四川师范大学

硕士学位论文

XML DTD到关系模式的映射研究

姓名：于光

申请学位级别：硕士

专业：计算机软件与理论

指导教师：杨春

20060405

XML DTD 到关系模式的映射研究

计算机软件与理论专业

研究生 于光 指导教师 杨春

可扩展标记语言 (eXtensible Markup Language，简称 XML)是一门新兴的面向 Internet 应用的标记语言，目前已迅速发展成为 Web 和数据交换的一种新标准。XML 数据的海量出现使得如何有效地对 XML 数据进行存储和管理成为研究热点。利用现有关系数据库的存储管理、并发控制、恢复、版本机制等技术可以有效地存储和管理 XML 数据。

本文首先介绍了 XML 的相关技术领域，对 XML 文档的各种存储方法作了一些简要比较，并着重研究了 XML DTD 到关系模式的映射。通过分析有关 XML 关系存储的经典算法可以发现，现有的 XML 到关系数据库的映射算法大多只考虑内容和结构的映射，而很少考虑 XML 所蕴含的语义。然而语义信息对于数据的存储模式设计、查询优化和更新异常检查等问题是十分重要的，因此如果 DTD 中蕴含函数依赖，在进行映射时予以考虑可以带来很多便利。

本文试图提出一种保持函数依赖的映射策略，在进行内容和结构映射的同时保持函数依赖。依据 XML 函数依赖定义，找出 DTD 中所蕴含的函数依赖，最终利用改进的内嵌算法 FD－Inlining 实现了从 XML 到关系数据库的保持函数依赖的映射。FD－Inlining 方法不仅考虑了 DTD 的内容和结构，还考虑了 DTD 所蕴含的函数依赖，对 DTD 中蕴含的其它语义约束也进行了分析，给出了保持部分语义约束的方法。最后对本映射策略的完整性进行了分析。

关键词：XML；DTD；函数依赖；语义；映射；关系

ABSTRACT

XML(eXtensible Markup Language)is a new markup language for internet application, and now it's one of the new standards of web and data exchange. Along with the mass-producing of XML data, how to store and manage the XML data efficiently becomes a hotspot in the study of XML storage strategy. Technology of storage and management, parallel control, recovery, and version mechanism of the existing relation database can be used to efficiently store and manage XML data.

In this thesis, field related to XML was introduced at first, then the existing storage strategies were compared, and at last the mapping of XML DTD to relational schema was discussed in particular. In general storage strategies of XML data, only contents and structures of XML data were considered, with the semantic information neglected. However, semantic information of XML data is of much significance for design of storage pattern, query optimizing and abnormity checking in updating. If there is some functional dependence in DTD, mapping with considering the functional dependence may bring much convenience.

We tried to provide a mapping strategy with functional dependency preserved in this thesis. According to the definition of XML, functional dependency contained in DTD was found out, and modified inlining strategy (FD-inlining) was adopted to map XML to relational schema. For FD-inlining, not only the content and structure, but also the functional dependency of DTD were considered. Other semantic information contained in DTD was also analyzed, and method to preserve semantic restriction was provided as well. Finally, integrity of the mapping strategy had been evaluated.

Keywords: XML; DTD; functional dependency; semantics; mapping; relational schema

1 前言

1.1 问题的提出及其研究意义

随着 Internet 的高速发展，人们要求对 Internet 上的数据进行深层次的处理。而现有的 HTML 语言的局限性越来越明显地暴露出来，它不具备大规模 WEB 应用所需的可扩展性、结构化等特性。于是，一种新型的标记语言 XML 应运而生。XML（eXtensible Markup Language 可扩展标记语言）是 W3C（World Wide Web Consortium 互联网联盟）提出的一套用于 Web 网络上的数据和文档结构的通用标记语言。XML 能实现平台无关性，即与数据的存储平台无关，与表示数据的模型无关，使得 XML 成为一种很好的数据交换格式。W3C 的原意是用 XML 来解决 HTML 信息的不确定性和网站之间的数据交流问题，但随着它的应用深入，特别是与 Java 语言相结合，在软件开发、网站运营、移动互联等方面逐渐呈现出它的优秀性能。尽管超文本标记语言（HTML）是目前创建 Web 页最常用的语言，但是 HTML 在保存信息方面的能力很有限。与之相反，XML 却有着巨大的伸缩性与灵活性，它具有极其灵活的语法，允许真实地描述各种类型的信息。可扩展标记语言 XML 将逐渐取代 HTML 而成为互联上的主要表示和交换工具。

随着 XML 数据的海量出现，如何有效地存储和查询这些 XML 数据就成为目前值得研究的一个重要问题。数据库系统具有强大的数据操作和处理能力，特别是关系数据库技术已经相当成熟，应用也极其广泛，是目前的主流。把关系数据库强大的数据操纵能力以与 XML 强大的数据表达能力相结合是存储管理 XML 数据一种切实可行又很有前景的方式。相对于关系数据库中用平面关系表来存储数据来说，XML 文档是一种比较复杂的树型结构，而且关系数据库也不能很好的支持 XML 中的一些本质关系，如层次、顺序、包含等。同时 XML 还蕴涵很多语义，特别是数据依赖，在映射时也要予以考虑。因此，怎样在关系数据库中有效的存储 XML 文档，既能保持 XML 文档的内容和结构，又能保持其语义约束特别是函数依赖成为一个难题。

许多大的数据库厂商都将它们的视线转移到 XML 上，并宣称在它们的新产品中提供对 XML 技术的支持。Oracle 早在 1999 年就率先推出支持 XML 的产品。虽然目前已经出现了各种支持 XML 的数据库产品，但是关系型数据库不能很好地支持结构化置标语言中一些本质的关系（如层次、顺序和包含等），

所以在开发中仍然存在很多问题。目前 XML 文档的有效存储还面临许多困难（如模式转换、结构变动和查询效率等问题），对其圆满解决还有很长的路要走。

1.2 国内外的研究现状

XML 文档数据属于半结构化数据，将它存放在面向对象数据库或原生 XML 数据库中是非常自然的。但是，面向对象数据库、原生数据库技术还不够成熟完善，限制了其应用范围。而采用关系数据库存放 XML 文档有几个优势：首先，同面向对象数据库和原生 XML 数据库相比，关系数据库具有数据结构化、最低冗余度、较高的程序与数据独立性、易于扩充、易于编制应用程序等优点。关系数据库已经相当成熟，能够提供对大量数据进行高效存取，是目前最为流行的数据库（如 SQL Server、DB2、oracle 等都是第三代的关系数据库）。如果 XML 数据也以关系的形式存放，利用关系数据库现有的存储管理、并发控制、恢复、版本机制等技术可以有效地对数据进行统一管理。利用关系数据库存储 XML 数据，以及如何充分发挥关系数据库成熟的数据管理技术优势已成为目前的研究热点之一。

目前用关系数据库存储 XML 数据存在多种不同的方法，主要可分为独立于文档的存储方法和依赖于文档的存储方法。采用独立于文档的存储方法，不需要 DTD，但关系中存储的数据缺乏语义。典型的独立于文档的存储方法如 Florecu 和 Kossman[6]提出的两种将 XML 文档映射为关系数据库的 Edge 和 Binary 方法及其 6 个变种。依赖于文档的存储方法即基于 DTD（Document Type Definition，文档类型定义）的关系映射，利用这种方法得到关系模式具有一定的语义。典型的依赖于文档的存储方法如文献[7]中给出的将 DTD 映射为关系模式的三种方法：基本内联技术、共享内联技术和混合内联技术。但在这些典型的映射算法中很少考虑 XML 所蕴涵的语义约束，特别是函数依赖。目前国内在此领域也基于以上依赖于文档的方法进行了一些有益的改进[4][5]。

1.3 本研究的思路及方法

目前依赖于文档的 XML 数据存储方法主要基于 XML Schema 与 DTD，Schema 是 DTD 的扩展，XML Schema 虽然功能强大，但过于复杂，因此在实际应用中 DTD 是 XML 文档使用最多和应用最成熟的模式。

通常XML存储的是半结构化数据，要想存储到关系数据库中必须把它转化为符合关系数据库的行-列结构。除此以外，XML还蕴含语义，现有的XML到关系数据库的映射算法大多只考虑内容和结构的映射，很少考虑XML所蕴含的语义，而语义信息对数据的存储模式设计、查询优化、更新异常检查等来说是十分重要的。如果DTD中蕴含函数依赖，则进行映射时应予以考虑。

本文提出一种保持函数依赖的XML DTD到关系模式的映射策略，首先根据参考文献[1]中基于DTD的XML函数依赖的概念，找出DTD中所蕴涵的函数依赖，然后对符合规则的元素应用FD-Inlining方法进行内联。最后分析DTD中所蕴涵的语义约束，并对这些语义约束进行处理，使得到的关系模式保持这些语义约束。

FD-Inlining方法在Abdel-aziz和Oakasha等人于2005年提出的新的内联方法[2]基础上进行改进得到，既保持了文档内容和结构，又保持了DTD所蕴涵的函数依赖。现有的算法一般是为每一个元素创建一个关系，这样造成文档碎片的形成。FD-Inlining算法中只为符合下列条件的元素创建关系，并把（0，1）及（1，1）约束中子元素属性，作为父元素的属性看待，不为子元素创建关系。从而尽量避免了文档碎片的形成。

本文按以下章节进行组织：

第一章为绪论，主要简单介绍本课题的研究意义，国内外的研究现状以及本课题的研究思路和研究方法等；

第二章为XML及其相关技术，主要简单介绍XML的基础知识和XML的一些相关技术规范；

第三章为XML文档的存储方法，详细介绍了当前已经存在的XML的存储方法，着重介绍了基于关系的XML存储的经典方法；

第四章为保持函数依赖映射策略的设计，提出了一种保持XML函数依赖的映射策略，既保持了XML的内容和结构，也保持了其所蕴含的函数依赖。对XML中的其它语义也进行了分析并给出了映射方法，最后进行了完整性分析，并与共享内联方法进行了比较；

第五章为基于XML的查询，包括XML的查询语言，将XML的查询语言转换为基于关系的查询语言SQL的方法，以及查询结果的返回，分析了基于关系数据库的XML查询过程。

2 XML 及其相关技术

World Wide Web 是最近几年 Internet 上最具生命力的一种应用，由于它操作简单而又功能强大，不仅能够传输文本数据，而且可以进行声音、图像、多媒体等数据的传输，因此受到越来越多的用户的喜爱。随着 Web 文件变得越来越大越复杂，Web 内容的提供商已经开始感受到普通的 HTML（超文本标记语言）已经无法提供用于大规模商业出版所需要的扩展性、结构和数据检查功能以及文件数据传输能力。

为了满足商业 Web 出版的需要，解决 Web 技术在新的颁布式文件处理领域的应用需求，W3C 开发了一种可扩展的标记语言，这就是 XML，以用于那些目前 HTML 无法满足需求的应用。

2.1 XML 的产生与发展

XML 同 HTML 一样，都来自 Standard Generalized Markup Language，即标准通用标记语言，简称 SGML。早在 Web 未发明之前，SGML 就早已作为文档描述和输出的标记语言存在多年。它最先是为文本处理而设计的。SGML 十分庞大，既不容易学，又不容易使用，在计算机上实现也十分困难。鉴于这些因素，Web 的发明者——欧洲核子物理研究中心的研究人员根据当时(1989 年)计算机技术的能力，提出了 HTML 语言。HTML 只使用 SGML 中很小一部分标记。为了便于在计算机上实现，HTML 规定的标记是固定的，即 HTML 语法是不可扩展的，它可以不包含 DTD（文档类型定义）。HTML 这种固定的语法使它简单易学，在计算机上开发 HTML 的浏览器也十分容易。正是由于 HTML 的简单性，使 Web 技术从计算机界走向全社会，走向千家万户，使 Web 的发展如日中天。Internet 提供了世界范围内网络互连和通信功能，到目前为止，几乎所有的 Web 页面都是用 HTML 编写的。HTML 简单易学又通用，句法简明紧凑，加上其扩充的表格、帧、脚本等功能，使它得以在 Web 主页上大显身手。尽管 HTML 在人机界面方面很成功，但却非常不利于机器之间的相互交流与信息传递。其不足具体体现在难以扩展、交互性差、语义性差以及单向的超链接等方面。

XML 和 HTML 一样是 SGML 的一个优化子集。XML 是 W3C 制定的用于描述数据文档中数据的组织和安排结构的语言。它类似于 HTML，但 XML 关

注的不是数据在浏览器中如何布局和显示，而是关注于怎么样描述数据内容的组织和结构以便数据在网络上进行交流和处理。

但是随着 Web 应用的越来越广泛，HTML 的弱点也越来越明显了，主要包括一下 5 个方面。①链路丢失后不能自动纠正；②动态内容需要下载的部件太多；③搜索时间长；④HTML 缺乏对双字节或多国文字的支持；⑤HTML 的可扩展性差。

近年来，随着 Web 的应用越来越广泛和深入，人们渐渐觉得 HTML 不够用了，HTML 过于简单的语法严重地阻碍了用它来表现复杂的形式。尽管 HTML 推出了一个又一个新版本，已经有了脚本、表格、帧等功能，但始终满足不了不断增长的需求。另一方面，这几年来计算机技术的发展也十分迅速，已经可以实现更为复杂的 Web 浏览器，所以开发一种新的 Web 页面语言既是必要的，也是可行的。有人建议直接使用 SGML 作为 Web 语言，这固然能解决 HTML 遇到的困难。但是 SGML 太庞大了，用户使用不方便。而且，要全面实现 SGML 的浏览器就非常困难，于是人们自然会想到仅仅使用 SGML 的子集，使新的语言既方便使用又实现容易。正是在这种形势下，Web 标准化组织 W3C 建议使用一种精简的 SGML 版本——XML 应运而生了。

2.2XML 的特性

XML 的优势之一是它允许各个组织、个人建立适合自己需要的标记集合，并且这些标记可以迅速地投入使用。这一特征使得 XML 可以在电子商务、政府文档、司法、出版、CAD/CAM、保险机构、厂商和中介组织信息交换等领域中一展身手，针对不同的系统、厂商提供各具特色的独立解决方案。XML 的最大优点在于它的数据存储格式不受显示格式的制约。一般来说，一篇文档包括三个要素：数据、结构以及显示方式。对于 HTML 来说，显示方式内嵌在数据中，这样在创建文本时，要时时考虑输出格式，如果因为需求不同而需要对同样的内容进行不同风格的显示时，要从头创建一个全新的文档，重复工作量很大。此外 HTML 缺乏对数据结构的描述，对于应用程序理解文档内容、抽取语义信息都有诸多不便。XML 把文档的三要素独立开来，分别处理。首先把显示格式从数据内容中独立出来，保存在样式单文件（Style Sheet）中，这样如果需要改变文档的显示方式，只要修改样式单文件就行了。XML 的自我描述性

质能够很好地表现许多复杂的数据关系，使得基于 XML 的应用程序可以在 XML 文件中准确高效地搜索相关的数据内容，忽略其他不相关部分。XML 还有其他许多优点，比如它有利于不同系统之间的信息交流，完全可以充当网际语言，并有希望成为数据和文档交换的标准机制。总体说来，XML 主要具有以下一些重要特性：

第一，可扩展性。XML 让使用者创建和使用他们自己的标记而不是 HTML 的有限词汇表，可扩展性是至关重要的，企业可以为电子商务和供应链集成等应用 XML 定义自己的标记语言，甚至特定的行业也可以一起来定义该领域的特殊的标记语言作为该领域信息共享与数据交换的基础。

第二，灵活性。XML 提供了一种结构化的数据表示方式，使得用户界面分离于结构化数据。在 XML 中可以使用样式表如 XSL(eXtensible Stylesheet Language，扩展样式表语言)、CSS2（Cascading Style Sheets Level 2，层叠样式表第 2 进阶）将数据呈现到浏览器中。

第三，自描述性。XML 文档通常包含一个文档类型声明，因而文档是自描述的，不仅人能读懂 XML 文档，计算机也能处理。XML 文档中的数据可以被任何能够对 XML 数据进行解析的应用所提取、分析、处理，并以所需格式显示，XML 表示数据的方式真正做到了独立于应用系统并且这些数据能重复使用。

第四，简明性。XML 只有 SGML 约 20%的复杂性，但具有 SGML 约 80%的功能。同完整 SGML 的相比，XML 简单得多、易学、易用，并且易实现。

第五，结构化。HTML 不支持表达数据库结构或面向对象的分级结构所需要的深层结构，XML 至少是结构化的。在结构化信息的组成要素中 DTD 或 XML Schema 是一个很重要的组成部分。它规定了资料的格式规范并且用这种规范来对资料进行解释。XML 自带一个 XML 语法分析器。语法分析器使用 DTD 或 Schema 来确定一个文件是否是有效的或结构化的。结构化信息的优点是允许不同格式的资料可以相互交换。用户可以使用结构化 XML 文件作为一种中介体实现两种数据库之间的灵活转换。

第六，交互性。据 W3C 的 XML 工作组主席 Jon Bosak 称：用户需要同 Web 上的应用交互操作，以及这种通过 HTML 很难实现的交互就是为什么 XML 需要存在的重要的原因。XML 支持交互性，用户可以自己制订和设计应用 XML

的文件格式，而不必须在使用固定的 HTML 格式，能得到比 HTML 更丰富的语法和更多的功能。

第七，语义性强。XML 可以自行设计有意义的标记便于异构系统之间的数据交换和信息检索。实现机器与机器之间的信息交换。

第八，纯文本。几乎任何工具都可以创建和编辑 XML，使得程序可以更简单读写它。从而提供了从小配置文件到企业级数据仓库的可扩展性。

第九，可格式化。可扩展样式语言 XSL 可以指定如何显示数据。数据和显示分离。可以为同一数据指定不同的样式表用于不同的输出。

第十，很强的连接能力。Xlink 可以定义双向连接，多目标连接，扩展连接和两个文档之间的连接。

第十一，与平台无关。XML 对格式的定义严格，具有层次结构而且与厂商和平台无关，XML 文档可无须任何更改直接移植到其他平台上。

当然，XML 作为一个新建立的标准，还有许多不足之处：它在强调了数据结构的同时，语义表达能力上略显不足。另外，XML 的有些技术尚未形成统一的标准，充分支持 XML 的应用处理程序还不多，甚至浏览器对 XML 的支持也是有限的。尽管如此，我们仍然可以预言，随着人们对 XML 认识的逐渐深入，XML 必将成为网络技术不可缺少的一员。

2.3 XML 模式语言

XML 模式语言就是用来定义 XML 的词汇表和文档结构的语言。XML 的一系列特性如可扩展性、结构化和自描述性等都要用 XML 模式定义语言来体现。XML 文档本身可以看作是数据库中的数据区，XML 模式可以看成是数据库模式。把 XML 映射到关系数据库时，就是把 XML 文档的模式映射为关系数据库中的关系模式。然后把 XML 文档存储到关系数据库的表中。所以我们必须熟悉 XML 模式定义语言。XML 模式定义语言有 XML-Data、文档内容描述（DCD）、面向对象的 XML 模式（SOX）、文档类型定义（DTD）、XML 大纲（XML Schema）等。后两种模式定义语言是 W3C 所推荐的标准，下文将予以详细介绍。

2.3.1 文档类型定义（DTD）

首先，文档类型定义是从 SGML 继承过来的，是 W3C 委员会推荐的 XML 模式定义语言之一，是目前用的最多的 XML 模式定义语言，其它定义可以看作对它的替代和扩展。

XML1.0 提供了一种机制——文档类型定义给你定义自己的词汇表，并将 DTD 作为规范的一部分。DTD 使用正规的语法来定义 XML 文档的结构和允许值，以上下文文法无关的方式描述文档中元素和属性间的嵌套关系，使用*（表示 0 个或多个），+（表示 1 或多个），？（表示 0 个或 1 个）和|（表示选择）等操作符描述元素和子元素间的关系。DTD 描述中的所有值均被假定为字符串，除非由关键字 ANY 定义（此时，值类型可以是任意的文档片段）。一个元素可以由多个子元素或属性来定义。其中，ID 和 IDREF 是两种特殊的属性类型。一个元素至多只能有一个 ID 属性，而一个 ID 属性唯一标识了一个元素，一个元素的 ID 属性可以被同一文档中的另一个元素的 IDREF 属性引用。IDREF 属性没有类型。DTD 描述中没有根的模仿，符合于一个 DTD 描述的 XML 文档的要结点可以为 DTD 中的任意一个元素。下面用 DTD 描述的一个简单的例子。

```
<?xml version="1.0" standalone="yes"?>
<DOCTYPE Research>
<!ELEMENT laboratory(labname,membet*)>
<!ELEMENT project(projname,(member|publication)*>
<!ELEMENT projname(#PCDATA)>
<!ELEMENT membet(name,email?,publication*,project*)>
<!ATTLIST member ID memberID #REQUIRED>
<!ELEMENT name(#PCDATA)>
<!ELEMENT email(#PCDATA)>
<!ELEMENT publication (author+,title,year)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT labname (#PCDATA)>
```

DTD 样例定义了一个实验室的文档类型。一个 laboratory 元素有一个子元

素 labname；0 或多个子元素 member。一个 project 有一个子元素 projname；0 或多个 member；0 或多个 publication 子元素。Member 由一个子元素 name ；0 个或一个 email；0 或多个 publication；0 或多个 project 子元素和 memberID 属性构成。这里的 memberID 属性由关键字#REQUIRE 描述，表示每一个 member 必须有一个 memberID 属性。Publication 又由 0 或多个 author，一个 title 和一个 year 子元素定义。期货元素都为原子元素，被定义为文本串#PCDATA。关于 XML 和 DTD 规范的详细描述见 W3C 的相关标准（http://www.w3c.org/XML/）。

2.3.2 XML Schema

W3C XML 模式工作组在 1999 年 12 月 17 日约定了两部分关于 XML 模式的工作草案。它们用 XML 语法写成，允许使用多个命名空间，它提供强大的内容分类。而且它们是 XML1.0 DTD 功能的超集。1999 年 12 月 17 日的工作草案被分成两部分：结构和数据类型。

1．结构部分

处理元素和属性的描述和声明。那里提供的材料允许 XML 设计者指定复杂的元素结构及设定这些元素内容数值的约束。这些描述部分可以在 http://www.w3c.org/TR/xmlschema-1/上找到。

我们能用 DTD 定义的每一件事在 XML 模式的结构部分得到了解释。XML 模式是由 XML 语法写成，结构是指我们能用来定义标记的 XML 命令。当然，这意味着 XML 模式实际上只是 XML 的另一个应用（一个为了定义 XML 文档类的词汇表），并且正是如此而拥有了一个模式可以来描述它自己。

于是规范的结构部分是定义模式的元素和属性出现的地方。更重要的是，元素的内容模型在这里得到描述。内容模型明确的描述了允许的元素内部结构，结构是 XML 模式的核心。

2．编写模式

一个模式由导言，不定数量的定义和声明组成。下面几部分介绍这些定义。

（1）导言

在根元素模式可找到导言。这一定至少包含属性的三部分信息：

- targetNS:它是正在使用的模式的命名空间和 URI。
- version：用来指定模式的版本。

- xmlns：为 XML 模式规范提供命名空间。
- 可选的 finalDefault 和 exactDefault，为两种后面将要常涉及的扩展提供缺省值。它可能包括转出和转入，包括结构。

（2）声明

XML Schema 的声明由元素声明、属性和属性组声明、简单类型声明等组成。在此不逐一进行介绍。

2.4 文档型 XML 文档与数据型 XML 文档

按照 XML 文件的结构可以将其分为数据型的 XML 文档和文档型的 XML 文档，下面简单进行对比：

表 2.1 文档型 XML 文档与数据型 XML 文档的比较

	数据型 XML 文档	文档型 XML 文档
文档结构	数据型的 XML 文档由非常规则的结构，它的特点是规则的结构和清晰的内容	文档型的 XML 文档具有不规则的结构，而且数据的粒度也比较大。特点是包含了不规则的结构，大量原始数据和大量的混合内容。
元素顺序	元素之间层次比较重要，而同一层次间的顺序关系并不重要。	元素之间的顺序往往非常重要（这点很好理解，因为以文档为中心的 XML 文档是给人阅读的）
对于 XHTML 片断的包括	不包括或很少包括	经常包括
针对对象	数据型的 XML 文档通常是为机器设计的，也就是说主要是方便机器进行处理	主要针对的对象是人
主要用途	常用来在数据库之间传递数据	文档型的 XML 文档主要用于（服务器）server 和（客服端）client 之间传递数据
用途举例	比如关于销售订单或者是销售发票的 XML 文档	具体的例子如书本，电子邮件，广告等等。

2.5 XML 语言的相关规范

2.5.1 XML 文档的链接功能

作为一种 Web 语言，XML 的链接能力是非常强大的。可扩展链接语言 XLL (eXtensible Linking Language)是专为 XML 文档设计的。XLL 分为两部分：XML 链接语言 Xlink (XML Linking Language)和 XML 指针语言 Xpointer(XML Pointer Language)。XML 中的链接可以分为简单链接和扩展链接，简单链接功能相当于 HTML 中的<A>标记。XML 允许用户根据需要在同一文档中加入不同的具有自己独有属性的链接元素，这充分体现了 XML 的灵活性和可扩展性。扩展链接是 Xlink 为支持多方向多目的链接而提出的，与简单链接不同，它可以有多个目标。多方向链接是指链接的操作可以从任一个链接资源开始。XML 为支持多方向链接，允许链接元素本身处于链接资源之外。链接资源可以是 XML 文档中的任何元素，也可以是整个文档。由于链接元素本身处于链接资源之外，因此这种链接可以自由组织多个文档之间的关系。当然 HTML 也可以通过在所有文档中加<A>来实现同样的功能，但若要对这些关系进行修改，HTML 需要修改所有文档，而外部链接只要修改一个文档中的一个链接元素就可以了，显然要比 HTML 方便得多。Xpointer 用组成定位器的定位项指定资源，，可以包含单一的定位器，定位器可以包含绝对、相对、范围和字符串匹配定位项。Xpointer 提供对 XML 文档的内部结构(如一个字符串或选择的一个段落)的定位。Xlink 提供功能强大的链接方法，可以在文档之间建立单向或多向的复杂链接关系，还有注释链接、概要链接、扩展链接集等多种链接功能，这些链接功能相对于 HTML 都有很大的增强。Xlink 不仅支持 HTML 的单向链接，还支持多目的、多方向链接，它甚至还允许链接单独提出来存放在数据库中，或者是单独的文档中。

2.5.2 XML 文档的解析

对 XML 的操作依赖于对 XML 文档的正确解析。XML 解析器要分离出 XML 文档中所有的标签、元素、属性和文字内容，解析结果要体现 XML 文档所隐含的层次结构。XML 的解析需要为用户提供一套可操作的接口，根据接口类型的不同，XML 的解析可分为基于事件的解析和基于树结构的解析。基于

事件的解析只根据用户的应用要求返回结果，结果只有用户所需要的元素以及它的属性和内容或其他的一些元素信息，因此在解析过程中不用首先映射出整个文档的结构，对用户不需要的元素也不用记忆，这种解析比较简单而且适于用较小的内存空间来解析较大的文档。基于事件的 API 的代表是 SAX，SAX 是一个非盈利的程序员组建议的 XML API 标准，它是基于流的、以事件处理方式工作的模型，它将分析 XML 文档时所遇到的某些事件和数据的通知通过回调函数报告给应用程序。而基于树结构的解析将 XML 文档映射为一个类似于树的结构，用户可以从中获得文档中所有的元素、属性和内容以及它们之间的关系，并能通过这样的树对文档的结构和内容进行动态地修改。基于树结构的 API 的代表是 DOM，DOM 是由 W3C 所制定的与平台、语言无关的程序接口，它提供了动态访问和更新文档的内容、结构与风格的手段。可以用 DOM 对文档做进一步地处理，并将处理的结果更新到表示页面。DOM level1 已于 1998 年 10 月推出，它包括核心、HTML 和 XML 三部分。DOM 的核心部分提供了能表示结构化文档的一组低层的基本接口集，并定义了用来表示 XML 文档的扩展接口。HTML 和 XML 部分提供了高层的接口，可以作为更方便的文档视图。DOM 规范由对象和方法组成，通过它们，程序员可以更容易、更直接地对特定类型的文档进行访问和操作。由上可以看出，DOM 是基于树形结构的 W3C 推荐的 API 标准，而 SAX 是事件驱动的 API 标准，它们适用于不同的场合，在实际应用中要根据具体的情况加以选择。DOM 适合结构化编辑 XML 文档，例如排序、记录移动和与其他应用共享 XML 文档等情形，SAX 效率高，适合处理大文档，执行与文档结构无关的任务，提取特定节点内容等。而 DOM 是整体装入和处理 XML 文档，因此对系统资源的占用很大，效率低，速度慢。采用 DOM 模型对于 Server 端带来的代价不可忽略。而对于 SAX 来说，DOM 的很多工作它很难完成，例如排序、移动等。因为它缺少对于 XML 文档的整体视图。

2.5.3 XML 文档的显示

XML 的一个最重要的特征是把内容和显示格式分开。这样做带来很大的好处，可以让不同的用户按照各自希望的格式显示同一 XML 文档的内容，这也就意味着 XML 文档本身并没有关于格式方面的信息。为 XML 文档提供格式信

息的是样式表，样式表可以控制文件内容在显示时的版面风格，如页面的边距、各式标题及文字的字体、颜色、对齐方式等。对同一份 XML 文档使用不同的样式表就可以得到不同的输出效果，这特别适合于将文档表示在不同的场合，如显示、打印、出版等。适用于 XML 文档的样式表语言有层叠样式表 CSS（Cascading Style Sheets Level）和可扩展样式表语言 XSL(eXtensible Stylesheet languang)。CSS 是一种比较简单的样式表语言，既可以用于 HTML 文档，也可以用于 XML 文档。CSS 用简单的语法描述元素的显示格式，决定了页面的视觉外观，但是不会改变源文档的结构。而 XSL 是专为 XML 设计的样式表语言，它使用 XML 的语法，但综合了 DSSSL（Document Style and Semantics Specification Language SGML 文件的样式表）和 CSS 的特点。XSL 的优势在于它可以用于转换，当然 XSL 也可以把 XML 文档转换为 HTML 格式。而且同一个样式表可以用于多个具有相似树结构的文档。处理 XSL 样式表的是 XSL 样式表处理器，样式表处理器接受一个 XML 文档或数据，以及 XSL 样式表，输出特定样式的显示，其显示格式根据 XSL 样式表确定。这个处理过程分两步进行，首先，从 XML 源树构建一棵结果树，这一步称为树转换。然后，翻译结果树，产生所需的显示。这一步称为格式化，通过大量的定样命令 FO(Formatting Object）完成。XSLT 是将一种 XML 文档转换为另一种的语言。这意味着它提供了单源 XML 数据的机制，可以在 Web 页面中创建出用户动态更改的丰富视图，可以为目标通信过滤数据。XSLT 对于业务规则编码已经足够强大。它可以从数据生成图形（不仅仅是 Web 页面），它甚至可以处理与其他服务器的通信，以及在 XSLT 自身内部生成适当的消息。另一方面，XSLT 也称为基于模板的语言，它允许将某种模式映射到源文档中，而源文档的输出是用 XML、HTML或纯文本书写的。使用XSLT可以将XML文档的结构转换为不同的XML文档，例如可以更改 XML 文档的顺序、添加或删除元素、执行条件测试或者用元素的集合进行迭代。

2.6 XML 的相关应用及使用前景

2.6.1XML 的相关应用

作为互联网的新技术，XML 的应用非常广泛，可以说 XML 已经渗透到了互联网的各个角落。下面主要列举其几个经典应用：

1、电子商务

电子商务就是利用电子手段尤其是互联网进行商务活动。从技术上说，电子商务是通过互联网传输和交换商务数据，并能根据商务数据进行人工或自动处理。XML 的可扩展性和自相容性等特点，使它成为电子商务活动中数据交换的有力工具。

2、网络出版

网络出版自从出现以来，用于信息发布的主要是 HTML 技术，但是这种方式在跨媒体出版时遇到了极大的困难，人们需要为不同媒体制作不同版本。根据 XML 的内容与显示分离的特点，人们可以一次性制作内容，配以不同的样式单，实现一次制作多次出版。

3、移动通信

为了满足人们随时随地与互联网连接的需要，Phone.com 联合了 Nokia、Ericsson、Motorola 在 1997 年 6 月建立了 WAP 论坛，旨在利用已有的互联网技术和标准，为移动设备连接互联网建立全球性的统一规范。在 1998 年 5 月，推出了 WAP 规范 1.0 版。并于 1999 年 11 月发布最新的 1.2 版。WAP 规范包括 WAP 编程模型、无线置标语言 WML、微浏览器规范、轻量级协议栈、无线电话应用（WTA）框架、WAP 网关几个组件。其中 WML 是利用 XML 定义的专为手持设备开发的置标语言。另外 W3C 也定义了一个基于 XML 的手持设备置标语言 HDML，WML 和 HDML 非常类似，因为 WML 脱胎于 HDML，可以说根在 HDML，而花开 WML。需要指出的是，虽然人们在提到 WAP 时首先想到的是手机上网，但掌上电脑等手持设备的上网也可以使用 WAP。

2.6.2 XML 的前景展望

XML 自从出现以来，一直受到业界的广泛关注。自从 1998 年 2 月成为推荐标准后，许多厂商加强了对它的支持力度，包括 Microsoft、IBM、ORACLE 和 SUN 等，它们都推出了支持 XML 的产品或改造原有的产品支持 XML。W3C 也一直在致力于完善 XML 的标准体系，然而由于 XML 的复杂性和灵活性，加上工具的相对缺乏，增加了 XML 使用的难度，因此 XML 很难在短期内完全替代 HTML 成为互联网的主角。另外，由于 XML 是元置标语言，任何个人、公司和组织都可以利用它定义新的标准，这些标准间的通信成为了巨大的问题，

因此人们在各个领域形成一些标准化组织以统一这些标准，但是这些努力并不一定能够形成理想的结果。无论如何，XML 的出现为互联网的发展提供了新的动力，终将成为互联网上全新的开发平台。它促使了新类型软件和硬件的形成和发展，而这些发展又将反过来促进 XML 的发展。

3 XML 文档的存储方法

随着 XML 数据的海量出现，XML 文档的存储成为研究的热点。本章对 XML 与数据库进行比较后，重点介绍了 XML 的存储方式，特别是在关系数据库中的存储方法。

3.1 XML 和数据库

在开始讨论 XML 和数据库之前，我们先回答许多人都遇到过的问题："XML 是数据库吗？"如果仅按数据库这个术语的本质来看，XML 文件就是数据库，它是数据的集合。

XML 与数据库有许多相似之处。在思想上，XML 与数据库都是试图通过将内容与应用分开的方式使得同一数据可以被不同的应用所复用。对于数据库，它本身只描述和存放数据内容，至于数据内容如何被应用则由应用程序来决定，不同的应用程序通过数据库接口如 ODBC 等就可以访问数据库中的数据。而 XML 文档则通过 DTD 或 XML Schema 描述文档里的数据是如何组织存放的，它也不涉及这些数据应被如何应用，比如说应该怎样显示。XML 解析器把 XML 文档中的数据解析成层次型的结构化数据，这样数据就可以被不同的应用程序根据自己的需要加以应用。从整个组织结构上看，XML 也可与数据库一一对应。数据库系统提供了数据定义语言 DDL（Data Definition Language）来让用户定义他想存放的数据的逻辑结构，用户输入数据库的数据是逻辑数据。XML 系统则以 DTD 或 XML Schema 的形式让用户自己定义数据的逻辑结构，而用户的实际数据以 XML 文档的形式存放，解析器根据 DTD 或 XML Schema 的描述输出层次型的逻辑数据。在这个意义上，XML 中的 DTD 或 Schema 相当于数据库中用 DDL 对数据的逻辑结构的定义，而 XML 文档则相当于数据库系统中的物理数据文件。XML 的层次型逻辑结构对应于数据库的数据结构，XML 解析器则对应着数据库管理系统 DBMS 中实现物理数据和逻辑数据之间映射的部分。XML 提供了许多数据库所具备的东西如存储(XML 文档)、模式(DTD，XML schema，RElAX NG 等)、查询语言(Xquery，Xpath，XQL，XML-QL，QUILT 等)和编程接口(SAX，DOM，JDOM)等。

不过 XML 还缺少一些作为实用的数据库所应具备的特性如高效的存储、索引、安全、事务和数据一致性、多用户访问、触发器和多文档查询等。因此，

在数据量小、用户少和性能要求不太高的环境下，可以将 XML 文档用作数据库，但是却不适用于用户量大、数据集成度高以及性能要求高的作业环境。而在大多产品的环境中，要求有许多用户使用、需要严格的数据完整性并且对性能有很高的要求，XML 就不能胜任了。而且，考虑到像 dBase 和 Access 等数据库既便宜又十分易用，因此甚至在第一种情况下 XML 都很少有理由充当数据库的角色。

3.2 XML 存储方式比较

XML 作为 Internet 数据表示和交换的工具，已经成为了众多应用领域中的标准数据格式。XML 文档的存储方式极大地影响了查询处理的效率，成为一个非常重要的研究方向。目前一些组织和学者已经提出了一些 XML 文档的存储策略，数据库厂商也纷纷推出了自己的 XML 数据库产品。

图 3.1 描述了处理 XML 数据的基本方式。根据存储方式的不同，我们可以将处理 XML 数据的基本方式分为四种类型：文件系统、半结构化数据仓库、数据库管理系统和原生的 XML 数据库。

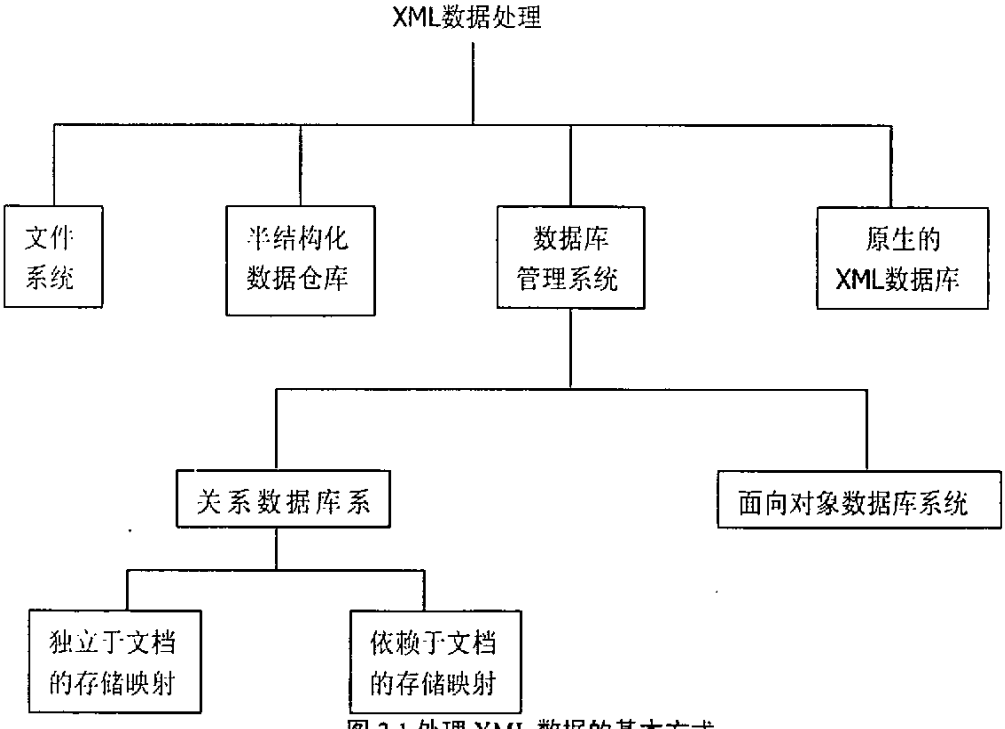

图 3.1 处理 XML 数据的基本方式

3.2.1 文件系统

XML 文件系统就是把 XML 直接存储为文件，访问时首先通过文件系统的目录结构，然后通过 XML 文档的元素结构来提供对数据的层次访问。基于文件的 XML 文件系统简单而容易实现，无需使用底层的数据库或对象存储管理。同时由于 XML 文档被直接存储为文本文件，文件的存储方式无需存储转换和重构查询结果，但是它存在如下四个方面的限制。

1、查询处理方面

由于文件系统本身不具备查询处理 XML 资料的能力，在处理查询时需要将 XML 文档解析为内存中的 DOM 树结构。然而这种存储方式在查询处理方面有明显的弱点，在每次浏览和查询时都要重复解析文档，并且在查询处理过程中整个文档都要驻留内存。尽管我们可以在内存中为文档建立索引，通过索引来定位查询所需的部分，然而维护这种索引的代价十分昂贵。

2、文件的大小方面

从上述的查询处理可知，要想查询首先必须把整个 XML 文档解析为内存中的 DOM 树结构，所以 XML 文件的大小受到内存大小的制约，当 XML 文件达到内存大小的三分之一时系统性能就将受到显著影响。

3、并发性

因为操作系统提供文件操作的最小单位是文件，所以 XML 的文件系统并发性很差。除了允许多人同时读 XML 文件外，不能提供任何其它并发操作。因此 XML 文件系统可用性受了极大的限制。

4、安全性

安全性方面，同样因为操作系统所用的安全机制通过是文件级的，所以不同的人可被赋予不同的处理权限处理 XML 文档，而文档的不同部分不能有不同的处理权限。

3.2.2 半结构化数据仓库

通常半结构化数据的内容与模式都包含在数据中，因此被称为“自描述的”。有些半结构化数据没有单独的模式，而有些只对数据做不严格的约束。由于 XML 数据与半结构化数据十分相似，利用半结构化数据仓库管理 XML 数据似

乎是比较自然的方式。在这种方法中，XML 数据被聚族存储为有向图。斯坦福大学的 Lore 项目在这方面作了初步尝试[39]，然而目前半结构化数据库技术尚不成熟，利用半结构化数据仓库处理 XML 的性能仍然难以让人满意。

3.2.3 原生的 XML 数据库

原生的 XML 数据库以某种逻辑的模型（如 XPath 模型或 DOM 模型）为基础来存储和检索 XML 数据，将 XML 文档作为一个整体来存储，并把 XML 文档作为数据处理的基本单位，而不要求有某种特定的物理存储模型，可以建立在数据库系统、对象管理器或文件系统之上。近年来出现了一些这样的数据库系统，如 Tamino[27]。

原生 XML 数据库有其优势，它以 XML 格式存储信息，完整地保留 XML 文档的信息，存储映射不需要 DTD 结构。因此原生 XML 数据库有利于文档存储和检索，消除了不必要的转换操作；以信息的原始 XML 格式检索文档，而不需任何附加编码，并且可以使信息以一定的样式显示；大多数 Native-XML 数据库具有完善的全文搜索的能力，包括整个同义字支持、字根（匹配一个字的所有形式：现在时、过去时和进行时）以及相近搜索（DTD NEAR XML Schema）。

但原生 XML 数据库也有其劣势：

1. 一些原生 XML 数据库需要选择哪些元素或属性用于索引，然后这个信息被用于建立索引，以便检索机制能用来快速定位相匹配的文档。当文档被添加到数据库中时，原生 XML 数据库就开始对文档内的所有信息建索引，这将导致对存储空间的需求飞速上升。当检索的信息位于大文档的末尾时，由于缺乏其他机制，原生 XML 数据库只能进行顺序扫描，一直到最后。

2. 当进行信息更新时，即使用户所需要的只是文档的很小一部分，很多原生 XML 平台也要从数据库返回整个文档。一些原生 XML 平台将 XML 文档保存到数据库之前进行了分解，但是如果所操作的 XML 文档具有复杂文档结构时，这种操作就显得笨拙不堪。

3. 原生的 XML 数据库目前尚不成熟，而关系数据库具有数据结构化、最低冗余度、较高的程序与数据独立性、易于扩充和易于编制应用程序等优点。XML 数据缺乏多重管理、协同工作和规划等能力，这些恰恰是大型关系数据库

所拥有的优点。

4. 缺乏明确的标准也是 XML 数据库领域的一个问题。XPath 查询语法不支持组、排列和摘要数据等功能，功能强大的 Query 查询语言仍然仅仅是一个设计表格。更有甚者当 Query 正式化时，依然不支持数据更新、插入和删除等功能。

3.2.4 面向对象数据库

面向对象数据库是用数据库管理系统存储 XML 的一种方式。面向对象的数据模型与 XML 很相似，它们都是层次树，都用的父子关系来描述数据之间的关系，都支持用户子定义复杂的数据类型。把 XML 映射到面向对象数据库中是很容易的事，但是这种存储 XML 的方法的最大缺点就是面向对象数据库本身的不成熟，无法提供快速的查询和高效的索引。

3.2.5 关系数据库

虽然关系模型与 XML 文档模型有一定的差距，但是利用关系数据库系统来处理 XML 数据的方式有如下优点：一方面当前的关系数据库的技术已十分成熟，商用的关系数据库系统都具有高性能的查询引擎、良好的可扩展性、安全性和健壮性，因此，利用关系数据库系统管理 XML 数据可以利用数据库的查询优化器和事务处理机制，保证 XML 数据的一致性和完整性。另一方面，目前大量的 WEB 数据主要存放在关系数据库中，XML—关系系统便于在关系数据库上建立适于二者的应用，并使关系数据库进入 Web 领域成为可能。但是由于数据模型上的差异，利用数据库系统管理 XML 数据也给数据库技术带来了许多新的挑战。近来，基于关系的 XML 数据处理技术受到了研究者和数据库厂商的关注，在这方面已经有了许多工作和成果。按照将 XML 数据转存为关系的映射方式的不同，可以将基于关系的 XML 数据存储分为独立于文档的关系存储和依赖于文档的关系存储两类。

3.2.6 支持 XML 的关系数据库产品

目前主流的数据库大多能够支持 XML，如微软的 SQL Server、Oracle 公司的 Oracle 9i、IBM 公司的 DB2 是三种具有代表性的支持 XML 的数据库，现分

述如下：

1、SQL Server2000

Microsoft 公司于 2000 年 5 月宣布 SQL Server 对 XML 提供支持，并且发布了一个预览版本。Microsoft SQL Server 的 XML 支持计划旨在产生一组功能强大的产品和服务来实现所谓的 BizTalk 框架。微软公司的 SQL Server2000 通过三种技术方式支持 XML 技术：

（1）在 SELECT 语句中增加了 FOR XML 条件子句；

（2）通过 Xpath 进行信息定位；

（3）在存储过程中使用 OpenXML 函数。

XML SQL 技术预览提供的 osulxm.dll 允许利用对象模型在 Script 脚本程序中实现基于 XML 的数据库操作。

2、Oracle 9i

Oracle 9i 对 XML 的支持主要体现在两个方面：内置 XML 开发包（XDK）和本机 XML 数据库支持（XDB）。

Java XML 开发工具包 XDK 预先装入了 Oracle 9i，而且链接了 CXDK。开发人员能够很容易地使用符合 W3C 的标准功能，在 Oracle 9i 里创建、操纵、解析和保存 XML 格式的数据。对于要通过内容管理程序保存、提取大量复杂 XML 数据的开发人员，Oracle 9i 里的 XML 数据库支持提供了本机的 XML 存储能力，从而能够优化性能。为了做到这一点 XDB 引入了 XML 类型，能够以本机方式保存 XML。

3、DB2

IBM 公司正在将 XML 查询技术集成到 DB2 数据库，这个名为 Xperanto 项目的软件计划于 2002 年底开发完成。IBM 对 XML 技术非常重视，它的很多数据管理软件都对 XML 提供了支持。2000 年，IBM 在发布 DB2V7 时，为其增加了一个支持 XML 的扩展接口——XML Extender。这是 IBM 在设计 DB2 时，基于面向对象的开放架构，为支持 XML 数据所做的延伸，包括与 XML 数据存储相关的 DTD，以及与 XML 数据管理相关的对完整性、安全性、可恢复性和可管理性等的支持。DB2 的 XML Extender 能很好的帮助用户方便的存储和检索 XML 数据。

3.3 XML 在关系数据库存储的映射策略分析

在关系数据库中存储管理 XML 文档的优点是可以直接利用关系数据库成熟的数据管理机制，不用为并发控制、安全性等问题做额外的工作。而且，关系数据库的数据也需要以 XML 的格式对外发布。在操作上需要做以下四步工作：

1、XML 文档结构（DTD 或 XML Schema）生成关系数据库模式；

2、将 XML 文档数据进行分割转存到关系数据库的表中；

3、将 XML 查询（如 Xpath 和 XQuery 查询）转化为对应的 SQL 查询；

4、将关系数据库查询结果重构成 XML 文档。

其中文档结构到关系表定义（即关系模式）的映射是最为关键的一步，因为不同的映射方法对性能的影响很大，而且这一步的好坏也决定了映射是否无损、正确，同时也制约了查询处理的性能。关于这方面的研究比较活跃[2][3][4][5][8][9][13][16][19]，当前给出的策略大致可以分为两类：独立于文档的存储映射、依赖于文档的存储映射。前一种是用固定的关系模式来存储所有的 XML 文档，这种方法不考虑 XML 文档本身的结构特点，对任何 XML 文档都产生一样的表结构，这对于一部分无模式的 XML 文档在关系数据库中的存储是非常有用的，文献[6]提出的方法即为此方法。后一种是从 XML 模式中导出关系模式，例如文献[8]中介绍了根据 XML 文档的结构 DTD 映射为关系模式的映射策略。

3.3.1 独立于文档的映射策略分析

采用独立于文档的映射策略，首先 XML 文档用一个有序的有向图来表示，文档的每个元素由图中的节点表示，并赋一个唯一的标识（ID），为了表示 XML 文档中每个 XML 对象的子元素的顺序，对图中的每一个节点引出的边也编上序号，叶子节点表示 XML 文档的值。Florecu 和 Kossman 提出了两种 XML 文档映射为关系数据库的存储方法 Edge 和 Binary 存储及其 6 个变种[6]。这里通过图 3.2 中给出的 XML 文档对其中的 Edge 存储和 Attribute 存储两种基本映射方法进行说明。

```
<project>
    <projname>XML</projname>
    <member member ID="&member3">
        <name>M.Franklin</name>
        <email>Franklin@sicnu.edu.cn</email>
        <publication>
            <author>M.Fanklin</author>
            <title> Query XML </title>
            <year>2005</year>
        </publication>
    </member>
    <member memberID="&member24">
        <name>J.Smith</name>
        <project>
            <projname>data mining</projname>
        </project>
        <publication>
            <author>J.Smith</author>
            <title> An Algorithm </title>
            <year>1999</year>
        </publication>
    </member>
</project>
```

图 3.2XML 文档

（1）边方法（Edge Approach）：Edge 映射将整个 XML 文档模型化为一个有向图，再将整个有向图存储在一个关系“Edge”中。XML 文档中的每个结点首先被赋予一个唯一的标识（ID）。Edge 表中的每个元组存储有向图中的一条边，包含边两端的结点标识（分别存储为 source 和 target）、目标结点的标记

tag 和一个表示结点在有向图中的次序的属性 ordinal，此外原子结点的文本串也被存储为一个属性 value。如此则表的结构为 Edge(source，tag，ordinal，target，value)，其中（source，ordinal）是 Edge 表的主键。表 3.1 中给出了图 3.2 中的 XML 文档对应的边，其中的结点标识为深度优先遍历 XML 的 DOM 树的次序。Edge 方法与 XML 文档的结构无关，在 XML 存储映射时不需要 DTD 或模式信息。然而，Edge 映射重复存储 XML 文档中的元素和属性的标记名，包含大量的冗余信息，同时将整个文档存储在一个关系表中的查询处理效率十分低下。

source	tag	ordinal	target	value
0	project	0	1	null
1	projname	0	2	“XML”
1	member	1	3	null
1	member	2	11	null
3	memberID	0	4	“&member3”
3	name	1	5	“M.Franklin”
3	email	2	6	“Franklin@sicnu.edu.cn
3	publication	3	7	null
7	author	0	8	“M.Franklin”
7	title	1	9	“Query XML”
7	year	2	10	“2005”
11	memberID	0	12	“& member24”
11	name	1	13	“J.Smith”
11	project	2	14	null
11	publication	3	16	null
14	projname	0	15	“Data Mining”
16	author	0	17	“J.Smith”
16	title	1	18	“An Algorithm”
16	year	2	19	“1999”

表 3.1 XML 文档对应的边表

（2）二元方法（Binary Approach）：二元方法中的 Attribute 映射将 Edge 表按照标记名 tag 水平划分成多个关系表，每个关系表对应一个元素/属性类型。例如图 3.2 中的 XML 文档将产生 10 个关系：project、projname、member、memberID、name、email、publication、author、title 和 year。每个二元表的结构为：Bname（source，ordinal，tag，target）。每个关系包含结点 ID、文本值及其在兄弟结点中的次序。与 Edge 方法相比，Attribute 映射更加简洁，然而由于没有保存边的信息，Attribute 方法在查询处理时需要用到文档的 DTD 来确定元素间的父子关系。同时，对于元素类型较多的文档，Attribute 映射将产生大量的关系表。

（3）通用表（Universal Table）：用单个表来存储所有边，相当于将所有二元表做全外联接操作。表的结构是：Universal(source，$dinal_{n1}$，$flag_{n1}$，$target_{n1}$，$ordinal_{n2}$，$flag_{n2}$，$target_{n2}$，$ordinal_{nk}$，$flag_{nk}$，$target_{nk}$)。

（4）不同值表（Separate Value Table）：为不同的数据类型创建一个表。表结构为：V_{type}（vid，value）。

（5）内联（Inlining）：将值和边存储在同一个表中，相当于将边映射方法中得到的表和不同值表做外联接。

这里给出了三种映射边的方法，两种映射节点的方法，通过它们产生的关系数据库的规模，执行不同 XML 查询和重构 XML 文档所需时间等方面的开销进行了大量实验，结果显示，以二元方法映射边和内联方法映射值这两种方法结合使用，能够达到最好的性能[9]。这些方法比较简单，容易掌握和使用，适用于一些简单的 XML 文档，特别是对于无模式的 XML 文档的存储是不错的选择。但缺点是没有考虑 XML 文档的结构和数据特性，而是直接给出了在关系数据库中的存储模式，包含了大量的空值，会造成空间的浪费，在查询时需要进行大量的连接操作，大大降低了查询的性能。

3.3.2 依赖于文档的映射策略分析

这种策略主要是依靠 XML 文档的 DTD 或者 Schema 信息，来映射其对应的关系模式，文献[7]中给出了将 DTD 映射为关系模式的三种方法：基本内联技术、共享内联技术和混合内联技术。在运用这些方法映射之前，要将文档模

式按照文献[7]中给出的方法进行简化，分为以下三种类型：

(1)扁平化转换，即变换每一个嵌套的定义到平坦的表示，使二元操作“，”和“|”不出现在任何操作之内，例如

$(e_1, e_2)^*\text{->}e_1^*, e_2^*$　　$(e_1, e_2)?\text{->}e_1?, e_2?$　　$(e_1|e_2)\text{->}e_1,e_2$

(2)简单化转换，将连续的多个一元操作转换成一个一元操作。例如

$e_1^{**}\text{->}e_1^*$　　$e_1^*?\text{->}e_1^*$　　$e_1?^*\text{->}e_1^*$　　$e_1??\text{->}e_1?$

(3)分组转换，将多个具有相同名称的子元素进行聚合。例如

…,e*,…e*,…->…,e*,…　　…,e*,…,e?,…->…,e*,…

…,e?,…,e*,…->…,e*,…　　…,e?,…,e?,…->…,e*,…

…,e,…,e,…->…,e*,…

另外，所有的“+”操作符都转换成“*”操作符。假设一个元素 a 的定义为：<!ELEMENT a((b|c|e)?, (e?|(f?, (b, b)*))*)>，通过运用上面给出的三种类型的简化方法对这个复杂的表达式进行转化，得到的结果应该是：<!ELEMENT a(b*, c?, e*, f*)>。

然后用模式图来表示简化的模式结构，例如图 3.4 给出的就是图 3.3 中的 DTD 对应的 DTD 图。图中的节点分别代表简化后 DTD 中的元素、属性和操作符。每个元素在图中仅出现一次，属性和操作符可以出现任意次。若元素 A 包含一个元素 B，则有一条从 A 指向 B 的边；若 A 可以包含 0 个或多个 B，则有一条从 A 经由一个“*”节点到 B 的通路。

（1）基本内联技术。首先为每个元素图生成一个对应的表定义，并为所有可以直接达到的元素生成对应的属性，对于那些通过节点达到元素则另外生成一个表定义。基本内联技术会产生大量的关系，其中有很多的重复部分，效率低下，很少采用。

（2）共享内联技术。其映射原则是：①将文档 DTD 图中入度大于 1 或者等于 0 的元素节点建立单独的关系；②入度等于 1 的元素或者属性内联到其它元素关系中，这样做的目的是使得 XML 文档尽量不要被分解得太碎，有助于提高查询的效率；③“*”操作符的子元素创建单独的关系；④如果模式图中有一个环，即形成相互递归的元素，则必须建立关系打破它。这样就确定了哪些元素必须分离出来创建一个单独的关系，对于其它元素的关系模式的创建就依赖于这些分离出来的元素。对于分离出来作为独立关系的每一个元素节点 A，

只要 A 可以达到节点 B，并且从 A 到 B 的路径上不包含分离出去作为独立关系的节点，则 A 将内联所有节点 B 到 A 的关系中。

```
<!ELEMENT book(booktitle,author)>
<!ELEMENT article(title,author*,contactauthor>
<!ELEMENT contactauthor (#PCDATA)>
<!ELEMENT monograph(title,author,editor)>
<!ELEMENT editor(monograph*)>
    <!ATTLIST editor name CDATA #REQUIRED>
<!ELEMENT author(name,address)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
```

图 3.3XML 文档的 DTD book.dtd

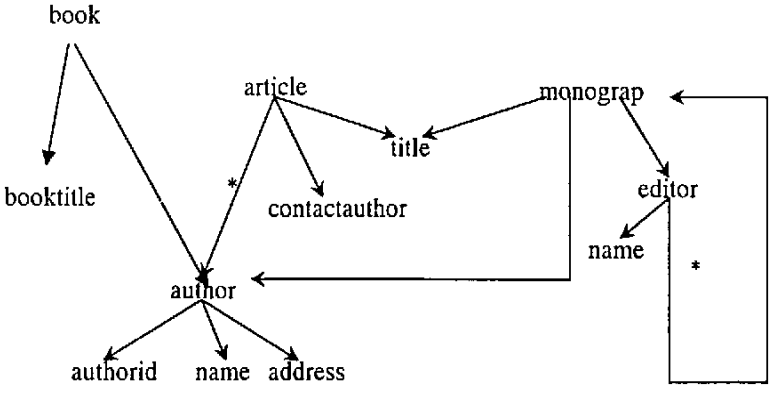

图 3.4 book..dtd 对应的 DTD 图

使用共享内联策略将图 3.4 中的 DTD 图进行映射，得到的关系模式为：

book(bookID,booktitle)

article(articleID,article.contactauthor)

monograph(monographID,monograph.parentid,monograph.parentCODE,monograph.editor.name)

title(titleID,title.parentid,title.parentCODE,title)

author(authorID,author.parentid,author.parentCODE,author.name,author.address,author.authorid)

这里每个关系都有一个 ID 列作为主键，其中的属性都是以从关系的根节点到该属性在 DTD 图中的路径来命名的。有父节点的节点元素所对应的关系都有一个 parentID 属性作为外键。例如：author 关系有一个外键 author.parentID 指向 article 关系的 articleID。parentCODE 用于区别关系对应结点的父结点(当存在多个父结点时)。

（3）混合内联技术。混合内联技术和共享内联技术基本一样，但是它还内联了一些更多的节点元素。在混合内联技术中，将模式图中节点入度大于 1，但是没有递归，并且不经过“*”到达的节点也内联到其它元素中，而不是作为单独的关系。如图 3.4 中的 title 元素，如果应用混合内联技术，则会被分别内联到 article 元素和 monograph 元素的关系中，即会得到下面的关系模式。

book(bookID,booktitle)

article(articleID,article,contactauthor,authorid,article.title)

monograph(monographID,monograph.parentid,monograph.title,monograph.author.name,monograph.author.address,monograph.author.authorid,monograph.editor.name)

author(authorID,athor.parentid,author.name,author.address,author.authorid)

共享内联技术相对减少了 XML 查询转换成的 SQL 语句数目，但是增加了每一个 SQL 语句查询中的连接操作的次数；混合内联技术相对减少了每个 SQL 查询中的连接运算，但是增加了 SQL 语句的数目。

4 保持函数依赖映射策略设计

文档类型定义(DTD)可以被看成是XML文档的模式，如图4.1展示了文档集合与关系数据库的对应关系[16]。从DTD到关系模式的转换是非常重要的。上一章主要介绍了XML到关系数据库存储的一些经典算法，特别是依赖于文档的算法。虽然依赖于文档的存储方法能保持DTD所蕴含的一些语义信息，但考虑的并不充分。语义信息对数据存储模式设计、查询优化和更新异常检查等来说是十分重要的，如果DTD上指定了XML的函数依赖及其它的语义信息，在映射到关系数据库中时予以考虑将会带来很多便利。目前许多考虑了函数依赖的算法都是在第三章所述几种方法的基础上改进得来的[4]。文献[2]提出了一种新的基于DTD的Inlining方法，这种方法只为符合规则的元素创建关系，把值元素及（0，1）、（1，1）约束关系中子元素的属性当作其父结点的属性处理，避免了文档碎片的出现，创建关系数目少。本章提出的FD-Inlining算法在文献[2]方法的基础进行了改进，既能保持函数依赖又能保持其内容和结构。在FD-Inlining的后续工作中还考虑了DTD所蕴含的其它语义信息，最后得到的关系保持了这些语义。

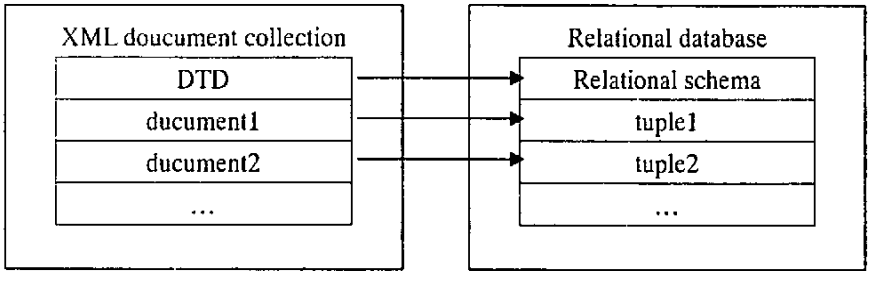

图4.1XML文档集合与关系数据库

4.1 DTD图和数据图

有效的XML文档通常带有描述文档结构的文档类型定义DTD，DTD通过正则文法的形式描述了XML文档中的元素及其子元素和属性间的嵌套关系。尽管DTD描述的文档结构是非精确的，利用DTD来生成关系模式不会影响XML到关系映射的有效性，但DTD的正则文法可能很复杂。例如，一个DTD中的元素定义可能为<!ELEMEMT (a,(b|c?|d)*,e)>，然而对于存储映射而言，我们只关心DTD描述的文档结构。因此在映射之前，我们可以先将这种复杂的文法描述简

化，简化的思想是将层状嵌套的DTD描述打平，并消除“*”和“？”之外的正则操作符，详细简化过程参见3.3.2节。经过简化以后，我们就可以将一个DTD文本描述模型化为一个有向图，称为DTD图[11]，它表示一个DTD的结构，DTD图中的每个非叶结点是DTD中的一个元素，叶结点表示原子元素或属性。结点用元素或属性名标记。每一个元素在图中出现且仅出现一次，属性和操作符在DTD图中出现的次数则与它们在DTD中出现的次数相同。DTD图中的边从元素结点指向子元素结点或属性结点。边可能带有标记*，？或不带标记，分别表示元素与其子元素或属性间的1对任意、1对0或1、1对1关系。图3.4即为图3.3所对应的DTD图。

与 DTD 相似，一个 XML 文档也可以模型化为一个有序的有向图，称为数据图。数据图中的每个结点对应文档中的元素或属性，每个结点用与其对应的元素或属性名标记。文档中的原子元素或属性带有字符串值，将字符串值作为原子元素或属性结点的儿子结点（叶结点）加入数据图，数据图中的边表示结点间的嵌套关系，如图 4.2 即为一个数据图。

4.2 基本概念：XML 的函数依赖

例 1 下面是一个描述学校信息的数据库 DTD D_1：

```
<!ELEMENT courses(course*)>
<!ELEMENT course(title,takenby)>
    <!ATTLIST course cno CDATA #REQUIRED>
<!ELEMENT title(#PCDATA)>
<!ELEMENT takenby(student*)>
<!ELEMENT student(sname,teacher)>
    <!ATTLIST student sno CDATA #REQUIRED>
<!ELEMENT sname(#PCDATA)>
<!ELEMENT teacher(tname)>
    <!ATTLIST teacher tno CDATA # REQUIED>
<!ELEMENT tname(#PCDATA)>
```

图 4.2 是一个符合 D_1 的 XML 文档树形结构。在此图中我们可以看到，在一门确定的课程(course)中，学号(sno)才能唯一地标识一个教师姓名(tname)，而

在不同的课程中，即使学号相同，教师姓名也不一定相同。即存在这样的约束：对于一门确定的课程(course)，学生的学号(sno)能唯一地标识教师名(tname)，表示成：

(courses.course,[courses.course.takenby.student.sno]->[courses.course.takenby.student.teacher.tname]).

上述约束有它成立的前提条件，即对于一门确定的课程。由此可知，函数依赖在 XML 中和在关系数据库中存在着明显的不同：关系模式表示了属性的平面关系，因而关系数据库中的函数依赖是全局的，不涉及函数依赖的成立范围；而 XML 文档是有层次结构的树形结构，函数依赖在 XML 中往往会涉及到它在 XML 树中成立的范围。

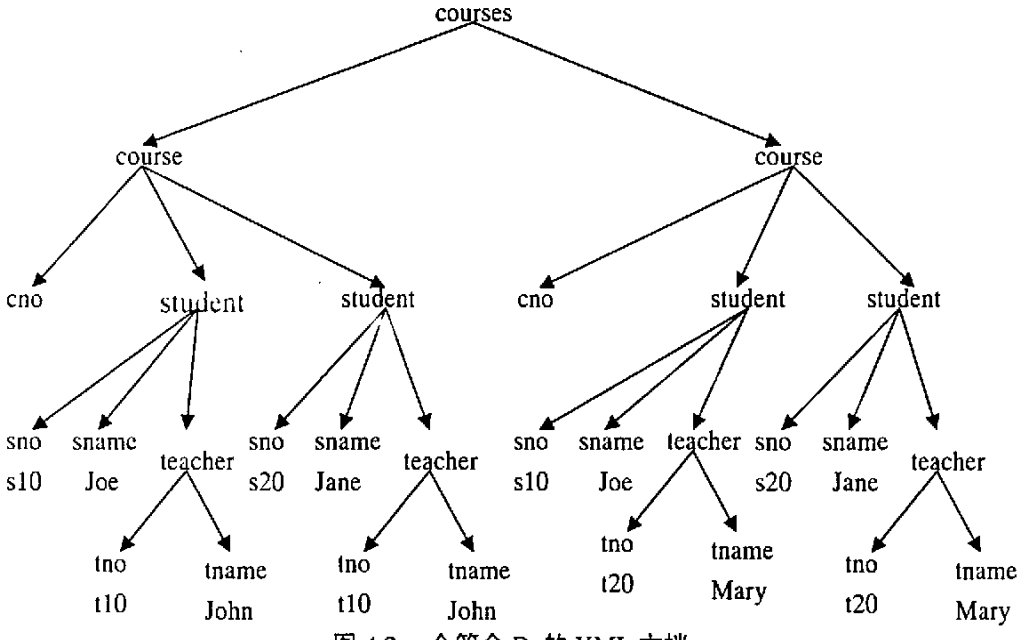

图 4.2 一个符合 D_1 的 XML 文档

在此给出 XML 函数依赖[1]的定义：

定义 4.1 给定 DTD D 和满足 D 的 XML 树 T，在 D 上的函数依赖 FD f 具有如下形式：(Q，[P_{x1}，P_{x2}...P_{xn}->P_y])，其中

(1) Q 是 XML 函数依赖的头部路径，是一个从 XML 文档的根节点开始的简单的 XPath 表达式，定义约束保持的范围；

(2) P_{x1}，$P_{x2}...P_{xn}$是XML函数依赖f的左部路径，P_{xi}（$1\leqslant i\leqslant n$）是一个左部实体类型(LHS)。一个左部实体类型由一个XML文档中的元素名字和一些关键操作属性组成；

(3) P_y是XML函数依赖f的右部路径，是一个右部实体类型(RHS)，一个右部实体类型由一个XML文档中的元素名和一个操作属性名组成。

在头部f路径约束保持范围内，对于树T中任意两个子树t_1和t_2，若对所有的P_{xi}（$1\leqslant i\leqslant n$）都有$t_1[P_{xi}]=t_2[P_{xi}]$，那么一定有$t_1[P_y]=t_2[P_y]$。

从FD的定义可知，任何一个LHS或RHS实体的值都是一个简单类型。如果一个LHS实体只有一个元素结点名字而没有任何的属性结点的话，那么该LHS实体其实就等于元素结点的文本孩子的值(PCDATA)。否则，如果该实体有一个元素名字而且后面带了属性结点，则该实体的值就是这些属性结点的值。而且必须强调的是，LHS实体是必须存在的。另一方面，如果一个RHS实体是只由一个元素结点名字组成而不带任何属性结点，那么该RHS实体的值就等于那个元素结点的文本孩子的值，而且必须保证在那个元素结点下面不能再有其它的子树，否则它的值就是等于所带的属性结点的值。

下面以第三章中的book.dtd为例（如图3.4所示），可以定义一些XML函数依赖如下：

FD1=[bookid->booktitle]

FD2=[bookid->author.id]

FD3=[bookid->author.name]

FD4=[bookid->author.address]

FD5=[monograp.id->editor.name]

FD6=[editor,monograp.id->author.id]

FD7=[editor,monograp.id->author.name]

FD8=[editor,monograp.id->author.address]

FD9=[editor,monograp.id->monograp.title]

FD10=[article.id->article.contactauthor]

FD11=[article.id->article.title]

FD12=[article,authorid->author.name]

FD13=[article,authorid->author.address]

就关系数据而言，上述函数依赖中的 FD6－FD9，及 FD12－FD13 实际上是分别与下面的函数依赖对应的：

FD6＝[editorid,monograp.id->author.id]

FD7＝[editorid,monograp.id->author.name]

FD8＝[editorid,monograp.id->author.address]

FD9＝[monograp.id-> monograp.title]

FD12＝[authorid->author.name]

FD13＝[authorid->author.address]

4.3 DTD 中的语义约束分析

现有的映射算法通常只注重结构上的映射而忽略其中的约束关系，从而导致语义丢失。图4.3为一种典型的结构映射算法[12]，这种映射算法丢失了“C|D”之间的结合结束：(C=null∧D≠null)∨(C≠null∧D=null)。基于结构的映射还可能丢失一些其他的约束。本节将对DTD中的约束进行分析。

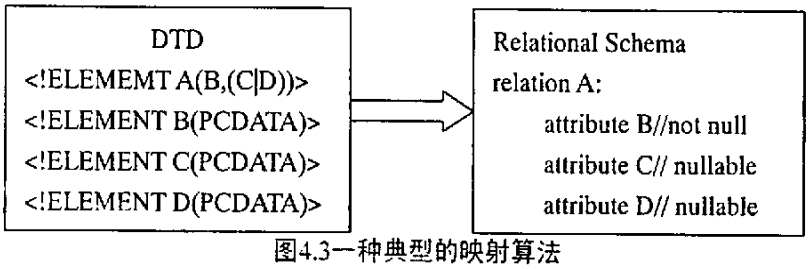

图4.3一种典型的映射算法

4.3.1 域约束

域约束是指一个元素或者属性的值被限制在一定的域范围内。具体有：

（1） 固定值约束，如<!ATTLIST form method CDATA #FIXED “POST”>

（2） 枚举值约束，如<!ATTLIST author gender(male female)>

（3） 默认值约束，如<!ATTLIST author gender(male female) “male”>

4.3.2 组合约束

组合约束是指同一元素中各子元素之间的约束关系。例如，A(C，D)中 C 和 D 之间的关系是“AND”，即（B∧C≠NULL），E（F|G）中 F 和 G 之间的

关系为“OR”，即（F∨G）≠NULL。

4.3.3 基数约束

基数约束是指父元素与各个子元素（或属性）之间的对应关系。例如：<!ELEMENT book(name，author+，reference*，price?)

- “name”表示“book”有且仅有一个“name”
- “price?”表示“book”可以没有或有一个“price”
- “author+”表示“book”可以有一个或更多“author”
- “reference*”表示“book”可以没有或有很多“reference”

基数约束一共有四种：“有且仅有一个”、“0 个或一个”、“一个或很多”、“0 个或很多”，可以分别用（1，1）、（0，1）、（1，N）、（0，N）来表示这 4 种基数约束。

4.3.4 包含约束

包含约束是指出现在某个元素（或属性）中的值必须包涵于另一个元素中，类似于关系模型中的参照完整性约束（Referential Integrity）。例如在 book.dtd 中对于元素 author 有 author.id ⊆ author，author.name ⊆ author，author.address⊆author。

4.4 FD-Inlining 算法

FD-Inlining 算法是在文献[1]方法的基础上改进而来，在保持文献[1]中方法优势的同时，又考虑了 DTD 所蕴含的函数依赖，映射过程既保持了函数依赖又能保持内容和结构。

4.4.1 几个符号表示

E 是元素名字的有限集；

A 是一个元素名字 e∈E 到多个属性名的映射；

T 是一系列不可分数据类型(例如 string，integer，date，ID，IDREF，IDREFS 等)。

M 是一个元素名字 e∈E 到元素类型定义的映射，M(e)∈α，α 是一个正

则表达式：$\alpha ::= \varepsilon \mid \tau \mid \alpha + \alpha \mid \alpha, \alpha \mid \alpha^*$，其中 ε 代表空元素，$\tau \in T$，“,”和“*”分别代表连接和闭包；

4.4.2 保持函数依赖的内联算法FD-Inlining

在本文的映射过程中，首先对满足条件的元素应用FD-Inlining算法，为其创建关系表。递归调用算法FD-Inlining，最后返回一个关系。在算法FD-Inlining中同样把（0，1）、（1，1）约束关系中子元素的属性当作父元素的属性处理，这些属性成为返回关系表中的属性。调用完FD-Inlining之后，处理（0，n）、（1，n）约束，最后进行关键字处理。在应用FD-Inlining之前为每个元素增加ID类型属性和唯一标识元素（如果已经存在ID类型属性，则不再添加）。

首先，对满足以下条件的元素应用FD-Inlining算法，并为它建立关系。

- Rule1：函数依赖(Q，[P_{x1}，P_{x2}...P_{xn}->P_y])，P_y中的元素；
- Rule2：没有在其它任何元素定义中出现的元素；
- Rule3：在其它元素类型定义中多次出现的非#PCDATA元素；
- Rule4：在另一个元素类型定义中多次出现，并带有*或+操作的非#PCDATA元素；
- Rule5：如果出现嵌套，则在嵌套中选一个元素来应用FD-Inlining算法。

FD-Inlining算法如下：

初始化：currE1=e，attSet=Φ；

输入：currE1,attSet，输出：ResultSet；

1、把A(currE1)中的所有属性（除IDREF，IDREFS外）置于attSet中；

2、Set ResultSet=Φ；

3、if (currE1是满足Rule1的结点且为1)

then attest=attSet$\cup$(P_{x1}，P_{x2}...P_{xn}，P_y)；

4、对M(currE1)中出现的带有(1，0)或(1，1)约束的元素进行简化，得到(e_1,e_2,...e_n)；

5、对每一个e_i进行如下操作：

{ if M(e_i)$\in$ T then attSet=attSet$\cup${e_i}；

else attSet=attSet$\cup$FD－Inlining(e_i，Φ)；}

If attSet=Φ,attSet={currE1}；

6、ResultSet=ResultSet∪atteSet;

7、Return ResultSet;

按照以上方法，图3.4中符合规则的元素分别是book、article、author、monograp应用FD-Inlining算法得到的关系分别是：

book (book.id，book.title，author.id，author.name，author.address)

article(article.id，article.title，article. contactauthor)

monograp(monograp.id，editor.id，monograp.title，author.id，author.name，author.address，editor.name)

author1(editorid，monograp.id，author.id，author.name，author.address)

author2(author.id，author.name，author.address)

4.5 约束处理

经过以上的 FD-inlining 算法中处理进行了保持函数依赖的内联，但并未对在 4.3 节中阐述的约束进行处理。本策略的目的是将 XML DTD 映射为关系模式，也就是最后要转化为 SQL 语言，然后在 DBMS 执行生成具体的关系模式。

(1) 关键字处理：

如果是为符合 Rule1 的元素建立的表，则(P_{x1}，P_{x2}...P_{xn})共同作为表的关键字，否则元素的 ID 类型的属性作为表的关键字。例如，表 author1 的关键字为(editor.id，monograp.id)，在表 book、article、monograph 及 author2 中则以 ID 属性作为表的关键字。可以用“PRIMARY KEY”语句来处理。

(2) 基数约束处理：

在 FD-inlining 算法中已经对(1，1)与(0，1)约束进行了处理，这里对(1，n)、(0，n)约束进行处理。如果存在一个表与(1，n)或(0，n)约束对应，则在该表中添加一个外键，指向其父结点元素。但如果表中存在父元素的 ID 类型属性，则不再添加外键。如果不存在对应的表，则为此元素建一个新表，并为其添加一个外键，指向其父结点元素。可以用“FORELGN KEY”、“CHECK”和“TRIGGER”语句处理。

(3)域约束处理：

对于固定值和枚举值约束可以用“CHECK（VALUE IN）”语句处理；而默认值约束用“DEFAULT”语句处理。

(4)组合约束处理：

对于“AND”类型的约束无需处理，对于“OR”类型的约束，使用“CHECK”语句处理。例如“A|B 映射后的语句为“CHECK((A is NOT NULL AND B is NULL) OR (A is NULL AND B is NOT NULL))”。

(5)包含约束处理：

在用保持函数依赖的内联算法 FD－Inlining 进行内联的过程，以及基数约束处理时添加外键的过程中，已经对包含约束进行了处理。例如，author 元素在应用 FD－Inlining 算法时生成其对应的关系 author，关系 author 中包含了属性 author.id、author.name，author.address。在基数约束处理时，对于 article 的子元素 author 所对应的关系中添加了 article.id，指向关系 article。

经过约束处理后，生成的关系变为：

book (book.id，book.title，author.id，author.name，author.address)

article(article.id，article.title，article. contactauthor)

monograp(monograp.id，editor.id，monograp.title，author.id，author.name，author.address，editor.name)

author1(editorid，monograp.id，author.id，author.name，author.address)

author2(author.id，author.name，author.address，article.id)

下面以表author2为例，给出其转化后生成的SQL语句：

```
CREATE TABLE author(
        author _id NUMBER NOT NULL,
        name VARCHAR(50) NOT NULL,
        addressVARCHAR(50) NOT NULL,
        PRIMARY KEY (author.id),
        FOREIGN KEY(article.id) REFERENCES article(article.id));
```

根据本文提出的保持函数依赖的映射策略，可以将符合一个 DTD 定义的文档的数据图对应地映射为与该 DTD 映射产生的关系模式对应的关系表。这样，通过 DTD-关系模式和数据图-关系表映射，我们可以将 XML 文档转化为关系表存储在关系数据库中。例如图 4.4 中的 XML 数据图在本映射策略下得到的关系表如图 4.5 所示。

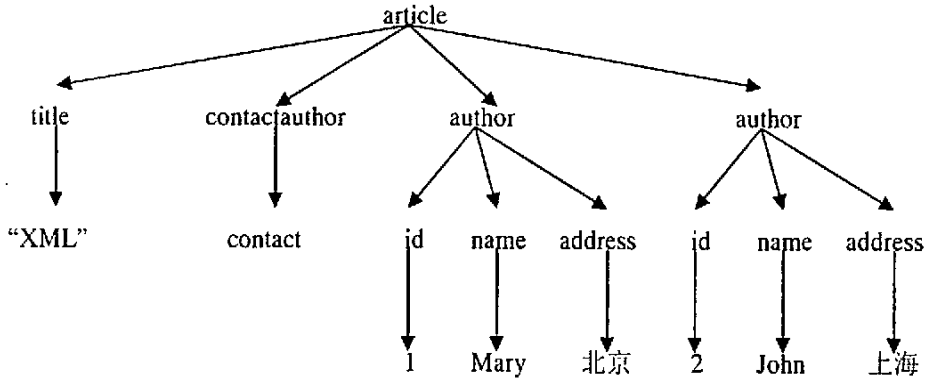

图 4.4XML 数据图示例

article

article.id	title	contactauthor
1	XML	contact

author

author.id	name	address	article.id
1	Mary	北京	1
2	John	上海	1

图 4.5XML 文档存储示例

4.6 数据完整性约束分析

数据完整性约束条件是在数据结构基础上对信息在语意和结构上的约束，可以说是数据结构的附属信息。在探讨 DTD 与关系数据模型间的映射时，为了保证信息的完整性和正确性，降低失真率，必须保证数据完整性约束条件。

在关系数据模型中，数据完整性定义有三类：实体完整性、参照完整性和用户自定义完整性。

4.6.1 实体完整性分析

关系数据模型中的实体完整性规则：若属性 A 是基本关系 R 的主属性，则属性 A 不能取空值。

对于实体完整性规则，在此需要探讨的其实就是实体完整性在 DTD 中的描述形式。DTD 中为属性界定了 ID 属性类型和#REQUIRED 属性默认类型。前者可以保障属性值永不重复，后者可以保障属性必须出现。在运用 FD-Inlining 算法之前，给 DTD 中每一个有属性或子元素且没有 ID 属性的元素添加一个 ID 属性。当运用完 FD-Inlining 算法后进行关键字处理时，对于符合规则的且生成了关系的元素，其 ID 或者作为主键出现，或者做为主健中的一个属性出现，从而使生成的关系模式保证了实体的完整性。

4.6.2 参照完整性分析

一般来说，在关系模型中实体及实体间的联系都是用关系来描述的，关系与关系之间存在着引用，即一个关系中的某个属性的取值需要参照另一个关系的属性取值。关系模型中参照完整性规则为：若属性（或属性组）F是基本关系R的外码，它与基本关系S的主码Ks相对应（基本关系R与S不一定是不同的关系），则对于R中每个元组在F上的值必须为：

- 或者取空值（F的每个属性值均为空值）；
- 或者等于S中某个元组的主码值；

在进行基数约束处理时考虑了参照完整性，即对外键的考虑。例如图3.4中的author元素，应用完FD-Inlining后得到的关系为author1 (author.id, author.name, author.address)，在进行基数约束处理后，为该表了添加了article.id属性，其为article元素所对应关系表中的主属性。因此本映射策略保持了参照完整性。

4.6.3 用户自定义完整性分析

任何关系数据库系统都应该支持实体完整性和参照完整性，除此之外根据不同需要往往还有一些特殊的约束条件，即用户定义的完整性。在 DTD 中，用户也可以自己定义一些约束，比如前面提到过的域约束和组合约束。在约束处理中对这两种约束均予以考虑，从而可以保证用户自定义完整性。

4.7 本策略与共享内联方法的比较

如前面所述，经典的映射算法没有充分考虑 DTD 蕴涵的数据依赖， 丢失了一些数据依赖。下面，以共享内联方法为例，分析共享内联方法丢失了哪些

数据依赖。

1、元素之间一对一的联系

共享内联方法确保一个元素只在一个关系中出现，为入度大于 1 的结点(元素)单独建立一个关系。子结点和父结点的联系通过 parentID 保持，丢失了元素之间一对一的联系。例如，图 3.4 Book.dtd 中元素 book 与 author 之间是一对一的联系，存在函数依赖 book.id－>author.id。由于结点 author 的入度大于 1，为 author 单独建立一个关系。author.parentID 指向 author 的父结点 book，丢失了函数依赖 book.id－>author.id，也就丢失了 book 与 author 之间一对一的联系。

2、元素之间一对多的联系

共享内联方法为 DTD 图中一对多的联系(如图 3.4 中*)下的元素单独建立一个关系，子结点和父结点的联系通过 parentID 保持。当子结点存在多个父结点时，通过 parentCODE 辨别与哪个父结点发生联系，丢失了元素之间一对多的联系。例如图 3.4book.dtd 中元素 article 与 author 之间是一对多的联系，存在函数依赖 author.id－>artcle.id。由于 author 存在多个父结点，需要通过 parentCODE 辨别与哪个父结点发生联系，丢失了函数依赖 author.id－>article.id，也就丢失了 article 与 author 之间一对多的联系。

由于共享内联方法映射得到的关系模式丢失了一些数据依赖，因此它不能保证 XML 文档的完整性。与共享内联方法相比，本策略保持了 DTD 蕴涵的数据依赖，保证了 XML 文档的完整性（如 4.6 节所述）。元素之间一对一的联系可以通过建立关系的主码(primarykey)来反映，从而保证了实体完整性（如 4.6.1 所述）。元素之间一对多的联系则可以通过建立外码(foreignkey)来反映，从而保证了参照完整性（如 4.6.2 所述）。

比较本映射策略和共享内联方法的映射结果，可以发现本策略比共享内联方法合并的关系更多，共享内联方法要求每个元素只在一个关系中出现，而本策略则允许一个元素在多个关系当中出现。例如，应用本策略得到的关系模式中，元素 author 同时出现在关系 book、article，monograph 和 author 中，而在由共享内联方法映射得到的关系模式中元素 author 只出现在关系 author 中。本策略虽然增加了数据的冗余度，但也提高了查询效率。例如，查询 booktitle 为“XML Database”的 author，共享内联方法需要连接两个关系 book 和 author 进行查询，而本策略只需查询关系 book 即可。因此采用本策略映射得到的关系

模式减少了查询的连接次数。

5 基于 XML 的查询

上一章给出了一种 XML DTD 到关系模式的映射方法，该方法保持了 XML 的函数依赖以及 DTD 所蕴含的其它语义约束的。通过这种转换，可以更好的管理 XML 数据，包括查询和提取数据，这要通过 XML 查询来实现。XML 查询是在复杂的 XML 树状结构中查询符合条件的结点，并把结果组合成一份 XML 文件或者一组结点后传回。XML 查询跟一般的 SQL 最大的不同就是 XML 查询是用来查询 XML 文件中的数据，而一般的 SQL 查询的是数据表中的信息。因为 XML 查询与 SQL 查询是不同的，所以当利用关系数据库作为存储 XML 文档的手段时就涉及到两种查询的转化问题。一般关系数据库的数据表结构是二维的，即所有的数据由行与列组成；而 XML 文件具有树状的结构，还涉及命名域的参考，以至于一份 XML 文件可能具有相当复杂的结构。因此，XML 查询与一般关系数据库的查询相比较为复杂。

5.1 XML 查询语言

每一种数据查询语言都相应地对应着一种数据类型，如结果化查询语言（SQL）对应着关系数据库模型，面向对象查询语言（OQL）对应着对象模型，XML 查询语言对应的数据模型既不同于关系模型，也不同于对象模型，而是和近几年来数据库研究人员进行了大量研究的半结构化数据模型相似。在半结构化数据模型中，大都将半结构化数据表现为某种图或树的结构，而 XML 文档也可表现为某种图或树的结构。

通常情况下，XML 查询语言具有如下一些特征：

（1）查询的对象为 XML 文档，查询返回的结果也应为 XML 文档，即 XML 查询语言应具有封闭性。

（2）查询过程应该是说明性的，而非过程性的。即用户只需要说明想得到什么，而不必说明如何得到。

（3）应具有潜在的代数基础。

（4）应简单、通用、易于被解析器进行解析。

（5）支持选择、重构、链接和析取等操作，还应该支持复杂选择条件和布尔操作。

（6）支持各种简单的数据类型，并提供一种可扩充数据类型的机制。

（7）查询范围应该能够覆盖单一的 XML 文档和一系列的 XML 文档。

（8）应使其能以多种方式使用，如在 URL 中使用，或以命令方式使用，或在程序中使用。以图形化界面方式进行使用。

与半结构化数据查询语言相类似，XML 查询语言通常也以下面这两种途径来进行研究[26][29]：

（1）以一种基于 XML 数据形式计算概念的语言为基础，将语法进行适当的变形，成为一种便于使用的查询语言，典型的如 XQL 语言[28]。

（2）以 SQL 或 OOL 为基础，添加必要的机制，使其能够表达一组有用的查询，典型的如 XML－QL 语言。

下面采用 XML－QL 的主要设计思想作概括介绍。

XML－QL[32]是由 AT&T 实验室为首的研究机构提出的 XML 查询语言，是在查询语言 UnQL 和 StruQL 基础上设计的。它能对 XML 文档进行查询、构造、转换和集成。XML－QL 集中了查询语言技术和 XML 语法格式，它通过说明路径表达式和模式的方式，给出 XML 数据的提取条件（where 子句）。同时，XML－QL 中可以给出构造查询输出的 XML 数据的模板，其输出结果仍为 XML 文档（Construct 子句）[30][31]。

XML－QL 有类似 Select－From－Where 的结构，与 SQL 很相似。但 XML－QL 有一些很重要的区别于基于结构化数据查询语言的特点。其 where 子句有两个部分组成——模式和条件表达式，这意味着被选出的数据项要满足两个条件：

（1）数据项的类型（或 Schema）和值必须与指定的模式匹配；

（2）数据项的值要满足条件表达式。

在查询条件中加入模式匹配是 XML－QL 与其它 Web 查询语言和结构化查询语言最大的不同之处。下面是 XML－QL 的标准结构。查询作者名为 Mary 的文章标题。

```
WHERE <article>                                    //模式
      <author>
          <name> Mary</name>
      </author>
      <title>$t </title>
```

</article>IN*CONFORMING TO book.dtd　　//条件表达式

CONSTRUCT <result>$t </result>　　//结果（XML 形式）

例子中的$t 所起的作用类似于程序设计语言中的变量一样，CONSTRUCT 是 XML－QL 中的保留关键字。返回结果为<result>XML</result>。

XML－QL 可以利用绑定变量、嵌套查询等特点实现关系代数中的选择、连接、投影、分组和排序等操作。

显然，由于 XML－QL 语言是以 SQL 或 OOL 为基础的，可以比较容易的将其转换为 SQL 语言，因为在本文中用关系数据库来存储 XML，所以将 XML－QL 作为查询语言，并在下一节中讨论如何将 XML－QL 语句转换为 SQL 语句。

5.2 路径表达式

XML 查询总可以转换为路径表达式，路径表达式可分为正则路径表达式和简单路径表达式。他们都是 XML 查询语言的基本成分。

1、正则路径表达式

一个正则路径表达式（PRS 表达式）定义如下[37]：

$$r = (r_1.r_2)\,|\,r^*\,|\,r+\,|\,r?\,|\,(r_1\,|\,r_2)\,|\,\#\,|\,name\,|\in$$

其中，$r_1.r_2$ 表示 r_1 到 r_2 的路径，r^*，$r+$，$r?$ 分别表示 r 的 0 或多、1 或多、和 0 或 1 次重复，$(r_1\,|\,r_2)$ 表示选择关系，# 表示任意的路径，*name* 表示结点标记，∈ 则代表空路径。表达式“＋（project.member）*.name”是正则（RPE）表达式。

2、简单路径表达式

与正则表达式对应，仅有节点标记构成（即不存在“*”、“＃”、“＋”、“？”、∈）的路径成为简单路径表达式（SPE）表达式。例如路径表达式“Labinformation.project.member.name”是 SPE 表达式。

若一个查询中包含正则路径表达式，则称该查询为正则路径查询（RPE 查询）。类似地，将仅包含简单路径表达式的查询称为简单路径查询（SPE 查询）。

5.3 将 RPE 转换为 SPE

在进行 XML 查询时一般的方法是把 RPE 查询重写为一组 SPE 查询，再将

SPE 查询改写成 SQL 查询。

RPE 重写为 SPE 的基本过程如下：

(1)消除查询中的表达式中出现的#。一般地，正则表达式 $e_1.\#.e_2$ 匹配从 e_1 开始，经一或多条边到达 e_2 的所有路径。文献[38]提出了一个求与从图中的一个源点到某个终点的子图等价的正则表达式的算法。采用文献[38]的算法，我们可以将表达式中的#部分转化为与之匹配的映射子图对应的正则表达式，从而消除#。

(2)扩展表达式中的*和+操作符。Kleene 表达式(p)*表示路径 p 的 0 到任意多次重复。由于*的无穷性，在模式层(即 DTD 图)无法将其重写为简单路径表达式，然而可以通过在模式层保存数据图中环的路径实例的统计信息来消除*操作符。类似地，我们可以消除+。

(3)消除?操作。一个表达式 p?可以简单地重写为 p|∈。

(4)消除|。该操作符可能出现于 RPE 查询的 FROM 或 WHERE 子句中。若表达式 s1|s2|…|sn 出现于查询 Q 的 FROM 子句中，则将查询 Q 重写为 n 个查询语句 Q1，Q2，…，Qn，在每个查询语句 Qi 中用 si 替换 s1|s2|…|sn，然后，用 Q1 UNION Q2 UNION…UNION Qn 代替语句 Q；若表达式 s1|s2|…|sn 出现于 WHERE 子句的某个谓词 p 中，则将谓词 p 重写为 n 个谓词 p1，p2，…，pn。在 pi 中用 Si 代替 s1|s2|…|sn，然后，在 WHERE 子句中用谓词 p1 AND p2 AND… pn 替换谓词 p 的析取。

利用上述方法，一个 RPE 查询可以转换为一个或多个 SPE 查询的 UNION 构成的查询。在每个 SPE 查询中的路径表达式都是 SPE 路径表达式。这些含有 SPE 路径表达式的查询需要进一步重写为 RDBMS 可实现的 SQL。

5.4 将 XML－QL 转换为 SQL

假设某条正则路径表达式转换后得到 n 个简单路径表达式 s1，s2，…，sn，假定该路径在 XML－关系映射下对应 k 个关系 r_1，r_2…，r_k，则将关系 r_1，r_2…，r_k 加入查询的 FROM 子句，关系间的关联条件加入 WHERE 子句。下面以 XML－QL 为例，说明如何将简单路径表达式转换为 SQL。

将 XML 数据存入关系数据库后，必须将 XML－QL 语句转换为 SQL 语句以便在关系数据库中根据 XML－QL 语句的语法规则和语义进行查询[32]。where

模块对应 SQL 中的条件语句，而 CONSTRUCT 模块对应 SQL 中的 select 部分。XML－QL 查询语句转换为 SQL 的规则是：先解析出 where 模块中的模式，利用模式中的元素间的层次关系与关系数据库中已有的元素关系表进行比较，倘若这个片段的层次关系在关系表中获得匹配，就可以根据需要（CONSTRUCT）模块获取相应的元素内容或属性的值。

以 5.1 节中的 XML－QL 语句为例进行说明。这个查询包括四个简单路径表达式即 article、X.author、X.articletitle、Y.name。其中 article 是根路径表达式，其它的是深层路径表达式。将查询转换为 SQL 通过如下方法：首先，识别出简单路径表达式对应的关系，将它们加入 SQL 的 FROM 从句中；其次，将深层路径表达式转换为关系中的连接，如果元素没有生成相应的表，而是在 FD－Inlining 算法中已经内联到其父元素生成的表中，这一步可以不执行。

转换后的 SQL 语句如下：

Select A.“article.title”

FROM article A，author B

Where A.articleid=B.articleid AND author.name＝“Mary”

其中在 Where 从句中包括一个关系条件，将 article 和 author 联系到一起；该查询在 article 关系中执行，返回作者名为“Mary”的文章名。

5.5 基于关系数据库的 XML 查询过程

以 XML 为基础的新一代 WWW 环境是直接面对 Web 数据的，不仅可以很好地兼容原有的 Web 应用，而且可以更好地实现 Web 中地信息共享与交换。XML 数据模型可以很容易地将 XML 的文档描述与关系数据库中的属性一一对应起来，实施精确的查询。

XML 已经成为正式的规范，开发人员能够用 XML 的格式标记和交换数据。XML 在三层架构上为数据处理提供了很好的方法。使用可升级的三层模型，XML 可以从存在的数据中产生出来，使用 XML 结构化的数据可以从商业规范和表现形式中分离出来。数据的集成、发送、处理和显示是下面过程中的每一个步骤[34][35]。

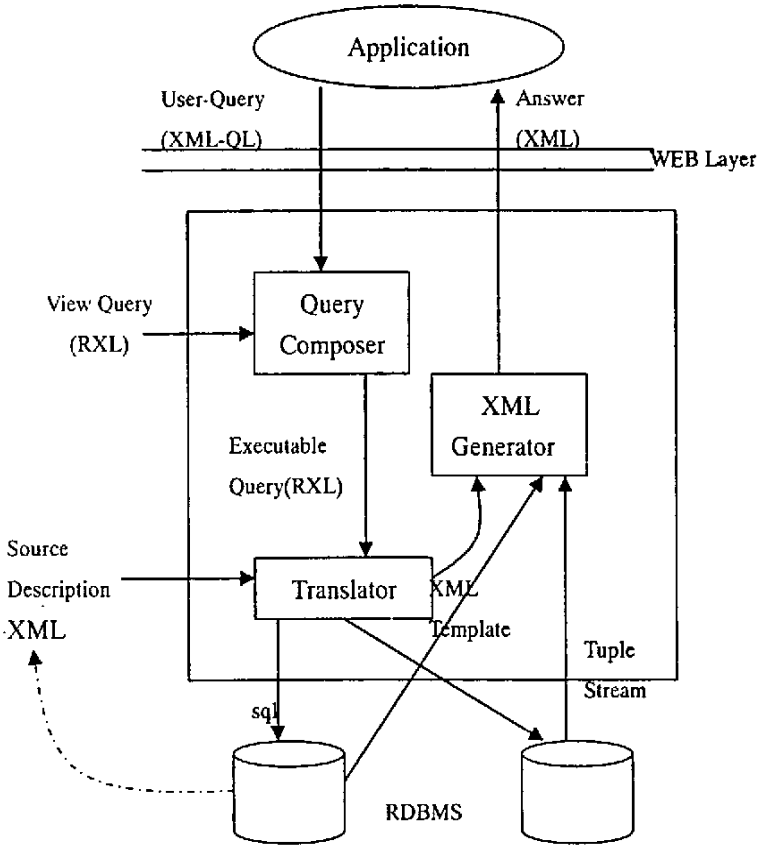

图 5.1 查询处理过程

在该查询处理过程中，用户通过 XML－QL 进行查询，XML－QL 是 XML 的查询语言。该处理过程的各个模块功能简单介绍如下：

（1）Query Composer。对于 XML－QL 请求在这里进行语法和语义的检查，判断是否符合 XML－QL 查询语言的语法，请求的 XML 数据是否存在。

（2）Executable Query。这个模块对 XML 文档库进行查询，如果 XML 文档库中没有找到符合条件的数据，该模块要把 XML－QL 查询请求传给 Translator 模块，并产生 XML 模板。如果在 XML 文档库中找到符合条件的 XML 文档，则直接返回给用户。

（3）Translator。该模块根据全局 XML 示图把 XML－QL 分解、翻译为针对局部处理数据源的查询请求。详细的处理参见 5.1 节。

（4）View query 和 source description。该模块占有很重要的地位，屏蔽异

构数据源异构性的工作主要在这里实现。在系统中用 XML 的 DTD 数据模式作为全局XML视图来描述各个异构数据源中的数据，并存储在全局XML示图中。全局 XML 示图屏蔽了异构数据源的异构性，呈现给用户统一的数据形式，这样用户就只需理解 XML 文档形式的数据，对 XML 中的数据进行访问。在全局 XML 示图中还要有数据源的物理存储空间，如数据库表在哪一个具体的数据库中，文本文件在哪一个数据源中。

（5）XML Generator 对于各个数据源的查询结果在这里进行合成，查询返回结果是从各个结点数据库上查询得到的，它们之间的数据类型是有差异的，因而需要转化为统一的格式（在这里，统一的格式是 XML 文档）。将各种格式的查询结果通过类型转换模块进行与 XML 之间的类型转换，并将查询结果存储在服务器上的 XML 文档中，然后由数据显示模块将其展现在客户端。

6 结论与展望

随着因特网技术的进步和用户规模的急剧膨胀，使得 HTML 逐渐不能适应用户的需要，在这种情况下 XML 应运而生。XML 已经成为 Web 上数据表示、交换和集成的标准，因此高效可靠的存储技术成为众多的研究人员所研究的方向，而关系数据库由于其成熟的技术和高性能，使得利用关系数据库来存储 XML 数据受到广泛的关注，成为当前 XML 数据存储的主流方向，并在各种应用中得到了广泛的使用。然而，XML 复杂的树形结构和简单平坦的关系数据模型有着很大的差异，给利用关系数据库存储 XML 技术带来了困难和挑战。虽然很多学者给出了一些关系数据库中存储 XML 文档的不同策略，主要关系数据库管理系统也都不同程度地支持 XML 存储管理，但仍存在很多限制。现有很多基于关系的 XML 存储很少考虑 XML 所蕴涵的语义约束，特别是函数依赖，针对这一问题本论文主要从如下几个方面进行了探讨：

1、分析了本领域研究现状，提出了本文的研究目标；

2、从 XML 的产生和发展入手，介绍了它的特性、模式语言、XML 文档类型及其相关规范，并对 XML 的使用前景进行了展望；

3、对存储 XML 文档的几种方式进行了比较，分析了 XML 在关系数据库中存储映射的经典策略；

4、提出了一种保持函数依赖的映射策略。首先依据 XML 函数依赖的定义，在文献[2]基础上提出了一种保持函数依赖的内联方法 FD－Inlining，然后分析了 DTD 中的其它语义约束，给出了保持部分语义约束的映射方法，最后对本映射策略进行了完整性分析；

5、研究了基于 XML 的查询，包括 XML 的查询语言，将 XML 的查询转换为基于关系的查询语言 SQL 的方法，以及查询结果的返回，并分析了基于关系数据库的 XML 查询过程。

受时间和本人水平所限，本文的研究还有很多空白和不完善的地方，个人认为在以下方面具有进一步进行研究的意义：

1、XML DTD 到关系模式的转换算法还有待在实践中继续验证和实现；

2、本映射算法没有考虑得到的关系模式的规范化；

3、转换模式是基于 DTD 的，鉴于目前的 XML Schema 正在日益成为新的

标准，应该进一步的探讨是否可以将本方法应用于从 XML Schema 到关系数据库的模式映射。

参考文献

[1] Lee ML, Ling TW, Low WL. Designing functional dependencies for XML. In: Jensen CS, et al., eds. Advances in Database technology—EDBT 2002, 8th International Conference on Extending Database Technology. Lecture Notes in computer Science 2287, Prague, Czech Republic: Springer-Verlag. 2002. 124~141.

[2] Abdel-aziz A A, Oakasha H. Mapping XML DTDs in Relational Schemas. Computer Systems and Applications, the 3rd ACS/IEEE International conference, 2005.

[3] Chen Y, Davidson S, Zheng Y. Constraint preserving XML storage in relations. Technical Report, MS-CIS-02-04, University of Pennsylvania, 2002.

[4] 王庆,周俊梅,吴红伟.XML 文档及其函数依赖到关系的映射.软件学报,2003,14(07):1275~1281.

[5] 何挺盈，王珊．从DTD映射到关系模式：一种保持数据依赖的映射方法．计算机研究与发展，2004，41（05）：868~873.

[6] Daniela Florescu, Donald Kossman:Storing and Querying XML Data using a DBMS.IEEE Data Engineering Bulletin 22(3):27-34(1999).

[7] J.Shanmugasundaram, K.Tufte, G.He et al.Relational databases for querying XML documents: Limitations and opportunities. In Proceedings of the International Conference on Very Large Data Bases(VLDB), 1999.302-314.

[8] A.Deutsch, M.Fermandez, D.Suciu.Storing semi-structured data with STORED.In Proceedings of the ACM SIGMOD International Conference on Management of Data, 1999.

[9] D.Florescu, D.Kossmann.A performance evaluation of alternative mapping schemas for storing XML in a relational database.Technical Report 3680, INRIA,1999.

[10] Yi Chen, Susan B.Davidson Yifeng Zheng.Constraint Preserving XML Storage in Relations. University of California at Los Angeles 1999.

[11] Natanya Pitts 著.XML 技术内幕.徐晓梅，龚志翔，王晓云译.北京：机械工业出版社，2002.

[12] 万常选.XML 数据库技术.北京：清华大学出版社，2005.

[13] R Bourret.Mapping DTDs to Databases[EB/OL].http://www.xml.com/pub/a/2001/05/09/dtdtodbs.html.Web page,2001-05.

[14] 萨师煊，王珊.数据库系统概论.2000 年 2 月第 3 版.北京：高等教育出版社，2000.

[15] 罗骏，何羽.SQL 实用简明教程.北京：清华大学出版社，2004.

[16] 刘云生，钟昊，陈明俊.XML DTD 到关系模式的映射研究.计算机工程与科学.2004，26(6)：73-76.

[17] 许卓明，刘琴，董逸生.基于关系数据库的 XML 存储技术评述.计算机工程与应用.2003，39（10），197-200.

[18] 吕腾，闫萍，王真星.XML 的函数依赖.小型微型计算机系统.2005，26（5），864-868.

[19] 曾宇昆，王清明，杨卫东，施伯乐.XML 模式到关系范式的映射.计算机工程.2005，31（8），37-39.

[20] 章义，黎峰.基于 XML 的数据库储存访问技术.计算机工程与设计.2005，26（1），208－211.

[21] 吴敏，徐德智.XML 数据的存储实现研究.计算机工程.2003，29（15），25－26.

[22] Shanmugasundaram J，Shekita E J，Barr R,et al.Efficiently Publishing Relational Data sa XML Documents[C].Proc .of the 26th VLDB Int.Conf.Cairo，Egypt，San Francisco：Morgan Kaufmann Publishers，2000.65-76.

[23] T.Bray，J.Paoli，C.Sperberg-McQueen，and E.Maler.Extensible Markup Language (XML) 1.0，October 2000. http://www.w3.org/TR/REC-XML

[24] Michael Classen. XML-the better HTML，Exploring XML.1999.12

[25] W3C，XSL Transformations（XSLT）Version 1.0 W3C Recommendation 16 November 1999，http://www.w3.org/TR/xslt

[26] R.Goldman，J.McHugh，and J.Widom.From Semistructured Data to XML：Migrating the Lore Data Model and Query Languages，1999.

[27] R.Bourret，XML and databases，http://www.rpbourret.com/xml/XMLAndDatabases.html. Nov，2000.

[28] J.Robie，J.Lapp，and D.Schach.XMLQuery Language（XQL）.1998. http：//www.w3.org/TandS/QL/QL98/pp/xql.html.

[29] Scholkopf B.Smola A. A tutorial on support vector regression.NeuroCOLT2 Technical Report Series 2NC－TR－1998－030，October，1998.

[30] Fernandez M，Suciu D，et al. A Query Language for a web－site Management System.

[31] Alin Deutsch，Mary Fernandez，Daniela Florescu，Alon levy Dan Suciu. XML-QL：A Query

致 谢

首先非常感谢我的导师杨春教授，杨老师无论是在学业上，还是在生活中，都给我很多指点和帮助。本论文从研究方向到选题立论，从资料收集到论文的最后完成，都得到了杨老师的关怀和帮助。杨老师治学严谨，对事业执著追求，在科研方面精益求精，对学生严格要求，认真负责，令我受益匪浅，终身难忘。他对我学习和实践工作进行的悉心指导，给予的无限帮助和鼓励使这篇毕业论文的完成成为可能。在此，谨向杨老师致以最诚挚的谢意。

十分感谢谭良博士，对我的理论和项目实践的指导，特别是对我发表的论文的指导。感谢计算机科学学院的全体老师及研究生处的老师。

感谢实验室的全体同学，特别是吴微、李黎同学对我的帮助与支持，感谢几年来与我朝夕相处的同学，他们在学习和生活上都给予我非常大的帮助。感谢室友—刘兰、伍希、李佩玲让我在他乡度过了愉快的时光。

感谢我的父母、兄姐等家人多年来一直支持我求学，关心我的生活。也要感谢男友梁书恩，他对我的关爱让我觉得很幸福。他们的关心和爱护是我永远的动力。

深深感谢、祝福所有曾给予我帮助和关心的人。

攻读硕士学位期间发表学术论文情况和取得的科研成果

发表论文：

[1] 于光，杨春. OGSA 及在具体宿主环境中其服务的实例化.计算机工程与设计. 已录用.

[2] 于光，杨春，谭良. 一种 DTD 到关系模式的映射方法.计算机与数字工程. 已录用.