第33卷 第21期 计 算 机 工 程 2007年11月
Vol.33 No.21 Computer Engineering November 2007
·软件技术与数据库· 文章编号：1000—3428(2007)21—0063—03 文献标识码：A 中图分类号：TP311.52

XML查询方法的设计与研究

沈剑沧[1]，鲍培明[2]
（1. 南京大学物理学系，南京 210093；2. 南京师范大学计算机科学系，南京 210097）

摘 要：提出了用Petri网建立XML语义结构模型的设计方法，根据XML的DTD结构建立描述XML语义结构的Petri网模型，DTD结构中的元素声明和属性声明对应Petri网模型中的一个或一组变迁，声明中的元素或属性对应Petri网模型中的库所，把XML查询中的路径表达式定义为Petri网库所中的Token。根据Petri网模型的结构生成存储XML数据的关系数据库模型，将XML的查询问题最终转化为数据库中数据的查询操作。

关键词：XML；Petri网；路径表达式；查询

Design and Research of XML Query Method

SHEN Jian-cang[1], BAO Pei-ming[2]
（1. Dept. of Physics, Nanjing University, Nanjing 210093; 2. Dept. of Computer Science, Nanjing Normal University, Nanjing 210097）

【Abstract】This paper proposes a method for modeling XML semantic structure with a Petri net. According to the DTD structure of XML, it constructs the Petri net model to describe the XML semantic structure, in which DTD element declaration and attribute declaration are described as some transitions, elements and attributes are described as places, and the path expressions in XML query are described as Tokens in Petri net model. Based on the structure of Petri net model, relational database model is set up to save XML data, and the query of XML is converted to the query of database.

【Key words】XML; Petri net; path expression; query

XML是W3C的XML工作组为适应Internet的发展，实现快速的电子商务和电子数据交换而推出的新型Web语言。工作组是这样描述该语言的[1]：XML是SGML的子集，其目标是允许普通的SGML在Web上以HTML目前的方式被服务、接收和处理。XML易于实现，且可在SGML和HTML之间互相操作。

由于XML提供了一种灵活的文档结构来描述数据，因此在网络上得到了迅速的普及，成为Internet以及各种信息集成中的数据交换格式。XML结构的灵活性是它与关系数据库最大的差别，因此，被普遍认为是一种典型的半结构 数据[2~3]。

1 现状以及思路

目前，XML数据的查询方法一般基于树结构和关系数据库结构。把XML描述成一棵树，虽然路径表达式很清晰，但这棵树的数据量很大，树中的结点通过边连接，边缺乏主动驱动和并发能力，这就造成树结构的路径表达式查询代价大。把XML直接存放在关系数据库中，虽然关系表的结构很规范，能提高查询速度，但可能降低XML本身的结构灵活性。

本文的思路在于3个方面：(1)XML DTD描述了XML数据应遵守的路径结构，它的信息量通常比XML本身小得多。因此，利用DTD建立XML语义路径模型，描述XML语义结构。(2)采用Petri网建模XML语义结构模型。在Petri网模型中，库所与库所之间通过变迁连接，利用变迁的自点燃能力和并发能力提高路径表达式的查询效率。(3)根据Petri网模型的结构生成关系数据库模型，将XML的查询问题最终转化为对数据库中数据的查询操作。

2 Petri网模型定义

2.1 XML的DTD结构

DTD是一套关于XML文档中标记符的语法规则。DTD主要由元素声明、属性声明、记法声明和实体声明组成。元素声明由元素内容模型来描述，该模型中的内容采取一组正则路径表达式（RPE）的形式。RPE中出现的元字符有“*”、“+”、“？”、“，”、“|”、“()”和“#PCDATA”，其中，决定XML语义路径构成的元字符是“，”、“|”和“()”。典型的XML图书列表的DTD示例如下：

```
<?xml version="1.0" encoding="GB2312" standalone="yes"?>
<!DOCTYPE 图书列表 [
  <!ELEMENT 图书列表 （书*）>
    <!ELEMENT 书 （作者*, 书名, 出版社, 价格, 评注*）>
      <!ATTLIST 书 书号 ID #REQUIRED>
<!ELEMENT 评注（作者, 身份？,（(标题, 内容）*|简评））>
      <!ELEMENT 作者 （#PCDATA）>
      <!ELEMENT 书名（#PCDATA）>
      <!ELEMENT 出版社 （#PCDATA）>
<!ELEMENT 价格 （#PCDATA）>
    <!ELEMENT 身份 （#PCDATA）>
      <!ELEMENT 标题 （#PCDATA）>
      <!ELEMENT 内容 （#PCDATA）>
      <!ELEMENT 简评 （#PCDATA）> ]>
```

基金项目：江苏省教育厅高校自然科学研究基金资助项目(04KJB520075)

作者简介：沈剑沧(1963－)，男，硕士、工程师，主研方向：XML数据管理，计算机网络，信息检索；鲍培明，硕士、副教授

收稿日期：2006-11-30 **E-mail：**jcshen@nju.edu.cn

2.2 XML 语义结构的 Petri 网模型

2.2.1 Petri网基本概念[4~5]

定义 1 三元组 $N=(P, T; F)$是一个网，当且仅当：

(1)$P\cap T=\Phi$，$P\cup T\neq\Phi$；

(2)$F\subseteq(P\times T)\cup(T\times P)$；

(3)$dom(F)\cup cod(F)=P\cup T$，其中，$dom(F)=\{t \mid p\in P\cup T, (t,p)\in F\}$；$cod(F)=\{p \mid t\in P\cup T, (p, t)\in F\}$。

它们分别为 F 的定义域和值域。

定义 2 四元组$\sum=(P, T; F, Mo)$是一个Petri网，当且仅当三元组$(P, T; F)$是一个有向网，$Mo: P\to N^{+}\cup\{0\}$，其中，N^{+}为正整数集；Mo称为初始Token。

如图 1 所示，其中，Mo 用黑点表示；F 用有向弧表示；P 用圆圈表示；T 用短划线表示。

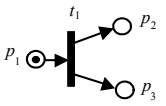

图 1 一个简单的 Petri 网

2.2.2 描述 XML 语义结构的 Petri 网模型

Petri 网是一种具有数学描述能力、又有图形直观表示的模型工具，它是信息处理系统描述和研究的一个强有力的工具。用于描述 XML 语义结构的 Petri 网定义如下：

定义 3 Petri 网是一个多元组：

$< E, P, T, M(p), I(t), O(t), f(p), B(p)>$

其中，

$e\in E$，$p\in P$，$t\in T$；

E 是 DTD 中元素和属性名称的有限集合；

P 是库所的有限集合，库所对应 DTD 中的元素或属性；

T 是变迁的有限集合，变迁对应 DTD 中的元素声明中 RPE 的“ ( ) ”操作符、“ | ”操作符或属性声明；

$M(p)$是 p 的一个标记值，反映该库所在当前操作中拥有的 Token 情况；

$I(t)$：$T\to P^{\infty}$是t的输入函数，反映变迁到库所的映射；

$O(t)$：$T\to P^{\infty}$是t的输出函数，反映变迁到库所的映射；

$f(p)$：$P\to E$ 是一个映射，反映库所对应 DTD 中元素或属性的名称；

$B(p)$：$P\to\Sigma^{*}$是一个映射，Σ是字符集合，Σ^{*}是Σ的闭包，用于指向库所对应的XML元素或属性的数据所在的关系表。

2.1 节的 XML 图书列表的 DTD 示例用 Petri 网建模，得到图 2 所示的 Petri 网模型。

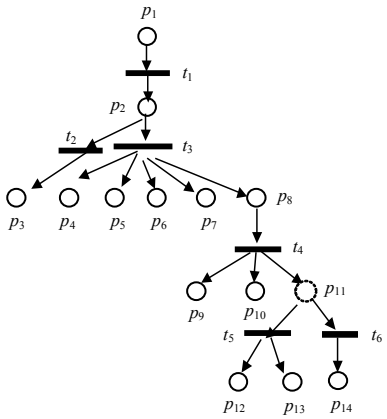

图 2 图书列表的 Petri 网模型

其中，p_{11}(图中用虚线圆圈绘制)定义为一个虚库所，表示为$f(p_{11})=\tau$，理解为一个复杂的元素声明语句：

<!ELEMENT 评注（作者，身份？,（(标题，内容）*|简评））>

将其分解为 2 条简单的元素声明语句：

<!ELEMENT 评注 （作者，身份？,τ）>

<!ELEMENT τ （(标题，内容）*|简评）>

对应于元素τ的库所定义为虚库所，τ不构成XML语义路径的成分。在Petri网中，可以把p_{11}看作p_8状态的持续。因此，在建立XML语义路径结构的Petri网模型时，利用虚库所，可以将任何复杂的元素声明简化为若干条简单的元素声明。

3 生成关系数据库模型

2.1 节的 DTD 示例中，一些元素后面带有通配符，一些元素之间存在或选关系；而图 2 所示的 Petri 网模型中定义了虚库所，因此，在进行关系数据库模型的设计时，共建 5 张数据表，关系模式分别为

(1)书表(ID，书号，书名，出版社，价格)；

(2)作者表(ID, PID，作者)；

(3)评注作者表(ID, PID，作者，身份)；

(4)标题内容表(ID, PID，标题，内容)；

(5)简评表(ID, PID，简评)。

除了作者表的 PID 字段（外键）与其同一级的书表的 ID 字段（主键）相关联以外，其他各张数据表的 PID 字段与其上一级的数据表的 ID 字段相关联。具体的多表关联方式为

(1)书表.ID=作者表.PID；

(2)书表.ID=评注作者表.PID；

(3)评注作者表.ID=标题内容表.PID；

(4)评注作者表.ID=简评表.PID。

4 路径表达式的查询方法

在 XML 数据处理中，把用户处理请求中的路径表达式以及查询条件作用在 Petri 网模型上；再根据 Petri 网模型与关系数据库结构的联系，生成访问数据库的 SQL 语句，执行数据库操作；操作结果通过 Petri 网模型重新构成结果的 XML 文档。路径表达式查询方法的具体实现步骤如下：

(1)把XML查询中的路径表达式定义为Petri网库所中的Token，而Token定义为一个二元组(x,y)，p的Token值为$M(p)=\{(x,y)\}$。x是查询中的路径表达式：$x=e_{i1}/e_{i2}/\cdots/e_{in}$，其中，$e_{ij}\in E$，$j=1,2,\cdots,n$。$y$是一个字符串，反映$x$在查询中的用途，当$y=\Phi$时，表明$x$所在的数据是查询结果；当$y\neq\Phi$时，表明$y$是查询条件，查询结果是使$x$符合$y$的数据。

(2)若存在t的输入库所p_i，Token值为

$M(p_i)=(e_{i1}/e_{i2}/\cdots/e_{in}, y)$

Token中x的第 2 项e_{i2}与t的至少一个输出库所p_j相关，则t是使能的并且可以点燃，而$M(p_i)$称为t的使能Token。

(3)当t点燃时，移去输入库所p_i中的使能Token：

$M(p_i)= M(p_i)-\{(e_{i1}/e_{i2}/\cdots/e_{in}, y)\}$

对t的输出库所p_j添加新Token：

$M(p_j)=M(p_j)\cup\{(e_{i2}/ e_{i3}/\cdots/e_{in}, y)\}$

对于 t 的点燃，不是每一个 t 的输出库所都会添加新 Token，只是与使能 Token 中的路径表达式相关的库所才会添加新 Token。

(4)当没有变迁可点燃时，将库所中的Token转化为数据库中的SQL语句，并优化SQL语句。$\forall p_i\in P$，$\forall(x, y)\in M(p_i)$：

1)$f(p_i)$加入SQL语句的select 项，若$y\neq\Phi$，则$f(p_i)$和$\&y$加入SQL语句的where项；

2)B(p_i)加入SQL语句的from项。

其中，“ & ”是一个替换符；&y 是将 y 的字符串内容替换&y 所在位置。在 from 项中，存在不同的数据表，牵涉到多表查询时，多表间的关联在定义表的关系模式时已经确定，并将连接条件加入到 where 项中。

(5)在数据库上执行 SQL 语句，生成数据库形式的查询结果，再结合 Petri 网的语义结构，最终生成查询结果的 X*M*L 文档。

如图 2 示，若需要查询价格大于 90 元的书的价格及其书名，可以置$M_0(p_1)$={(图书列表/书/价格, Φ), (图书列表/书/书名,Φ), (图书列表/书/价格, “>90”)}；t_1点燃，置$M_1(p_1)$={ }，$M_1(p_2)$={(书/价格,Φ), (书/书名, Φ), (书/价格, “>90”)}；t_3点燃，置$M_2(p_2)$={ }，$M_2(p_5)$={(书名, Φ)}，$M_2(p_7)$={(价格,Φ), (价格, “>90”)}，$B(p_5)$= B(p_7)；没有变迁可点燃，根据库所中的Token情况，生成访问数据库的SQL语句：

select 书名, 价格 from &B(p_5) where 价格>90

5 系统设计方案以及实现技术

根据上文的 XML 图书列表示例，设计了一套基于 Petri 网的 XML 图书查询系统。最终，给用户提供一个应用软件，用户只要输入查询的路径表达式，就可以对基于 Petri 网的 XML 图书查询系统进行快速的数据查询并自动输出查询结果的 XML 文档。Petri 网模型和数据表在内部处理时使用，对用户来说是不透明的，面对用户的只是 XML 文档的 DTD 结构、XML 文档本身以及查询结果的 XML 文档。

使用 VB.NET 对用户使用界面和 Petri 网模型的结构进行程序设计，并将 XML 的数据导入根据 Petri 网模型结构事先设计好的 Access2003 中。在 VB.NET 中，通过 ADO.NET 进行数据访问。

6 系统分析以及实验结果

6.1 系统分析

Petri 网规模不受 XML 中的数据量及数据量变化的影响，因此，Petri 网稳定且规模小。若 XML 文档共包含 m 个不同的元素名和属性名，则 Petri 网的空间复杂度只有 $O(m)$。

Petri 网不仅具有良好的层次结构，而且层次深度也很小。若层次深度为 n，查询处理花费在 Petri 网上的时间复杂度为 $O(n)$。Petri 网的变迁点燃具有并发操作能力，并且只发生在与查询路径相关的变迁与库所上，因此，效率比树结构上所有结点依次搜索高得多。

6.2 实验结果

对 XML 图书查询系统使用基于 Petri 网的查询方法和使用基于树结构的遍历查询方法，分别进行查询。如图 3 所示，查询条件都选取：图书列表/书/价格, “>90.00”，而 Q1~Q8 是 8 个不同查询路径。

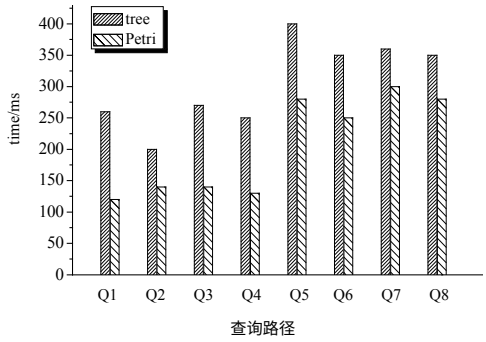

图 3 15 000 本 XML 图书的查询时间比较

7 结束语

在 XML 图书查询系统中，当书的数目大于一定数目时，不管是对简单路径表达式进行查询，还是对复杂路径表达式或者多个路径表达式进行查询，基于 Petri 网的查询时间总是快于基于树结构的遍历查询时间。由此得出结论：Petri 网实现了 XML 路径表达式的准确定位，并且有效提高了路径表达式的查询效率。

参考文献

1 World Wide Web Consortium. Extensible Markup Language(XML) 1.0(3rd ed.)[DB/OL]. (2004-02-04). Http://www.w3.org/TR/REC-xml/T.

2 Magalhaes K V, Alberto H F L, Silva A S. Storing Semistructured Data in Relational Databases[C]//Proceedings of the 8th International Symposium on String Processing and Information Retrieval. 2001.

3 Abiteboul S. Semistructured Data: From Practice to Theory[C]//Proc. of the 16th Annual IEEE Symposium on Logic in Computer Science, Boston, MA, USA. 2001-06.

4 袁崇义. Petri 网原理与应用[M]. 北京: 电子工业出版社, 2005.

5 蒋昌俊. Petri 网的行为理论及其应用[M]. 北京: 高等教育出版社, 2003.

（上接第 62 页）

这里只是举例说明如何打包，实际上把所有文件打包在一起是不可取的，这样即使只升级一个文件也要重新发布整个包，且无法只安装软件的一部分，实际工程中应将其细分成多个更基本的软件包并建立适当的依赖关系。例如 opie 项目把 Qt 系统的每一个应用程序、每一种语言库、每一种字体都单独打包，体现了 ipkg 软件包细分的特点。

5 结束语

综上所述，ipkg 作为一种小巧便利的包管理程序，非常适用于有一定软件规模且存储空间较小的嵌入式 Linux 系统，如掌上电脑、信息终端机、工控设备等。笔者所开发的信息终端机采用 S3C2410 / 64MB RAM / 64MB NAND flash 硬件平台，集 Web 服务器、网络管理、人机界面于一体，是一个适用 ipkg 的典型系统。随着开源社区的持续努力，ipkg 体系将会像 Debian Linux 一样得到长足发展，不久之后，在各种平台的嵌入式设备上开发、安装和使用 Linux 软件都会像 PC 一样容易了。

参考文献

1 ipkg Wiki[Z]. [2006-10]. http://handhelds.org/moin/moin.cgi/Ipkg.

2 Yaghmour K. Building Embedded Linux Systems[M]. USA: O'Reilly, 2003.

3 Lutz M. Programming Python[M]. 2nd ed. USA: O'Reilly, 2001.

4 Qtopia Online Reference Documentation[Z]. [2006-10]. http://doc.trolltech.com.