，和已知的标识；
- Yellow Pages：包括了基于分类学的工业划分；
- Green Pages：包括了关于该企业提供的 Web Service 的技术信息，其中还包含了指向特定的 Web Service 的链接或是 URL 等[24]。

通过这样一个信息结构，企业可以详细地将自身与提供的 Web Service 的信息描述清楚，并让“发现“它们的人能清楚的知道，这些是否是他们所需要的，如果是，那么如何去调用它们。同时，它实现了一组使企业能将自己提供的 Web 服务登记并让别的企业用户能够查询并访问到的标准。

2.3.2.3 Web Services 和 SOA 的关系

SOA 很早就已经出现了，而在软件开发中真正应用 SOA 却是从 Web Service 技术的成熟和广泛应用开始的，很长时间，很多人都将这两者混淆不清，认为二者是等同的。严格来说，SOA 和 Web Service 并不完全相同，但确实存在着非常紧密的关系。

(1) Web 服务是适合执行面向服务体系结构(SOA)的一种技术。

本质上，Web 服务是自描述和模块化的应用，将业务逻辑公开为可以在 Internet 上发布、发现和调用的服务。基于 XML 标准，Web 服务可以开发成松散耦合的应用组件，可以采用任何编程语言、任何协议、或任意平台。满足业务应用作为服务传递给任何人、在任何时间、任何位置、通过任意平台调用。Web 服务的出现产生了根本的改变，因为很多 Web 服务项目的成功显示这种技术事实上确实存在，借此可以实现真正的面向服务的体系结构[25]。

(2) Web 服务不等同于面向服务的体系结构。

Web 服务是包括 XML、SOAP 、WSDL 和 UDDI 在内的技术的集合，它使您能够针对特定的消息传递和应用程序集成问题构建编程解决方案。随着时间的推移，有理由相信这些技术将逐渐成熟并最终为更好、更有效、更健壮的技术所取代，但是，就目前的情况而言，它们可以发挥作用。至少，它们是 SOA 能够最终实现这种观念的证明。应该指出的是 Web 服务不是用于实现 SOA 的唯一技术，有许多组织成功应用其他技术实现 SOA 的例子，Web 服务也可以用于实现其他非面向服务的体系结构。服务可以用任何一种技术实现：JMS、WebService、RPC、或是简单方法调用，但却不应该将一个服务和具体的技术绑定起来[26]。

(3) 面向服务体系结构不只是 Web 服务

SOA 只不过是一种体系结构，也不是任何诸如 Web 服务这样的特定技术的集合；而是超越它们的，在理想的情况下，是完全独立于它们的。在业务环境

中，SOA 的纯粹的体系结构定义可能会是这样“SOA 是一种应用程序体系结构，在这种体系结构中，所有功能都定义为独立的服务，这些服务带有定义明确的可调用接口，可以以定义好的顺序调用这些服务来形成业务流程“。

2.4 本章小结

本章首先对 SOA 进行简单的介绍，包括其概念，基本结构和操作，然后讨论了目前主要的的数据模型和数据建模理论，最后介绍 SOA 实现技术 XML 和 Web Services 的特点和相关技术标准。这些方法和技术，都是构建基于 SOA 分布式应用系统的基础。

第 3 章　基于 SOA 的分布式应用系统数据模型的建立

3.1　基于 SOA 的分布式应用系统参考架构

分布式系统的特点是运作的敏捷性、信息需求的多样性、信息系统的异构性以及良好的扩展性。在建立分布式应用系统软件体系结构的时候，必须提高其对业务变化的适应性，可扩展性，可配置型，可维护性，才能够适应企业机构调整和业务需求的不断变化。

为了保证软件质量，提高软件可靠性、可重用性和可维护性，软件开发越来越注重系统的总体结构即软件体系结构的设计和规范。通过建立软件体系结构，合理划分子系统和模块，可以清晰地勾画软件系统的组织结构，用基于组件开发和组装的方式来分解复杂问题[27]。

根据 SOA 技术架构面向服务的思想，针对分布式应用系统的特点，各大 IT 巨头厂商都提供了相应的参考架构，图 3-1 所示是一个典型的基于 SOA 的分布式应用系统参考架构。

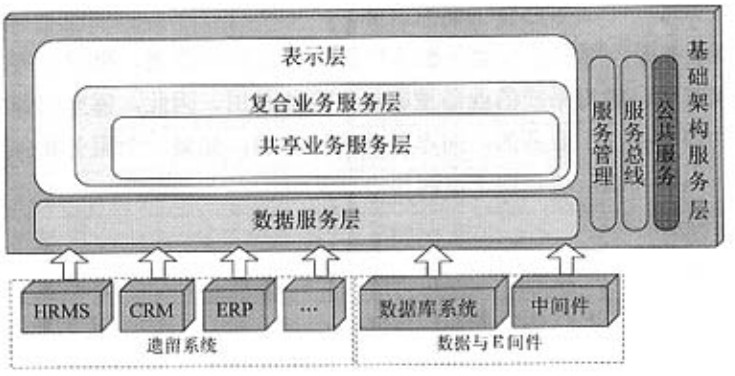

图 3-1 基于 SOA 的典型分布式应用系统参考架构

Figure 3-1 Service-oriented Distributed system reference architecture

基于 SOA 的分布式应用系统参考架构从一个较高层次来描述开发人员如何开发基于 SOA 的分布式应用系统。它分为数据服务层、业务服务层、表示层和复合应用层，并由基础服务架构层支撑系统的运行。

- 数据服务层

任何系统，数据都是核心，构建分布式应用系统，其中最基础的工作就是进行数据的集成和重构，在基于 SOA 的分布式应用系统参考架构中，最基础、

最底层的就是数据服务层。

数据服务层，建立统一的综合的数据模型，为整个信息数据提供一个统一的数据视图，在全企业范围内得到一致性的使用；隔离应用与底层数据源，以标准存取方式提供服务给其它层服务或用户调用，使得应用界面与各数据源是松耦合的；完成企业数据服务的定制和封装。数据服务层的关键是封装（资源接入接口）、数据集成（数据的完整视图获取）和标准（应用标准的结构规范）

通常情况下，以适当颗粒度的组织的上层服务能够通过一定的接口合约对这层的服务进行访问。

- 共享业务服务层

共享业务服务层的主要职责在于根据业务逻辑，对核心业务进行梳理、整合和划分；为上层应用提供相对独立的业务功能服务；从业务活动剥离、抽象可共享的、基于标准的服务，以及这些核心业务的验证和对于事务的控制。共享业务服务层是系统的业务逻辑核心处理区，在一个典型的应用系统开发中，它处于中间层次，利用和构建于数据服务层之上。

- 复合业务服务层

利用企业已经存在的服务组件，通过组合与协调下层服务组件使得更容易组合成复合应用；服务是粗颗粒度的，面向业务支撑的；服务实现高层的、多步骤业务流程逻辑，而这些业务逻辑通过组合其他服务层提供的服务而成；其表现形式就像一个服务组合了其他服务。

共享业务服务层和复合业务服务层的主要目的是协调、组织业务，为上层应用提供基础，使得系统的业务逻辑组件更加易用。因此，需要强调的是这两层提供的服务是基于业务的，而不是基于技术的；如果一个服务面向业务或者面向流程，它一般应该存在于这两层之一。

- 表示层

随着现代计算的普适性，计算无处不在，那么表示就无处不在，所以有必要为一个系统提供多种表示方式。

表示层是集成框架的用户接口部分，是用户与集成框架应用接口层交互信息的窗口。表示层的职责是处理用户的请求/响应过程，即表示层接受用户输入，把用户请求移交给业务层的代理，然后接受业务处理结果并显示给用户，其间还包括对用户输入进行有效性检查，对后端层次抛出的异常做友好处理等任务。复合应用层通过表示层向用户暴露其所提供的功能，用户可以通过表示层发起一个整合流程。现实中表示层可以是一个浏览器，也可以是个瘦客户端，也可以是一个手机。

一个设计良好的表示层至关重要，而现在普遍存在的问题是开发人员容易把大量本属于业务层的逻辑处理放到表示层来实现。这带来了一时的便利性，

但也带来了潜伏性问题，给系统的后期维护、扩展带来了巨大的伤害。因此，必须把表示层的职责严格限制在使其提供标准化、个性化的信息和功能展现方式范围之内。

- 基础架构服务层

基础架构服务层的功能主要包括三个方面。公共服务：服务认证、身份验证、授权和访问控制、日志管理和异常处理等。服务总线：提供服务交互所需的消息传输、翻译、转换和消息路由，异步或同步传输模式的管理。服务管理：提供所有 SOA 参与者的管理能力，对服务进行集中管理和监控，包括服务的目录、版本、配置等。

基于 SOA 的分布式应用系统参考架构是以数据服务层为基础，以业务服务层为驱动的，以表示层为窗口，使用复合应用层灵活组合、调用服务来满足企业应用。

从上面基于 SOA 的分布式应用系统参考架构可以看出，它相对现在通用的软件架构有很多不同，对此我们作一简单比较，结果如表 3-1 所示：

表 3-1 现有架构和面向服务架构

Table 3-1 General system architecture and SOA

通用架构	面向服务架构
面向功能性	面向业务流程
为最终功能而设计	为方便更改而设计
长开发周期	交互式和重用式开发
成本为中心	业务为中心
应用包	服务编排
紧耦合	松耦合的、敏捷的和可适应的
同构技术	异构技术
面向对象	面向消息
明确的实现方式	抽象的实现方式

3.2　数据服务层

在分布式企业应用系统中异构数据的有其自己的特点，主要表现为分布性、自治性和异构性。如何集成这些异构数据，使得既实现数据透明性和操作透明性，又使各个源数据系统仍保有自己的应用特性、完整性控制和安全性控制。这些问题一直是研究的热点。

目前，对分布式异构数据的集成主要有以下几种技术：

(1) 数据仓库

在数据仓库集成结构中，来自几个数据源的数据被抽取出来，合成一个全局模式，然后数据存储在数据仓库中，这在用户看来与普通数据库无异[28]。组织方式如图 3-2：

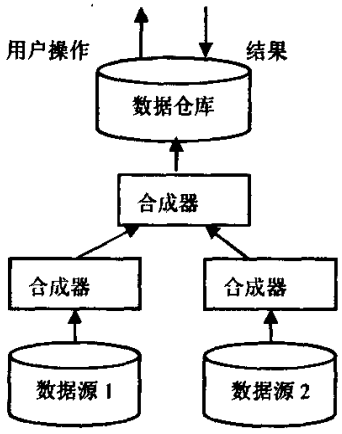

图 3-2 数据仓库集成结构图

Figure 3-2 Data warehouse Integration Architecture

(2) 联邦式数据库系统

各个异构的局部数据库之间仅存在着松散的联邦式耦合关系，没有全局统一模式，各局部数据库通过定义输入、输出模式进行彼此之间的数据访问[29]。在某些情况下，联邦系统可能是最容易建立的，特别是在各个系统之间的通信本来就有限时。组织方式如图 3-3：

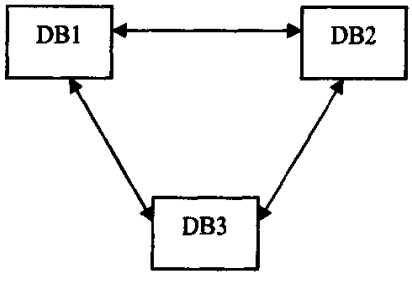

图 3-3 联邦数据库结构图

Figure 3-3 Federation Integration Architecture

(3) 协调器

协调器支持虚拟视图或视图集合，它不存储任何数据。当用户向协调器发出操作请求时，协调器向对应的包装器或适配器发送消息，包装器对数据源进行操作，返回结果给协调器，协调器再返回用户[30]。在协调系统中允许用户象使用一个集中的数据库一样透明地访问各异构的、自治的、分布的数据库，组织方式如图 3-4。

在这三种方案中，应用最广泛的就是协调器系统，其特点是：对原有应用

系统不必加以改造，各应用系统可在原模式下运行，集成起来比较灵活，易扩展，更新组件数据库时非常灵活。

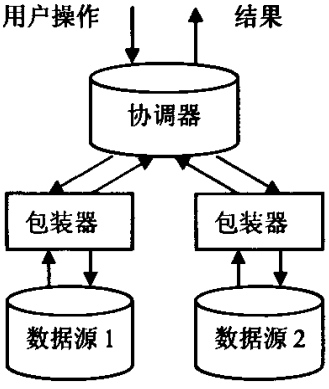

图 3-4　协调器结构图

Figure 3-4 Mediation Integration Architecture

把面向服务的思想引入分布式应用系统的开发中，就是将数据模型、业务策略和服务定义结合起来，以建立统一数据模型为中心，在系统的分层体系结构中抽象一个数据服务层，专门用来解决 SOA 的数据服务要求[31]。数据服务层主要采用协调器集成方式，通过建立统一的综合数据模型，实现一个虚拟数据库系统，提供业务数据的完整视图，从而隔离应用与底层数据源，并通过统一的 API 存取所有数据源。如图 3-5 所示。

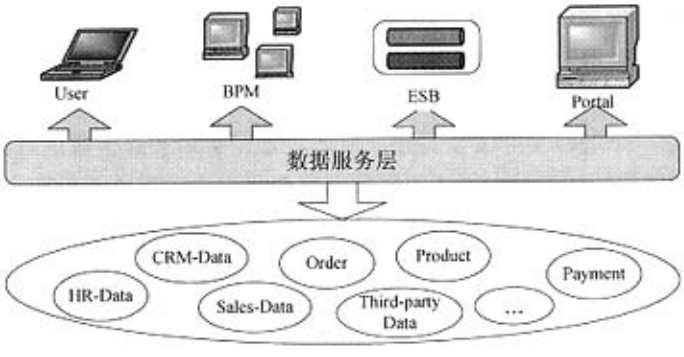

图 3-5 数据服务层示意图

Figure 3-5 Data Service Layer

数据服务层的工作原理是：先构造中心虚拟数据库，通过数据映射模块，将对中心虚拟数据的构造与操作映射为对各应用系统的分布计算，由应用系统的分布计算模块向应用系统集成框架发送数据，而由数据组装模块将分组数据“组装“至虚拟数据库。其工作方式采用“拉”的工作方式，由应用系统集成框架将各应用系统的数据“拉”至中心虚拟数据库。

在基于 SOA 的分布式应用系统参考架构中，数据服务层是最基础和最关键

的一层，它的主要功能有：

- 全局数据模式(Global Data Model)：建立能够描述各局部成员数据库的全局数据模式，并建立各局部源数据类型及计量单位间的对应转换关系，以便向用户提供统一的数据视图，从而使用户感到如同访问一个单一的数据库，实现系统的透明性。
- 名字服务(Name Service)：为源数据提供统一的名字注册、地址查询、结点位置与配置信息等服务，这些信息都存放在一个公共服务的名字数据库中。
- 全局数据字典(Gloabal data Dictionary)：描述异构数据库系统中各局部成员数据库的数据库、表结构和属性，以及全局表对应的局部表名、局部表的所在结点位置等信息。这是实现全局查询的基础。
- 应用程序接口 API: 为应用层开发提供一组规范化的访问异构数据的功能调用接口。其中关键在于以现有的 SQL 标准为基础，定义支持异构数据库访问的联邦结构化查询语言。

3.3 数据模型的建立过程

企业应用系统地开发过程通常是从企业建模开始的。在企业模型中最重要的就是数据模型。数据模型是系统企业数据的统一表现实体，在全企业范围内得到一致性的使用；提供业务数据的完整视图和详细描述信息，完成企业数据服务的定制和封装[32]。目前企业建模和企业数据模型的概念正在为越来越多的人所接受。

要设计一个适合实际需要的数据模型，首先必须了解企业信息结构的特点和信息处理的方式。通过对多个企业的考察和对多个大型企业应用系统的分析，可以总结出一个企业信息结构的普遍规律：

- 组织关系明确：企业一般都有比较明确的组织关系。一般采用分层管理的金字塔型组织结构。也就是说，组织的局部与整体之间存在明确的隶属关系，而组织与组织之间一般不存在这样的关系。
- 访问控制严格：既然组织之间存在明确的隶属关系，那么不同组织掌握的数据实际上也就有了不同的属主。这个属主与具体的组织结构有关，而与数据库中数据对象的属主是没有关系的。也许多个组织掌握的数据在数据库的概念中应该是同一个属主，但是就数据的逻辑归属而言，可能应该是不同的属主。因此由于企业内部存在明确的组织分工和，严格的组织结构关系，所以数据的访问不能是无限制进行的，必须具有比较严格的访问控制。

- 业务关系紧密：企业的数据通常都是与其业务内容密切相关的，数据模型必须能够很好地反映企业的业务状况，能够对业务工作给予强大的支持。

通过上面的分析可以看出，如果用纯粹的关系数据模型或对象数据模型来表达和构造企业数据模型并不是一个很好的选择。于是我们基于 SOA 面向服务架构的思想提出主题域数据模型和服务数据模型两阶段构造方法来构造分布式应用系统数据模型。

建立分布式应用系统的数据模型，就是要突破传统的像“烟囱”一样的数据模型，在设计数据模型的时候，直接从所在领域出发，先建立面向主题的主体域 E-R 模型，即领域内扁平化的数据模型，这种数据模型并不依赖某个具体业务，可以在领域内通用。然后，在此础上再根据具体的业务流程，组合或封装数据实体，形成数据服务模型。

3.3.1　面向主题—主题域 E-R 实体数据模型

主题域 E-R 实体数据模型义了企业级的主题域、层面、主要的实体及实体间的业务关系，它是独立于数据库技术的，即在设计时不需要考虑使用什么样的数据库、字段类型、长度、索引等，也不需要考虑应用程序的性能等因素。当数据模型的主题涵盖企业的各个方面并提供了企业级的业务视图时（Enterprise-wide Business View），就成为企业主题域数据模型。企业主题域数据模型是一个关于整个组织业务信息的完整模型。

数据模型是把权限管理和数据归属等繁琐问题的处理从应用层下放到数据服务层来实现，为设计人员屏蔽数据的物理结构和逻辑结构之间的映射关系[33]。而主题域 E-R 实体数据模型所体现出来的数据逻辑结构，可以很好地反映分布式系统应用领域中的数据对象的组织结构，使程序员不再需要将他们从用户那里得到的应用需求翻译成关系型或对象型的数据模型。程序员可以很直接地用非常符合现实情况的主题域 E-R 实体数据模型来表达他们获取的需求，使得对数据的管理更加合理有效。主题域数据模型是结构化数据管理方式和基于主题归属关系的分类视图的结合。

(1) 主题域 E-R 实体数据模型的基本概念

主题域数据模型有如下 5 个基本概念：

- 实体：现实世界和信息世界的事物都可以抽象为实体，实体是概念世界的基本单位，它们是客观存在且又能相互区别的事物。具有共性的实体可以组成一个集合称为实体集。
- 属性：事物均具有一些特性，这些特性用属性这个概念来表示。属性的

取值是同类实体进行个体区分的依据。

- 联系：实体集间的一些固有关系被称为联系。
- 主题域：一种可以容纳不同种类实体的特殊的实体集合。
- 域关系：域和域成员之间的一种访问控制关系，可以简单理解为访问权限。

在上述的 5 个基本概念中，实体、属性、联系是来源于 E-R 模型的概念，其含义和用法与 E-R 模型并无不同，在此只举例说明主题域和域关系。

在电大教务管理系统中，所有与学生主题应用相关的实体组成的实体集合就可以组成学生主题域，同样，所有与课程主题应用相关的实体组成的实体集合组成课程主题域。因为学生选课这个业务活动，这两个主题域之间存在一种数据访问的关系，这就是一种域关系。

(2) 主题域 E-R 实体数据模型的建立

主题域 E-R 模型建立的主要过程包括两个步骤：第一步，借鉴数据仓库按主题建模的思想，使用自顶向下的分析过程，从整个系统的高度对企业业务数据进行分析，确定主题域，建立反映企业主题数据关系的高层次结构模型；第二步，用传统的 E-R 分析方法建立业务数据的实体关系模型，分部门、分组织找出分析并找出所有数据实体，然后按主题域再对实体进行综合的处理，使得数据实体仅包含最基本的业务属性，并消除冗余和重复。

第一步所要完成的工作有：

- 界定系统的边界：明确建模的目标和所涉及的数据，即把系统的一些基本的、方向性的数据需求以系统边界定义的形式表示出来。一般系统的需求分析中已有比较明确的描述。
- 确定主要的主题域及其内容。这一步中，要确定系统所包含的主题域，然后对每个主题域的内容进行比较明确的描述，例如描述主题域的核心实体、主题域之间的联系以及充分代表主题的属性组等。
- 关系模式的定义：企业应用系统的每个主题都包含多个实体，对应各个数据源的多个表，这些表之间依靠主题键码联系在一起，形成一个完整的主题。因此我们要确定各个表的关系模式。

第二步所要完成的工作有：

- 确定所涉及的实体，具有不同主题属性或特殊行为的实体需要分为不同的主题域，实体可以改变，但域则是固定的。
- 定义实体的主题属性，包括属性命名，定义属性域（缺省值、有效数值的范围、有效代码范围、有效性规则）。
- 定义实体的特殊行为，如名称、功能、参数、返回值。
- 定义实体之间的关系。

(3) 主题域 E-R 实体数据模型建立原则

- 忠实性：首先得也是最重要的，建模应当忠实于原有的具体要求。也就是说，这个阶段建立的实体关系模型应当客观的反映现实，应当根据每个系统、每个业务，每个功能去真实的建模。
- 避免冗余：在分布式系统中，由于以前各个模块或子系统是单独开发的，只是从自身去考虑，虽然就单个模块来说不存在冗余，但从企业全局来看，往往存在重复描述和信息存储。因此，在建立企业的全局的、统一的数据模型时，必须避免冗余和重复。
- 简单性考虑：尽量使模型简单，如果没有绝对必要，不要添加更多冗余成分。
- 选择正确的联系：主题域之间、实体之间有很多联系，也可以用多种方式表示。但把每种可能的联系都加到模型设计中并不是好办法。首先、它会导致冗余，一个关系联结起来的实体或者实体集可从一个或多个其他联系中导出；其次，可能使得在最后的实现中需要更多的空间来存储冗余元素，而且会使修改更加复杂。因此，尤其是从全局的高度去建模，就必须注意这个问题。
- 选择正确的元素种类：在建模中，对实际中的一个元素，往往可以选择多种建模元素来表示，例如很多选择介于使用属性还是使用实体或者联系之间。一般来说、属性比实体集或联系都易于实现。然而，并不能把一切事物都作为属性。

(4) 主题域 E-R 实体数据模型和 E-R 模型的区别

总的来说，主题域数据模型也是建立在实体和关系的基础上的，所以它与 E-R 模型有一定的联系，同时它们之间也有很大的区别。

域数据模型与 E-R 模型的区别主要有：

- 面向主题的数据模型从一个新的角度提出了实体集的概念，那就是主题域。在 E-R 模型中，实体集本质上也是一种抽象的容器，它在逻辑上把相同结构的实体组织在一起，构成通常所说的表[34]。在主题域 E-R 实体数据模型中，主题域也是一种实体集，不过这种实体集有一定的特殊性，它可以容纳不同种类的实体。
- 主题域 E-R 实体数据模型中所说的关系不同于 E-R 模型中的关系。E-R 模型中的关系是体现实体数据项之间的联系，是一种结构上的关系，它是一种共性的体现；而主题域数据模型中的关系是指主题域和主题域成员之间的访问控制权限，是实例间的关系。
- 主题域 E-R 实体数据模型是在 E-R 模型的基础上，按照一定的应用语义，在数据服务层对数据实体进行了重新的划分，构成了一种异构的

实体集——主题域。所以，从应用角度来看，主题域和表在逻辑上具有基本相同的地位和作用。主题域是异构的，表是同构的。

- 采用 E-R 模型的建模方法通常是自底向上的，而基于主题域数据模型的建模方法更合适采用自顶向下的方法。对于小型的企业应用系统而言，自底向上还是自顶向下的建模方法区别不是很大，而对于大型应用系统而言，采用自底向上的建模方法容易产生“盲人摸象”的局部效应，而自顶向下的建模方法对把握全局非常有效。

(5) 主题域 E-R 实体数据模型的优点

建立主题域 E-R 实体数据模型有助于：

- 找出多个系统相关和重合的信息。
- 减少系统之间数据的重复定义和不一致性，从而减小应用集成的难度。
- 由于建立了企业级的数据架构，在以后进行高层模型设计时，可以从企业主题域 E-R 实体数据模型中进行映射并检查信息的完整性。
- 主题域 E-R 实体数据模型明确定义出了对信息的需求如何转化为数据结构。更好地满足企业信息处理的需要，为企业管理者、业务用户和开发人员提供了一个一致的模型。
- 为高层管理人员清晰地定义出基本业务概念，改善各部门各系统之间的的沟通，提高系统开发效率。
- 在企业购买或开发新的企业应用系统、或进行 IT 战略规划时界定出信息需求的范围。

面向主题的数据模型的一个非常好的特性就是可扩展的特性，因为这些数据类型都是通过全局统一考虑的不是根据某个业务或者某个平台建立的。这样就形成了 SOA 所需要的扁平化的数据模型，它们是领域内的通用数据模型。这种设计为跨越不同应用程序建立统一的数据模型、数据视图提供良好的数据访问基础。随着很多中间件厂商推出跨越平台和数据库建立数据视图，完成异构系统数据整合，数据展示和系统间业务模块的复用，这种数据模型正好顺应了这些新产品的应用要求。

3.3.2 面向业务—服务数据模型

从软件的发展过程来看，面向对象的应用构建在类和对象之上。随后发展起来的建模技术将相关的对象按照业务功能进行分组，就形成了组件的概念；对于跨组件的功能调用，则采用接口的形式暴露出来。进一步的将接口的定义与接口的具体实现进行解耦，就催生了 SOA[35]。而作为业务和 IT 之间的契约的服务，是 SOA 最重要的概念。因此，面向服务技术与面向对象技术和面向组

件技术紧密相关。对象、组件和服务都是对现实世界的抽象描述，所不同的是：首先，在概念层面上，对象是对客观世界中实体的描述，组件描述的是现实世界中细粒度的服务，而服务则是相对组件更粗粒度的表示；其次在复用的策略上，对象是通过继承实现复用，组件则是通过合成实现复用，而服务是通过流程的编排来实现粗粒度复用。因此面向对象、基于组件、面向服务是三个递进的抽象层次。

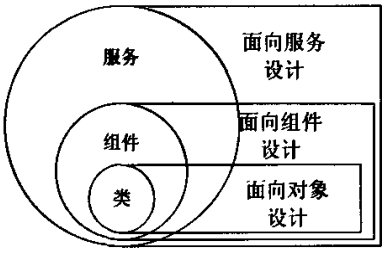

图 3-6　面向对象、基于组件、面向服务关系层次图

Figure 3-6　OOD、COD and SOA

从概念上讲，SOA 中有三个主要的抽象级别：

- 操作：代表单个逻辑工作单元的事务。执行操作通常会导致读、写或修改一个或多个持久性数据。SOA 操作可以直接与面向对象 (OO) 的方法相比。它们都有特定的结构化接口，并且返回结构化的响应，特定操作的执行可能涉及调用附加的操作。
- 服务：代表操作的逻辑分组。
- 业务流程：为实现特定业务目标而执行的一组长期运行的动作或活动。业务流程包括依据一组业务规则按照有序序列执行的一系列操作。操作的排序、选择和执行称为服务或流程编排，典型的情况是调用已编排服务来响应业务事件。业务流程通常包括多个业务调用。

在 SOA 架构下，不管是数据访问、组件访问还是业务访问，都是对于服务的访问，并可在此基础上组合和共享[36]。所以在寻找了系统主要实体，建立主题域数据模型后，接下来就需要根据不同的业务应用来建立服务数据模型，即面向服务的建模（Service Oriented Modeling Architecture，SOMA）。

需要特别指出的是，SOMA 的出现并不是要替代 OOAD（Object Oriented Analysis Design）和 CBD（Component Based Development），正如 CBD 需要借助 OOAD 一样，SOMA 也要借助 OOAD 和 CBD 进行实现层面的建模。与 OOAD 和 CBD 相比较而言，SOMA 贯穿整个 IT 建设的生命周期，在项目规划、设计、实施、运行中都起到重要的作用。其另外一个显著的特点就是将 IT 与业务对齐。具体来看，业务组件模型（或者类似业务分析方法论的结果）、端到端的业务流程以及关键业务指目标是 SOMA 的三项主要输入，SOA 的实现则是 SOMA 的输出。

SOMA 分为服务发现、服务规约以及服务实现三个阶段[37]。

(1) 服务发现：采用自上而下、自下而上和中间对齐的方式，得到服务的候选者。

自上而下 （业务领域分解）方式从业务着手进行分析，我们将业务进行领域分解、流程分解，以及进行变化分析。

业务组件模型是业务领域分解的输入。根据业务组件模型的详细描述，我们可以将业务领域按照业务职责细分为业务范围，并直接其映射到 IT 范畴的子系统，实现业务与 IT 的无缝连接。

顶级的业务流程是流程分解的输入。将业务流程分解成子流程或者业务活动，逐级进行，直到每个业务活动都是具备业务含义的最小单元。流程分解得到的业务活动树上的每一个节点，都是服务的候选者，构成了服务候选者组合。在大部分情况下，服务候选者组合都是一个很长的列表，加上自下而上和中间对齐方式还有可能发现新的服务，因此将服务候选者按照某种方式进行分类是一件非常必要的事情。业务领域分解的结果——业务范围是一个业务概念，一个分类原则。根据业务范围，服务候选者组合可以被划分服务候选者目录。

变化分析的目的是将业务领域中易变的部分和稳定的部分区分开来，通过将易变的业务逻辑及相关的业务规则剥离出来，保证未来的变化不会破坏现有设计，从而提升架构应对变化的能力。变化分析可能会从对未来需求的分析中发现一些新的服务候选者，这些服务候选者需要加入到服务候选者目录中。

自下而上（已有资产分析）方式的目的是利用已有资产来实现服务，已有资产包括：已有系统、套装或定制应用、行业规范或业务模型等。

通过对已有资产的业务功能、技术平台、架构以及实现方式的分析，除了能够验证服务候选者或者发现新的服务候选者，还能够通过分析已有系统、套装或定制应用的技术局限性尽早验证服务实现决策的可行性，为服务实现决策提供重要的依据。

中间对齐（业务目标建模）方式的目的是帮助发现与业务对齐的服务，并确保关键的服务在流程分解和已有资产分析的过程中没有被遗漏。

业务目标建模将业务目标分解成子目标，然后分析哪些服务是用来实现这些子目标的。在这个过程中，为了可以度量这些服务的执行情况并进而评估业务目标，我们会发现关键业务指标、度量值和相关的业务事件[38]。

结合这三种方式的分析，我们发现服务候选者组合，并按照业务范围划分为服务目录。同时为服务规约做好其他准备，如：通过业务目标建模发现的业务事件等等。

(2) 服务规约：定义实现服务的服务组件的细节，包括，数据、规则、服务、可配置概要、可能的变更，同时还会涉及到消息、事件的定义和管理。

经过服务发现的阶段，我们得到了候选服务目录，接下来就需要决定暴露哪些服务。理论上所有的服务候选者都可以暴露为服务，但是一旦暴露为服务，该服务候选者就必须满足附加的安全性、性能等方面的要求，企业还必须为服务的规划、设计、开发、维护、监管支付额外的开支，因此我们会根据一定的规则来决定将哪些服务候选者暴露为服务。

这些规则包含以下几个方面：

- 业务对齐：该服务候选者可以支持相关的业务流程和业务目标。
- 可组装：该服务候选者满足技术中立、自包含以及无状态等特点，同时还满足复合应用的相关非功能性需求。
- 可重用：该服务候选者可以在不同的应用、流程中重用，从而减少重复的功能实现，降低开发和维护的成本。

在决定暴露特定的服务候选者为服务以后，服务规约还需要定义服务的消息、非功能性需求以及服务之间的依赖关系、组合关系。

(3) 服务实现：根据对业务领域的理解和现有 IT 系统的分析，将服务的实现分配到相应的服务组件，并决定服务的实现方式。具体的实现方式，可以由已有系统暴露相关功能为服务，或者重新开发相关功能提供服务，也可以由合作伙伴来提供服务。

在上述过程中，需要说明的是：

- 服务分级和分类：这个活动在服务被指定时开始。将服务分级为服务层次是非常重要的，反映了服务的复合或者不规则的本性：服务可以也应该由良好粒度的组建和服务组成，分级帮助决定合成和分层，以及基于层次的相互依赖服务的协同构建。
- 服务粒度：选择正确的抽象级别是服务建模的一个关键问题，应该在不损失或损害相关性、一致性和完整性的情况下 尽可能地进行粗粒度建模。
- 服务分类和聚合：服务有不同的用法和用途，一个公用的服务可以用于不同于业务服务。此外，还可以将原子服务编排成级别更高、功能齐全的服务。
- 流程建模：定义和建模业务流程是服务建模的关键因素。业务流程是一种可变的交互模式，当某个组织在实现特定的业务目标时，在该组织的组件及其环境之间发生了这些交互[39]。业务流程通常很复杂，因为在应对独特而瞬息万变的环境时，人们会不断进行大量的更改。没有正式的流程文档和流程管理系统的话，这些流程复杂性就会使组织遇到不必要的障碍和瓶颈。流程建模从组织结构、业务流程及相关资源的角度来看待业务，流程建模关注业务活动之间的流动。

3.4 数据模型层次描述

数据模型采用基于层次的建模方法，层次建模的主要思想是把模型排列成一个聚类树，最初每个对象是一个层次，然后根据它们之间的相似性，对这些原子的模型进行合并，在需要的层次上对其进行划分，相关联的部分构成一个层次。模型可以有粒度的层级关系，粗粒度的模型可以用细粒度的模型来组合，不同的建模范型可以用来建立各种粒度层次的模型，每种建模范型都有其更加适应的表达粒度层次[40]。

依据这种分层建模的思想，数据模型主要从数据实体服务层、服务聚合层、跨组织服务聚合层这三个层次来描述，这三个层次是从具体的组织内数据实体视图到虚拟化的、面向用户的跨组织数据视图进行区分的。建立数据模型的过程，就是数据服务抽象的过程；不同阶段、不同层次模型的转换和映射的过程，就是服务封装和组合的过程。

3.4.1 数据实体服务层

数据实体服务层为系统提供数据实体的统一视图，并将数据都封装成为定义简单的、原子的数据实体服务模型。

表 3-2 数据实体服务层描述元素

Table 3-2 Data entity layer

元素	说明
数据实体服务名字	数据实体服务的名字标志
数据实体服务种类	数据实体所属节点、组织、部门等
数据实体模式映射	组织内数据实体全局模式和实体模式之间的映射关系
服务接口	数据实体服务接口描述

在不同的业务系统中，数据以不同形式存在，使用不同的标准进行建模和编码，对整个系统来说，数据实体有全局的，有局部的，有原子的，有组合的。因此，在数据实体服务层，要重新建立一个全新的、统一的、集成的数据模型，重新定义新的关联和数据结构，对数据实体的描述也要进行扩充，除了其本身的固有的属性，还应包括每个数据实体的位置、来源、用处、限制和数据存储模型，以及对这些数据实体服务的描述，当然，这些数据服务仅仅是对数据实体的一些简单操作。在分析抽象时数据实体时要从整个系统的高度去看，而不是从某个业务领域去看，主要使用自顶向下的分析建模方法，要按照数据实体的不同功能和来源进行分类和分层，分析抽象出最原子、最底层的数据实体，

对每一种数据实体要描述清楚其局部模式和全局模式之间的映射关系。新的数据模型将以全新的体系结构图开始，是系统内所有数据实体从各个角度的描述，是对数据实体服务的描述。

数据实体服务层向上发布其元数据信息，提供的是较低层的、细粒度的数据服务。

数据实体服务层描述元素包括的元素如表 3-2

3.4.2　数据聚合服务层

数据聚合服务层：基于数据实体服务层，按照某个部门或特定业务领域制定的某种聚合策略，建立聚合服务模型。

每个聚合服务对应唯一的一种聚合策略，有唯一的全局标示。对每个聚合服务的描述，包括其标识、种类、功能、聚合策略和该聚合服务向下层服务的映射和转换模式，也包括对服务接口的定义。每种聚合服务可以对应到任意多个数据实体服务的组合，也可以对应到数据实体服务和底层聚合服务的混合组合，也可以是多个子聚合服务的再组合。当聚合服务被调用时，聚合服务模型把服务调用映射、转换到各数据实体服务或底层子聚合服务，生成服务的实例，并与这些服务进行交互。

聚合服务是进行整个企业内跨部门、跨系统数据集合的最小单位。数据聚合服务层描述元素见表 3-3：

表 3-3　数据聚合服务层描述元素

Table 3-3　Service aggregation layer

元素	说明
聚合服务名字	聚合服务的名字标志
聚合服务种类	该聚合服务完成何种局部功能，它所属的组织
聚合服务功能具体描述	调用该服务的输入、输出、前提条件、后置条件等上下文描述信息
聚合服务 Qos 属性	服务的响应时间、支持的最大用户数目
聚合服务接口	根据聚合策略定义的通用接口
数据实体服务引用	该聚合服务聚合的数据实体

3.4.3　跨组织服务聚合层

跨组织服务聚合层：基于数据实体服务层和聚合服务层，建立一个逻辑模型，主要用于解决跨组织提供信息的统一视图问题。

这一层要建立的模型不是一个固定的全局模型，而是一个动态可扩展逻辑模型，把下层提供的数据实体服务和数据聚合服务映射到该逻辑模型中。需要注意的是在映射过程中，要保证聚合服务在跨组织时的名字、结构、语义、并行等的透明性。跨组织聚合层在引用下层的聚合服务时，还要考虑它们的各方面的属性，尤其是非功能属性，如选取不同组织提供的相同服务时要考虑各组织所提供服务的 QoS 属性。跨组织服务聚合层可以灵活调用任何层次的数据服务，例如跨组织服务聚合层也可以直接调用数据实体服务。

跨组织服务聚合层描述元素见表 3-4：

表 3-4　跨组织服务聚合层描述元素

Table 3-4　Cross-department service aggregation layer

元素	说明
服务视图名字	视图所包含的组织
聚合服务引用	视图所包含的对聚合服务的应用
聚合服务关系	聚合服务之间的关系

以上是数据模型的建立过程和描述方法，我们以一个简单例子进行说明。如图 3-7 所示：

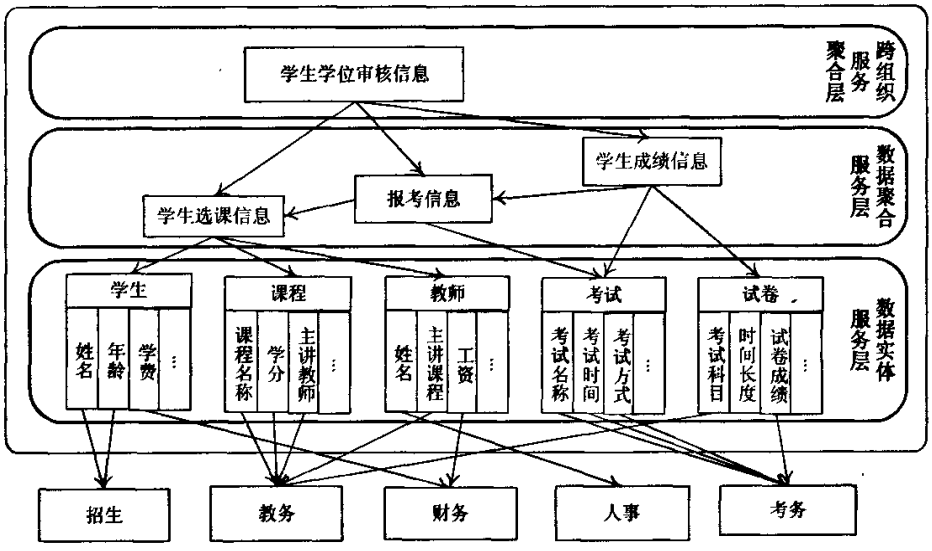

图 3-7 中央电大教务管理系统业务数据模型示例

Figure 3-7 An example for business data model in CRTVU

在电大教务管理系统中，可以简单抽象出学生、课程、教师和考试等主题，其中学生、课程、教师、考试、试卷等是从远程开放教育信息系统中抽象的部分数据实体，分别是上述主题域的核心实体，这些实体是从整个信息系统的高度来抽象地，其属性可能来源于多个不同的部门，在数据实体服务层描述它们；学生选课信息、学生报考信息和学生成绩信息是主题域之间按一定业务规则聚

合的服务，属于数据聚合服务层；学生毕业审核信息是一个跨部门的高层聚合服务，它根据动态的业务要求聚合下层服务，属于跨组织服务聚合层。

3.5　本章小结

本章是本论文的重点章节，这个章节的重点在于建立数据模型，在本章中详细论述了基于 SOA 建立数据模型的过程，通过把面向服务的思想引入数据模型的建立，将数据模型、业务策略和服务定义结合起来，以建立统一数据模型为中心，在系统的分层体系结构中抽象一个数据服务层，为上层应用提供服务，并讨论数据模型的层次描述和两阶段数据模型建立过程。无论从三个层次描述数据模型，还是分两个阶段分析和建立数据模型，都是对数据模型逐层抽象，逐层组合的过程，也是逐层定义数据服务并进行封装的过程。

第4章　电大教务管理系统数据服务的分析与实践

4.1　建立基于 SOA 的中央电大教务管理系统数据模型

4.1.1　中央广播电视大学教务管理系统架构

远程开放教育的教务管理体系是以满足学生个性化学习需求为宗旨的。与传统的校园内教务管理不同，远程开放教育的教务管理体系是向全社会开放的，因为每一个社会成员都是潜在的学习者，他们随时可能进入开放教育教务管理系统，进行信息查询、课程预览、课程注册等。同时，分散在各地的学习者也需要经常通过开放的教务管理体系接受教学过程的指导与管理[41]。为了满足远程开放教育发展的需要，必须依据 SOA 策略重新对远程开放教育信息系统进行规划。而在远程开放教育信息系统中最重要的一部分就是中央电大教务管理系统。

中央广播电视大学教务管理系统是一个遍布全国城乡的远程广播电视大学开放教育系统，该系统分为两级平台，四层管理，五种角色。两级平台为：中央电大平台，省级电大平台。四层管理为：中央管理，省/市管理，分校管理，教学点管理；五种角色为：中央，省/市，分校，教学点，学生。其组织管理机构图如图 4-1 所示。

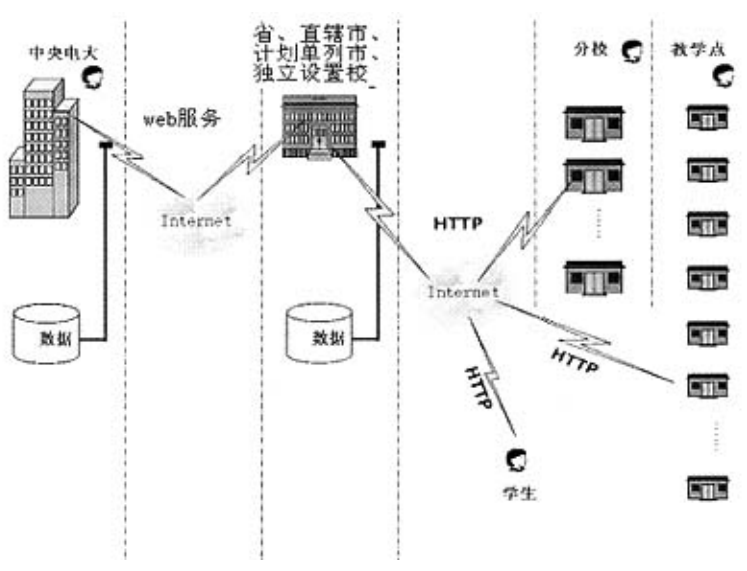

图 4-1 中央电大组织管理结构图

Figure 4-1 CRTVU Administration Structure

由于系统涉及范围广，业务需要复杂，系统需求需要在各级单位得到讨论协调形成最终的一个广而杂的用户需求。另外如此庞大的机构组织，其业务发展本身又是处于一个频繁变化的过程中。总之，对中央电大教务管理系统的需求进行分析得到系统特点如下：

- 用户面广，业务需求复杂，且不稳定。
- 功能繁多，贯彻中央电大整个教学过程中。
- 系统分布广，同时又要互联互通，并且中央对所有省平台具有集中控制的功能。
- 随着深入教学改革，系统需要经常更新维护且时常会有业务需求变动，同时还会有新的需求被提出。
- 教学管理系统支撑着中央电大的核心业务发展，要求系统具备高性能高可靠性、可用性、易操作性、高度安全性。

针对以上中央电大教务管理系统的特点，使用面向服务架构的思想，确立电大教务管理系统的系统架构如图 4-2 所示。

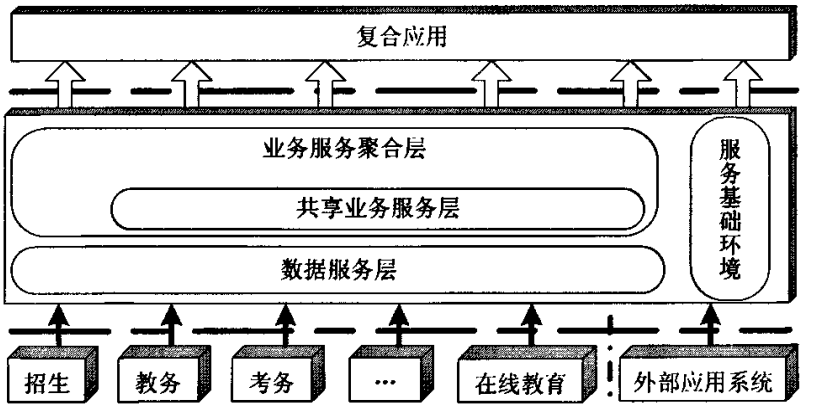

图 4-2 中央电大教务管理系统参考模型架构图

Figure 4-2 Reference Model Architecture of CRTVU-EDU System

数据服务层：建立统一的数据模型，为整个信息数据提供一个统一的数据视图；隔离应用与底层数据源，以标准存取方式提供服务给其它层服务或用户调用。

业务服务聚合层：根据业务逻辑，对核心业务进行梳理和整合，为上层应用提供相对独立的业务服务，同时从业务活动分离抽象可共享的、基于标准的服务。

复合应用层：根据业务流程的变化，面向客户需要和业务过程组成较高层次的复合应用，通过调用下层提供的业务服务，最后展示给用户。

服务基础环境：提供服务交互所需的消息传输、转换和路由，对服务进行集中管理和监控，包括服务的目录、版本、配置等。

基于SOA的远程开放教育信息系统参考架构是以数据服务层为基础，以数据模型为驱动的，良好的数据模型将为整个系统的开发和运行提供保障。

4.1.2 中央电大教务管理系统数据模型典型实例

中央电大的教育教学是以学生为主线的，贯穿从招生到毕业整个过程。在中央电大教务管理系统的开发中，也是着重分析学生从入学到毕业这过程中的核心教学环节。根据学生在教学活动过程的业务流程及其相应的支撑管理系统，中央电大教务管理系统初步划分出如下子系统：

- 学籍注册：负责学生的学籍注册信息的采集。
- 学生选课：负责学生的选课信息的采集。
- 学生报考：负责学生的报考课程信息的采集。
- 毕业申报：负责学生毕业申报信息的采集。
- 学籍管理：负责建立并维护学生的档案信息，同时，还需要负责学籍信息的异动。
- 专业规则管理：负责定义专业规则，专业规则包括：培养定位、课程设置规则、选择课程的规则和毕业规则。
- 课程管理：主要用于建立并维护课程资源，同样，由于电大系统存在多级管理问题，课程资源的建立也存在从上下级的申报审批过程。
- 选课管理：系统负责维护当前学期可选课程。学生既可以按照专业规则的定义选课和又可以从课程超市选课。同时，管理人员要完成学生选课信息的确认、收费，并且划分课程学习班。
- 考务管理：负责维护考试信息，制订考试计划，公布考试课程，配备考试资源，学生根据考试计划选择报考课程，管理人员负责对学生的报考记录确认、收费。然后组织安排考试。
- 毕业审核：管理人员需要根据既定的专业规则中的毕业规则，来审核学生的学习成果，合格者，授予毕业证书和学位证书。
- 收费管理：在许多业务环节中都涉及到费用的发生，因此有必要独立开发一个收费子系统，专门用于处理收费业务，做好收费明细记录等工作。
- 成绩管理：负责学生考后的试卷判分，成绩录入，更动维护，报表分析。

从用例出发的方法把整个系统划分为子系统，可以把复杂问题分解进而分而治之，但同时导致各个子系统间的依赖关系，各个子系统往往不能独立工作，因为其需要彼此的数据支撑。因此必需使用面向服务架构的思想，建立统一的

数据模型。

如第三章基于 SOA 的分布式应用系统数据模型的建立所述，首先就是采用自顶向下的分析方法，建立主题域数据模型。

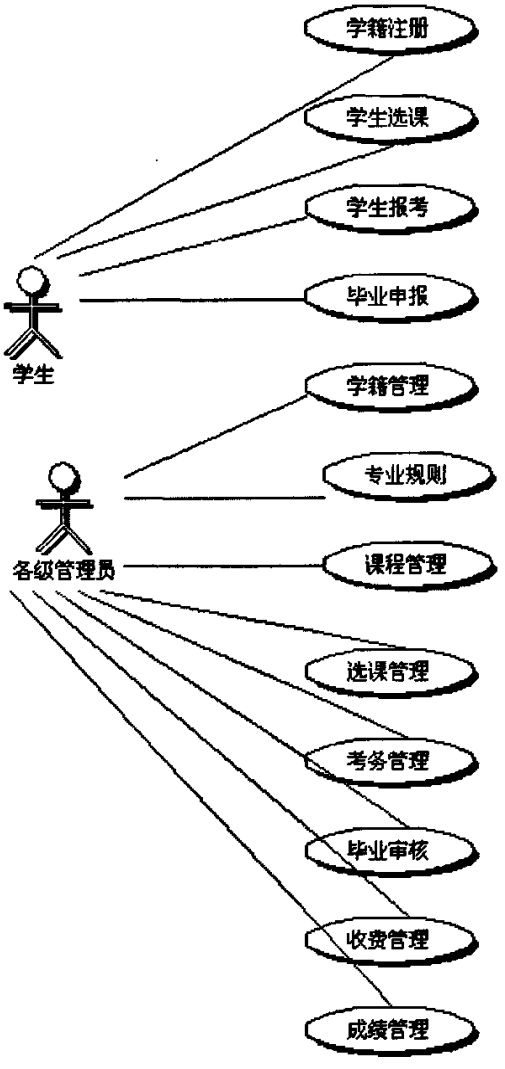

图 4-3 电大教务管理系统总体功能用例图

Figure 4-3 CRTVU User Case

主题域 E-R 数据模型义了企业级的主题域、层面、主要的实体及实体间的业务关系，其目的是要涵盖企业的各个方面并提供了企业级的业务视图，是一个关于整个组织需要信息的完整模型。

深入分析电大教务管理系统的各个系统和核心业务，不从某个业务来看而是从整个系统进行分析，可以发现在教务系统，所有的教学活动都是在学生、课程、考试、教师和学校及其之间完成的，因此可以提取出整个系统存在的主

题，即学生、课程、考试、学校、专业规则、教师和帐户。如图 4-4

在寻找了系统主要的主题域之后，就要摆脱某个平台或某个业务逻辑从全局考虑，进而分析该主题域的核心实体。

从学生主题出发，派生和创建关于学生个人的各种实体，可以找到学生（Student），学生的奖惩(RewardsPunishment)，学籍(StudentStatuesStates)，学生复学（StudentResume），学生休学（SuspendingFlunking），学生变更信息（StudentChangeInfo）。

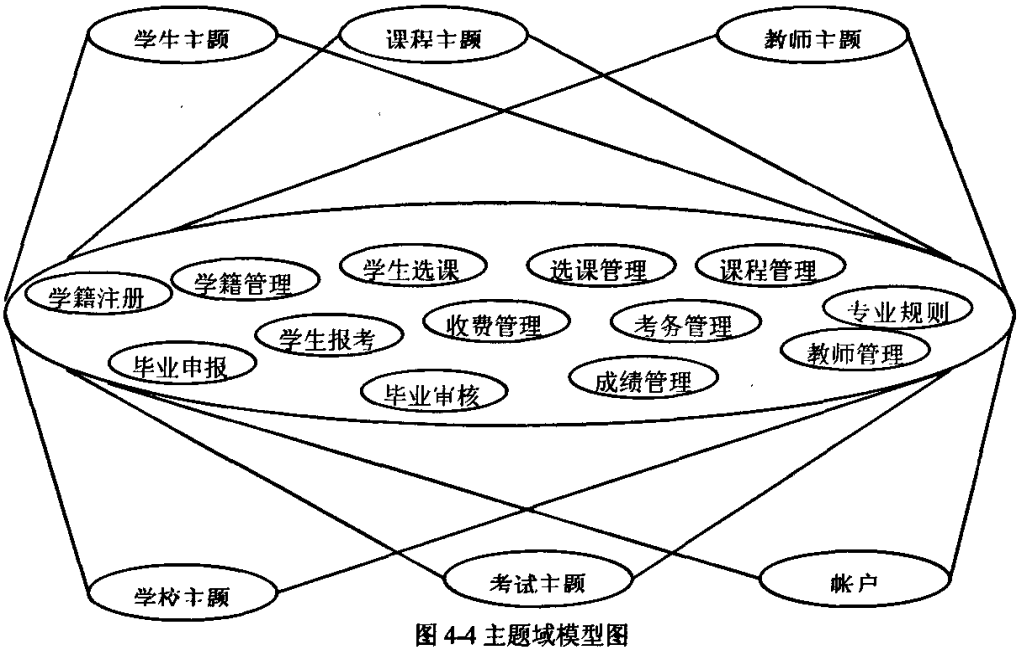

图 4-4 主题域模型图

Figure 4-4 Subject domain Diagram

从课程主题出发，派生和创建关于课程操作的各种实体，可以得到课程(Course)，课程资源(CourseResource)，专业规则（SpcialtyRule），模块（Module），模块课程（ModuleCourse），相似课程（SimilarCourse），互斥课程（MutexCourse）。

从考试主题出发，派生和创建关于考试管理操作的各种实体，考试定义(ExamDefine)，成绩(Score)，试卷(ExamPaper)，试卷订单（PaperOrder），考试科目（PlanExamSubject），形考（FormItem），考点（ExamPlaceInfo），考场（ExamRoom），考试时间单元（ExamTimeArranged）。

从教师主题出发，派生和创建关于教师管理操作的各种实体，可以找到教师(Teacher)，教师的工资（TeacherWage），部门（Dept），教师的奖惩(TeacherRewardsPunishment)，教师的职称(TeacherStates)，教师变更信息（TeacherChangeInfo）等。

从学校主题出发，派生和创建关于学校管理操作的各种实体，可以找到学校（School），学期（Semester），班级 (Manager) ，学位（Degree）专业(Specialty)等。

从帐户主题出发，派生和创建关于帐户管理操作的各种实体，可以找到帐户（Count），收费定义（FeeDefine）等。

这样就形成了 SOA 所需要的扁平化的主题域数据模型，它们是领域内的通用数据模型。这种设计为跨越不同应用程序建立统一的数据模型、数据视图提供良好的应用层数据访问模型基础。随着很多 SOA IT 厂商推出跨越平台和数据库建立数据视图，完成异构系统数据整合，数据展示和提供数据服务，这种数据模型正好顺应了这些新产品的应用要求。

深入分析各个系统之间和业务之间的可公用部分，提取出整个系统提取业务活动过程中的公共业务实体组成数据实体服务层，它放置于系统的较低层次，接下来的工作就是在主题域数据模型的基础上面向具体业务，建立服务数据模型。

建立服务数据模型，我们需要分析和定义整个业务应用的工作流步骤，尤其需要找到业务的操作、业务中所使用的数据和数据发生操作的转换点。将业务的流程和数据联系起来，并且映射这两者之间的关系。

以电大招生业务为例，下面是它的业务流程图（图 4-5）和说明：

- 中央电大 OA 制定本年度的招生计划，包括各省各分校各专业的招生人数。
- 中央电大向各省下达招生计划。
- 分校根据中央电大制定的招生计划和本地区的实际情况，提出招生人数调整意见并上报中央审批。
- 中央电大审批通过即为最终的招生计划，省校在网上将其公布。
- 电大教务管理系统根据电大 OA 制定的招生计划制定招生简章，包括学生入学要求，水平测试科目，交费标准，报名日期等。
- 学生在网上或到教学点报名入学，注册用户名、密码等。提供本人基本信息和相关材料和证书，并交报名费及考试费。
- 分校对学生进行初审。包括检查学生提供的材料是否真实，是否符合要求，是否已交费等。
- 初审未通过，则通知考生，退还考试费等。若初审通过，则成为考生。
- 为了保证学生统一登陆电大在线系统进行考试，电大在线系统需在考试前从教务管理系统接收考生基本信息，包括姓名、身份证号、用户名、密码等。
- 分校组织考生统一参加入学水平测试，考生携带本人证件到分校指定

考场（机房），通过注册的用户名和密码登陆电大在线网站进行网络在线考试。

- 考试完毕后，电大在线系统计算考生各科成绩。

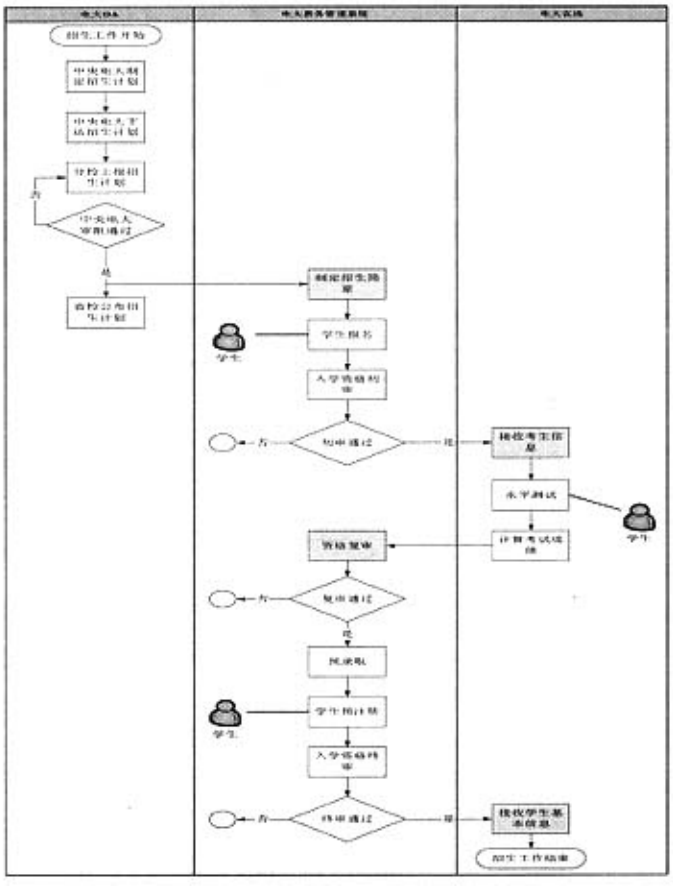

图 4-5 招生业务流程

Figure 4-5 Enrolling operation process

- 省校根据入学水平成绩标准和考生的考试成绩进行比较，包括考核课程（必须达标的某一门课程，各专业不同）成绩和总成绩，复审出合格的考生，并保存成绩。
- 复审通过则分校根据招生实施方案中的招生人数进行预录取，以及专业调剂，向学生发录取通知书。
- 学生登陆网站进行预注册，核对并补充录入本人详细信息，如职业、婚姻状况等。分校进行学生照片采集。
- 中央对各省学生进行抽查，检查其各项资格是否符合入学标准。并统一生成学号，成为电大正式学生。
- 电大在线系统接收入学学生的基本信息，包括学号等信息，以备参加

今后各项电大在线形成性考核，期中期末网络考试等，以及成绩录入系统之用。

从图 4-6 可以看出，这个业务流程，涉及学生、课程、考试和学校四个主题域，用到了学生基本信息、学生入学测试收费、测试试卷、测试成绩、教学点等多个数据实体，核心数据学生信息在业务流程中多次变换，横跨三个系统。因此，根据其业务流程和应用特定，需要三个服务，且都是系统间服务，其中电大 OA 需提供根据各省、各分校、各专业等条件查询招生实施方案的服务。电大教务管理系统提供根据年度学期查询考生基本信息服务。电大在线系统提供的服务为，根据各省、各分校、各专业、考生报名号等条件查询考生入学水平测试成绩。如图 4-6 所示：

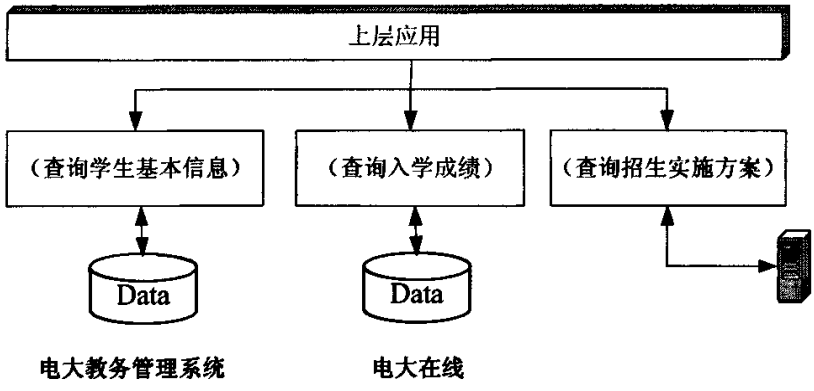

图 4-6 招生业务服务结构图

Figure 4-6 Enrolling operation Service

其中电大教务管理系统查询学生基本信息，可以从学生实体对应的学生表中查，数据实体服务即可完成功能；电大 OA 查询招生方案与此相似。但电大在线查询考生入学成绩，跨学生主题和考试主题，是学生入学测试这一业务活动产生的结果，需要学生、考试、试卷、成绩等实体信息按照学生入学测试组合为学生入学测试成绩。

与电大招生业务类似，电大教务管理的另一个核心业务活动学生选课也是学生、课程、学校和教师主题域相关联产生的，其用到的数据实体也都集中在这三个主题域，但又因参与具体业务，会产生一些聚合服务数据。如学生选课结果数据，学生选课收费数据、模块课程数据，专业规则数据、教学点开设课程数据和学生选课次数数据等。

其他如学生报考、毕业审核等业务，也与电大招生、学生选课类似。在此不再详述。

分阶段，只是建立数据模型的过程，而最终的数据模型必须根据前面对数据模型的描述方法，进行分层的描述，把系统中的公共业务实体组成的数据实体服务层放置于系统的较低层次，而把与业务紧密相关的聚合服务数据搭建在

较高层次的之上。

下面是电大教务管理系统数据模型的三层描述，如图 4-7 所示：

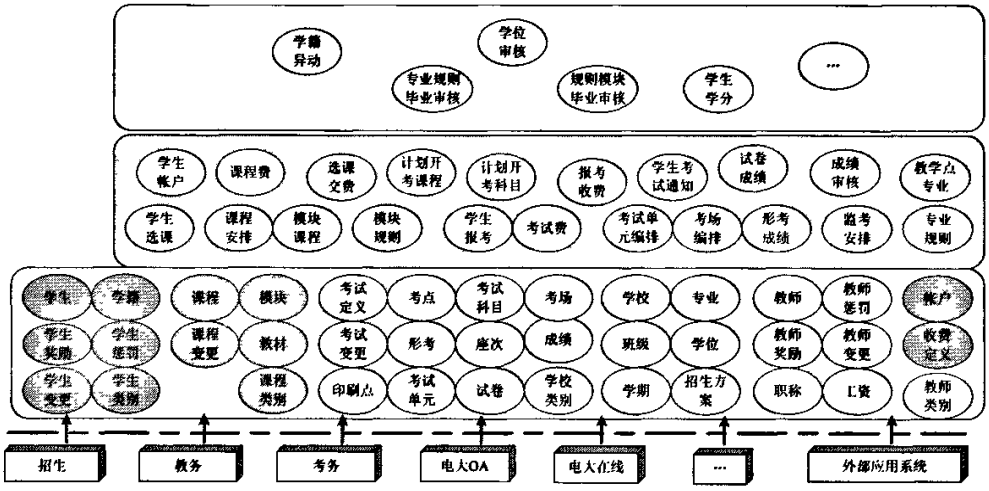

图 4-7 中央电大教务管理系统数据模型三层描述

Figure 4-7 Business data model in CRTVU

4.2　数据服务实现框架

4.2.1　实现框架

建立数据模型仅是实现数据服务的第一步，还必须整合利用各种技术进行实现，图 4-9 是我们实现远程开放教育系统数据服务的逻辑框架。

数据集成接口：屏蔽底层数据源，向上提供统一标准格式的数据。一般采用 API 来实现，可直接访问各种源数据库及其元数据，也可访问其它数据源，包括应用系统 API、用户数据、Web 服务数据及其它外部的各种数据源。

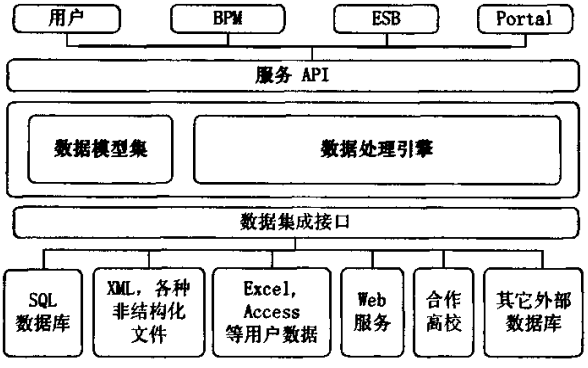

图 4-8 远程开放教育信息系统数据服务逻辑框架图

Figure 4-8 Implementing the CRTVU through Business data mode

数据模型集：以一致、标准和统一的格式描述数据模型，并进行模型的管

理和监控，可以对模型进行修改、添加和删除，类似于协调器集成系统中的虚拟中心库。

数据处理引擎：按照上层具体应用要求，从数据模型生成数据服务实例；完成服务调用和响应。主要负责两方面的工作：一是解释数据模型和服务调用，完成各层模型的转换和映射；二是从服务调用和响应消息中抽取业务数据，完成数据操作和计算，并把结果以标准格式写入响应消息。

服务 API： 按照系统业务功能，划分基于标准的服务接口，接受外部服务调用请求，将请求信息进行转换后发送给数据处理引擎，同时把调用请求映射到一个数据模型。快速构成服务组件，适应业务流程需求的变化。

在系统工作中，采用“拉”的方式，将各应用系统的源数据“拉”至中心虚拟数据库。

这个实现框架应用到了多种技术，实现是非常复杂的。但随着 SDO 及其附属工具的出现，尤其是当 SDO 提供了大量的数据编程模型和 API 后，已经大大简化了该框架的实现。

4.2.2 服务数据对象

服务数据对象（Service Data Objects，SDO）是 Java 平台的一种数据编程架构和 API，它统一了不同数据源类型的数据编程，提供了对通用应用程序模式的健壮支持，并使应用程序、工具和框架更容易查询、读取、更新和检查数据[42]。

4.2.2.1 SDO 驱动因素

虽然 Java 平台和 J2EE 提供了大量的数据编程模型和 API，但是这些技术都是分散的，而且通常不能服从于工具和框架。此外，其中的一些技术可能难以使用，而且可能功能不够丰富，无法支持常见的应用需求。SDO 旨在创建一个统一的数据访问层，以一种可以服从工具和框架的易用方式，为不同的数据源提供一种数据访问解决方案，而不是为了满足替换底层数据访问技术的需求而开发的。SDO 的目标是：

- 对异构数据源的统一数据访问。当前的数据编程技术或多或少都是特定于数据源类型的。但是，在实际应用程序中，数据通常来自于各种数据源，包括关系数据库、定制的数据访问层、Web 服务、数据存储区、JMS 消息和企业信息系统。这种异构性对应用程序开发人员提出了严重的挑战，因为他们需要学习和使用大量不同的编程模型。因此，一个与数据源类型无关的、表示数据集合的通用工具可以为应用程序员提供更简单的统一编程模型。

- 对静态和动态数据 API 的统一支持。静态的强类型接口可以为应用程序员提供一种使用编程模型的简单方式[43]。例如，实体 EJB、JDO、Hibernate 以及其他对象/关系持久性机制提供类型化的 JAVA 数据接口，这为开发人员提供了一种方便的编程模型。相比之下 JDBC 的 ResultSet 和 RowSet 接口只提供了动态的非类型化数据 API。类似地，JAXB 为 XML 数据提供了代码生成的 JAVA 接口，这使 XML 数据编程比使用 DOM 或 SAX API 更简单。但是，只有静态数据 API 或只有动态数据 API 是不够的：两者都是必需的。静态数据 API 提供应用程序员所需的易用性。但是在某些情况下，静态 JAVA 数据接口既不可行也不适合。例如，对于事先不知道结果数据的类型的动态查询，静态 JAVA 数据接口就不可行。因此，统一的数据编程技术需要同时无缝地支持静态和动态数据 API。
- 对断开编程模型的支持。许多应用程序自然地包含数据访问的断开使用模式：应用程序读取一组数据，将其在本地保留一段短的数据，操纵数据，然后再将更改应用到数据源。例如，这在基于 Web 的应用程序中是非常常见的模式：一个 Web 客户端请求查看一个表单，一个 Servlet 或 JSP 请求本地读事务中的数据，并将数据呈现在 HTML 表单中，Web 客户端提交对表单的更新，Servlet 或 JSP 使用新的事务更新数据。如果有任何的底层数据在客户端应用更改之前被更改，更新就会被拒绝，应用程序就必须采用纠正措施。最佳实践通常要求对这种场景使用乐观并发语义，从而提供具有适当业务级语义的高级别并发。但目前没有哪一种数据访问技术能够同时提供这种断开的乐观模型以及前述的其他特性。JDBC 扩展提供了断开模型，但是如前所述，JDBC 不能满足异构数据访问或易用性的要求。类似地，许多对象/关系持久性机制（例如，许多实体 EJB 实现、JDO 等）虽然支持乐观并发语义，但是它们不提供必需的统一数据访问特性。
- 对基于常见设计模式的定制数据访问层的支持。许多应用程序使用常见的设计模式（例如，Transfer Object、Transfer Object Assembler、DataAccess Objects 以及 Domain Store）来构建定制的数据访问层。当应用程序需要与物理数据源隔离时通常都使用这些模式。实现数据访问层通常需要大量的定制代码，其中许多都是可以自动化的。此外，这些定制的数据访问层应该能够集成到工具和框架中。
- 应用代码与数据访问代码的去耦合性。为了获得可重用性和可维护性，应用代码应该与数据访问代码分离。数据访问技术应该服从这一关注点分离。

4.2.2.2 SDO 架构

SDO 具有一个可组合的架构。它提供了一组核心组件和服务，然后使用 SDO 支持的工具和框架进行扩展。核心 SDO 规范提供了适用于各种类型的数据源的基本 API。例如，核心 SDO 规范没有指定一种特定的查询语言或特定的后端存储区。因此，查询语言可以使用 SQL，也可以使用 XPath 或 Xquery，或者其他任何查询语言。存储区可以是关系数据库，也可以是对象数据库或者 XML 数据源。SDO 架构的基本原则是，尽可能的使用常见资源，必要时允许使用特定于数据源类型的资源。核心 SDO 规范创建了使这种灵活性和简单性成为可能的内核。SDO 框架如图 4-10：

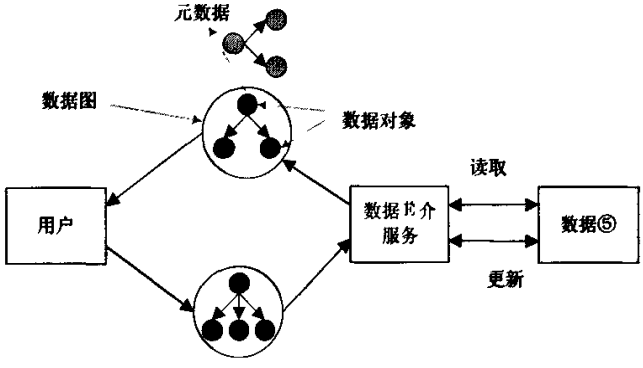

图 4-9 SDO 架构图

Figure 4-9 SDO Architecture

SDO 核心。核心 SDO 规范包含程序员常用的基本组件，包括数据的数据对象（ Data Object）和表示数据视图的数据图（Data Graph）。SDO 规范还提供了一个元数据 API，它允许客户端自建数据模型。SDO 元数据 API 使工具和框架能够以统一的方式处理不同的数据源。元数据 API 支持来自其他综合性更强的元模型的基本概念。

SDO 数据中介服务。数据中介服务提供对数据源的访问。数据中介服务通过从后端数据源读取数据而创建数据图，还可以基于对数据图所做的更改而更新数据源。

SDO 支持的工具。 SDO 支持的工具包括代码生成器、元模型转换器、模式转换器、数据建模工具、模式建模工具等等。

该架构中的各种组件包括数据对象、数据图和元数据：

- 数据对象。数据对象保存实际数据，包括基本值以及对其他数据对象的引用。数据对象还拥有对其元数据的引用，这允许基本数据对象以获取有关数据类型、数据关系和数据约束的信息。
- 数据图。从概念上来说，数据图是一组提供组件之间或层之间的传输

单元的数据。具体来说，数据图是一个多根的数据对象集合。数据图记录所有对数据的更改，包括新的数据对象、被更改的数据对象以及被移除的数据对象。

- 元数据。关于数据对象的元数据使开发工具和运行时框架能够内省数据，包括数据类型、关系以及约束。 SDO 提供了一个通用的跨异构数据源类型的元数据 API 来支持一般的工具和框架[44]。

SDO 中对象的关系也可以用 UML 模型表示，如图 4-11：

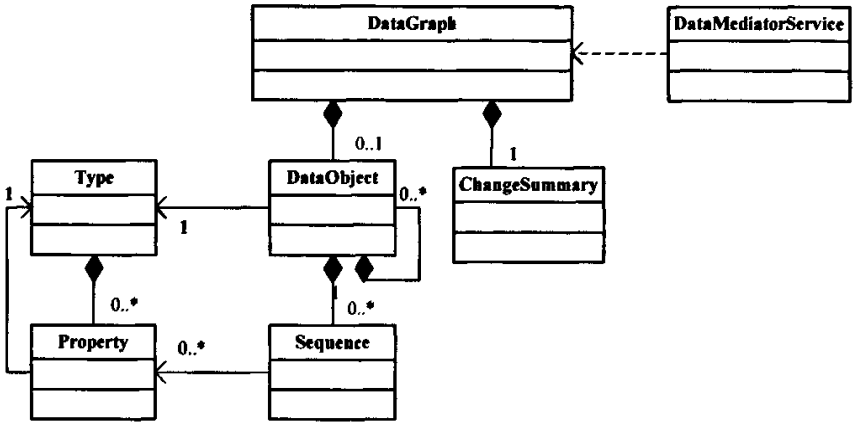

图 4-10　SDO 组件 UML 模型图

Figure 4-10　SDO Component UML Diagram

4.2.2.3 SDO 应用

SDO 支持多种常见应用，下面介绍主要的几种。

- 虚拟数据访问

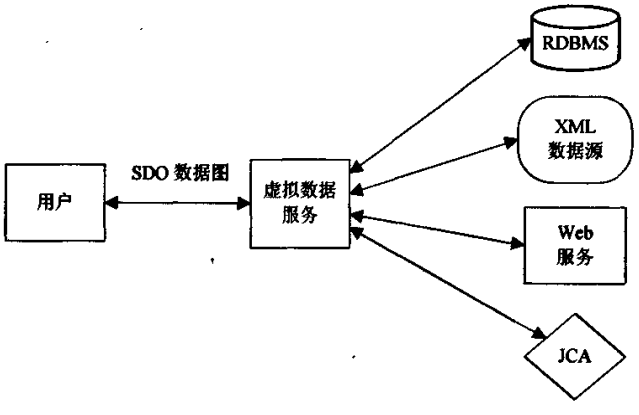

图 4-11 SDO 虚拟数据访问方案

Fig4-11 SDO Vitrual Data Access

可以用于提供来自于多个异构数据源的统一聚合数据的虚拟数据访问架构中。这是一种常见的架构模式，甚至底层数据源全部都是通过 JDBC 或实体访问的关系数据。一个“虚拟数据中介”可以支持丰富的查询语言。

然后可以观察、内省、更新来自虚拟数据中介的数据图，然后将其发送回原始数据源——程序员不必知道或了解后端跨各种类型的数据访问的复杂性。这种分离在面向服务架构的实现中非常重要。

- 关系数据库访问

SDO 编程模型在对关系数据源编程方面的表现引人注目，因为它提供了断开的访问模型，以及静态和动态数据 API。实际上，SDO 架构包含了许多广泛使用的设计模式。SDO 还可以与对象/关系持久性机制相集成，包括实体 EJB、JDO 和 Hibernate。

- 读写 XML

SDO 支持一个综合的 XML 编程模型，包括查询、读取和更新。在该架构下，XML 数据源可以是 XML 文件、本地 XML 数据存储区或者是具有 XML 功能的关系数据库。SDO 打算提供比 Xquery API 更高级的抽象。如图 4-13 所示。

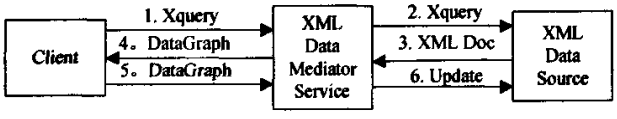

图 4-12 SDO XML 访问方案

Fig4-12 SDO XML Data Access

4.2.3 数据映射

电大教务管理系统中的数据多种多样，采用了 XML 来描述数据模型。XML 以一种开放的自描述方式定义数据结构，在描述数据内容的同时能突出对结构的描述，从而体现出数据之间的关系，提供了一种连接关系数据库和面向对象数据库及其他数据库管理系统之间的纽带。正是 XML 的这种无模式和自描述的特点适宜于描述缺乏统一固定模式、自述性和动态可变性极高的 Web 信息数据，使得 XML 成为目前多数据信息集成框架的首选[45]。

在集成框架中，各类数据源都具有各自不同的数据结构。要有效地把这些不同数据结构的数据结合起来，向上层提供一个统一的、XML 化的数据结构，需将来自于异构数据源的关系数据模式、对象数据模式、HTML 格式的 Web 数据模式和 XML 文档等转换成集成系统能进一步处理的统一模式，实现不同数据源中各式数据的集成和共享。

4.2.3.1 XML 数据模式与关系数据模式的双向映射

异构数据源集成中，必然会涉及到关系数据库和 XML 文档之间的数据传递。XML 数据是半结构化树状结构，数据是可嵌套的混合类型。关系数据是结

构化的，字段值是不可分割的基本实体属性。因此，二者间的直接数据交换是不合适的。

XML 是一种树形结构的数据格式，其元素可以嵌套，也可以完全包含在另一个元素中。XML 文档从根元素开始，以下是层次的嵌套元素结构。元素可以是子元素也可以是属性。DTD 描述了 XML 文档的结构，定义了所允许的元素类型、属性和实体，并表述了对它们组合方式的约束条件。DTD 定义的这些规则保证了所有 XML 文档有一个一致的逻辑结构。因此，XML 文档是其 DTD 文档的一个实例。XML 文档中的数据信息则通过 DTD 中的元素和元素属性来实现。

关系数据主要是由一系列的数据表来表示。表由结构相同的元组（行）构成，而元组则由带有类型、长度的字段（列）构成。对于每一个表来说，都包含一个主键字段 Key，数值由系统自动生成。

通过对两者形式化结构的比较，可看出 DTD 中的属性可以和关系中的字段对应，DTD 的元素既能和关系中的字段相对应，也能对应关系中的表。所以，一个 DTD 文档可能对应存储于关系数据库中的几张表。

(1) 关系模式到 DTD 结构的映射

从关系数据库结构生成 XML 文档的规则是：

- 为数据库中的每个表新建一个元素；
- 为表中的每列建立一个属性或只含 PCDATA 的子元素；
- 为每个属于主键的列新建一个子元素。

映射后 DTD 文档名被赋予关系数据的表名，同时也是虚根下的唯一直接子元素的名字。除主键外的全部字段均被映射成为文档中唯一直接子元素的 PCDATA 类型的子元素。关系表与 DTD 文档的映射实例如下：

Table Exam	<!ELEMENT Exam(Exam Num,Date,StudentNum)>
Column Exampkey	<!ELEMENT ExamNum(#PCDATA)>
Column ExamNum	<!ELEMENT Date(#PCDATA)>
Column Date	<!ELEMENT StudentNum(#PCDATA)>
Column StudentNum	

(2)　DTD 结构到关系模式的映射

从 DTD 树图的虚根开始，按照广度优先搜索对每种元素或属性进行映射。

- 为 DTD 中的每个元素建立一个表，元素的 PCDATA 子元素或属性直接映射为表中的字段。如果子元素是混合型元素或是多值元素，则在建表时暂不考虑。
- 如果元素含有父元素，则新建一张表，将其父元素转换为新建表的主键。

- 元素映射完成后，按广度优先算法重复 前两个步骤映射下一个未映射的元素，直至所有的元素和属性映射完毕。

由于 DTD 元素的嵌套和多值特性，一个 DTD 文档可能会生成几张表。例如：

```
<!ELEMENT Items(ItemNum,Quantity,Part)>      Table Items
<!ATTLIST Items Exam IDREFS #IMPLIED>          Column ItemNum(pk)
<!ATTLIST Items Name CDATA #REQUIRED>          Column ExamNum(fk)
                                               Column Quantity
                                               Column PartNum
                                               Column Name
                                           Table Exam
                                               Column ExamNum(pk)
```

4.2.3.2 DTD 与对象模式的双向映射

(1) 类结构到 DTD 结构的映射

XML 与对象数据模型最为相似，因为它同样也是由节点元素组成，并且每个节点都可以包含异类数据。另一方面，节点异类的程度大部分取决于用于定义 XML 文档结构的特定 DTD。DTD 是面向数据记录的，这使得 XML 文档和层次数据库可以一样严格。只要 DTD 模式正确，它足以完整地表示对象或层次结构。

- 为每个类建立一个元素；
- 类中包含的每个属性以及简单的数据域都对应地生成新的元素；
- 类中包含的每个复杂的子类域递归地使用前两个步骤生成新的元素，新元素作为当前元素的子元素。

```
Class A {                    <!ELEMENT A (B,C)>
  String b;                  <!ELEMENT B (#PCDATA)>
  C    c;                    <!ELEMENT F (#PCDATA)>
  string f; }
Class C {                    <!ELEMENT C (D, E)>
  String d;                  <!ELEMENT D (#PCDATA)>
  String e; }                <!ELEMENT E (#PCDATA)>
```

(2) DTD 结构到类结构的映射

同 DTD 至关系结构的映射类似，从 DTD 树的虚根开始，按照广度优先搜索对 DTD 树中的每种元素和属性进行映射。

- 为每个元素建立一个类；
- 元素中所有的属性或 PCDATA 子元素直接映射为类的数据域；

- 若元素的子元素是混合型或多值元素，则递归地使用前两个步骤生成新类，新类的实例化对象作为当前类的数据域。
- 按广度优先算法，映射下一个未映射的元素直至所有的元素和属性被映射。

```
<!ELEMENT A (B, C)>                    Class A {
<!ATTLIST A D CDATA #REQUIRED>           String   b;
<!ATTLIST A G IDREFS #IMPLIED>           String[] g;
<!ELEMENT B (#PCDATA)>                   C    c; }
<!ELEMENT C (D, E)>                    Class C {
<!ELEMENT D (#PCDATA)>                   String   d;
<!ELEMENT E (#PCDATA)>                   String   e; }
```

4.2.3.3 XML 和 HTML 的映射

目前 Web 环境中的数据主要以 HTML 格式发布，HTML 有效数据与 XML 结构化数据的转换是 HTML 内容数据重用和支持 XML 系统与 HTML 系统互操作的一个关键问题。HTML 中既有表达特性语义的内容数据，也有用于表现数据的格式数据，而 HTML 内容和 XML 数据之间的转换才是有意义的转换，所以实现 HTML 内容数据与格式信息的分离是实现二者之间映射的关键问题[46]。

HTML 内容数据分析 HTML 页面由内容数据和表现内容的格式数据组成。内容数据是具有特定语义的数据，是需要被映射成为 XML 模式的数据。而格式数据则负责内容的表现形式，通过具有特定含义的标签及属性值来格式化数据。

DTD 为 HTML 内容数据定义了相应的元素和元素关系，DTD 为 HTML 内容数据定义了相应的元素和元素关系，实现了 HTML 内容数据的标签规则到 XML 模式的映射。DTD 定义如下：

```
<!ELEMENT Block (Text*, TextLink*,Img*,Form*,Table*,Frame*)>
<!ELEMENT Form (Text*, TextLink*, Img*)>
<!ELEMENT Table (Text*, TextLink*, Img*)>
<!ELEMENT Frame (Text+)>
<!ELEMENT Text (#PCDATA)>
<!ELEMENT TextLink(#PCDATA)>
<!ELEMENT Img (#PCDATA)>
```

系统实现 HTML 至 XML 的映射，首先需要设计者按照标签规则对 HTML 网页进行逻辑块和内容块的划分，对 HTML 网页源代码添加标签。系统依据 DTD 对内容标签规则的定义，通过 DOM 接口模型将带有自定义标签的 HTML 源文件转换为 DTD 格式所限定的 XML 文档。

XML 至 HTML 的转换

XML 实现了数据与显示的分离，在不同的应用之间传输数据。XML 数据的显示是由 XSLT 实现的。XSLT 是一种专门描述结构文档表现方式的样式单语言，能组合多个源码中的数据或根据用户或会话的特征实现个性化地显示输出内容。论文使用 XSLT 将 XML 文档转换成 HTML，并使用 HTML 浏览器作为格式化引擎。HTML 实际上只是 XML 词汇表的一个示例，而 XSLT 可以使用任何 XML 词汇表，很好地实现 XML 文档向任何一个其它格式的文档的转换。

4.3 中央电大教务管理系统数据服务实践分析

SOA 并不是一种现成的技术，而是一种组织和架构 IT 基础结构及业务功能的“方法论”。SOA 的思想就是利用现成的信息技术，将企业网络变成一个大的软件开发环境。因此，在实际运用中，企业需要根据具体情况综合利用其它技术，才能更好的实现 SOA。中央电大教务管理系统涉及范围广，业务流程复杂，对数据服务的要求也多种多样，不可能采用一种技术或方案去实现。因此，我们根据不同的数据服务要求，在统一建立数据模型和实现框架的基础上，灵活采用多种实现技术，充分发挥各种技术的优势，以满足不用业务的需求。

4.3.1 基于 Hibernate 技术实现关系数据库表到数据对象的映射

关系型数据库，具有简单灵活的数据模型、较高的数据独立性、能提供良好性能的语言接口、并且有比较坚实的理论基础等优点。电大各个系统的数据存储都采用的是关系数据库。由于关系数据库是属于集合理论范畴，主要处理数据集合的存储问题，但在后面的业务开发中，需要按照建立的数据模型对这些数据表进行映射，并根据业务活动把数据封装成业务数据对象，业务数据对象包含了数据和对数据的处理方法。如何处理面向关系数据表和数据模型的映射以及业务数据模型的封装，这是电大教务管理系统实现数据服务的一个重要问题。

J2EE 领域的对象/关系映像产品有很多，现在使用最广泛的是 Hibernate。Hibernate 功能强大，支持集合、对象关系、组合类型，对继承、多态，各种复杂的关系也都有很好的支持[47]。同时，Hibernate 还拥有丰富的 HQL（Hibernate Query Language）查询语句支持几乎所有 sql 功能，诸如：连接查询、统计函数等。Hibernate 的一大特色则是可以支持多种环境下工作，可以通过简单的配置文件修改来移植应用系统到不同的数据库（比如从 SQL SERVER 数据库移植到 ORACLE 数据库），其配置信息可以通过 properties 或者 xml 文件来配置，同时 Hibernate 还提供了本地化缓存方案等来进一步提高系统性能。图 4-14 是它的体

系结构图。

中央电大教务管理系统采用了基于 hibernate 的解决方案。用 XML 完成关系数据库表之上数据模型的描述并进一步映射封装为 JAVA 值对象

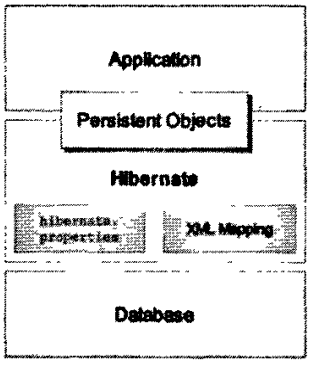

图 4-13 Hibernate 体系结构图

Figure 4-13 Hibernate Architecture

VO(Value Object)。VO 封装数据对象，在各个层次间被传递、处理、显示，如图 4-16 所示。

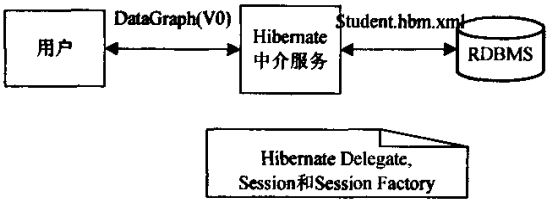

图 4-14 Hibernate SDO 实现

Figure 4-14 Hibernate SDO Implemention

我们以学生表为例做单表映射。图 4-15 是学生表，它对应的 Student.hbm.xml 及说明如下：

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 2.0//EN" "http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd">

HibernateMapping 的标准和使用 DTD 的数据结构框架

<hibernate-mapping>

在这对标签中间就是值对象的属性与数据库表的字段的对应

<class name="rtvu.student.common.vo.Student" table="xsb">

Java 的值对象类 rtvu.studnet.common.vo.Studnet 对应数据库中的学生表（xsb）

<id column="xh" name="xh" type="java.lang.String">

<generator class="uuid.hex"/>

</id>

标签指定值对象中的标识符就是学生表（xsb）的学号（xh），类型从数据库指定的 nchar(32)映射成为 java 中的字符串类型。

<version column="Bbh" name="bbh" type="java.lang.Integer"/>

version column 用来持久对象更新和控制 dirty 状态，如果版本号发生紊乱必然说明有两个以上的操作同时更新了数据。

Column Name	Data Type
Xh	nchar(13)
Xm	nvarchar(20)
Yhm	nvarchar(20)
Zydm	nchar(8)
Zygznd	nchar(4)
Zygzxq	nchar(2)
Xslxdm	nchar(2)
Zyccdm	nchar(1)
Bdm	nchar(15)
Nd	nchar(4)
Xqdm	nchar(2)
Xjztdm	nchar(1)
Bbh	int

图 4-15 学生表

Figure 4-15 XSB Table

<property column="Xm" length="13" name="xm" type="java.lang.String"/>

<property column="zydm" length="7" name="zydm" type="java.lang.String"/>

这两条属性，分别是数据库表学生表（xsb）中的两个字段，也对应 rtvu.student.common.vo.Student 类中的两个属性。type 字段代表映射后的 java 数据类型如 String，Float。

对应的 StudentVO 是由系统设计的 VO 基类派生并添加属性，在应用中还建立 VO 的操作类 StudentHibernateDelegate 类，这个类也是从系统中 HibernateDelegate 基类派生并添加针对数据类的操作方法，它将应用 hibernate 提供的数据持久化类，Session 类和 SessionFactory 类，同时拥有数据操作的方法，方法将使用 hibernate 提供的 HQL 语句。之后，就可调用 VOHibernateDelegate 类中的方法来完成工作。

当然，Hibernate 也可以完成多表映射，例如学生选课结果信息中的部分信息可从学生表和课程总表映射出来，其 XML 描述 SelectedCourse.hbm.xml 如下

```
<class name=" SelectedCourse.Student" table="Student">
    <composite-id name="xh" class=" SelectedCourse.xh">
        <key-property name="name" column="NAME" type="string"/>
<key-many-to-one name="xxdm" class=" SelectedCourse.xxdm"
```

```
        column="xxdm"/>
        </composite-id>
    <version name="varsion" column="VERSION" unsaved-value="0"/>
    <set name="cousrse" lazy="true" inverse="true">
        <key>
            <columncolumn="NAME"/>
            <columncolumn="kcid"/>
        </key>
    </set>
</class>
<class name=" Course" table="kczb">
    <many-to-one name="student" class=" SelectCourse.Student">
            <columncolumn="NAME"/>
            <columncolumn="kcid"/>
    </many-to-one>
</class>
```

查询服务是数据服务提供的主要服务之一，通过书写 Hibernate 查询语言（HQL）可以很好完成上层应用需要的各种查询操作。HQL 语句可以很好的封装数据对象，隔离数据库表的细节。以查询学生选课信息为例，以下是 HQL 与 SQL 的写法上的区别：

```
Select * from SelectedCourse, Student where SelectedCourse.xh=Student.xh
Select * from xsxkjgb, xsb where xsxkjgb.xh=xsb.xh
```

可以清楚看出它们的区别，SQL 中应用系统开发人员仍然需要知道数据库表和它的字段，而 HQL 中开发人员可以完全忽略数据库表细节，只要根据系统中 SelectedCours 和 Student 类和及它们内部的属性就可以操作数据了。HQL 就是把 SQL 进行了一次应用对象和对象属性的封装。使得应用系统开发人员更关注系统和业务逻辑而不是数据库结构和数据库设计。

4.3.2　电大教务管理系统数据服务应用实例

在中央广播电视大学的日常四层教学管理中，涉及大量的查询操作，而且这些查询往往横跨中央、省校两级平台，且为多种角色提供如学生、教师、管理人员、合作高校。同时、为支持高层管理人员决策分析，还要进行必要的数

据统计分析。因此，在整个教务管理系统中报表服务就显得必不可少。

报表查询和数据分析一般都是跨多个业务，或者从某一个主题出发统计分析数据，需要从统一的综合数据模型出发，提供业务数据的不同用户视图。因此，建立了电大教务管理系统的数据模型并实现数据服务，为报表服务提供了强大的数据源支持。

数巨报表 MAX Reports 是一套功能强大且简单易用的专业报表开发工具与数据分析工具。它支持 Web 方式下的应用，并成功地解决了 B/S 架构软件中报表制作、预览、打印及导出文件等难题，可以做到所设计的报表与 Web 应用程序的紧密集成。值得注意的是 Max Report 也支持与 XML 数据源的集成，可将整合后的数据进行分组、过滤，或建立多维数据集，进行钻取、排序、虚拟分层、复杂计算等。

中央电大教务管理系统采用 Max Report 实现报表服务，每个报表的设计包括两项主要任务：

- 数据源设计：建立与报表要求符合的数据源，从原理上类似于在数据库的基础上建立视图
- 报表设计：报表样式的设计，做成报表模板

我们主要讨论如何完成报表数据源设计。在建立数据模型和实现数据服务服务过程中，我们使用 XML 文件描述统一的数据模型，并将其映射为数据对象，封装数据操作。因此，在设计报表数据源时，根据用户需求，我们利用 Max Report 支持 XML 数据源支持的特性，以统一的数据模型为基础，通过数据对象为辅助数据处理类定义报表的数据源。其实现结构如图 4-16。

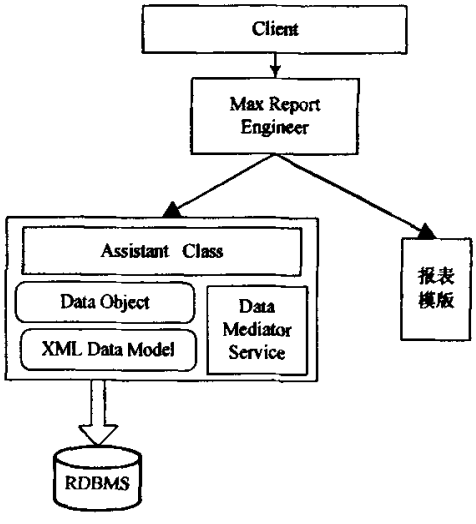

图 4-16 电大教务管理报表实现

Figure 4-16 Report Implement for CRTVU

在统一数据模型的基础上设计实现报表的数据源，当用户调用报表时，报

表引擎向数据服务层发送数据处理请求，数据服务调用数据中介服务完成数据映射，并从数据库中读取数据，返回给报表数据源辅助类，由辅助类进一步转换为报表所用数据，返回给报表引擎。最后、报表引擎加载报表显示模版生成报表，返回给用户，如图 4-17。

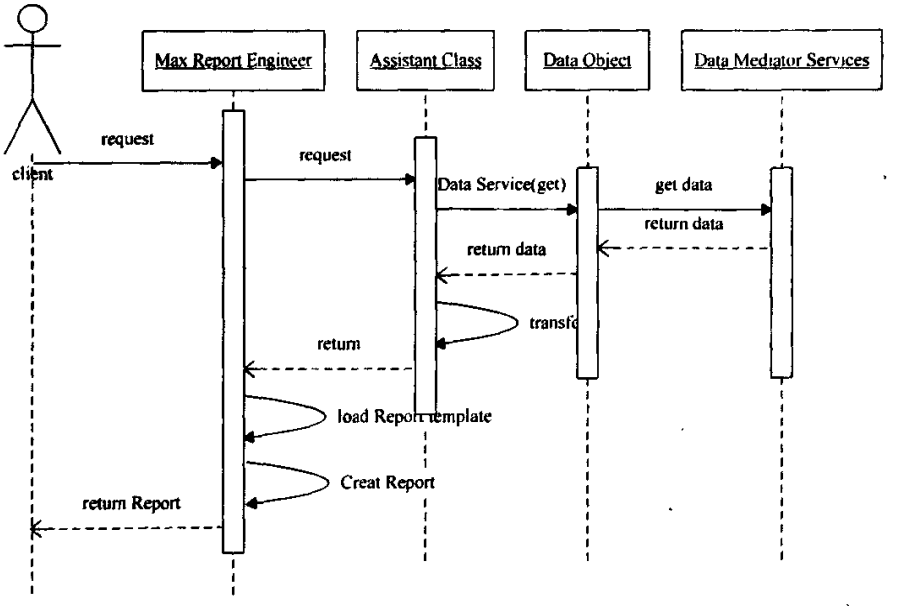

图 4-17 电大教务管理报表服务顺序图

Figure 4-17 Report service sequence diagram of CRTVU

4.4 本章小结

本章也是论文重点，在上一章中已经阐述了如何基于 SOA 建立分布式应用系统的数据模型，本章应用这种方法进行实践，详细讨论了如何建立中央广播电视大学教务管理系统的数据模型，并提出了实现数据服务的实现框架，对一些关键技术进行了分析，设计了多种实现方案，以满足中央电大多种多样的数据服务要求。

结 论

作为通用的软件系统开发的需要，目前的数据模型都是基于关系模型或者面向对象模型的，而且都已经比较成熟。但由于它们只是针对数据进行常规组织和管理的通用模型，并不能结合不同类型应用系统的数据特点和业务流程进行有针对性的建模和管理。因此，随着SOA的出现，单纯的使用其中一种或两种建模方法已经不能满足应用的需要。尤其是在开发基于SOA的分布式应用系统过程中，为了更好的解决各个应用系统中数据分布性、自治性和异构性的问题，就需要一种基于服务的建模方法，进行面向服务的分析和设计。

随着开放教育改革的展开，中央电大原有相互独立的“电大在线教育”、“电大 OA”和电大教务管理系统已经不能满足业务需求，需要重构、整合和扩展，建立一个远程开放教育平台。

本论文在中央电大教务管理系统开发过程中，重点研究了如何将数据模型、业务策略和服务定义结合起来，提出基于SOA建立分布式应用系统数据模型的方法和理论。论文的主要工作包括以下几点：

1. 分析了数据模型建模的相关理论和技术，对各种建模方法进行讨论和比较。

2. 分析了基于 SOA 的分布式应用系统的参考架构，提出在系统的分层体系结构中抽象一个数据服务层，以建立统一数据模型为中心，专门用来解决SOA的数据服务要求。

3. 把面向服务的思想引入数据模型的建立，论述了从数据实体层、数据聚合层和跨组织服务聚合层三个层次来描述数据模型，并分主题域E-R实体数据模型和服务数据模型两个阶段来建立基于SOA的数据模型。无论从三个层次描述数据模型，还是分三个阶段分析和建立数据模型，都是对数据模型逐层抽象，逐层组合的过程，也是逐层定义数据服务并进行封装的过程。

4. 应用上述方法对中央电大教务管理系统的数据模型进行分析和设计，研究并提出了基于SOA实现中央电大教务管理系统数据服务实现框架，分析设计了多种的解决方案，以解决不同的实际问题，并给出应用实例。

把数据模型、业务策略和服务定义结合起来，通过建立统一的数据模型，提供业务数据的完整视图，并通过统一的API存取所有数据源，从而隔离应用与底层数据源。建立数据模型的过程，就是数据服务抽象的过程；不同阶段、不同层次模型的转换和映射的过程，就是服务封装和组合的过程；而模型间的转换模式和映射模式，就是服务调用和响应的模式；整个过程都是按照一定业

务策略来进行的。

基于 SOA 开发中央电大教务管理系统，是面向服务体系结构这一领域一次很好的技术研究和应用探索，建立好数据模型后，如何以它为基础和驱动，逐步推进远程开放教育信息系统的建设，将是我们继续研究的内容。

另外 SOA 作为一个新兴的软件体系架构，其包含的内容非常广泛，其技术也处于不断的发展与更新之中。本文仅对基于 SOA 建立分布式应用系统数据模型进行了较为详细的论述，其他的问题并没有涉及和讨论。以后还需要进一步的研究。

参考文献

1 Thomas Erl. SOA 概念、技术与设计. 王满红 陈荣华译. 机械工业出版社, 2007: 218-230

2 操云甫. 基于Internet/Intranet的资源共享模型及技术研究. 中国科学院研究生院. 博士学位论文. 2002:50-65

3 Thomas Erl. Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services. Prentice Hall PTR,2004:105-106

4 朱正杰. SOA 的关键技术的研究与实现. 电子科技大学. 硕士学位论文. 2006:15-20

5 K. Votis, C. Alexakos, B. Vassiliadis and S. Likothanassis. An ontologically principled service-oriented architecture for managing distributed e-government nodes. Journal of Network and Computer Applications, In Press, Corrected Proof, Available online 12 June 2006:241-256

6 麻志毅, 陈泓婕. 一种面向服务的体系结构参考模型. 计算机学报. 2006, 29(7):1011-1015

7 Roger L. McIntosh. Open-source tools for distributed device control within a Service-Oriented Architecture. Journal of the Association for Laboratory Automation, Volume 9, Issue 6, December 2004, Pages 404-410

8 HectorGarcia, Jeffrey D., Jenifer. 数据库系统全书. 岳丽华, 杨冬青, 龚育昌等译. 机械工业出版社, 2003: 15-26 667-671

9 Johnson, James L. 数据库：模型、语言与设计. 李天柱, 肖艳芹, 杨文柱等译. 电子工业出版社, 2004:203-210

10 Silberschatz, Abraham.数据库系统概念. 杨冬青,马秀莉,唐世渭等译.机械工业出版社, 2006:149-153

11 C.D. Tarantilis, C.T. Kiranoudis and N.D. Theodorakopoulos. A Web-based ERP system for business services and supply chain management: Application to real-world process scheduling. European Journal of Operational Research, In Press, Corrected Proof, Available online 20 November 2006

12 陆静平. 基于 XML 的产品数据模式、存储及共享模型的研究. 重庆大学. 博士学位论文. 2003:26-28

13 聂铁铮. 服务网格中资源需求描述方法与服务选择策略的研究. 东北大学. 硕士学位论文. 2005:35-38

14 Daniela Florescu, Andreas Grünhagen and Donald Kossmann. An XML programming language for Web service specification and composition. Computer Networks, Volume 42,

Issue 5, 5 August 2003, Pages 641-660

15 Marcel Frehner and Martin Brändli. Virtual database: Spatial analysis in a Web-based data management system for distributed ecological data. Environmental Modelling & Software, Volume 21, Issue 11, November 2006, Pages 1544-1554

16 M. Weiss, B. Esfandiari and Y. Luo. Towards a classification of web service feature interactions. Computer Networks, Volume 51, Issue 2, 7 February 2007, Pages 359-381

17 吕庆中，韩燕波，麦中凡. Web 服务环境中的业务过程建模语言比较框架. 计算机工程与应用. 2003, 23:7-10

18 Leo G. Anthopoulos, Panagiotis Siozos and Ioannis A. Tsoukalas. Applying participatory design and collaboration in digital public services for discovering and re-designing e-Government services. Government Information Quarterly, Volume 24, Issue 2, April 2007, Pages 353-376

19 李冠宇，刘军，张俊. 分布式异构数据集成系统的研究与实现. 计算机应用与研究. 2004,3:96-99

20 Eric Nercomer, Greg Lomow. Understanding SOA with Web Services. 徐涵译. 电子工业出版社, 2006: 126-127 (Soap Http)

21 Rakesh Agrawal, Roberto J. Bayardo, Daniel Gruhl and Spiros Papadimitriou. a service-oriented architecture for rapid development of Web applications. Computer Networks, Volume 39, Issue 5, 5 August 2002, Pages 523-539

22 Dr. Ali Arsanjani. 基于服务的建模和架构. http://www.ibm.com/developerworks/cn/webservices/ws-soa-design1

23 Jiachen Hou and Daizhong Su. Integration of Web Services technology with business models within the total product design process for supplier selection. Computers in Industry, Volume 57, Issues 8-9, December 2006, Pages 797-808

24 Deependra Moitra and Jai Ganesh. Web services and flexible business processes: towards the adaptive enterprise. Information & Management, Volume 42, Issue 7, October 2005, Pages 921-933

25 叶应贤. 面向服务体系结构 SOA 技术研究及其在企业集成系统的应用. 重庆大学. 硕士学位论文. 2005: 22-23

26 Andrew N.K. Chen, Sagnika Sen and Benjamin B.M. Shao. Strategies for effective Web services adoption for dynamic e-businesses. Decision Support Systems, Volume 42, Issue 2, November 2006, Pages 789-809

27 Ivar Jacobson, Grady Booch, James Rumbaug. 统一软件开发过程. 周伯生，冯学民，樊东平译. 机械工业出版社，2002：46-63

28 陈京民. 数据仓库原理、设计与应用. 中国水利水电出版社, 2004:31-35

29 Clifton Nock. 数据访问模式. 鄢爱兰, 王安鹏译. 中国电力出版社, 2004: 24-25 39-40

30 Zhang Yingchao, Zhang Weiming and Xiao Weidong. Research on Unified Information Description Model for Information Sharing in Virtual Organization. Journal of Systems Engineering, Vol 20. Beijing, Changsha. 2005:67-75

31 Laura Bocchi and Paolo Ciancarini. On the Impact of Formal Methods in the SOA. Electronic Notes in Theoretical Computer Science, Volume 160, 8 August 2006, Pages 113-126

32 余彤鹰. 企业建模的目的、范围及"模型驱动系统"（MDS）. http://www.ee-forum.org/bbs/bbsview2.asp?type=2&id=46

33 Ricardo Jardim-Goncalves, Antonio Grilo, Adolfo Steiger-Garcao. Challenging the interoperability between computers in industry with MDA and SOA. Computers in Industry. Volume 57, Issues 8-9, December 2006, Pages 679-689

34 赵雷. 域数据模型的研究与实现. 苏州大学 博士学位论文. 2006:53-55

35 Edson Murakami, Antonio M. Saraiva, Luiz C.M. Ribeiro Junior, Carlos E. Cugnasca, Andre R. Hirakawa and Pedro L.P. Correa. An infrastructure for the development of distributed service-oriented information systems for precision agriculture. Computers and Electronics in Agriculture, Volume 58, Issue 1, August 2007, Pages 37-48

36 Serena Pastore. The service discovery methods issue: A web services UDDI specification framework integrated in a grid environment. Journal of Network and Computer Applications, In Press, Corrected Proof, Available online 30 May 2006

37 Wang Guiling, Li Yushun and Jiang Jinlei. A Dynamic Information Aggregation Model and Its Application in Service Grid Environment. Chinese Journal of Computers, Vol 28. Beijing, China. 69-75

38 Kuo-Ming Chao, Muhammad Younas and Nathan Griffiths. BPEL4WS-based coordination of Grid Services in design. Computers in Industry, Volume 57, Issues 8-9, December 2006, Pages 778-786

39 Issam Chebbi, Schahram Dustdar and Samir Tata. The view-based approach to dynamic inter-organizational workflow cooperation. Data & Knowledge Engineering, Volume 56, Issue 2, February 2006, Pages 139-173

40 张英朝, 张维明, 肖卫东, 沙基昌. 虚拟组织中面向共享的信息统一描述模型研究. 系统工程学报. 2005 20(1): 62-71

41 中国广播电视大学教育统计年鉴.中央广播电视大学出版社,2002.5:1-200

42 Sinuhe Arroyo, Alistair Duke, José-Manuel López-Cobo and Miguel-Angel Sicilia. A model driven choreography conceptual framework. Computer Standards & Interfaces, Volume 29, Issue 3, March 2007, Pages 325-334

43 徐罡, 黄涛, 刘绍华, 叶丹. 分布应用集成核心技术研究综述. 计算机学报. 2005,

28(4):433-436

44 S. C. Hui and G. Jha. Data mining for customer service support. Information & Management, Volume 38, Issue 1, October 2000, Pages 1-13

45 黄斌. 数据网格系统中数据统一访问和管理的研究与实现. 国防科学技术大学. 硕士学位论文. 2003.40-43

46 Matjaz B. Juric, Ivan Rozman, Bostjan Brumen, Matjaz Colnaric and Marjan Hericko. Comparison of performance of Web services, WS-Security, RMI, and RMI–SSL. Journal of Systems and Software, Volume 79, Issue 5, May 2006, Pages 689-700

47 F.L. Gutiérrez Vela, J.L. Isla Montes, P. Paderewski Rodríguez, M. Sánchez Román and B. Jiménez Valverde. An architecture for access control management in collaborative enterprise systems based on organization models. Science of Computer Programming, Volume 66, Issue 1, 15 April 2007, Pages 44-59

攻读硕士学位期间的科研成果

1 邸瑞华, 王虎. 基于 SOA 的远程开放教育信息系统数据模型的研究. 计算机与信息技术. 2007, 15(5): 82-84

2 邸瑞华, 李维明, 王虎, 邵飞. 贸易出口业绩分析软件 V1.0. 软件著作权. 编号: BJ7587. 首次发表日期: 2006.11.10

3 邸瑞华, 邵飞, 王虎. 贸易地图生成软件 V1.0. 软件著作权. 编号: BJ7586. 首次发表日期: 2006.11.10

致 谢

转眼间，三年的研究生学习已经接近尾声。在三年里，我最感谢我的导师邸瑞华教授。邸老师渊博的学识、严谨的作风、谦虚的风范、正直的处事原则以及孜孜不倦的工作热情对我树立正确的治学观和价值观都产生了直接而深远的影响。本论文也是在她的悉心指导下完成的。从论文的选题，撰写到定稿，邸老师给了我很多有价值的建议、指导和帮助。在此，谨向邸老师致以深深的敬意和衷心的感谢。

我还要感谢李维铭老师。他对我的帮助也是巨大的。他细致认真地帮助了解系统需求，为我指出研究中出现的种种不足，当我面对一些错综复杂的问题感到难以入手时，李老师以他丰富的经验给我指出了正确的前进方向。

感谢三年来与我一起研究课题的同学聂靖松、梁路和邓奇，感谢实验室的其他师兄师弟。在与他们多年的交流与合作中，我的知识和能力都得到了长足的提高和发展，并取得了一定的科研成果。

最后，衷心感谢我的家人。他们总是在我遇到困难时鼓励我前进，在我困惑时为我解惑，我得到的一切成绩都和他们为我付出的艰辛是分不开的。