第33卷 第16期 计 算 机 工 程 2007年8月
Vol.33 No.16 Computer Engineering August 2007
·软件技术与数据库· 文章编号：1000—3428(2007)16—00—0 文献标识码：A 中图分类号：TP301.6

基于DTD的XML与SQL查询转换算法

卜 莉，李军怀，张 璟
（西安理工大学计算机科学与工程学院，西安 710048）

摘 要：针对如何将XML查询转换成SQL查询提出了一个XSLT查询到SQL查询的转换框架和算法，研究了文档类型定义(DTD)和关系模式的相互映射方法及基于XML DTD且不使用任何中间语言将XSLT查询转换为SQL查询的具体过程和算法：从XSLT代码中抽取指令集合，合并、简化并分割其中的XPath，最后抽取出SQL语句的各个组成部分。该算法具有较高的通用性。

关键词：文档类型定义；XSL转换；模式映射；SQL；查询转换

Translation Algorithm of XML and SQL Query Based on DTD

BU Li, LI Jun-huai, ZHANG Jing
(School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048)

【Abstract】 In order to translate XML queries to SQL queries, this paper proposes a translation framework and algorithm from XSLT queries to SQL queries. It studies how to map DTD to relation schema and its reverse. The key is the process and algorithm of translating XSLT queries to SQL queries which is based on XML DTD and without any intermediate language. The algorithm extracts statement set from XSLT program, and then merges and simplifies XPath in statement. After splitting XPath, each part of SQL statement is extracted. And the algorithm has high generality.

【Key words】 DTD; XSLT; schema mapping; SQL; query translation

1 概述

近年来，XML已经成为Internet上发布与交换数据的事实标准。但是由于要对XML进行解析和文本转换，因此数据访问速度较慢。同时，单纯的XML数据管理技术无法满足商业应用中高效的存储组织、数据的完整性和一致性、并发控制等更高的技术要求。因此，在实际应用中往往使用关系数据库系统作为商业数据的存储介质，而将XML作为数据交换和发布的介质以结合二者的优势，即RDBMS中的数据被封装成XML文档呈现给用户，用户使用XML查询语言（XSLT、XPath、XQuery、XML-QL等）查询感兴趣的数据，而事实上数据储存在RDBMS中。但要实现用户的查询，就必须将XML查询转换成SQL查询。

XML查询到SQL查询的转换技术主要有2类方法：(1)开发一个查询转换引擎以支持应用系统中XML查询到SQL的查询转换；(2)在现有的关系数据库管理系统中增加对XML查询的支持。

文献[1]提出了第1类转换方法，并用实例说明将关系数据库的数据发布为XML文档后，如何将XSLT查询转换成SQL查询，但没有给出相应的查询转换算法，并且缺乏普遍性。文献[2]采用XML-QL作为XML查询语言，定义了一种名为RXL的中间语言。XML-QL查询首先被转换为RXL然后再将RXL转换成SQL查询。文献[3]在XML文档和关系数据库之间建立通用虚拟关系模式GVRS，先将XQuery查询转换成对GVRS的查询，然后进一步转换为对RDB的SQL查询。文献[4]采用T-expression重写技术，但未给出XSLT到SQL的转换算法。

在第2类方法中，现有的商业数据库厂商诸如Oracle和Sybase，已经开发出支持XML存储和简单XPath查询的版本。然而，对XPath查询的支持有限且用户必须了解关系模式。即查询不是基于DTD而是基于关系模式的。任意复杂的基于DTD的XPath表达式并不被这些产品所支持。

2 XML与SQL查询转换的实现框架

查询转换首先通过模式映射将DTD映射为关系模式或者将关系模式映射为DTD，然后基于DTD将用户的XSLT查询翻译成SQL查询。本文选择XSLT作为示例XML查询语言，它的表达能力强于其他查询语言，如果本算法适用于XSLT，则同样能适用于其他XML查询语言。具体实现框架见图1。

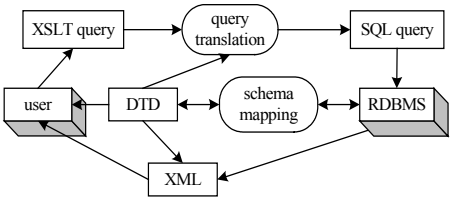

图1 XML与SQL查询转换实现框架

图1中查询转换的前提是DTD，DTD和RDBMS通过schema mapping实现相互映射。user提出的XSLT query通过query translation转换成SQL query，RDBMS执行SQL query交给并将结果封装成XML文档呈现给user。user无须了解商业数据实际存储位置和存储结构，只须根据DTD提出XSLT查询即可得到XML文档，从而实现了user和RDBMS之间的透明访问。很多文章讨论过如何将RDBMS中的数据封装

基金项目：国家“863”计划基金资助项目(2002AA414060)；陕西省自然科学基金资助项目(2005F05)
作者简介：卜 莉(1978－)，女，硕士研究生，主研方向：Web应用；李军怀，副教授；张 璟，教授、博士生导师
收稿日期：2006-11-10 **E-mail：**lilian.bu@hotmail.com

为 XML 文档，本文主要研究 XSLT 查询到 SQL 查询的转换，即图 1 中的 schema mapping 和 query translation。

3 DTD 和关系模式的相互映射

XML DTD 和关系模式分别是描述 XML 和关系数据库数据结构及数据约束关系的方法。而 XSLT 与 XML 的关系如同 SQL 和关系数据库的关系。显然，XSLT 和 SQL 分别依赖于 DTD 和关系模式。因此，查询转换的前提是模式映射。

(1)将 DTD 映射为关系模式。将 DTD 中可多次出现的节点（带*或+的节点，以下称为复合节点 n）映射为一个关系 r，将 n 的简单孩子节点（只能出现一次的节点）映射为 r 的属性，如果 n 包含值，则在 r 中增加 Value 属性；如果 n 没有 ID 属性，在关系 r 中添加 ID 字段作为 r 的唯一标识。对于 n 的每个父亲节点 p，如果 p 不是根节点，则在 r 中加入字段 pID 储存外键 k，k 的值为 p 的 ID 属性。

(2)将关系模式映射为 DTD 则较为简单，首先建一个只包含根节点的 DTD，然后将每个关系 r 添加为根节点的复合右孩子 n，r 的属性则被映射为 n 的简单孩子。

图 2 是一个示例 DTD。由于关系数据库无法表示递归，因此这里的 DTD 没有递归。示例 1 是满足图 2 DTD 的 XML 文档，示例 2 是一段 XSLT 示例，将 schema mapping 应用于图 2 中的 DTD 可以得到表 1，为使表达更加直观，表 1 中加入了示例 1 中的数据。

示例 1

```
<Company>
<Product ID="p1"></Product>
<Product ID="p2"></Product>
<Product ID="p3"></Product>
<Sell ID="s1" Date="2006-01-20">
  <Goods ID="p1" Amount="10"/>
  <Goods ID="p3" Amount="20"/>
  <Sum>660</Sum>
</Sell>
<Sell ID="s2" Date="2006-04-11">
  <Goods ID="p2" Amount="2"/>
  <Sum>61</Sum>
</Sell>
<Buy ID="b1" Date="2006-03-23">
  <Goods ID="p1" Amount="50"/>
  <Sum>1150</Sum>
</Buy>
</Company>
```

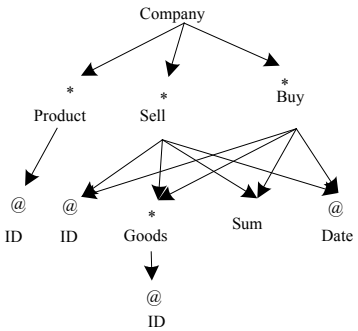

图 2　示例 DTD

示例 2

```
1：<xsl:variable name="sel" select="//*[Sum]"/>
2：<xsl:template match="Product">
3：  <xsl:variable name="pro" select="."/>
4：  <xsl:for-each select="$sel[Goods/@ID=$pro/@ID]">
5：    <xsl:sort select="@Date"/>
6：    <xsl:copy-of select="Sum"/>
7：  </xsl:for-each>
8：</xsl:template>
```

表 1　图 2 DTD 的关系模式

rID	Node	ID	Attribute	
r01	Product	ID		
		p1		
		p2		
		p3		
r02	Sell	ID	Sum	Date
		s1	660	2006-01-20
		s2	61	2006-04-11
r03	Buy	ID	Sum	Date
		b1	1150	2006-03-23
r04	Goods	ID	pID	
		p1	s1	
		p3	s1	
		p2	s2	
		p1	b1	

4 查询转换

XSLT 查询由一组模板规则组成，规则可以简单地描述为：如果在输入中遇到此条件，则生成下列输出。模板是文本结果元素和 XSLT 指令的集合。XSLT 指令描述如何转换源 XML 文档，其使用 XPath 引用输入树中的节点，可以按任何方向浏览树以选择节点，并根据节点的值和位置应用谓词。它还包括用于基本字符串处理、数字计算和布尔代数的工具。

查询转换的目的是使 XSLT 查询和 SQL 查询产生等价的输出，即根据 XSLT 指令类型将 XPath 路径表达式对节点的匹配转换成对 RDBMS 中表的操作。查询转换的细化过程如图 3 所示。

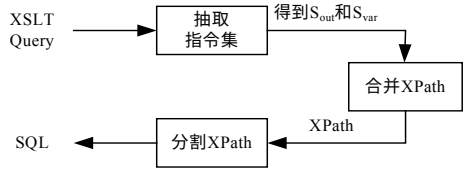

图 3　查询转换细化过程

4.1 输出指令集合和变量指令集合的抽取

在 XSLT 指令中，可能产生输出的指令包括 xsl:value-of, xsl:copy-of, xsl:variable, xsl:apply-template, xsl:call-templates。其中，xsl:apply-template 是特殊的输出指令，当有其他匹配模板时，它输出满足其他匹配模板匹配规则的节点值；反之，输出与自身所在模板规则匹配的节点值；xsl:call-template 是调用另外一个模板进行匹配。本文暂不考虑后 2 个指令，它们可以转换为其他几个输出指令；xsl:variable 指令的输出是节点集，它的结果并不直接体现在 XML 中，因此，对其进行单独处理。

定义 1 输出指令集合S_{out}

$S_{out}=\{s_1, s_2,\ldots,s_n\}$，其中，$s_n$是xsl:value-of或xsl:copy-of，s_1是s_n所在的模板，其余为在s_1和s_n之间的指令顺序序列并且s_i嵌套在s_{i-1}之中（$2\le i\le n$）。

定义 2 变量指令集合S_{var}

同上可得$S_{var}=\{s_1, s_2,\ldots,s_n\}$，其中，$s_n$是xsl:variable指令，它可以不包含在模板内。

4.2 Xpath 的合并

S_{out}和S_{var}中的指令一起作用于XML文档才能输出最终结

果。得到S_{out}和S_{var}后，根据指令类型将XPath合并。

算法 1 合并 XPath，得到无分支的 XPath

UnionXPath(指令集合 S){
path=“”
For each s_i∈S
If(s_i是xsl:sort指令&&s_n不是xsl:variable指令)
p=“[ORDER: ”+ s_i的XPath+“]”
Else If(s_i是xsl:if指令或者xsl:when指令)
p=“[”+s_i的XPath+“]”
Else If(s_i是xsl:for-each-group指令)
p=s_i的select中的XPath+“[GROUP:”+s_i group中的XPath+“]”
Else p=s_i的XPath
If (p≠“.”)
If(p 的首字符是“[”) path+=p
Else path+=“/”+p
Return path}

如果合并后的XPath中有“ // ”和“ * ”，则会导致多个分支路径。设P含有*k*个分支，首先将P简化成*k*条不重复的路径$\{p_1,\ldots,p_k\}$，简化的方法参考文献[5]。

4.3 Xpath 的分割

无分支的XPath的一般形式为$P_1[C_1]/P_2[C_2]/\ldots/P_n$，其中，$P_i$表示路径节点；$C_i$为过滤条件，$C_i$也是形如XPath的一般形式。首先将XPath用最外面的“ / ”分割为*n*部分，每一部分形如$P_i[C_i]$。根据P_i是复合节点还是简单节点会分别转换成SQL中的表和列；接下来处理C_i，若C_i中包含“ / ”，则对C_i重复以上过程中；如果不包含“ / ”，则C_i是过滤条件对应WHERE子句。以上步骤重复直至整个XPath被处理完。

算法 2 分割 XPath，得到 S{}, F{}, W{}, G{}, O{}

SplitXPath(XPath 路径 path，当前节点 Node){
用path中最外层的“/”分割path得到$\{p_1,\ldots,p_n\}$
For each p_i∈path
If(p_i是最后一部分)
令N=p_{i-1}中的路径节点，将Node.p_i中的@去掉后插入S
Else If(p_i含有比较运算符)
令L=p_i运算符左边，R=p_i运算符右边，将p_i中的复合节点插入F；去掉p_i中的过滤条件和@，将“/”用“.”替换，将p_i中的“复合父亲.”去掉后插入W
If(L 中有过滤条件||R 中有过滤条件)
令 N=过滤条件之前的节点，C=过滤条件，将 C 中的元素或属性加上前缀 N. 并去掉@后插入 W
Else N=p_i中的路径节点，C=p_i中的最外层过滤条件集合
If (N 是复合节点) 将 N 插入 F 中
For each c∈C
If(c 中有“/”) SplitXPath(c,N)
Else If(c 以“ORDER: ”开头)
将“ORDER: ”之后元素或属性加上前缀 N. 并去掉@后插入 O
Else If(c 以“GROUP: ”开头)
将“GROUP: ”之后元素或属性加上前缀 N. 并去掉@后插入 G
Else 将 c 中的元素或属性加上前缀 N. 并去掉@后插入 W}

完整的查询转换过程可以描述为：抽取输出指令集合S_{out}和变量指令集合S_{var}，根据算法 1 合并XPath，替换变量并简化为无分支的 XPath。对每一条无分支的 XPath 调用算法 2，得到一组集合 S{}, F{}, W{}, G{}, O{}，分别对应 SQL 中的 SELECT, FROM, WHERE, GROUP BY 和 SORT BY 子句。

4.4 示例

示例 2 只有一条输出语句，该语句的S_{out}={2,4,5,6}，数字代表XSLT指令的行号。路径合并后得到Product/$sel[Goods/@ID=$pro/@ID]/Sum，替换变量并简化为“ 无分支的 ”后变成 2 条：

Product/Sell[Sum][Goods/@ID=Product/@ID]/Sum
Product/Buy[Sum][Goods/@ID=Product/@ID][ORDER: Date] / Sum

对第 1 条路径，转换之后S、F、W、G、O的值分别为{Buy.Sum}, {Product,Buy,Goods,Product}, {Buy.Sum,oods.ID= Product.ID}, {}, {Buy.Date}。但此时的结果集中还有冗余信息，不能直接构建SQL。首先去掉其中的重复项，如果F中的任意 2 个节点t_1, t_2之间存在父亲-孩子关系，还要在W中添加t_1.ID=t_2.pID。按上述方法处理之后得到的最终结果为{Buy.Sum}, {Product, Buy, Goods}, {Buy.Sum, Goods.ID= Product.ID, Buy.ID=Good.pID}, {}, {Buy.Date}。此时，就可以根据S, F, W, G, O构造一条SQL语句了。构造方法省略。对于W中的条件c如果不是表达式则可以转换成c IS NOT NULL，形如contain(x, y)的函数可以转换成x LIKE ‘%y’的形式。最后给出结果SQL1：

SELECE Buy.Sum
FROM Product, Buy, Goods
WHERE Buy.Sum IS NOT NULL and Goods.ID=Product.ID and Buy.ID=Good.pID
SORT BY Buy.Date

同理可得 SQL2（略）。SQL2 与 SQL1 的区别在于 SQL2 将 SQL1 中的 Buy 换成了 Sell。显而易见，SQL1 和 SQL2 的结果集之和与示例 XSLT 的运行结果一致。

5 结束语

如何结合 XML 和关系数据库的优点是近年来关系数据库技术研究的热点问题。越来越多的商业应用之间通过将关系数据发布为 XML 文档实现 Internet 上的数据共享。这就需要将对 XML 的查询转换为对关系数据的查询。本文给出基于 XML DTD、将 XSLT 查询转换成 SQL 查询的算法。然而，本文提出的算法仅使用了 DTD 中节点的父子关系，如何借助更多 DTD 信息，以实现查询转换算法的优化是下一步要考虑的问题。

参考文献

1 Jain S, Mahajan R, Suciu D. Translating XSLT Programs to Efficient SQL Queries[C]//Proc. of WWW’02, Honolulu, Hawaii, USA. 2002.

2 Fernandez M F, Tan W C, Suciu D. Silk Route：Trading Between Relations and XML[J]. Computer Networks, 2000, 33(1/6): 723.

3 孙宏伟, 张树生, 周竞涛, 等. 模式映射弱依赖的 XQuery 到 SQL 转换算法[J]. 计算机辅助设计与图形学学报, 2004, 9(16): 1301.

4 Liu J X, Vincent M. Querying Relational Databases Through XSLT[J]. Data & Knowledge Engineering, 2004, 48(1): 103-128.

5 高 军, 杨冬青, 唐世渭, 等. 一种基于DTD的XPath逻辑优化方法[J]. 软件学报, 2004, 15(12): 1860-1868.