摘 要

万维网的相关技术是互联网领域发展最快的技术之一。网络上已经和正在产生的大量的数据主要是超文本标识语言格式的数据，可以预计将来的网络上还将产生大量的数据，不过越来越多的将是扩展标识语言格式。如何有效地存储和检索网络上的扩展标识语言格式的数据成为一个研究热点。由于数据库技术不断趋于成熟，一种自然的想法是采用数据库对扩展标识语言格式的数据进行存储和查询等操作。其基本思想就是利用扩展标识语言格式的数据模式到数据库的数据模式的映射来存储。但是因为使用的数据库不同，在具体实现技术上不尽相同。关系数据库是当前最成熟、最流行的数据库技术。选择了关系数据库作为研究扩展标识语言格式的数据存储的数据库环境，主要研究数据模式映射和存储的方法。

描述扩展标识语言文档的模式/结构的方式有文档类型描述、扩展标识语言模式、简化的扩展标识语言数据和面向对象的扩展标识语言模式等，其中使用得最广泛的是文档类型描述和扩展标识语言模式。虽然文档类型描述是现在使用的最多的扩展标识语言结构的文件，但是扩展标识语言模式取代文档类型描述成为扩展标识语言的主要模式文件已是大势所趋。选择了扩展标识语言模式文件作为扩展标识语言的模式/结构描述文件来进行研究。

在分析国内外典型的基于关系模式的扩展标识语言格式的数据存储方法的基础上，给出了利用扩展标识语言模式映射扩展标识语言格式的数据模式到关系模式，从而把扩展标识语言格式的数据存储到关系数据库中的方法。描述了整个映射过程的步骤，设计了一个扩展标识语言格式的数据存取的模型，并且提供了通过扩展标识语言模式来把扩展标识语言格式的数据存储到关系数据库的算法，最后利用微软.NET 环境对这个方案进行了初步实现。

关键词：扩展标识语言，数据库，扩展标识语言模式，关系模式，文档类型描述

Abstract

The technologies on WWW are the fastest developed technologies on Internet. There are plenty of data on Web now, most of them are of HTML format. And it is sure that there will be more and more data appearing on Web, but most of them will be of XML format. How to store and query XML data efficiently is becoming urgent. Because the database technologies are highly developed, it's natural to store XML data in databases. And the basic principle is to process the storage by mapping the XML data mode into the database data mode. But all of the methods differ in detail for the different databases in use. The relational database is the most mature and popular database technology nowadays. The relational database is chosen as the database environment for storing XML data, and the research focuses on the method for mapping the XML data mode to relational mode.

There are some files which descript the mode or structure of XML data, such as DTD, XML Schema, XDR or SOX and so on. DTD and XML Schema are the most popular ones. Although DTD is the dominant XML data mode description file, it is certain that XML Schema will be the main XML mode file in the near future. The XML Schema file is selected as the XML mode/structure description file.

Based on the analysising all the research of storing XML data in relational databases home and abroad, a model is proposed which is for storing XML data in relational databases by XML Schema, and the steps to make it is put forward whereafter. Then a algorithm on mode mapping is given. In the end, this model is implemented in a project name AMS which is developed in Microsoft .NET framework.

Key words: XML, database, XML Schema, relational mode, DTD

独创性声明

本人声明所呈交的学位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，除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：肖必强

日期：2004 年 4月 29日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密□，在_____年解密后适用本授权书。

本论文属于

不保密□。

（请在以上方框内打“√”）

学位论文作者签名：肖必强　　指导教师签名：

日期：2004 年 4月 29日　　日期：2004 年 4月 29日

1 绪论

随着 Internet 的普及，它已经与人们的生活息息相关。随着 Web 技术的发展 XML 语言应运而生并且得到了广泛到应用。本文主要是在分析 XML 数据的结构特点的基础上，研究了 XML 数据存储到关系数据库中的方法。

1.1 课题背景

为了在 Web 上表示信息[1]，人们先后发明了 SGML（Standard Generalized Markup Language，标准通用标记语言）、HTML（HyperText Markup Language，超文本标记语言）语言。SGML 具有可扩展性、结构性以及可支持性，但是过于繁复，价格昂贵，不太适合 Web 开发，难以得到大范围的应用。HTML 语法精炼，使用容易，在扩展性方面、可交换性等方面却存在许多不足。随着信息来源渠道及信息显示媒体的日益多样化，我们需要寻找新的信息表示方法，使得信息更易于存储和查找[2]。这对于充分高效地利用信息资源有着重要的意义。在这种背景下，XML 应运而生[3]，成为 W3C 的推荐标准[4]。XML 也是 SGML 的一个子集，对 SGML 进行了精简，而在语言的易用性、易懂性等方面做了较大改进 [5,6]。

尽管 XML 设计的初衷是用来进行 Web 数据的标识，但是由于 XML 优良的特性，使得 XML 的应用远远超过 Web 信息的标识。基于 DTD 或者 XML Schema[7]，几乎所有的数据都可以用 XML 来表示。目前 XML 将成为新一代 internet 数据组织和交换的事实标准，并且大量的 XML 数据将很快出现在 Web 上。因此，对于正在发展的 XML 技术和未来以 XML 数据为主体的 Web 数据，如何利用已有的计算机技术（例如数据库技术）对它们进行有效的存储和查询，使我们要迫切解决的问题。实质上，XML 为 Web 的数据管理提供了新的数据模型，可以预见，很多成熟的数据库技术将进入 Web 信息处理领域，把 Web 变为一个巨大的数据库。XML 是朝这个方向迈出的第一步。这种变化给数据库研究界带来了巨大的机会，使得将数据库技术和 Web 数据管理相结合成为可能。其中，利用 XML 数据的结构性和数据库的数据结构性的映射来进行存储是切实可行的，并且得到了大量的研究。

本课题来源于湖北清江水电开发公司资产维护系统(AMS)，该系统是基于Microsoft 的.NET 平台的 B/S 结构的企业信息系统，它模拟电力企业生产过程，提供对生产流程的控制及生产数据的信息化管理。AMS 由设备维护、运行、物资、权限等业务上相互独立子系统构成，各子系统之间通过 XML Web Services 进行交互并实现信息共享，系统中大量的基于 XML 的数据通信和信息处理，本文力图在 SQL Server2000 的关系数据库管理系统环境中通过扩展实现对 XML 数据的存储和检索，且提出了一个基于 XML Schema 的对关系模式的 XML 数据存取模型和映射算法[8]。

1.2 国内外研究概况

利用数据库技术来对 XML 数据进行存储和查询等管理，这是当前对 XML 数据的存取技术研究的主要研究方法。

1.2.1 XML 数据存取技术研究成果概况

XML 数据的存取方法的研究现在有很多种，并且所构造的系统中 XML 数据自身的结构和数据库数据模式的磨合程度不尽相同[9]，因而构造的系统也各有特点，归纳来看有以下几种：

1. XML 本原数据库（Native XML Database，NXD）

数据库的根本设计目的是为存储和处理 XML 文档[10]。它的基本存储单元是 XML 文档，通过 XML 相关的标准进行数据库的存储。这种数据库维持原有 XML 文档的数据结构和相关的元数据，而不关心数据的底层存储格式（关系数据库或是面向对象数据库），只能通过 XML 特有的相关技术对数据进行存储。Tamino、dbXML 和 X-Hive 都采用这种方法。

2. 支持 XML 的数据库（XML-Enabled Database，XEDB）

它的基本数据存储单位是 XML 数据（XML 文档所提供的数据），主要是通过增加一个映射层来管理 XML 数据的存储。数据首先要与一个明确的格式相匹配，符合要求的才能根据预先定义好的规则映射到数据库中（关系数据库或是面向对象数据库），但可能会损失一部分元数据和最初的结构。同时可从现有的数据库中生成 XML 页面，但不能保证与当初存入的原始页面完全符合。Oracle、Microsoft 和许多 XML

工具软件都不同程度上支持这种功能。比如，Burret[11]的 XML-DBMS 就是一种 XML 文档与关系数据库之间进行转换的中间件。

在 XML-DBMS 中，将 XML 文档看作一个有详细数据的对象树，应用对象—关系映射理论将对象映射到关系数据库中。它提出了一个映射工厂的概念，使得要存入数据库中的 XML 文档与一个映射模型进行匹配，当需要进行数据转换时，协调模块首先根据 DTD 文档通过映射工厂的一个映射实体来描述 DTD 与关系视图的映射关系，并对于需要特殊映射处理的数据产生元数据（如列的数据类型）和一些 Insert、Select 等命令的一个实体对象，最后应用组件（DBMS To DOM 或 DOM To DBMS）进行有详细数据的文档对象树与数据库的转换。在将 XML 数据转换到数据库中时，协调者应根据 XML 文档建立一个对象模型树（DOM），并与映射实体一起提交给数据对象模型到数据库管理系统（DOM To DBMS）组件，该组件返回给协调者关于怎样重新获取数据的信息。在从数据库中获取数据时，协调者将映射实体和文档获取信息提交给数据库管理系统到数据对象模型（DBMS To DOM）组件，并获取文档对象模型树。

3. 混合 XML 数据库（Hybrid XML Database，HXD）

根据具体的要求和应用，既可做 Native XML Database，又可做 XML-Enabled Database。Excelon 和 Ozone 就采用了这种思想。以 XML 文档为基本存储单位，保留了文档的原始结构和 XML 原有的优点，存储简单，同时可以将 XML 文档看作对象树，有利于对单个 XML 文档进一步的数据挖掘。但信息的格式、内容相对繁杂，检索策略的复杂度高，处理起来比较困难。以数据为中心的方法将 XML 文档的数据进行重新组织，使数据的组织相对规范，有利于在此基础上对信息的进一步应用，比如智能检索、电子商务等，但破坏了原文档的结构，也很难保证文档恢复到原来的结构，在存入关系数据库中的预先处理工作量比较大，尤其是当 XML 文档不规范化时就需要对它进行复杂的分解、映射等处理。

1.2.2 主要关键技术

为了实现对 XML 数据的存储和查询等管理功能，可以把 XML 数据通过转化和扩展数据库把数据直接放在数据库中[12]，可以通过中间件，或者 Web Services 等技术来构造一个中间转换的层，从而实现 XML 数据和数据库的透明转换，XML 查询技术可以利用数据库自身的查询语言或者对半结构化查询语言加以改进而成。

1. 基于 XML 的中间件技术

中间件是一种独立的系统软件或服务程序，分布式应用软件借助这种软件在不同的技术之间共享资源，中间件位于客户机服务器的操作系统之上，管理计算资源和网络通讯[13,14]。中间件屏蔽了底层操作系统的复杂性，使程序开发人员面对一个简单而统一的开发环境，减少程序设计的复杂性，将注意力集中在自己的业务上，不必再为程序在不同系统软件上的移植而重复工作，从而大大减少了技术上的负担。中间件带给应用系统的不只是开发的简便和开发周期的缩短，也减少了系统的维护、运行和管理的工作量。基于 XML 的数据格式转换中间件是一个提供数据格式转换的平台[15]，它能够以 XML 作为数据交互的中间格式，使得异构数据可以进行交互。

XML 的开放和数据独立性保证了结构化数据对于应用、平台和商家来说是统一和独立的。同时，XML 加强了数据完整性，提供了在工业标准和本地数据定义之间的数据交换导航，也提供了松耦合数据界面，给格式要求严格的应用带来了灵活性。因此 XML 被看作是自 SQL 以来最重要的数据交换标准。

利用 XML 和 CORBA 等分布式信息处理和交换技术，设计基于 XML 的分布式数据交换中间件模型，解决 Web 应用程序之间的数据交换问题。使用该中间件模型，可屏蔽各系统之间的差异，实现各系统间透明的数据交换[16]。同时开放式的系统架构使中间件模型可适用于各种不同的应用环境。总的来说，XML 基于 XML 的中间件技术要解决一下几个问题：

(1) 研究一个能适应分布式异构环境的数据交换中间件系统架构。这个系统架构应用于多层分布式异构环境，能够在理论和技术上扩展 C/S 模式，使系统具有良好的可伸缩性，便于系统的开发和升级。

(2) 研究一种跨平台、易于扩展的信息交换格式的描述语言。这种信息格式描述语言应当具有通用性、开放性和可扩展性，能够方便地与外部应用交互或者相结合。换言之，它不仅可以作为信息格式，同时可以作为存储格式。

(3) 研究可以适应于各种应用环境的、用户可定义的信息交换模板。根据各应用环境的需要，系统可以自定义数据类型和信息格式。

(4) 研究如何使系统具有较强的可移植性。

2. XML Web Services

XML Web Service 是基于 XML 数据交换标准的 Web 服务[17,18]，是通过标准的 Web 协议可编程访问的 Web 组件[19]。它的核心观念是将软件作为服务，即可以将软件作为

Web 服务并在 Internet 上进行发布，真正的使企业只面向核心的商务逻辑，企业可以使用标准的 Web 协议与远程应用通信。软件向服务转化是目前的业界重点，基于 Internet 的 Web 服务是实现这一转变的最佳选择。

Web 服务以开放的标准为基础，构建在 Internet 之上，为企业应用提供了一个灵活的、松散耦合的分布式计算环境；同时，Web 服务屏蔽了底层的企业应用平台，具有良好的平台无关性，并且通信数据都是标准的 XML 数据，具有良好的语言无关性，为企业应用集成提供了一个便捷而快速的途径。

(1) Web 服务的体系结构

如图 1.1 所示，一个典型的 Web 服务体系结构包含三个实体。即服务提供者、服务代理和服务请求者。服务提供者创建 Web 服务并通过服务代理注册该项服务，从而把 Web 服务发布到 Internet 上去。服务代理维护已发布服务的注册信息，以供服务请求者查询。服务请求者通过搜索服务代理所维护的注册表以找到所需的 Web 服务，然后连接并使用该服务。

图 1.1 Web 服务体协结构

(2) Web 服务核心技术

构建 Web 服务的三大技术基础或者说核心技术是 SOAP、WSDL 和 UDDI[20]。

SOAP（Simple Object Access Protocol，简单对象访问协议）是一种简单、轻量级的协议，用于在 Web 上传输、交换 XML 数据。提供了一个基于 XML 的形式在分布式环境中交换结构化信息的机制。客户应用程序正是通过 SOAP 协议来访问 Internet 上的 Web 服务的。SOAP 协议可以构建在 TCP、SMTP、HTTP 等协议之上，最常用的是 HTTP。SOAP 协议是一个简单易用的协议，完全以 XML 为基础，具有良好的可扩展性。SOAP 协议定义了一个消息处理的模型，但没有定义任何的应用程序，如编程模型或特定的实现语义等。

WSDL（Web Services Description Language，Web 服务描述语言）是一种描述 Web 服务的 XML 语言，它定义了描述 Web 服务接口规范的标准格式。如果说 SOAP 是一种用来交换纯内容信息的通信协议，那么 WSDL 就是一种描述信息内容的语言。从概念上讲，就像在 CORBA 中使用 IDL 文件一样，WSDL 是用来描述 Web 服务的编程接

口的，WSDL 文件是客户与服务器之间的一个协约。

UDDI（Universal Description,Discovery and Integration，统一描述、发现和集成协议）是一套基于 Web 的、分布式的、为 Web 服务提供信息注册中心的实现标准规范[21]，同时也包含一组使企业能将自身提供的Web服务注册以使得别的企业能够发现的访问协议的实现标准。简言之，UDDI 标准定义了一个 Web 服务发布与发现的方法。

(3) Web 服务发布及应用的请求/应答过程

XML、WSDL、UDDI、SOAP 共同支撑了 Web 服务，它们在一起相互协作，完成了 Web 服务的发布和应用与 Web 服务之间的请求/应答过程。在发布 Web 服务时，服务提供商把所提供服务的接口和需要的数据类型和结构用 WSDL 进行描述，然后和服务商自己的描述信息结合起来构成 UDDI 注册中的相关资料，在全球的 UDDI 注册中心进行注册。当一个应用系统需要调用 Web 服务时，它首先通过 Internet 在 UDDI 注册机构中查找需要的 Web 服务，取得 Web 服务的地址和用 WSDL 描述的接口，根据 WSDL 描述的接口信息准备数据，依据相应的地址进行连接。完成连接后，应用系统通过 SOAP 协议和 Web 服务中的远程对象绑定在一起，进行请求的发送和应答的接收。一个 Web 服务也可以同样请求其他的 Web 服务为自己完成某些服务。

3. XML 数据的查询技术研究

随着 XML 技术不断成熟，越来越多的数据或以 XML 格式存储，或以 XML 格式交换，或者用 XML 来表现。因此，要构建基于 XML 的各种应用，科学的从 XML 数据源中查询并获取数据就成为其中很关键的一步。XML 数据库查询语言就是实现这一功能的主要工具[22,23,24,25]。这些语言都有自己的数据模型、符号集、算法以及实现方式，但同时也表现出很多共同的特性。

(1)根据需要构建输出。XML 查询的结果不一定作为最后的结果输出到界面，它可以是一个中间结果，以对它进行进一步的查询。也可以作为符合某一格式要求的数据存入到数据库中，或者成为众多查询结果中的一部分。正因为这样，自如的控制查询结果的结构和式样就十分重要。XML 查询语言将对数据的查询和对结果的控制同步进行，既可以将查询结果保持原文的结构，也可以进行改变，形成新的符合用户需要的结构。

(2)不同 XML 数据源的集成。XML 的查询语言可以同时从多个数据源获取数据，将相关的数据进行必要的连接，并形成一个整体的查询结果。通过 XML 查询语言的这一特性，使很多独立的零碎存在的数据在再次利用过程中产生了意想不到的价值，

而且使得查询本身也变得简单便利。

(3)模糊查询。这里的模糊查询主要指在不了解 XML 数据的具体结构或不明确数据格式等情况下所进行的查询。

(4)对嵌套循环数据结构的支持。由于 XML 数据本身允许嵌套、循环、递归等结构，而且这种结构的数据大量存在，在此基础上的查询就要求能够到达任意路径上的任意结点，XML 查询语言采用规则路径表达的方法，使得定位、查询及相应的查询优化等都十分便利。

XML 的迅速发展激发了对 XML 查询语言的需求。迄今为止，工业界和学术界提出了不下十种 XML 查询语言。按照提出者的研究背景，这些查询语言可以分成两大类：一类来自文档研究领域，如 XSL[26]和 XQL[27]；另一类来自数据库研究领域，如已 Lorel[28]、XML-GL、XML-QL 和 Quilt。各种语言在语法形式和查询能力上都不尽相同。一般来说，由文档研究者提出的查询语言功能较弱，如 XSL 不支持连接操作。而由数据库研究者提出的查询语言功能相对强大，一般都支持复杂的路径表达式、连接、嵌套查询等数据库查询语言的典型特性，但在对一些高级特性的支持程度上也存在一定的差别，如 Lorel 和 Quilt 支持全称量词，而 XML-QL 和 XML-GL 则不支持。同时，在语法形式上也有所差别：Lorel 和 Quilt 类似于 SQL 和 OQL，XML-QL 的主体部分为标准的 XML 格式；XML-GL 则是一种图示化的查询语言。

为了推动 XML 语言的发展，W3C 于 1998 年底成立了专门的查询语言工作组，着手进行 XML 查询语言的规范化工作。但是，由于 XML 本身表达范围非常广，涵盖了结构化文档、半结构化文档、关系数据库和对象数据库，从而使得设计一种对各类数据源上都同样有效的 XML 查询语言的工作变得非常困难。所以，直到 2001 年 2 月 W3C 才提出了 XQuery 语言的第一个工作草案。

4. XML 数据的存储技术研究

利用 XML 数据结构化的特征进行 XML 文档的数据库存取，其基本思想是一致的，即把 XML 标记的数据放入一定的数据模式中，从而利用这个数据模式对这些数据进行管理。但是在具体实现技术上不尽相同。根据所采取的数据模式来看，可以分为以下几类：

(1) 文件系统结构。这种方式是将 XML 数据存放到文件系统中去，使用文件是存储 XML 的一种最简单的方法。以这种方式保存的 XML 数据文档可以用普通的文档编辑器来编辑它，也可以用专门的 XML 编辑器来编辑。基于 XML 的文件系统主要包括

三部分：文件组织、安全系统和文件的基本管理功能。

使用文件系统来存储文档的优点是简单易用，而 XML 且也适合于文档内容少、数量小的场合，但其在系统规模、并发与安全等方面的局限性 XML 也是显而易见的。

(2) 半结构化数据模式。半结构化数据是界于严格结构化的数据（如关系数据库中的数据）和完全无结构的数据（如声音，图象文件）之间的数据形式。半结构数据通常称为“无模式”或“自描述”[29]。也就是说，半结构数据不存在独立的结构和类型描述。一般来说，在对数据进行存储或处理时，应事先描述出数据的结构（类型、模式）然后再生成该类型或模式的实例值。XML 数据模型与半结构数据模型有着很多的相似性。可以说，XML 是 WWW 上的半结构数据。它既为半结构数据的研究提供了广阔的应用前景，同时也推动了半结构数据研究的发展。例如，利用 XML 与半结构化数据的相似性，在原先半结构化数据的研究成果上支持 XML 的存储与查询功能。这主要有 Stanford 大学的 Lore 数据库研究[30]。

(3) 面向对象模式，把 XML 元素当作对象看待存放在面向对象数据库中。面向对象数据库用自身的方法、关系和语义来管理 XML 数据，同时提供了强大的导航和连接功能，例如 Ozone 是建立在对象数据库系统 O_2 之上，它集成了半结构化方法。面向对象的模型支持复杂数据类型，因此可以方便、直观地建立 XML 数据的对象模式，进而利用对象查询语言 (OQL)实现对 XML 数据的结构化查询。面向对象的方法已普遍地应用于软件开发的各个阶段，OODB 能够与面向对象的程序设计方法无缝结合，因此 OODB 很有可能成为未来数据库的主流。以一致的方法访问 XML 数据和 OODB 中的数据无疑将使应用系统的开发得到简化。

(4) 关系模式。关系数据库具备完备的理论基础、简洁的数据模型、透明的查询语言和方便的操作方法等。到目前为止，在各个领域使用最广的还是要数关系型数据库。将 XML 数据保存在关系数据库中，利用关系数据库系统的查询机制实现对 XML 数据的查询。由于关系模型不支持复杂类型的属性，因此采用这种方法处理 XML 数据存在一定的局限性。一个文档通常被转换为多个表，元素与表之间的关系不够直观，查询常常涉及多个表的连接，导致效率降低。

XML 关系存储的核心是 XML 文档到关系数据库的模式映射方法。XML 的“树/图”数据模型与平坦的关系数据模型间存在固有的不匹配[31]；XML 固有的“文档根”，使得 XML 与关系数据在语法、结构与约束的众多方面均存在许多异构性。现在使用的方法主要是模型映射方法。它的原理就是利用 XML 文档中的数据模型的结构（如

DTD 或者 XML Schema）显性或者隐性地将其映射成数据库的结构，它的优点是简单易用。现在研究得比较多得是利用 DTD 模式来把 XML 数据存储到关系数据库中，但是随着 XML Schema 的出现和它相对于 DTD 的优点，将来会基于关系模式存取 XML 数据会越来越多的利用 XML Schema 来映射。

目前，XML 到关系模式的典型映射方法有三类：用固定的关系模式存储 XML 数据[32]、从实例数据提取关系模式[33]、从 DTD 导出关系模式[34]。商业的 RDBMS 的映射方案不能够大量地自动映射，所以不在研究之列。本文重点分析并且借鉴了从 DTD 到处关系模式的方法。DTD 是 XML 文件的模式类型描述文件，根据 DTD 可以得到它所描述的所有 XML 文档的结构。因此根据 DTD 文件映射存储 XML 数据完全可行。但是 DTD 是树/图状结构，而关系模式是 E-R 模型的结构，它们的结构不相同，同时 DTD 的结构比较复杂，不能满足转换需要。因此在进行 DTD 的模式映射时，首先要对 DTD 进行简化，然后根据 DTD 文件产生 DTD 的树状元素图，再深度优先算法遍历 DTD 元素图产生关系模式。最后利用利用 DOM 将 XML 文档解析生成 DOM 树结构，利用 DOM 树存储算法来存储 XML 数据。

1.3 课题主要研究工作

本课题主要是研究 XML 数据的存取技术。在课题的研究过程中，我们认真参阅了国内外对 XML 数据的存取技术，特别是利用关系数据库来存放 XML 数据相关研究。虽然基于关系数据库来存取 XML 数据有一些局限和弱点，但是至少在现阶段有充分的理由需要研究 XML 在关系数据库中如何存储的问题。结合本课题的实际情况，我们采用关系数据库来作为存放的环境，这样可以利用关系数据库成熟的管理、存取、更新和检索技术来对 XML 数据进行管理。

通过将 XML 文档看作为有具体数据的对象树，应用对象-关系映射理论映射到关系数据库中。XML 文件一般都附有模式或者结构描述文件，如 DTD 和 XML Schema。这两种模式/结构描述文件差别很大，在语法、结构等方面都不一样。XML Schema 相对于 DTD 具有很多的优势，并且在丰富的数据类型、继承和复用、命名空间等方面来说更是 DTD 所不能媲美的，现在已经有很多研究者正在研究 DTD 向 XML Schema 转换的方法和工具。所以本文主要研究利用 XML Schema 来映射存储 XML 数据到关系数据库中的方法。

本文借鉴了利用 DTD 来映射存储 XML 数据的方法。整个映射过程包括 XML Schema 的简化、XML Schema 元素图的生成、关系模式的生成以及 DOM 树的存储。本文设计了从 XMLSchema 到关系模式的模型，然后给出了从 XML 文档到关系模式的转换算法。最后，本文在 Microsoft .NET 框架和 SQL Server 2000 的环境下进行了初步应用。

2 XML 数据的结构分析

XML 文件通常由两部分组成。一部分是 XML 标签及其内容，另一部分是定义标签及其相互关系的数据模式/结构文件，如 DTD 文件和 XML Schema 文件。XML 数据的结构分析主要是分析数据模式/结构文件 DTD 和 XML Schema 的特点和区别，从而更好地对 XML 数据进行处理。

2.1 XML 数据的结构特点

XML 不是一种编程语言，因此不能创建二进制可执行文件。XML 能让使用者简单地定义标签和它们之间的关系。使用 XML 编码的文章的结构使得进行跟踪、格式化和操纵都相当容易。

一个 XML 的文档示例如图 2.1 所示，它来自于本文研究的项目——水电站资产管理系统（AMS），从中可以看出 XML 数据是一个图/树状结构。

```
<?xml version="1.0" encoding="gb2312" ?>
……
<department>
   <deptName>物资科</deptName>
   <operator id =”xiaohui”>
      <name>
         <firstname>张</firstname>
         <lastname>三</lastname>
      </name>
      <contact>5556154</contact>
   </operator>
</department>
……
```

图 2.1　部门信息的 XML 文档

正如从上面的例子中看到的，XML 文档类似于树型分层结构，元素可以完全嵌套在另一个元素中，且只有单个顶层元素——即文档元素或根元素（它包含了所有其他元素）。因此，可以很容易使用 XML 定义分层结构文档，如上面提到的可分为 department、deptName、operator、name 和 contact。XML 对于大型和复杂的文档是理想的，因为数据是结构化的。

XML 有强大的数据描述能力，使得复杂数据的表达变得方便。还有自我扩展能力，把对数据的约束减到最少。XML 使数据可以进行自我描述。应用系统间交换的 XML 数据可以通过其文档中的标签推断其涵义。XML 是数据与表达分离的。XML 标记描述的是文档的结构和意义，它不描述页面元素的格式化。可用样式单为文档增加格式化信息。文档本身只说明文档包括什么标记，而不是说明文档看起来是什么样的。XML 文件通常有两个部分。一部分是 XML 标签及其内容，另一部分是定义标签及其相互关系的 DTD（文件类型定义，Document Type Definition）或者 XML Schema。DTD 可以和 XML 资源存放在同一个文件内，但也可以单独存放。

从文档管理的角度来看，XML 是一种结构化文档标记语言；但从数据管理的角度来看，由于 XML 体现了各种数据表示格式的最大共同点，因此可作为一种通用的数据表示格式；此外，若忽略一系列诸如元素次序、元素与属性的区别、元素中内容与子元素的混合、以及评注和处理指令等 “文档” 特性，则 XML 可作为半结构化数据的串行化表示语法而用于数据交换。例如，图 2.2 就是图 2.1 文档的一种可能的边标注的有向树/图表示。每个节点表示一个对象（XML 元素）并赋予一个唯一标识（OID），节点的每一出边表示相应对象的一个属性并用属性名（XML 子元素/属性名）标注，叶子节点有相应的原子数据值（XML 子元素内容/属性值）。

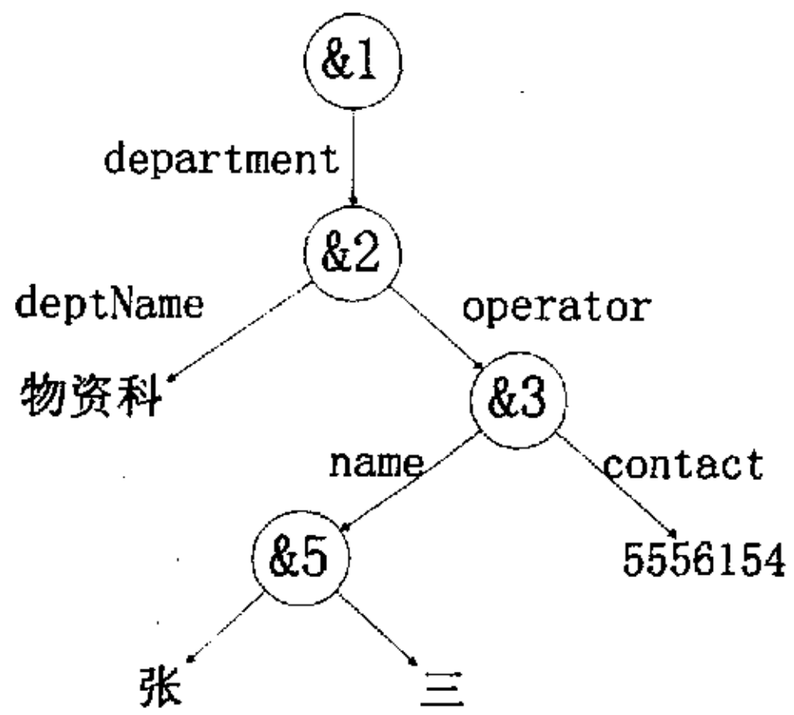


图 2.2 部门信息 XML 文档的数据结构图

如果 XML 文档实例中有（且考虑）元素间的引用，则它是一个图模型，否则是一个树模型；如果考虑 XML 元素次序，则成为一种有序模型。显然，这样的模型与无序、平坦的关系模型之间存在固有的不匹配。另外，由于模式信息（数据类型、结构及其约束）对数据的有效存储、索引和查询求值至关重要，因此，XML 放松的类型强制和无模式（即自描述）性虽有利于数据交换，却不利于数据存储。

2.2 ML 数据的模式

模式是关于标记的语法规则，它详细描述了 XML 文档的结构，从而确定了文档的框架。一个模式文件严格地规定了以它为标准的所有 XML 文档的树状层次结构的全部细节。当某一 XML 文档引用该模式文件时，它必须通过有效性检验[35]。XML 文档的模式有 DTD、XML Schema、XDR(XML Data Reduced，简化 XML 数据)和 SOX（Schema for Object-Oriented XML，面向对象 XML 语言的模式）等。本文只讨论使用得最广泛的 DTD 和 XML Schema 模式。

XML 的一个重要特性就是其可扩展性，XML 文档的作者可任意定义文档数据的结构以及元素的名称和属性。虽然这种可扩展性给文档的制作提供了很大的灵活性，但是同时它也使得不同组织的应用程序间的数据交换变得难以实现，原因是不同组织的应用程序对同样的标记名称可能有不同的理解[35]。另外，对于一个将不断发展的文档模型，XML Schema 或 DTD 可指导此发展过程[36]。

2.2.1 文档类型定义（DTD）

DTD 源于 SGML 规范，它描述了一个 XML 文档的结构。一个 DTD 通过指定一个元素的属性以及子元素的名称来规定其结构。所有的值都是字符型的，可以用特殊属性 ID 来唯一指定一个元素。图 2.3 给出了与图 2.1 对应得 DTD 的例子。

```
……
<!ELEMENT department( deptName, operator )>
<!ELEMENT material( desc, operator*, assistent )>
<!ELEMENT assist EMPTY>
<!ATTLIST assistent operatorID IDREF IMPLIED>
<!ELEMENT room( desc, operator, keeper )>
<!ELEMENT keeper( room* )>
<!ELEMENT operator( name, contact )>
<!ATTLIST operator id ID #REQUIRED>
<!ELEMENT name( firstname?, lastname )>
<!ELEMENT firstname(#PCDATA) >
<!ELEMENT lastname(#PCDATA) >
<!ELEMENT contact(#PCDATA) >
……
```

图 2.3　部门信息 XML 文档对应的 DTD

DTD 中的声明分为四类，它们分别是：元素类型声明、属性列表声明、实体声明、记号声明。DTD 指定了文档结构的一系列规定，并且将文件的结构和文件的内容完全分开，这样有很多的好处：

1. 使 XML 文档标准化变为可行。任何人可以依照 DTD 的结构，编写出合乎标准的 XML 文档。

2. DTD 严格的规范使不同的应用程序或用户可以读取彼此的文件。

3. 外部的 DTD 可以被不同的文件或是网站所分享。

4. DTD 中只包含结构，设计者可以在不影响数据的情况下优化 DTD。

5. 以适当的 DTD，可以转换 XML 文件成为不同的文件格式。

6. DTD 中的实体参照用途更多，设计者可以利用它来将外部的数据或是图文件加到 XML 文件中，使 XML 文件内容更加丰富。

7. 它可以代表某种类型的文档，因而在文档的结构化检索、分类、比较以及转换等处理方面起非常重要的作用[37]。

DTD 包含了定义元素的元素声明和属性列表声明，这些元素组成了词汇表，而属性列表声明则指明了这些元素的属性。内容模式描述了什么样的内容可以包含在元素当中。实体声明则描述了那些可以包含在用来构造 DTD 文档中的文本和二进制实体。下面将分别说明其具体结构。

1. 元素声明

元素声明指的是单一的标记元素，在文档中所使用的每个标志，都通过 DTD 中的元素声明而定义。结构如下：<!ELEMENT 元素名元素内容描述>。XML 标准将元素按内容划分为四类：空元素类型（EMPTY）、ANY 元素类型、子元素类型和混合元素类型。

2. 元素属性声明

元素属性声明是用来声明整套属性的，每个属性列表声明都用来为指定的元素定义一套属性，列表中的属性只能在该元素中使用。使用下面的格式来给一个元素定义一组合适的属性，同时指定这些属性的类型和缺省值：<!ATTLIST 元素名(属性名属性类型缺省值) >。可以分为下面几种情况：必须赋值的属性（REQUIRED）、属性值可有可无的属性（IMPLIED）、固定取值的属性（FIXED“缺省值”）和其他类型（定义缺省值的属性如果不使用上面任何一种关键字的话，该种属性就是属于这种类型。

3. 实体声明

实体声明创建一个实体，它本质上是一组数据唯一的别名，实体分为一般实体和参数实体两种类型，它们都可以定义为内部的，也可以用关键字 SYSTEM 定义为外部的。

4. DTD 的逻辑结构

DTD 用来描述其内部的逻辑结构和置标规则，它定义了逻辑结构中上层结点和下层结点间的从属关系：在 DTD 描述中，非叶子节点定义部有一个从属结点产生器。用以定义子结点的构成规则，根据 ODA 的定义。这些规则包括：

(1) 可选，optional(0PT，0 次或 1 次发生)；

(2) 要求，Required(REQ，1 次发生)；

(3) 重复，Repetitive(REP，1 次或多次发生)；

(4) 可选并重复，OPtional and Repetitive(0PT&REP，0 次或多次发生)；

(5) 顺序，Sequence(SEQ，顺序发生)；

(6) 聚集，Aggregate(AGG，无次序发生)；

(7) 选择，Choice(CHO，仅其中一次发生)。

这些规则在XML 1.0中被分别表示为下属要素的出现顺序和出现回数。

2.2.2 XML Schema

在 XML Schema 出现以前，DTD 一直是 XML 技术领域所使用的最广泛的模式。但是，由于 DTD 在 XML 之前就已经出现，因而它不可避免地不能完全满足 XML 处理的要求，例如不支持名字空间，缺乏对文档结构、属性、数据类型等约束的足够描述等。而 XML Schema 弥补了 DTD 的不足，逐渐成为 XML 的标准模式，并在很多应用中取代了 DTD。

W3C 于 2001 年 5 月 2 日正式推荐使用 XML 的规范语言“XML Schema”。XML Schema 是一些规则的集合（也称为语法或者语汇），其中包括了类型定义（简单和复杂类型）以及元素和属性声明。XML Schema 用来描述 XML 文档的合法结构、内容和限制，定义了可共享的词汇表，使用这些词汇表的 XML 文档结构提供了它们之间的联系手段。以下程序清单给出了与图 2.1 对应得 XML Schema 的例子。

```
<?xml version="1.0"?>
<!--命名空间-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
<!--以下是简单元素-->
<xs:element name="deptName" type="xs:string"/>
<xs:element name="desc" type="xs:string"/>
<xs:element name="room" type="xs:string"/>
<xs:element name="firstname" type="xs:string"/>
<xs:element name="lastname" type="xs:string"/>
<xs:element name="contact" type="xs:string"/>
<!--以下是属性-->
<xs:attribute name="id" type="xs:ID"/>
<xs:attribute name="ref" type="xs:IDREF" use=”required”/>
<!--以下是复杂元素-->
<xs:element name="name">
  <xs:complexType>
  <!--sequence 表示它包含的元素必须显示-->
  <xs:sequence>
    <!--minOccurs 表示最小出现次数-->
      <xs:element ref="firstname" minOccurs="0"/>
      <xs:element ref="lastname"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="operator">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element ref="contact"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name=”operator-ref”>
  <xs:complexType>
    <xs:attribute ref=”ref”/>
  </xs:complexType>
</xs:element>
<xs:element name="keeper">
  <xs:complexType>
  <xs:sequence>
    <!--maxOccurs 表示最大出现次数,unbounded 表示没有上限-->
      <xs:element ref="room" minOccuus="0" max0ccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="room">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="desc"/>
      <xs:element ref="operator"/>
      <xs:element ref="keeper"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="assistent">
  <xs:complexType>
```

```
      <xs:sequence>
        <xs:element ref="operator-ref"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="material">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="desc"/>
        <xs:element ref="operator" minOccuus="0" max0ccurs="unbounded"/>
        <xs:element ref="assistent"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="department">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="deptName"/>
        <xs:element ref="operator"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

XML Schema 用来描述 XML 文档合法结构、内容和限制。XML Schema 由 XML1.0 自描述，并且使用了命名空间，有丰富的内嵌数据类型和及其强大的数据结构定义功能。XML Schema 充分地改造了并且极大地扩展了 DTD（传统描述 XML 文档结构和内容限制的机制）的能力，将逐步替代 DTD，成为 XML 体系中正式的类型语言，同 XML 规范、Namespace 规范一起成为 XML 体系的坚实基础。

1. XML Schema 组成

(1) Primer。一个非标准化的文档，提供了 XML Schema 的一个简单可读的描述，目的是快速地理解如何利用 XML Schema 语言创建一个 Schema。

(2) Structures。这一部分详细说明了 XML Schema 定义语言，这个语言为描述 XML1.0 文档的结构和内容限制提供了便利，包括开发了 XML Namespace 的使用。

(3) Data Types。这一部分定义了可用于 XML Schema 和其它 XML 规范中的定义数据类型的方法。这个数据类型语言本身由 XML1.0 自描述，提供了说明元素和属性数据类型的 XML1.0 DTD 的一个超集。

2. XML Schema 的逻辑结构

XML Schema 与关系数据库中的 DDL（Data Description Language，数据定义语言）思想相似。在关系数据库中，使用 DDL 去创建表以及描述作用于这个表的规则和限制。XML Schema 提供了创建 XML 文档必要的框架，详细说明了一个 XML 文档的不同元

素和属性的有效结构、限制和数据类型。XML Schema 采用元素类型声明和属性表声明来限制 XML 文档的结构[38]。

(1) 元素类型声明

```
<ElementType name= "element_name"
      content= "empty | textOnly | eltOnly | mixed"
      dt:type = "data_type"
      model = "open | closed"
      order = "one | seq | many">
</ElementType>
```

图 2.4　元素类型声明

name 域规定元素的名字；dt:type 域规定元素的数据类型；content 域描述元素中的内容类型，empty 表示元素内容为空，textOnly 表示元素内容只能为文本，eltOnly 表示元素内容只能为子元素，mixed 表示元素的内容包括上述 3 种情况；model 域规定元素的内容是否要严格遵守 Schema 中的定义，open 表示元素内容可添加未定义的特征，closed 表示元素内容只能添加定义过的特征；order 域规定该元素的子元素的排列顺序，one 只允许子元素按一种顺序排列，seq 允许子元素按指定的顺序排列，many 表示可按任意顺序排列。

(2) 属性类型声明

```
<AttributeType name= "attribute_name"
      dt:type= "data_type"
      default= "default_value"
      dt:values= "enumerated values"
      required= "yes | no">
</AttributeType>
```

图 2.5　属性类型声明

name 和 dt:type 域的含义同上；default 域规定属性的缺省值；dt:values 域规定了当 dt:type 域为枚举类型时的各个枚举值；required 域规定该属性是否必须出现，yes 表示必须，no 表示可以不出现。

2.2.3 DTD 和 XML Schema 的比较

XML Schema 与 DTD 相比，有其独特的特点，提供了丰富的数据类型，实现了继承和复用，与命名空间紧密联系，易于使用。

1. XML Schema 的丰富数据类型。与 DTD 不同，XML Schema 规范提供了丰富的

数据类型。其中不仅包括一些内嵌的数据类型，规范还提供了定义新类型的能力，如complexType 和 simpleType。在 DTD 中，基本上只支持文本类型。

2. 继承和复用。可以利用从已经存在的 Schema 中获得某些类型而构造新的Schema，也可以在不需要时将获得的类型使之无效。同时，XML Schema 能将一个Schema 分成单独的组件。

3. 与命名空间紧密联系。XML Schema 与 XML Namespace 紧密联系，使得在一个命名空间中创建元素和属性非常容易。

4. 易于使用。XML Schema 是由 XML1.0 自描述的。可以利用 XML 解析器对Schema 进行解析，用 XML DOM 和 XML SAX（简单 API，Simple API for XML）对其进行操纵，使用 XML 编辑器编辑，或者利用 XSL 进行转换。

虽然 XML Schema 在许多场合中胜过 DTD，但仍然还有一些 DTD 更胜一筹的领域。在短期内 DTD 还是有它的优势。DTD 有着广泛的工具支持。从更实际的角度来说，用于 XML 模式的工具不如用于 DTD 的工具来得成熟，所有的 SGML 和许多 XML 工具都支持 DTD。在应用上来看，有很多的文件形式都支持 DTD。DTD 已经使用多年，人们已经积累了很多经验。

即使 DTD 有着大量应用的优势，由于 XML Schema 的优点，以及它广泛的应用领域，XML Schema 取代 DTD 已成大势所趋。但作为一种最简单的 XML 模式，XML DTD 也还将会在一段时间内发挥它应有的作用。

2.2.4 DTD 向 XML Schema 的转化

为了实现 XML DTD 到 XML Schema 的转换，首先必须透彻理解 DTD 的内容和语法，其次必须透彻理解 XML Schema 的内容、它的组成部分的 XML 表示以及对于 XML 表示的约束。然后找出从 DTD 到 XML Schema 的映射关系。

对于从 DTD 到 XML Schema 的映射关系，为清晰起见，可以根据 DTD 中的声明的类型分类列出从 DTD 到 XML Schema 的映射表。对于 DTD 中的每一类声明，列出 DTD 中的所有的 Pattern 以及相对应的 XML Schema 的 XML 表示。例如，对于 DTD 中的元素类型声明，可列出一张如附录 3 所示的从 DTD 到 XML Schema 的映射表。对于 DTD 中的属性列表声明、记号声明可列出类似的表格。由于 XML Schema 不支持实体，所以对于 DTD 中的实体声明需区别对待。对于参数实体声明，可以将其展开

后再转换成 schema。对于通用实体声明的转换，还没有很好的解决办法，一种可选的解决办法是将其转换为 XML Schema 的元素声明。对于 DTD 中的注释和处理指令，可不经转换直接写人 schema 中。另外需指出的一点是：在 DTD 中，对于某一元素可以有多个关于此元素的属性列表声明。并且，属性列表声明不一定位于它所引用的元素的元素类型声明下面。

而在 schema 中，关于某一元素的元素声明和属性声明是在同一<element name="...">下，先是元素声明，接着是属性声明。所以在转换时，当在 DTD 中遇到元素声明时，应取得此元素的所有的属性列表声明，然后将此元素的类型声明与所有属性列表声明一起转换成 schema。所以还需定出将 DTD 中的元素类型声明和属性列表声明一起转换为 schema 的转换规则。有了以上这些，我们就可以将 DTD 转换为 schema 了。

我们可以设计一个 XML DTD 到 XML Schema 的转换工具，它的功能是：解析输入的外部 DTD 文件或是 XML 文件中的内部 DTD 子集和外部 DTD 子集，将 DTD 转换为 schema。XML DTD 到 XML Schema 的转换工具的实现可分解为两大部分：一是解析部分，功能是解析所输人的外部 DTD 文件或是解析 XML 文件中的内部 DTD 子集和外部 DTD 子集；二是转换部分，功能是将解析好的 DTD 转换成 schema。

2.3 利用 DOM 来解析 XML 文档

出于 XML 文档是结构化的，因此如果使用 XML 外部处理器[39]，也可正确地取出多需要的数据。但在使用 XML 文档全部应用中，用来组成 XML 处理器的难点在于代价过高。因此对从应用程序调用 XML 处理器的接口做了规定。这种 API 应用程序接口称为 DOM 文档对象模型方法。

作为 W3C 中的一个规范，DOM 为 XML 应用提供了一个标准的应用编程接口。DOM 设计为可以使用任何编程语言来实现。为了提供准确的、独立于语言的规范，DOM 工作组使用了 OMG(Object Management Group，对象管理组织)的 IDL(Interface Definition Language，接口定义语言)来定义 DOM 接口，然后由厂商来具体地实现这些接口。这样既实现了标准的统一，同时又使标准的实现成为可能。

DOM Level1 包括两部分：DOM Core 和 DOM HTML。DOM Core 是 XML 功能所要求的，并且是 DOM HTML 的基础。核心又分为底层基本接口和 XML 扩展接口。

基本接口定义了编写使用 XML 的应用程序的界面和开发者更容易地使用 XML 的界面；扩展接口是用于方便开发者的，并不必须。DOM HTML 定义了 HTML 的高级接口，同时与 DOM 核心保持类似于继承的关系。

2.4 XML 数据模式的形式化定义

为了方便后面的理论研究和表达，现对 XML 作一个形式化的定义。XML 是一个树/图状的结构[40]，图中入度为零的根结点对应于 XML 文档的根元素，中间结点与 XML 文档中带子元素的非根元素相对应，叶结点与 XML 文档中无子元素的元素相对应。

定义 2.1　任何 XML 文档都可以表示为一个 XML 有向图 G= (V,{A})。其中，G 是一个连通、有向、有序且置标的图，包括一个节点集 V 和一个弧集 A。$V = \{V_1,V_2,\ldots V_n\}, n \geqslant 0$。$A = \{(V_i, V_j)\}, V_i,V_j \in V$, 且 $i \neq j$.有向图的简单表达为：

<XML 图> ::= <节点>|<弧>.

定义 2.2　XML 图中度（degree）的衡量。对于任意一个节点 $V_i \in V$，有：

1. 度，V_i 的度是和 V_i 相关联的弧的数目，记为 $TD(V_i)$；

2. 入度，以节点 V_i 为头的弧的数目称为 V_i 的入度，记为 $ID(V_i)$；

3. 出度，以节点 V_i 为尾的弧的数目称为 V_i 的出度，记为 $OD(V_i)$。

并且有下面的公式成立：$TD(V_i) = ID(V_i) + OD(V_i)$.

定义 2.3　XML 图的节点可以分为根节点、内部节点和叶节点。对于任意一个节点 $V_i \in V$，有：

1. 如果 $ID(V_i) = 0$，则 V_i 为根节点；

2. 如果 $OD(V_i) = 0$，则 V_i 为叶节点；

3. 如果 $ID(V_i) \neq 0$ 且 $OD(V_i) \neq 0$，则 V_i 为内部节点。

即：<节点> ::= <根节点>|<内部节点>|<叶节点>.

2.5 本章小结

在本章中主要分析了 XML 的文件的结构特点。对于 XML 数据文件，有专门的模式文件在结构上对它进行约束。本章讲述了两种 XML 数据模式文件：DTD 和 XML

Schema。因为 XML Schema 的日渐流行，文中也简述了 DTD 向 XML Schema 转换的基本原理。DOM 对于解析 XML 文档，生成 DOM 树非常有用。最后，为了方便后面的分析和设计，本章最后给出了 XML 文档和 XML 数据的部分形式化定义。

3 XML 数据的存储技术分析

如何有效地存储 XML 文档，这一直是一个很受研究者注意的问题。但是因为 XML 数据是半结构化的数据，并且独立为单个的文件，因此对 XML 文档的处理和检索查询等都很麻烦，一直以来都没有很有效的办法。本章主要分析研究国内外相关研究方法的原理、特点和优缺点，以利于后面提出改进的方法。

3.1 Internet 与数据库的结合

数据库是从 60 年代初发展起来的计算机技术。经过二三十年的发展，数据库技术已经趋于成熟。在 Internet 发展的今天，数据库技术首先要解决的是，在数据库与 Internet 之上所架构的信息系统之间有没有可结合的共同点[2]。

1. Web 数据库

原有的数据库技术面对的空间是若干数据库在一定的网络空间上连接起来的一个世界。在这个世界上它可以实施相应的技术，使用户有效地管理各自的数据库，并集成起来去完成一定的任务。在 Web 世界中，这一情况发生了本质的改变。Web 上存储的是一些无结构，而且无法有能力去组织管理的数据。每个站点上的数据都是在无序的状态下自生自灭。如何在这上面有效地实施数据库管理技术，两者之间本质上存在着一定的差异。

1998 年的一份报告形成了这样的论点[41]：Web 在改变一切，Web 是一个巨大的数据库。但数据库技术没有在 Web 技术的发展中扮演重要的角色；Web 世界迅速发展，但大量的数据并没有依赖在数据库技术架构上。

现在从 Web 数据库的角度出发，在 Web 数据组织、数据查询等方面都在展开研究。Web 数据组织是研究 Web 信息的特点，找出适合 Web 信息的合理组织模式，目前的研究成果主要体现为半结构化数据模式的研究。Web 上的信息集成是 Web 数据管理的最现实的问题。Web 上诸多数据源中的信息如何构成一个为用户可用的整体，是目前很多应用亟待解决的问题。Web 查询是指能根据更丰富的语义信息在有效数据组织模式下找出更准确的信息。

2. XML 数据源分类

XML 数据源有多种多样，根据具体的应用，大概可分为下面三种：

(1) XML 纯文本文档。XML 纯文本文档将数据存储于文件中，其最大的优点在于可以直接方便地读取，或者加以样式信息在浏览器中显示，或者通过 DOM 接口编程同其他应用相连。

(2) 关系型数据库。关系数据库来源是对第一种来源的扩展，目的是便于开发各种动态应用。其优点则在于通过数据库系统对数据进行管理。这种方式最适合于当前最为流行的基于三层结构的应用开发。

(3) 其他各种应用数据，如邮件、目录清单、商务报告等。

3. XML 文档与 DBMS 的对比

狭义的 XML 仅仅指一种语言和采用该语言所描述的 XML 文档，广义的 XML 包括 XML 语言、XML 文档以及所有与 XML 相关的工具和技术。广义的 XML 与 DBMS 大致等价，XML 与 DBMS 相同之处是：

(1)提供数据存储。XML 采用 XML 文档来存储数据。

(2)提供数据的模式描述。XML 采用 DTD 或者 XML Schema 来描述数据的逻辑结构。

(3)提供对数据的直接存取访问。XML 采用 XQL、XML-QL、QUILT 等查询语言作为直接操作 XML 文档中数据的工具。

(4)提供应用逻辑接口。XML 采用 SAX 和 DOM 定义应用编程接口，使应用程序能够访问和更新 XML 文档的样式、结构和内容。

尽管 XML 文档与 DBMS 具有一些共同的特点和功能，但二者之间也存在不少差异。相比较而言，DBMS 具有更加强大的数据管理功能：DBMS 具有的索引功能极大地提高数据查询的速度、自动的并发访问控制机制和强大而灵活的用户权限管理功能。相比较而言，XML 在开放性、结构柔性和 Internet 支持方面优于数据库。目前在实际应用中，大量数据的存储管理还是依靠数据库管理系统，XML 的核心作用主要体现于共享数据的实现和直接的 Web 操作。

4. XML 数据存储研究综述

XML 数据存储的基本思想大致相同，即将 XML 标记数据放入一定的结构中，这样对数据的检索、分析、更新和输出就能够在更加容易管理的系统和较为熟悉的环境下进行。一种比较自然的想法是采用数据库对 XML 数据进行存取和操作，将半结构

化数据转化为结构化数据。这样可以利用成熟的数据库技术。根据存储 XML 数据的系统所使用的技术来分类，综合来看有以下几种方法：

(1) 文件系统结构；

(2) 半结构化数据模式；

(3) 面向对象模式；

(4) 关系模式。

下面分节讲述每种方法，重点讲述基于关系模式的 XML 数据存储。

3.2 基于文件系统的 XML 数据存储

使用文件系统来存储 XML 文档的优点是简单易用，而 XML 且也适合于文档内容少、数量小的场合，但其局限性也是显而易见的：

(1)大小受限制。对于当今 Web 上的海量数据，无论是数据传输还是数据维护都变得难于操作。

(2)并发性。与数据库不同的是，文件系统不能提供并发性控制。当多个人同时编辑同一个文档时，容易引起会引起信息丢失。

(3)安全控制。对于文件系统，其安全控制的基本单位是单个文件，因而无法对一个文件的局部进行安全控制。

1. XML 文件系统模型

基于 XML 的文件系统主要包括三部分：文件组织 (Organization of Files)、安全系统 (File Security System)和文件的基本管理功能 (File Management Functions)。如图 3.2 所示。

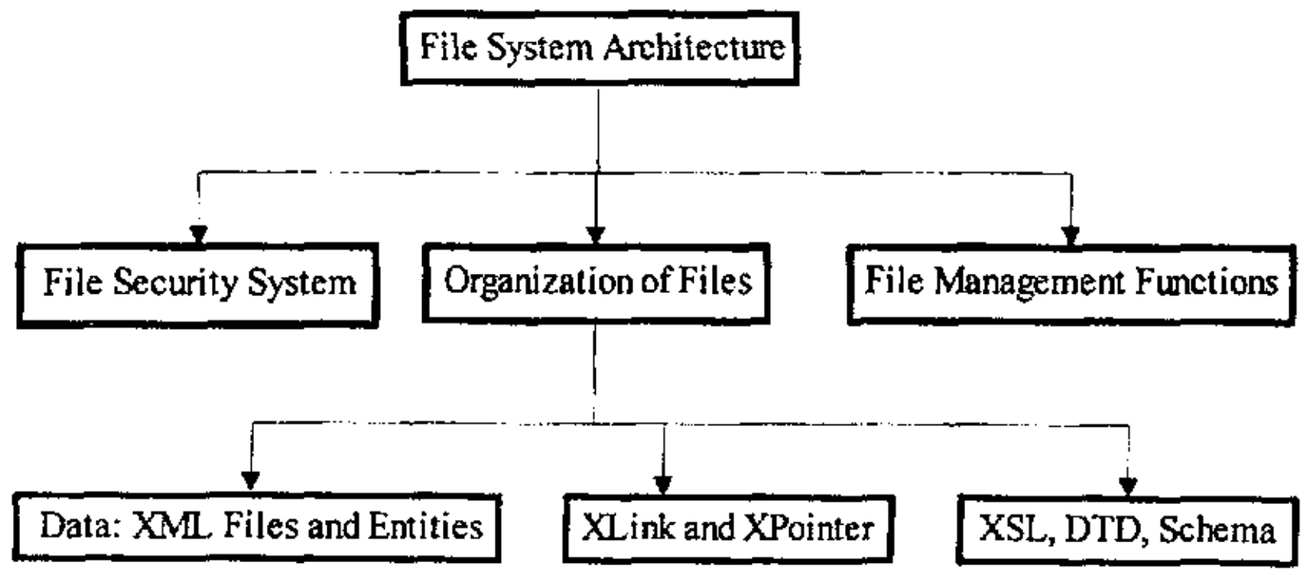


图 3.2 XML 文档系统的逻辑结构

在文件组织方面，设计的目标是将几乎所有的文件和数据库都转换为 XML 文件。不能转化的，以实体的形式存放。文件按种类划分，包括 XML 数据文件和实体、单独存放的 Xlink 和 Xpointer 文件以及 DTD、Schema 和 XSL 文件。

2. 文档和 XML 文件的关系

在这种文件系统中，文档不再仅仅是文件， XML 文件往往并不是一个逻辑上独立的文档。XML 文件之间以链接的方式，将小文档组织成大文档，每个 XML 文件仅仅是更大的文档的一部分，同时又包括（链接）其它文档。XML 文件更像一个数据的容器，最终文档的数据往往是多个 XML 文件共同提供的。

3. XML 文件系统的安全控制

目前的文件系统，安全控制只可以达到文件规模，数据库系统一般可以达到表规模。以 XML 实现的文件系统，理论上安全控制可以达到任意级别。

3.3 基于半结构化数据库的 XML 数据存储

Web 可以被看成是一个巨大的、异构的、分布的、由超文本链接所连接的文挡的集合，对这样的数据进行查询与传统的数据库查询有着明显的不同。首先，已有的数据模型不能很好地适应网上数据的特点，需要引入新的数据模型；其次，由于 Internet 上的许多数据经常缺乏明确的模式，存在不规则的数据形式，这就给查询和处理提出了新的挑战，由此人们提出了半结构化数据的概念。

半结构数据通常称为“无模式”或“自描述”[31]。半结构化数据的模式与传统的关系或面向对象数据的模式不同，对半结构化数据来说，是先有数据，后有模式；半结构化数据的模式是用于描述数据的结构信息，而不是对数据结构进行强制性的约束；半结构数据的模式是非精确的，它可能只描述数据的一部分结构，也可能根据数据处理的不同阶段的视角而不同；半结构化数据的模式可能规模很大，甚至超过源数据的规模，而且会由于数据的不断更新而处于动态的变化过程中[42]。

与普通的关系或者面向对象数据库相比，半结构化的数据库有以下特点[39]：

1. 层次模型。一些数据更适合于表示为层次结构。对这些数据，层次结构语言能够简化操作。

2. 标签和路径操作。传统的数据库允许对元素的操作，但是不能对元素名称操作。半结构化数据库提供了检测标签名字的操作符和对路径进行处理的操作符。

3. 不规则结构。一个节点的内容不需要符合类型描述，例如一个节点不需要有预先决定好的属性列表。

4. 变化的结构，有些结构甚至是预先不知道的。

5. 序列。与关系数据库的表不同，文档部分是按顺序排列的，所以序列的特性要表现出来。

虽然半结构化数据与 XML 数据之间有很多的相似点[43]，但是毕竟还是有很多区别。为了使已有的半结构化数据库能够支持 XML 数据的存取，已有的半结构化数据库必须加以修改。Stanford 的 Lore 系统是一个比较典型的半结构化数据库，为了利用现有系统存取 XML 数据，研究者们通过构造一个 Lore 系统的 XML 数据模型整合 Lore 和 XML[44]，达到在 Lore 中存储和查询 XML 数据的目的。

3.4 基于面向对象模式的 XML 数据存储

面向对象数据库用自身的方法、关系和语义来管理 XML 数据，同时提供了强大的导航和连接功能，例如 Ozone 是建立在对象数据库系统 O_2 之上，它集成了半结构化方法（主要采用 Lore 系统中的方法）。

面向对象数据库适合存取 XML 数据主要是基于以下考虑：

(1) 面向对象的模型支持复杂数据类型，因此可以方便、直观地建立 XML 数据的对象模式，进而利用 OQL（Object Query Language，对象查询语言）实现对 XML 数据的结构化查询。

(2) 面向对象的方法已普遍地应用于软件开发的各个阶段，OODB 能够与面向对象的程序设计方法无缝结合，因此 OODB 很有可能成为未来数据库的主流。以一致的方法访问 XML 数据和 OODB 中的数据无疑将使应用系统的开发得到简化[45]。

在数据库系统中，模式定义了数据的类型和组成关系等约束条件，是表达查询和查询优化的基础。为了通过对象数据库系统查询 XML 数据需要建立 XML 数据的 OODB 模式。XML 数据是多样的，按照约束条件的不同可以分为仅满足规范性约束的数据、有效的数据和遵从 XML Schema 的数据。对于不同类型的数据需采用不同的方法建立其 OODB 的模式。具体可以分为 3 类。

1. 根据 DTD 建立 XML 数据的 OODB 模式。根据 DTD 建立 XML 数据的 OODB 模式的方法是直观的：为所有元素类型建立相应的对象类型，根据元素的内容模型建

立从属联系，根据引用属性建立引用联系，根据属性列表建立对象类型的属性等。描述文本结构是 XML 应用的一个重要方面。对于文本密集的应用，对象的组成与引用关系是模式中的核心内容。对于声明了 DTD 的 XML 文档可以根据 DTD 中定义的有关元素类型、元素的属性、组成与引用关系等方面的结构信息建立 XML 数据在 OODB 中的模式。与采用标注的有向图模式的方法相比[46]，建立对象模式的代价是较低的。

2. 无 DTD 的 XML 文档的结构信息的提取。在实际中有很多数据是没有 DTD 的，例如通过自动工具从已有的数据库、HTML 页面等资源中转换而来的数据。对于无 DTD 的 XML 文档首先要根据文档的内容推测出各元素类型的内容模型[47]，这一过程的结果是产生一个 DTD（称这种通过提取无 DTD 的 XML 文档的结构信息而建立的 DTD 为临时的 DTD，以下简称 TDTD）。因为建立 TDTD 的目的是为了建立 XML 数据的对象模式并最终为查询服务，所以可以根据查询的需要来确定元素的内容模型，从而使求解 TDTD 的问题得到简化。另外，效率是需要考虑的一个重要因素。这里，将 XML 元素的次序分为两种类型，确定的和非确定的次序。确定的次序是以顺序表定义且表中没有“?”和“*”操作符， 如(a，b+，c)。其它类型的次序为非确定的次序，如“(a，b*，c)”或“a，(b|c)”。对于非确定的次序，对象的前趋(或后继)类型只有在运行时才能确定，因此，在 TDTD 中一律作为无序处理。

3. 根据 XML Schema 建立 XML 数据的 OODB 模式。可以在 XML Schema 中定义原型，从而实现封装、继承等面向对象的方法。XML Schema 本身也采用 XML 格式编写，从而方便了数据的建立和处理。XML Schema 克服了 DTD 的许多局限，有望在数据密集的应用中取代DTD成为定义XML数据模式的主要手段。XML Schema与OODB的模式较接近，因而根据 XML Schema 建立 XML 数据的 OODB 模式更加方便。在 XML Schema 标准中提供的基本数据类型(如 int，float，string 等)与对象数据库系统中的相似，因此可以直接地采用 XML Schema 中的定义或转换为接近的类型。在定义元素的组成与引用关系方面，XML Schema 与 DTD 的功能相似，可以采用与 DTD 类似的方法根据 XML Schema 中的结构信息建立 OODB 的模式，这里不再赘述。

3.5 基于关系模式的 XML 数据存储

关系数据库是当前最流行也是最成熟的数据库技术，大量的业务数据已经并将继续在关系数据库中存储与管理，而基于关系数据库平台的应用之间需要以 XML 格式

交换、共享与集成数据。RDBMS（关系数据库管理系统）已提供了有效、完善及用户熟悉并乐于使用的功能服务。因此，需求与技术两种驱动力导致近来对 XML 的关系存储方法进行了大量的研究与原型开发[33,48]；主要的 RDBMS 厂商如 IBM 等也纷纷进行了 XML 扩充，推出了自己的 XML-Enabled 产品。

一个 XML 文档可遵循一个 DTD 或 XML Schema 的约束。DTD 以上下文无关语法规定了一个 XML 文档的结构，但其很弱的数据类型和结构约束机制并未给 XML 文档中的数据提供足够且合适的模式信息。XML Schema 在这些方面进行了扩充和加强，但由于其本质上定义的还是文档模式，与关系数据库模式之间在语法、结构与约束的众多方面存在着固有的异构性，因此难以直接用于产生关系模式[46]。另外，即使多个 XML 文档实例遵循同一个文档模式（或类型）定义，它们也可有不同的结构。XML 数据的以上特点致使 XML—关系数据库映射存在固有的困难。映射时往往可考虑以下因素而采用各种不同的策略：

(1)在多大程度上忽略 XML 的文档特性；

(2)在多大程度上忽略 XML 文档实例中数据的半结构化性；

(3)是否或如何利用可能有的 XML 文档模式（或类型）定义信息。

目前，XML 到关系模式的典型映射方法有三类：

(1)用固定的关系模式存储 XML 数据：定义固定的关系模式直接存储 XML 半结构化实例数据[32]。

(2)从实例数据提取关系模式：对 XML 文档实例数据进行分析，从中提取规则的模式信息用于产生“好的”关系模式，不规则部分的数据存储在“溢出模式”中[33,49]。

(3)从 DTD 导出关系模式：对 XML 文档模式定义中的语法进行分析、简化、转换，产生关系模式 [48,50]。

3.5.1 用固定的关系模式存储 XML 数据

该方法包含几种方案[32]，三种存储边的方案（边方法、二元方法、通用表）和两种存储叶子值的方案（单独值表、内联）。

从产生的关系模式规模、执行不同的 XML 查询以及重构 XML 文档所需的时间代价等方面对几种方案进行了比较分析，发现二元方法和内联方法相结合是“最优的”映射方案[32]。用固定的关系模式存储 XML 数据的映射方法比较简单，对任何 XML

文档都映射成固定的 关系模式。它适合于存储没有文档模式定义的 XML 文档，但模式定义中的类型信息有利于优化存储。此法生成的关系模式不是数量众多、包含大量空值，就是在查询时需要执行大量的连接操作。

3.5.2 从实例数据提取关系模式

STORED(Semistructured To Relational Data)方法将产生“好的”关系模式问题看做是一个输入参数为 XML 文档实例的优化问题[33,49]，这样的优化是 NP 难问题。STORED 方法利用了一种启发式数据挖掘算法自动产生“好”的关系模式和 STORED 映射。这种映射是无损的，部分不规则数据由 “溢出映射”对应到 “溢出模式”中。也就是说 STORED 方法把全部 XML 数据存储到由规则的关系模式和不规则的溢出模式组成的混合模式中。当然，这样的模式选择不是唯一的，也不一定对所有的查询是最优的。另外，这些关系模式也许不能满足 XML 更新的需求。此时，可将数据存储在溢出模式中。STORED 技术虽然可以不需要 XML 文档模式定义，但文档模式信息的存在有利于最小化溢出图、提高系统的效率。

从实例数据提取关系模式能根据不同的 XML 文档实例产生“好的”关系模式。但是，模式的好坏取决于 XML 文档实例的结构“规则”程度，同时，“好的”模式也未必对所有的 XML 查询是性能最优的。另外，实例数据的模式提取算法的最坏时间复杂度较高，因而降低了其实用性。

3.5.3 从 DTD 导出关系模式

因为 DTD 的流行和大量运用，现在的大部分研究是基于 DTD 进行的[48,50]。首先依据简化规则对 DTD 进行简化；然后将 DTD 表示为 DTD 图。

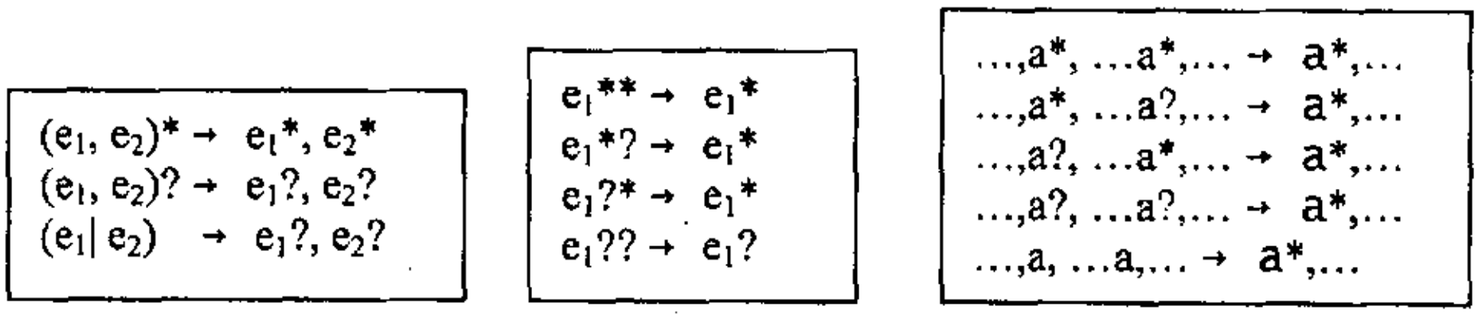
2. 简化 DTD

根据简化后的 DTD 生成关系模式 RS 有如下的原则：任何符合某个 DTD 的文档必须存在该 DTD 生成的关系模式 RS 中；任何关于 XML 文档的符合 DTD 的查询都可以通过计算关系模式 RS 的实例来实现。

DTD 的复杂性主要是由于元素定义的复杂性引起的。比如可以定义这样的元素“a”：<ELEMNT a((b|c|e)?,(e?|(f?,(b,b))))>，这里 b、c、e 和 f 也都是元素。查询语言

更关心的是元素在文档中的位置以及元素间的相互关系，于是我们采用 DTD 变换的方式来简化任意复杂的 DTD。DTD 简化变换有三种类型：

(1) F 变换(Flattning Transformation)。F 变换把嵌套定义转为平面表示。变换规则如图 3.3a 所示。



a. F 变换　　b. S 变换　　c. G 变换

图 3.5　简化 DTD

(2) S 变换(Simplication Transformation)。S 变换把多个一元操作符转换为单个的一元操作符号。变换规则如图 3.3b 所示。

(3) G 变换(Grouping Transformation)。G 变换聚簇同名的子元素。例如把两个形如 e 的子元素聚为一个 e 元素。变换规则如图 3.3c 所示。

所有的“+”操作符均替换为“*”操作符，因为“*”的语义涵盖了“+”的语义。通过简化变换，上述元素定义简化为<!ELEMENT a(b,c?,e,f)>。DTD 简化变换保留了“一或多”以及“空或非空”两种语义。

3. 生成元素图

对于一个 DTD 文件 d，可以通过对每一个元素生成它的元素图，从而构造整体的元素图。对于 d 中的某个元素 e，整个构造方法如算法 3.1 所示。

算法 3.1　构造 DTD 元素图。

以深度优先遍历 d，以 e 为根节点。

(1) 设置 e 为前一节点 p。

(2) 若 d 未遍历完，则以深度优先顺序从 d 中取结点 n，n 的父节点为 n0；若 d 已遍历完则转到(6)。

(3) 若 n 未被标记为 visited，则在元素图中创建一个新的结点 n'，n'与 n 同名；创建一条从 n0 在元素图中对应的结点 n0'到 n'的有向边，将 n 标记为 visited。若 n 已被标记为 visited，则创建一条从元素图中最新创建的节点到元素图中与 n 同名节点 n'的逆向边。

(4) 若 n 未被标记为 visited 且 n 的所有子结点都已经遍历，则取消 n 的标记。

(5)转到(2)。

(6)结束。

4. 生成关系模式

当我们为 DTD 中每个元素 e 创建了元素图之后，e 所对应的关系模式 RS 可以根据 e 的元素图来生成。方法是把元素图中所有 e 的后继节点均作为属性被包含到 RS 中，但以下节点例外：

(1)节点“*”的直接后继节点。这类节点将映射为独立的关系，因为它们代表了父节点的集合属性。

(2)逆向边所指向的节点。这类节点也将映射为独立的关系，因为它们表示了 DTD 定义中的递归情况。

我们把这种方法称为基本内联方法，直观地看，该方法把某元素的后继元素都“包含”到该元素对应的关系中。关系模式中属性是根据从根节点开始到它的路径来的命名的。每个关系模式均有一个 ID 属性，作为该关系模式的键。有父节点的元素生成的关系模式还含有 ParentID 属性，作为外键。

内联方法主要的缺点在于它为每个元素都创建一个关系，导致关系的数量相当庞大。以下两种方法是对基本内联方法的改进：

(1)共享内联技术：为 DTD 图中入度大于 1 或等于 0 的元素节点建立关系，入度等于 1 的元素节点内联到其它元素关系中，“*”的儿子元素节点单独建立关系，相互递归的元素节点选择其中之一单独建立关系；

(2)混合内联技术：它在上述技术的基础上进行了改进，将 DTD 图中节点入度大于 1、但不递归、不经过“*”到达的节点元素内联到其它元素中。所有的关系都由映射自动赋予 id 属性，非根元素对应的关系都有 parentid 属性（parentid 是外键，指向父元素），且存在 order 列指明元素在兄弟元素中的顺序。

比较分析显示[45]：共享内联技术相对减少了 XML 查询（XML-QL）转换成的 SQL 语句的数目但增加了每个 SQL 查询中的连接运算；混合内联技术相对减少了每个 SQL 查询中的连接运算但增加了 SQL 语句的数目。

5. 利用 DOM 实现 XML 数据的存储动作[51]

将 XML 文档经过解析器处理生成可编程的 DOM 树结构，根据 DOM 规范可以对 DOM 树进行操作。XML 存储模块完成对 DOM 树的存储。DOM 树包含内容的节点必然是叶节点，因此我们需要遍历 DOM 树，找到每个叶节点元素对应的关系属性，存

储叶节点的内容。DOM 规范中节点都是 Node 类的实例，DOM 树便由 Node 类实例构成的。DOM 树的存储算法如算法 3.2 所示。其中过程 Store(RID,A,a)是一个自定义的数据库操作调用，功能类似于 SQL 中 INSERT，它的三个参数的含义分别为：关系表记录的 ID、存储的属性名及属性的值。同时队列 Queue 增加一项：节点 N 对应的记录标识 RID。我们的系统支持 GPE(General Path Expression，通用路径表达式)查询，但必须转换为 SQL 查询。转换需要利用元素、元素环的统计信息。显然，统计信息是随着 XML 库的变化(增加、删除数据)而变化的，这就要求系统增量地维护统计信息。另外还有一种路径表达式即 SPE（Simple Path Expression，简单路径表达式）。一个 SPE 是形如：$E_1[V_1].E_2[V_2],...,E_n[V_n]$的表达式，其中 E_i 是元素名，V_i 是可选的变量。由于展开了 GPE 中的形如(R) 的正则表达式，SPE 中可能出现 $E_i=E_j(i\neq j)$。

算法 3.2　DOM 树的存储算法。

(1)取根节点(ROOT,NULL)入队列 Queue；

(2)如果 Queue 为空，则转到(6)，否则转到(3)；

(3)Queue 中一节点(N,R)出队列；

(4)N 如果是生成独立关系的节点，则根据 N 映射的关系名新生成一个对应的新记录 R'，否则 R'=R。如果 N 是原子元素节点，则取 N 的内容 content，找到 N 元素在 DTD 字典中的映射属性名 A，Store(R,A,content)。如果 N 是复杂元素节点，则其所有子节点(N_i,R)入队列 Queue。对于 N 中的每一个属性 A_j（值为 a_j），取 A_j 在 DTD 字典中的映射关系属性名 A_j'，Store(R,A_j',a_j)。

(5)转到(2)。

(6)结束。

在增加 XML 数据时，DOM 树存储算法扫描整个文档，因此可以保留扫描过程中的任何一条路径以计算元素环的最大、最小重复数(MaxDN,MinDN)。整个文档处理完毕后，将原有的元素环的 MaxDN,MinDN 与当前计算出的元素环的 MaxDN'、MinDN'作比较，选择恰当的值存储。

3.6 本章小结

对 XML 数据的存储研究一直是一个热点问题，当前的研究方法主要是利用各种数据库技术来存储 XML 数据。本章主要是分析了各种 XML 数据存储方法的特点原理，包

括：

(1)利用 XML 文件系统来存储 XML 数据，文件系统虽然直接，但是有许多问题；

(2)利用半结构化数据库来存储 XML 数据，半结构化有层次特征，这种技术大部分已经落后了；

(3)利用面向对象数据库来存储 XML 数据，理论上来说很适合，但是现在的面向对象数据库技术还不成熟；

(4)利用关系数据库来存储 XML 数据，具体分析了三种方法，其中重点讲述了利用 DTD 生成关系模式的方法。关系数据库是最流行的数据库，也是本文重点研究的存储 XML 数据的环境。检验 XML 到关系模式映射优劣的重要标准是考察 XML 查询重写为 SQL 查询后的执行效率如何[55]。总体来说，充分利用 XML 文档的模式定义信息有利于获得好的查询性能和紧凑的数据表示。

4 利用 XML Schema 进行 XML 数据存储的设计与实现

尽管 XML Schema 与 DTD 主要功能相同，但是它具有许多 DTD 所不具有的特点和优势。.NET 框架中集成了 XML Schema 类，并且提供 DOM/SAX 对其访问。本章将讲述在.NET 环境中对利用 XML Schema 来存储 XML 数据到关系数据库中的方法进行设计和实现。

4.1 映射模型的设计

根据系统的需求和结构设计，我们可以生成大量的 XML Schema，然后根据这些 XML Schema 文件来产生大量的 XML 文件。同时，在网络上已经存在大量的 XML 文件，它们都附有自己的 XML Schema 文件，如图 2.4 所示。

4.1.1 XML Schema 的简化

为了利用 XML Schema 把 XML 文件中的数据映射存储到关系数据库中，我们重点需要研究和处理的是 XML Schema 文件。和 DTD 文件一样，XML Schema 文件中对元素的限定有许多关键词，为了能够方便计算机系统更好地解析 XML Schema 文件，XML Schema 文件需要得到一定程度的简化，并且根据简化了的 XML Schema 文件和 XML 文档来产生 XML 数据的模式。

XML Schema 中的数据总的来说，分为属性声明、简单数据类型和复杂数据类型三类。因为属性和简单数据类型都没有嵌套元素，它们已经是最小的单位，不能再简化。所以，对 XML Schema 的简化实际上就是对复杂数据类型的简化。

复杂数据类型（complexType）中可以嵌套其他元素，根据嵌套的实际情形，可以分为几种类型，根据不同类型可以相应的简化，为了节约篇幅，这里我们省略了整个 element 的框架代码，只显示 complexType 内的嵌套情况，这里的 F、S、G 变换和第 3 章中用 DTD 来映射存储 XML 数据到关系模式的 F、S、G 变换的概念是一样的，但是变换的主体是 XML Schema。

表 4.1 简化 XML Schema 的 F 变换

complexType 内嵌套情况	简化 complextType
<xs:choice> <xs:element ref="A"/> <xs:element ref="B"/> </xs:choice>	<xs:element ref="A" minOccurs="0"/> <xs:element ref="B" minOccurs="0"/>
<xs:choice max0ccurs="unbounded"> <xs:element ref="A"/> <xs:element ref="B"/> </xs:choice>	<xs:element ref="A" minOccurs="0" max0ccurs="unbounded"/> <xs:element ref="B" minOccurs="0" max0ccurs="unbounded"/>
<xs:choice minOccurs="0" max0ccurs="unbounded"> <xs:element ref="A"/> <xs:element ref="B"/> </xs:choice>	<xs:element ref="A" minOccurs="0" max0ccurs="unbounded"/> <xs:element ref="B" minOccurs="0" max0ccurs="unbounded"/>
<xs:sequence minOccurs="0"> <xs:element ref="A"/> <xs:element ref="B"/> </xs:sequence>	<xs:element ref="A" minOccurs="0"/> <xs:element ref="B" minOccurs="0"/>
<xs:sequence minOccurs="0" max0ccurs="unbounded"> <xs:element ref="A"/> <xs:element ref="B"/> </xs:sequence>	<xs:element ref="A" minOccurs="0" max0ccurs="unbounded"/> <xs:element ref="B" minOccurs="0" max0ccurs="unbounded"/>

表 4.2 简化 XML Schema 的 S 变换

complexType 内嵌套情况	简化 complextType
<xs:choice minOccurs="0">… <xs:element ref="A" minOccurs="0"/> …</xs:choice>	… <xs:element ref="A" minOccurs="0"/> …
<xs:choice minOccurs="0" max0ccurs="unbounded"> …<xs:element ref="A" minOccurs="0"/>… </xs:choice>	… <xs:element ref="A" minOccurs="0" max0ccurs="unbounded"/> …
<xs:choice …>不论其属性如何设置 …<xs:element ref="A" minOccurs="0" max0ccurs="unbounded"/>… </xs:choice>	… <xs:element ref="A" minOccurs="0" max0ccurs="unbounded"/> …

表 4.3 简化 XML Schema 的 S 变换

complexType 内嵌套情况	简化 complextType
<xs:element ref="A" minOccurs="0" max0ccurs="unbounded"/> …<xs:element ref="A" minOccurs="0" max0ccurs="unbounded"/>	<xs:element ref="A" minOccurs="0" max0ccurs="unbounded"/> …
<xs:element ref="A" minOccurs="0" max0ccurs="unbounded"/> …<xs:element ref="A" minOccurs="0"/>	
<xs:element ref="A" minOccurs="0"/> …<xs:element ref="A" minOccurs="0" max0ccurs="unbounded"/>	
<xs:element ref="A" minOccurs="0"/> …<xs:element ref="A" minOccurs="0"/>	
<xs:element ref="A"/> …<xs:element ref="A"/>	

1. F 变换。变换规则如表 4.1 所示。

2. S 变换。变换规则如表 4.2 所示，这里仅列举<xs:choice>的情况，<xs:sequence>的 S 变换和<xs:choice>相同。

3. G 变换。变换规则如表 4.3 所示。

4.1.2 XML Schema 元素图的生成

XML Schema 经过简化变换后保留了两种语义，即“minOccurs=”0” ”和“minOccurs="0" max0ccurs="unbounded"”。这样的变换损失了部分语义，如元素的顺序。不过我们可以在根据关系模式装载 XML 文档过程中附加一些信息来表示类似语义。简化 XML Schema 后紧接着就应该将 XML Schema 文件形式化，产生元素图。但是 XML Schema 中的语法不能直接运用到元素图中去，因此我们必须对简化后的 XML Schema 文件在进行一次转换。这里，类似于第 3 章中对 DTD 的转换方法，将“minOccurs=”0” ”用“?”代替，将“minOccurs="0" max0ccurs="unbounded"”用“*”代替。

一般来说，关系模式由数据模型生成，例如 E-R 模型（Entity-Relationship Model，实体关系模型）。这种转变是直接的，因为在实体和属性之间的区别很明显，每个实体和它的属性都被映射为一个关系。但是当我们试图把 XML Schema 中的元素和元素的属性转变为 E-R 模型中的实体和实体的属性时，却难以找到这种对应关系。在 E-R 模型中的属性在 XML Schema 中往往被当作实体对待。图 2.4 的 XML Schema 就是一个例子，在 E-R 模型中，operator 是实体，而 firstname 和 lastname 是属性。但是在 XML Schema 中却不是这样。事实上，如果把 firstname 和 lastname 当作属性看待，那么它们则和 name 冲突，因为在 XML 中属性是不能嵌套的。因此直接把 XML Schema 中的元素映射为关系将导致 XML 文档的混乱。

在元素图中，每一个元素当作一个结点处理，复杂类型的数据元素包含有子结点，而简单数据元素或者属性则作为复杂数据元素的子结点出现。这样互相关联，通过“?”和“*”来表示元素之间的比例和包含关系。

这里我们采用第 3 章中利用 DTD 生成关系模式的生成 DTD 元素图的方法（算法 3.1）来生成 XML Schema 的元素图。

一个与图 2.1 部门信息 XML 文档相对应的 XML Schema 元素图如图 4.1 所示。

从图中可以看出，每个元素只出现了一次，并且表示出了各元素间以及元素与属性间的关系。这里有两点需要说明：

1. XML 元素图的产生可以来自于对一个 XML 文档的解析，或者是对一个存在的 XML 元素图进行变换或查询的结果。

2. 一个 XML 文档可能表示为不止一种 XML 元素图。在 XML 中属性的顺序并不重要，因此如果一个文档中所有的属性都按照字母顺序排列，而另一个相同的文档以不同的方式保存属性，尽管其物理表示不同，但对于 XML 1.0 而言两个文档是等价的。

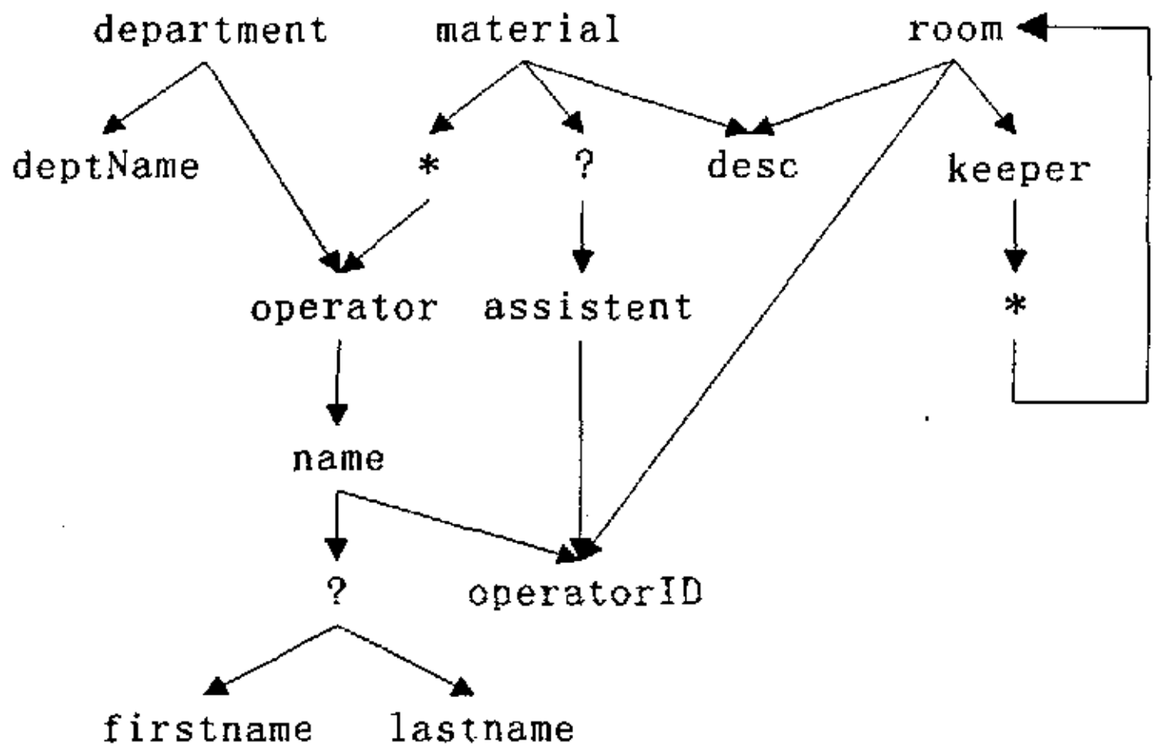


图 4.1　与部门信息 XML 文档对应的 XML Schema 元素图

4.1.3 关系模式的映射

如第 3 章中利用 DTD 导出关系模式，对 XML Schema 元素图映射到关系模式的处理也有几种内联方法。基本内联方法是最简单，但是是效率最低，并且效果最差的。它不能用在实际的映射转换中，共享内联方法也有它的弱点。结合各种内联方法的特点，这里我们采用混合内联方法。映射转换的关系模式如图 4.2 所示。

对于映射后的关系模式，结合 XML 文档和简化后的 XML Schema 文件[52,53]，我们可以利用 DOM 来对 XML 数据进行存储。DOM 提供了比较丰富的 API 用来对 XML 文档进行解析处理[54]，同时如第 3 章中所述，可以利用 DOM 树存储算法来存储 XML 数据。在 XML Schema 映射转变模型中，我们把 DOM 树的存储算法和处理程序融合在模式转换中，作为 XML Schema 映射到关系模式整个过程的一步，经过 DOM 的解

析后，最后 XML 文档中的数据就可以方便地存储到关系数据库中去了。图 4.2 是对 XML Schema 元素图处理后产生的关系模式。

```
department(departmentID:integer,department.deptName:string,
        operator.name.firstname:string,operator.name.lastname:string,
        operator.contact:string)
material(materialID:integer,material.assistOperator.operatorID:integer,
        material.desc:string)
room(roomID:integer,room.parented:integer,room.desc:string,
        room.keeper.operatorID:integer,operator.name.firstname:string,
        operator.name.lastname:string,operator.contact:string,
        operator.operatorID:integer)
operator(operatorID:integer,operator.parented:integer,
        operator.name.firstname:string,operator.name.lastname:string,
        operator.contact:string)
```

图 4.2　对元素图处理后产生的关系模式

4.1.4 利用 XML Schema 模式映射模型的设计

综合前面所述，通过 XML Schema 来把 XML 数据映射存储到关系数据库中一般要经过以下几步：

(1) XML Schema 的简化，这样可以产生一个嵌套层次较少，并且语法简练，元素和属性间的关系简单的 XML Schema 文件。

(2) XML Schema 元素图的生成，这是映射到关系模式的一个很关键的步骤，我们利用第 3 章中所述的方法来生成。

(3) 将 XML Schema 元素图映射到关系模式，采用混合内联方法实现。

(4) 利用 DOM 树来存储 XML 数据。

图 4.3 是初步设计的模型图，DOM 树的存储包含在模式转转中。

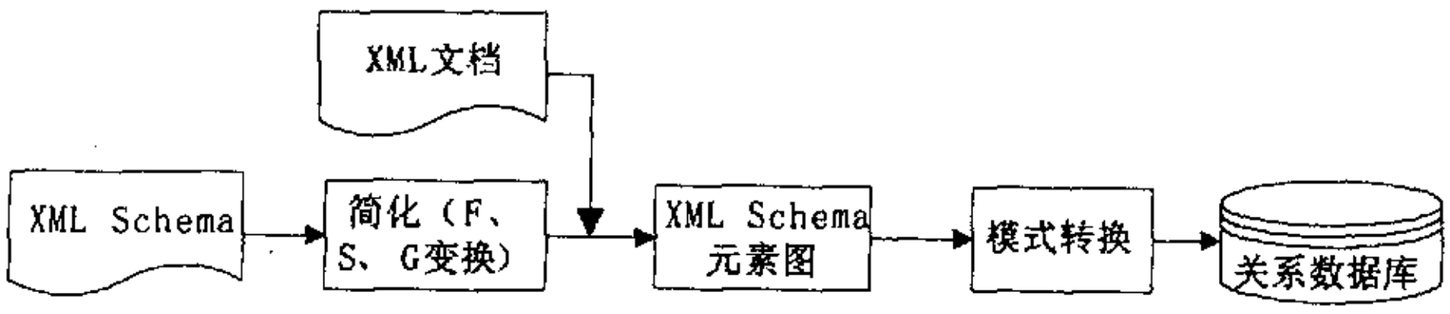


图 4.3　XML Schema 到关系模式的映射处理模型

在第 3 章中讲述了利用 DTD 来生成关系模式的方法，这里我们采用 XML Schema 进行模式转换，它的转换算法跟 DTD 的算法大致相同，但是因为语法、数据类型以及

文件结构不尽相同，在具体转换上还是有所区别。算法 4.1 是 XML 模式转化为关系模式的 XSToRS 算法[55]。

算法 4.1　XSToRS 算法（用 XML Schema 生成关系模式）。

1．总的算法分 3 步完成。

(1)把简化后的 XML Schema 文件进行转化为 XML Schema 元素图。

(2)由 XML 元素图生成一个包含所有将要转化为关系的元素节点表。

(3)由元素节点表转化生成关系模式。

下面分别讲述这 3 步算法的操作内容。

2．把 XML Schema 文件进行转化为 XML Schema 元素图的算法。

(1)读入 XML Schema 文件，并且初始化 XML 元素图（命名为 Graph）；

(2)如果已到文件尾，结束算法，否则读入一个元素；

(3)如果这个元素在 Graph 里已经存在，结束算法，否则在 Graph 中创建这个元素（命名为 eNew）；

(4)如果 eNew 没有子元素，结束算法，否则读取 eNew 的下一个子元素（命名为 eSub）；

(5)如果 eSub 在图 Graph 中存在，那么在 Grpah 中连接 eNew 和 eSub，然后转到第 4 步。

3．由 XML 模式图生成一个包含所有将要转化为关系的元素节点表的算法。

(1)读入 XML 模式图 Graph，并且初始化元素节点表（命名为 Table）；

(2)对于 Graph 中的任何一个元素 V_i，如果满足下列条件之一，则把 V_i 加入 Table：V_i 的入度 $ID(V_i) = 0$，或者 $ID(V_i) > 1$ ，并且存在子元素 V_j，有(V_i, V_j)的弧的个数没有上限。

(3)利用深度优先算法遍历 V_i，并且把它们链接到 V_i 上，如果 V_i（命名为 nNew）有子元素，读入（命名为 nSub），注意以下问题：(a)如果 nSub 是“*”或者在 nNew 和 nSub 之间有环路，那么设置 nNode.parentID 为 nSub 的 ID，读取其父元素，继续遍历。(b)如果 nSub 是“?”，那么读入 nSub 的子元素并且设置其为 nSub。

4．由元素节点表转化生成关系模式的算法。

(1)对元素节点表 Table 中的每一个节点，创建一个关系，关系名为节点名字，把关系的标志(ID)作为其属性，如果附加属性里有父节点 ID，将其也添加进去；

(2)对每一个作为关系名的 Table 的节点，遍历其所有子节点，并且将它们的名称

作为关系的属性。

4.2 系统实现

到目前为止，还没有任何系统能够完全地把 XML Schema 通过映射关系模式来在关系数据库上进行存取。在 Microsoft™发布的.NET 框架和 SQL Server™ 2000 中较好地集成了对 XML 的支持，我们制定了一个实现方案，并且在项目“水电站资产维护管理系统（AMS）”中进行了初步应用。

4.2.1 完整的系统体系结构框架

在一个完整的利用 XML Schema 来映射存取 XML 数据的系统中，一般要达到这样几项功能：利用 XML Schema 来映射存取 XML 数据到关系数据库中，并且在关系数据库中对 XML 数据进行查询，最后把输出结果转换为 XML 文档。整个系统的体系结构如图 4.4 所示。

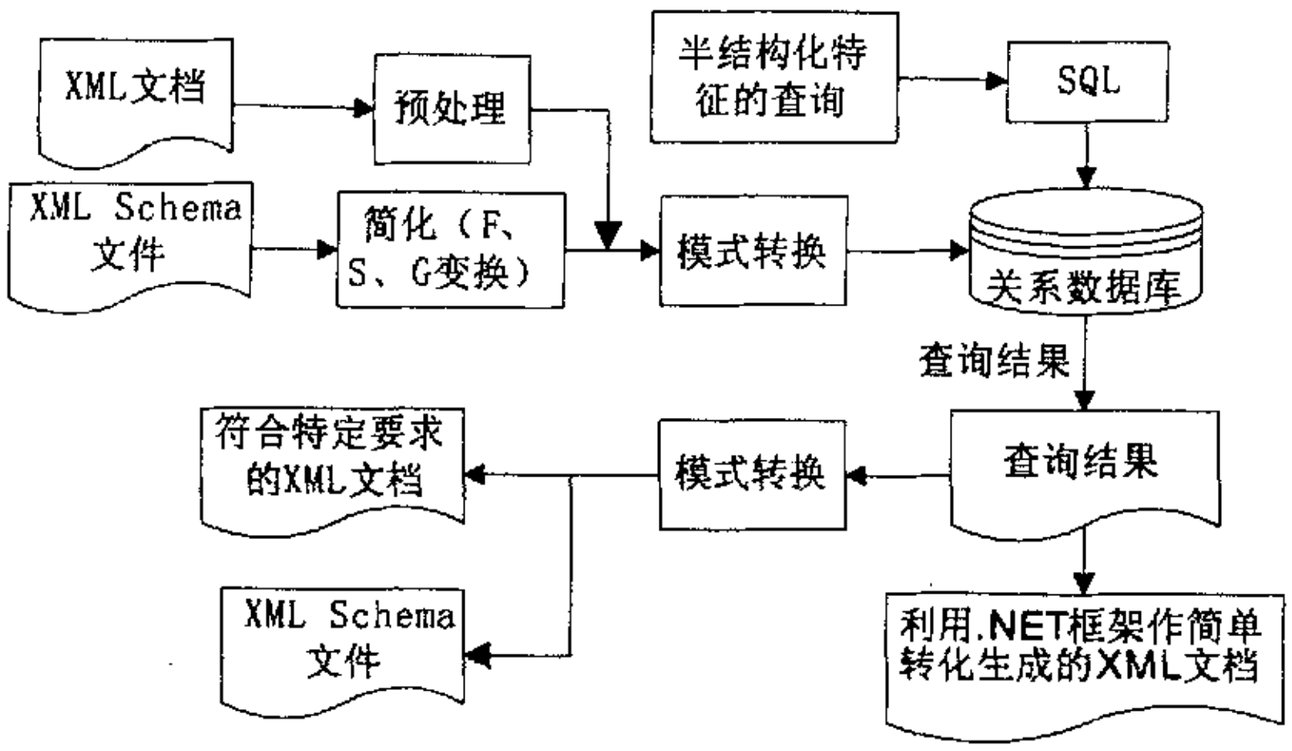


图 4.4 基于.NET 框架的 XML 存取体系结构图

整个系统分为以下几个大的模块：

1. XML 文档的存储，把 XML 文档存放到关系数据库中。它包括 XML 文档的预处理（即对一些结构不好的文档进行处理，清除异常、错误等不符合 XML Schema 的信息）、XML Schema 文件的简化、模式转换和 DOM 树的存储几个方面，其中 DOM 树的存储包含在模式转换中。

2. 查询，把基于 XML 的半结构化查询转化为基于关系数据库的 SQL 查询。这是我们需要深入研究的模块。XML 查询不可避免的要用到 XML Schema 的相关信息。XML 查询最本质的特点在于查询中含有 GPE[45]。一般地，GPE 查询转化为 SQL 查询有两种途径：(a)GPE 查询→SQL 查询；(b)GPE 查询→SPE 查询→SQL 查询。方法(a)由于是直接转化成 SQL，因此效率较高，但通用性不强。方法(b)通过 SPE 过渡,更具有普遍性（SPE 既可以转化为 SQL，也可以转化为 OQL 或其它）。

3. 输出，把查询结果转化为 XML 文档输出[51]。由于在 XML 查询转换过程中得到了关于元素的信息，所以关系查询结果的转化工作变得十分简单。得到关系查询结果 R 后，可以给 R 附加上若干标签，即<E_1><E_2>...<E_n>R</E_n>...</E_2></E_1>，作为 XML 文档返回给用户。

4.2.2 系统的设计

这里我们将分析实现一个简单的原型系统，它没有实现 4.2.1 中的整个系统，只是实现了部分功能。原型系统来自于我所参与的 AMS 项目，抽取了 AMS 系统中的报表系统和物资管理系统构成演示系统。

1. 项目背景

湖北清江水电开发公司是一家拥有多个电厂的大型发电公司，他们先后构建了各类业务处理系统，包括生产管理系统、物资管理系统、人事权限系统和办公自动化系统等，这些系统的开发和实施为提高企业的生产效率和运行管理水平发挥了相当重要的作用。不过，随着 IT 技术的发展和公司业务操作的不断变化以及整个公司计算机应用水平的不断深入和提高，原有的应用系统开始呈现出许多弊端。各应用系统采用的数据库管理系统和开发工具各不相同，使得系统间不能相互访问，各系统的后台数据不能实现共享，形成了公司内部的一个个“信息孤岛”；公司内部数据存在大量冗余，各应用系统的部分功能也相互重叠，容易造成全局数据不一致和投资浪费；这些问题制约着公司信息化的进一步发展。

为了解决上述弊端，使各系统之间可以互联互访，彼此共享数据和功能，进而形成一个统一的整体系统，需要开发一个基于分布式集成项目。这也就是 AMS 项目的项目背景。AMS 系统基于.NET 平台，采用易于维护和升级的 B/S 结构开发；AMS 系统大量采用最新的 IT 技术，如 XML，Web 服务等，使得系统具有松散耦合的结构和可扩展

性；整个 AMS 系统集成了企业原有的生产管理系统，物资管理系统，人事权限系统以及工作流系统等各系统的功能，为水电行业企业的生产和维护管理提供了全面的解决方案。

2. 设计的实现

AMS 系统是基于 B/S 模式的，所以在客户端没有安装程序，所有的程序和运行环境工具都在服务器端。因此，报表系统也是如此。我们采用水晶报表作为报表系统的开发工具，并且把报表系统放在服务器端。在运行 AMS 的时候，客户端在浏览器里直接可以打印报表也可以设置和预览报表。整个报表系统的逻辑结构图如图 4.5 所示。

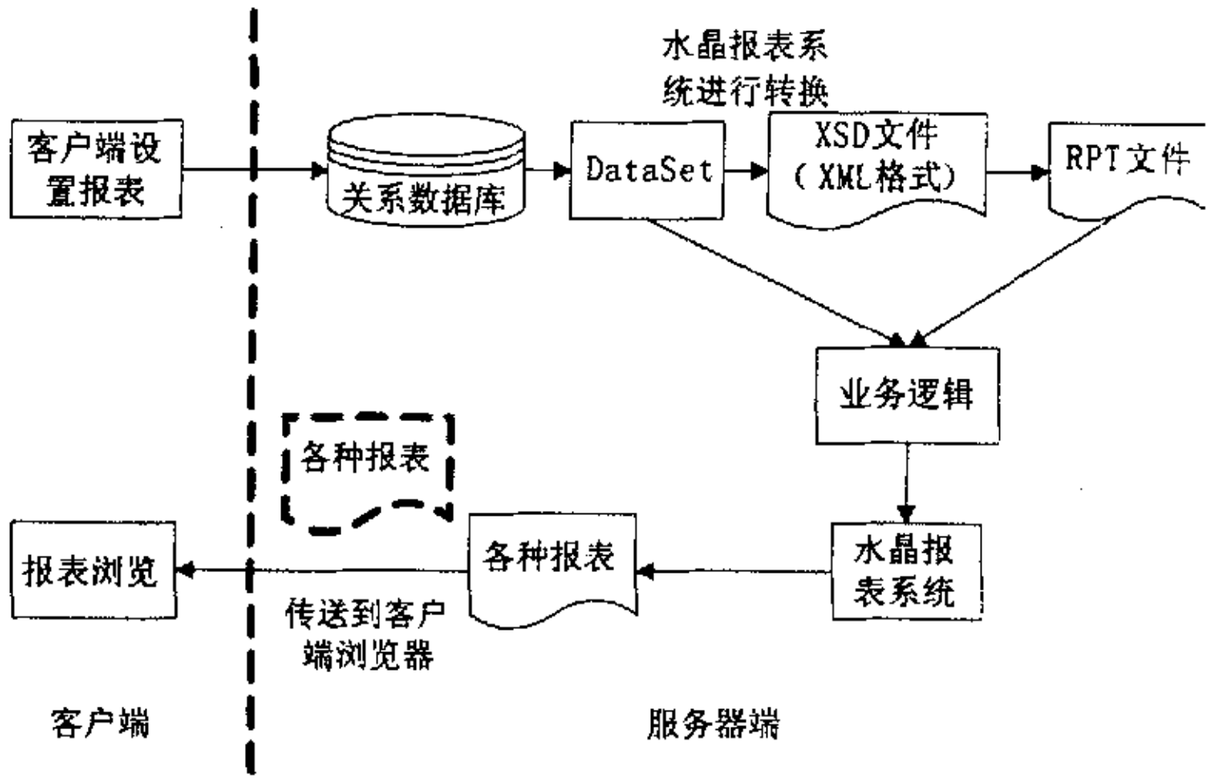


图 4.5　AMS 报表系统逻辑结构图

报表系统中包含以下几部分：

(1)生成 XML 格式的数据源：XSD 文件。客户端引发报表设置请求，然后把请求传回服务器端，服务器端受理请求，产生一个 DataSet，然后开发人员利用水晶报表把 XSD（XML Structure Definition，XML 架构定义）文件中的数据取出来，构造 RPT 文件。

(2)报表的打印。在客户端的浏览器里产生报表打印动作，然后服务器端的业务逻辑把关系数据库中的数据和 XSD 文件以及 RPT 文件结合起来产生报表，然后把报表传送到客户端的浏览器中。业务逻辑在处理的时候，找到对应的 XML 数据源文件（XSD 文件），根据 XML 文件的结构解析出 XML 文件中的结构和数据，然后根据 DataSet 结合产生报表。

AMS 系统开发的一个目的就是为了集成以前多个单独的子系统，使得它们之间能

够互相通信和交换数据，打破各个系统相互孤立的情况。但是以前的系统使用的数据库环境各不相同（有文本数据库、Sybase 和 SQL SERVER 等），在 AMS 集成各子系统时，就要考虑怎样来有效地集成异种数据库环境。我们这里提供的一种方案就是利用 XML 来作为中间的文件格式来进行数据交换。AMS 和其他应用子系统通信时，各个子系统把自己的数据库表、数据等信息都转换成 XML 数据，并且附加有结构良好的 XML Schema 文件，然后发送到 AMS。AMS 通过 XML Schema 来解析 XML 文档，从而把 XML 数据存放到 AMS 的 SQL SERVER2000 中来。以物资子系统为例，以前湖北清江水电开发公司有单独的物资子系统，早期版本是基于文本数据库的，后续版本是基于 SQL SERVER 的。不但数据库的环境不一样，数据库中表的结构也不尽相同。因此在集成以前的物资系统时，必须采取有效方式来通信。我们采取的 XML 中间数据交换方式如图 4.6 所示。

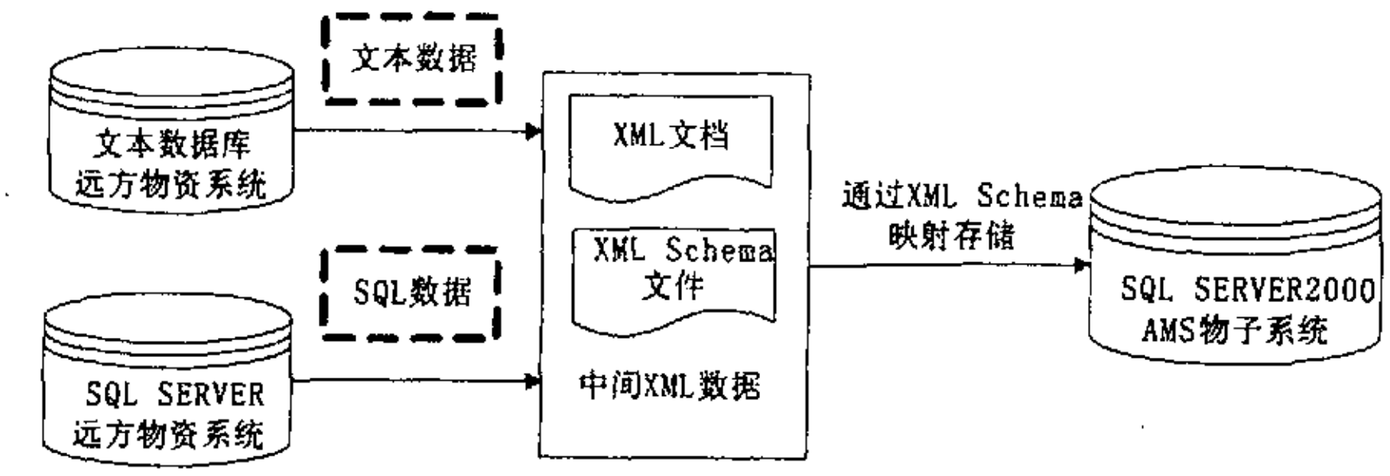


图 4.6　AMS 物资管理子系统的集成逻辑框架

AMS 物资子系统在集成以前清江公司的远方物资子系统的时候，基本步骤有以下几步。远方物资子系统把自己的数据库表和数据都转化成 XML 文档和描述 XML 数据模式和结构的 XML Schema 文件，然后这些文件被发送到 AMS 物资子系统。AMS 物资子系统对接收到的 XML 文档的 XML Schema 文件进行分析，解析出数据结构，然后根据这些结构来把 XML 文档中的数据利用前面提到的存储算法存储到关系数据库中。

3. .NET 集成开发环境

我们的 Web 服务集成模型基于微软的.NET 平台，其目的是要利用.NET 平台对 Web 服务的强大支持。Microsoft.NET 作为微软公司主推的面向 Web 服务的产品策略，提供了一套完整的解决方案，用于支持用户的企业级应用开发。从技术层面上说，Microsoft.NET 平台包含两个主要组件：即公共语言运行库和.NET 框架类库。.

ADO.NET 和 .NET Framework 中的 XML 类集中于 DataSet 对象。无论 DataSet 是文件还是 XML 流，它都可以使用来自 XML 源的数据来进行填充。无论 DataSet

中数据的数据源是什么，DataSet 都可以写为符合万维网联合会 (W3C) 的 XML，并且将其架构包含为 XML 架构定义语言 (XSD) 架构。由于 DataSet 固有的序列化格式为 XML，它是在层间移动数据的优良媒介。

.NET Framework 中的 XML 具有符合 W3C 标准、可扩展、可插式结构、较高的性能和与 ADO.NET 紧密集成等特点。.NET Framework 中的 System.Xml 命名空间和其子类如 XmlDocument、XmlTextReader、XmlTextWriter 和 XslTransform 等极大地简化了对 XML 文档的读、写等操作。

SQL Server 2000 中也集成了对 XML 的支持，如用 for XML 输出 XML 格式的数据，而 openXML 则用来把 XML 数据存储到数据库中。

在 AMS 物资管理子系统中，对于被集成的子系统发送来的 XML 数据，AMS 能够通过 XML Schema 来映射为关系模式，从而从 XML 文档中读取数据，把数据存放到 SQL SERVER2000 中的相应表中去。

在这个系统中，我们的 XML Schema 文件都是结构良好的，而且 XML 文档要严格符合 XML Schema。在.NET 框架中我们采用 C#作为编程语言，目前基本实现了从 XML 模式到关系模式的转换。

4.3 本章小结

本章提供了一个 XML 数据映射存储到关系模式的改进模型，是利用 XML Schema 来进行映射存储的，并且给出了基于 XML Schema 来映射关系模式的具体算法，最后给出了在 Microsoft .NET 环境下进行利用 SQL SERVER2000 进行存储的系统初步实现。

5 结束语

Internet 的出现和快速发展已经改变了我们这个时代的生活方式，作为 Internet 的主要技术 Web 也一直不停的发展，Web 数据的标记语言的发展就是其中一例。由 SGML 到 HTML 再到 XML 的发展历程，就表现了 Web 技术发展的历程，也表现了相关计算机技术的发展。

XML 数据有着比 SGML 和 HTML 更适合于 Web 的特点，特别是随着相关计算机技术的蓬勃发展（如数据库技术、搜索技术和电子商务等等），XML 的数据独立性、可扩展性等特点使得它越来越受到人们的热切关注。现在 XML 在数据交换中间件、XML Web Service、XML 数据查询和 XML 数据存储等方面有着大量的应用和研究活动。

本文对如何存储 XML 数据进行了一些分析和设计工作，主要有如下方面：

1. 分析了 XML 文件系统的架构、特点和优缺点，因为利用文件系统来存储 XML 文档是最直接的方法。

2. 针对 XML 数据具有半结构化的特征，分析了利用半结构化数据库来存储 XML 数据的模型和具体系统实例。

3. 分析了利用面向对象数据库技术来存储 XML 数据的模型和模式的建立，面向对象概念和 XML 数据的结构有着很多相似点，但是面向对象数据库技术现在还有很多不成熟的地方。

4. 因为关系数据库是当前最流行的数据库技术，所以这也是本文的重点。分析了基于关系模式来存储 XML 数据的原理、模型和方法，并且给出了基于 XML Schema 的 XML 数据存储。

由于 XML 的结构特征类似于树/图，具有半结构化特征，它和关系模式的特征还是有很大区别，因此在基于关系模式的 XML 数据存储上还需要进行更加深入的研究，具体来说有以下几方面：

1. 现在的基于关系模式的存储方法都或多或少地丢失了部分信息，因此如何更加完整地把 XML 数据存储到关系数据库中需要更加精细的模型来支持；

2. 因为 XML 数据的层次很多，需要改进算法来提高 XML 数据在关系数据库上存储的效率；

3. 同样的，如何把关系数据库中的数据生成为 XML 数据现在也处于研究阶段；

4. 另外，如何把基于 XML 的查询语言同样有效地移植到关系数据库中来也是一个很重要的研究课题；

5. 最后，如何在 Web 的环境下自动化的进行 XML 数据到关系数据库的转化以及关系数据到 XML 数据的转化，如何把各种特征各异的环境有效集成在一起值得我们继续去研究。

由于时间仓促和作者水平有限，论文中难免存在错误和缺陷，敬请各位老师和同学批评指正。

致 谢

“书到用时方恨少”，在这次写毕业论文的时候，我才感觉到研究生三年匆匆而过，平时所学的知识还远远不够。毕业论文给了我一个很好的机会，让我来好好认识自己，系统地梳理知识，认认真真地进行一次理论的研究。

在此我首先要感谢是我的导师宋善德老师，他给了我关心和照顾，更给了我学习和研究的机会。宋老师在学习中给予我以指导，在平时的生活中给以关怀，他对待学生、工作和生活的态度值得我学习。

同时我要感谢实验室的陈传波老师、欧阳星明老师、胡迎松老师等老师给予的无私的指导和帮助。

感谢带领我参加项目的实验室老师刘小峰、喻知斌，和清江公司信息中心的陈中新和王伟川，他们给予了我很多帮助。

感谢郭翔、戴路、朱刚、何力、刘黎志、李涛、陈明、张勇、徐峰、张道杰、方雄、况湘玲、陈韬、黄刚、涂文昊、赵颢等，我和他们一起度过了一段难忘的时光，从他们身上我学到了很多！

感谢我的同班同学，和他们在一起生活和学习让我感到非常的快乐和幸福。

感谢同寝室的邹复好、杨兵、陈辉、陈帆，我和他们朝夕相处，同甘共苦，一起分享生活学习中的快乐和幸福。

感谢华工赛百公司为我提供了一个科研和工作的环境，感谢这里的所有同事在工作上给予的支持和帮助！

特别要感谢的是我的父母，是他们对我一贯的宽容、支持、鼓励、关心和教导，才使我有了今天在华中科技大学深造的机会。我只有更加积极的生活，更加努力的学习才能报答他们无私的亲情。

最后，感谢论文评审委员会的老师们百忙之中对我论文的悉心指正。

感谢所有关心和爱护我的人！

参考文献

[1] Charles F, Goldfarb, Paul Prescod 著. 张利, 王显著译. XML 实用技术. 北京：清华大学出版社, 1999. 12～50

[2] 孟小峰. Web 数据管理研究综述. 计算机研究与发展, 2001, 38(4)：385～395

[3] 陈球. 标准通用置标语言SGML语言概况. 国际电子报, 1996. 6, 第79版. 131～138

[4] 郑红, 李德河. 标识语言的发展、特点及其在 WWW 中的应用. 微机发展, 1999, (1):26～27

[5] 周杰韩, 曾庆良, 谢金崇等. 基于 XML 的互连网信息资源描述及其应用研究. 计算机工程与应用, 2002, (3):65～67

[6] 曹亮, 王茜, 卢菁. XML 数据在关系数据库中存储和检索的研究和实现. 东南大学学报, 2002, 32(1):124～127

[7] Walmsley P. XML Schema 权威教程. 陈维军, 乔安平, 英宇译. 北京：清华大学出版社, 2003. 20～208

[8] Valentine C, Dykes L, Tittlel E. XML Schema 数据库编程指南. 毛选, 魏海萍等译. 北京：电子工业出版社, 2002. 13～147

[9] 李由, 黄凯歌, 汤大权. XML 的数据库存储技术研究. 计算机应用研究, 2002, (4):60～62

[10] P. Buneman, S. Davidson, G. Hillebrand, D. Suciu, "A Query Language and optimization Techniques for Unstructured Data," SIGMOD International Conference on Management of Data, 1996.506~516

[11] 汪源, 孙建玲. XML 查询语言. 计算机时代, 2002, (1):1～5

[12] 朱韵箎, 程代杰. 基于 XML 的分布式数据交换中间件模型设计. 计算机工程与设计, 2003, 24(8):35～40

[13] 明仲, 曾新红, 倪宏业. 连接 XML 与数据库的中间件的实现. 计算机工程, 2003, 29(20):47～49

[14] Paul Benjamin Lowry. Data mediation and collaboration: a proposed comprehensive

architecture and query requirements for using XML to mediate heterogeneous data sources and targets. In: Proceedings of the 34th Hawaii International Conference on System Sciences. 2001.322~328

[15] 周泽华, 金戈, 黄涛. 基于 XML 的分布式 Web 应用框架. 计算机工程与应用. 2001, (18) :60～65

[16] 胡萍, 李文华. 基于 XML 的数据转换中间件的研究与实现. 微机发展, 2003, 13(6) :72～74

[17] 周小平, 胡小鹏, 石向月等. 分布式 Web 系统的设计研究. 计算机应用研究, 2001, (11) :123～125

[18] 阳红星, 彭志红. 基于 Web 的 XML Web Service 的研究. 华东地质学院学报, 2003, 26(4) :387～393

[19] Steven J. Vaughan Nichols. Web Service: Beyond the Hype. IEEE Computer, 2002, (2): 18~21

[20] David Ehnebuske, Dan Rogers, Claus VonRiegen UDDI Data Structure Reference. 2001,(6): 6~17

[21] 闫新庆, 李文锋, 陈定方. Web 服务的体系结构和应用. 武汉理工大学学报, 2002, 24(5) :28～31

[22] 吴敏, 徐德智, 陈学工. XML 模式蕴含及查询优化研究. 计算机应用, 2003, 23(8) :7～8, 30

[23] 童咪娜, 金远平. XML 数据库查询语言特性研究. 计算机应用, 2001, 21(9) :69～71

[24] 袁和金, 王翠如, 王晚霞. XML 文档的信息查询算法的实现. 计算机应用, 2001, 21(12) :57～58

[25] V. Christophides, S. Abiteboul, S. Cluet, and M. Scholl. From structured documents to novel query facilities. In: Proceedings of ACM SIGMOD Conf on Management of Data. Minneapolis, Minnesota, 1994. 313~324

[26] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom, "Lore: A Database Management System for Semistructured Data," SIGMOD Record, 26(3), September 1997.54~66

[27] 杨建武, 陈晓鸥. 半结构化文档集的结构模式提取的研究与实现. 计算机工程, 2001, 27(10) :19～21, 113

[28] Fredj Dridi, Gustaf Neumann. Towards Access Control for Logical Document Structures. In: Proceedings of the Ninth International Workshop of Database and Expert Systems Applications (DEXA'98), Vienna, August 1998. 322~327

[29] Len Seligman and Arnon Rosenthal. The Impact of XML on Databases and Data Sharing. IEEE Computer, June ,2001.127~138

[30] Y.Chen, S. Davidson, Y. Zheng, Constraint preserving xml storage in relations, In: Proceedings of the International Workshop on the Web and Databases (WebDB), Madison, WI, 2002.

[31] Alin Deutsch, Mary Fernandez and Dan Suciu. Storing semistructured data in relations. In: Proceedings of the Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats, 1999. 431~442

[32] Lee D and Chu W W. Constraints-preseving transformation from XML document type definition to relational schema. DKE 2001, 39(1):3~25

[33] R. Bourret, C. Bornhovd, A. Buchmann. A Generic Load/Extract Utility for Data Transfer between XML Documents and Relational Databases. 2nd Int. Workshop on Advanced Issues of EC and Webbased Information Systems (WECWIS), San Jose, California, 2000.

[34] 周建洪，吴以群，引明，楼荣生.XML 文件系统的设计.计算机工程与科学, 2001, 23(2):72～75

[35] S. Abiteboul, P. Buneman, D. Suciu. Data on the Web: From Relations to Semistructured Data and XML. Morgan Kaufmann Publishers.San Francisco, California, USA, 2000. 18~121

[36] Buneman P., Davidson S, Fan W, Hara C, Tan W. Keys for XML.WWW 10, 2001. 201~210

[37] M Garofalakis, A Gionis, R Rastogietal. XTRACT: A system for extracting document type descriptors. In: Proceedings of 2000 ACM SIGMOD Int'l Conf on Management of Data. Dallas, Texas, 2000.165~176

[38] Phil Bernstein, Michael Brodie, Stefano Ceri, et al. The Asilomar report on database research. SIGMOD Record, 1998, 27(4):74~80

[39] S Nestorov, S Arbiteboul, R Motwani. Extracting Schema from Semistructured Data.

In: Proceedings of ACM SIGMOD Int'l Conf on Management of Data, Seattle, Washington, USA, ACM Press, 1998. 295~306

[40] Roy Goldman, Jason McHugh, Jennifer Widom. From semistructured data to XML. Markup Languages: Theory & Practice, 2000, 2(2): 153~163

[41] R Goldman, J Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. In: Proceedings of the 23rd Int'l Conf on Very Large DataBases. San Francisco, CA: Morgan Kaufmann, 1997. 436~445

[42] Jayavel Shanmugasundaram, Kristin Tufte, Gang He, Chun Zhang, David DeWitt, Jeffrey Naughton. Relational Databases for Querying XML Documents: Limitations and Opportunities. In: Proceedings of the 25th VLDB Conference. Edinburgh, Scotland. 1999. 302~314

[43] Masatoshi Yoshikawa, Toshiyuki Amagasa,Takeyuki Shimura and Shunsuke Uemura. XRel: A path-based approach to storage and retrieval of XML documents using relational databases. ACM Transactions on Internet Technology, August, 2001, 1(1):110~141

[44] Deutsch, M. Femandez, D. Suciu. Storing Semistructured Data with STORED. In: Proceedings of ACM SIGMOD Int'l Conf on Management of Data, Philadelphia, PA, USA, ACM Press, 1999. 431~442

[45] Florescu D, Kossmann D. Storing and querying XML data using an RDMBS. IEEE Data Engineering Bulletin, 1999, 22(3):27~34

[46] R G G Cattell. The Object Databases Standard: ODMG-93 San Mateo, CA. Morgan Kaufmann, 1994.

[47] R. Goldman, J. McHugh, J. Widom. From semistructured data to XML: Migrating the LORE data model and querylanguage. In: Proceedings of the 2nd Int'l Workshop on the Web and Databases (Web DB '99). Philadelphia, 1999. 25~30

[48] F Tian, J Dabid, J Chen. The Design and Performance Evaluation of Alternative XML Storage Strategies. ACM SIGMOD Record, 2002 31(1):5~10

[49] Eric van der Vlist. XML Schema. O'Reilly. June, 2002.

[50] 周傲英, 张龙, 梁宇奇等. 基于关系的 XML 数据存储. 计算机应用, 2000, 20(9):9～12

[51] 周竞涛, 张树生, 孙宏伟等. 关系模式到 XML Schema 的约束保留映射. 西北工业大学学报, 2003, 21(3):372～376

[52] 邓成玉, 王超, 贺琪. XML 数据存储管理技术. 燕山大学学报, 2002, 26(2):119～123

[53] 陈维斌, 喻小光. 一种 XML 数据到结构化数据的转换方法. 华侨大学学报, 2003, 24(2):201～207

[54] 黄莹, 杨明福. 使用 DOM 对象实现 XML 和数据库的交互. 微型电脑应用, 2001, 17(4):14～16

[55] 宋善德, 肖必强. 关系模式下的 XML 数据存取技术研究. 计算机工程与科学, 2004, (8). 录用待发.

附录 1　攻读硕士学位期间发表论文目录

[1] 宋善德, 肖必强. 关系模式下的 XML 数据存取技术研究. 计算机工程与科学, 2004.

（第一署名单位:华中科技大学）

附录 2　词汇缩写表

AMS(Asset Management System)	资产管理系统
ANSI(American National Standards Institute)	美国国家标准协会
C/S mode(Client/Server Mode)	客户/服务器模式
DAD(Data Access Definition)	数据访问定义
DDL(Data Description Language)	数据定义语言
DOM(Document Object Model)	文档对象模型
DTD(Document Type Definition)	文件类型定义
E-R Model(Entity-RelationshipModel)	实体关系模型
GPE(General Path Expression)	通用路径表达式
HTML(HyperText Markup Language)	超文本标记语言
HXD(Hybrid XML Database)	混合 XML 数据库
IDL(Interface Definition Language)	接口定义语言
NXD(Native XML Database)	XML 本原数据库
OEM(Object Exchange Model)	对象交换模型
OQL(Object Query Language)	对象查询语言
OMG(Object Management Group)	对象管理组织
RDBMS(Relational Database Management System)	关系数据库管理系统
SAX(Simple API for XML)	简单 API
SGML(Standard Generalized Markup Language)	标准通用标记语言
SOX(Schema for Object-Oriented XML)	面向对象 XML 语言的模式
STORED(Semistructured To Relational Data)	半结构化到关系数据的转换
WWW(World Wide Web)	万维网
XDR(XML Data Reduced)	简化的 XML 数据
XEDB(XML-Enabled Database)	支持 XML 的数据库
XML(eXtensible Markup Language)	可扩展标识语言

附录 3　从 DTD 到 XML Schema 的映射表

DTD	XML Schema
<!ELEMENT ROOT (#PCDATA)>	<xs:element name="ROOT" type="xs:string"/>
<!ELEMENT ROOT ANY>	注：当 ROOT 无属性声明时，对应 schema 如下： <xs:element name="ROOT" type="xs:anyType"/> 注：当 ROOT 有属性声明时，对应 schema 如下： <xs:element name= "ROOT"> <xs:complexType mixed=”true”> <xs:complexContent mixed="true"> <xs:restriction base="xs:anyType"> <xs:sequence> <xs:any processContents="lax" minOccurs="0" maxOccurs="unbounded"/> </xs:sequence> <xs:attribute.../> </xs:restriction> </xs:complexContent> </xs:complexType> </xs:element>
<!ELEMENT ROOT EMPTY>	<xs:element name= "ROOT"> <xs:complexType/> </xs:element> 注：当 ROOT 有属性声明时，complexType 元素有 attribute 子元素
<!ELEMENT ROOT (A,B)>	<xs:element name= "ROOT"> <xs:complexType> <xs:sequence> <xs:element ref="A"/> <xs:element ref="B"/> </xs:sequence> </xs:complexType> </xs:element>
<!ELEMENT ROOT (A|B)>	<xs:element name= "ROOT"> <xs:complexType> <xs:choice> <xs:element ref="A"/> <xs:element ref="B"/> </xs:choice> </xs:complexType> </xs:element>

<!ELEMENT ROOT (#PCDATA|B)*>	<xs:element name= "ROOT"> <xs:complexType mixed=”true”> <xs:choice min0ccurs="0" max0ccurs="unbounded"> <xs:element ref="B"/> </xs:choice> </xs:complexType> </xs:element>
<!ELEMENT ROOT (A|(B,C))>	<xs:element name= "ROOT"> <xs:complexType> <xs:choice> <xs:element ref="A"/> <xs:sequence> <xs:element ref="B"/> <xs:element ref="C"/> </xs:sequence> </xs:choice> </xs:complexType> </xs:element>
<!ELEMENT ROOT (A?,B+,C*)>	<xs:element name= "ROOT"> <xs:complexType> <xs:sequence> <xs:element ref="A" minOccurs="0"/> <xs:element ref="B" maxOccurs="unbounded"/> <xs:element ref="C" minOccuus="0" max0ccurs= "unbounded"/> </xs:sequence> </xs:complexType> </xs:element>
<!ELEMENT ROOT (A|B)+>	<xs:element name= "ROOT"> <xs:complexType> <xs:choice max0ccurs="unbounded"/> <xs:element ref="A"/> <xs:element ref="B"/> </xs:choice> </xs:complexType> </xs:element>
<!ATTLIST TestDTD testAttr1 CDATA# IMPLIED>	<xs: attribute name=”TestAttr1” type=”xs:string” usd=”optional” default=”3”/>
<!ATTLIST TestDTD testAttr2 CDATA# REQUIRED>	<xs: attribute name=”TestAttr1” type=”xs:string” usd=”prohibited” />
<!ATTLIST TestDTD testAttr1 CDATA# FIXED “3”>	<xs: attribute name=”TestAttr1” type=”xs:string” usd=”required” fixed=”3”/>