维普资讯 http://www.cqvip.com

XPostgreSQL 对 XML 支持的策略

刘长浩　程　伟　孙玉芳
（中科院软件所　北京100080）

摘　要　随着XML的应用越来越广泛，对XML的支持已经成为数据库系统功能的一个重要组成部分。本文在PostgreSQL的基础上，提出了一个全面支持XML的新版本XPostgreSQL。并从XML文档的存储、关系数据到XML数据的转换、XML数据到关系数据的转换三个方面详细介绍了XPostgreSQL对XML支持的策略。其中，MAP文件、XPathRewriter、索引表(Side Table)等技术和概念都是第一次在PostgreSQL对XML的支持研究中提出。

关键词　XPostgreSQL，XML，PostgreSQL，XType，数据库管理系统，XPath，MAP文件，索引表，XPathRewriter

Strategies on the XML Support of the XPostgreSQL

LIU Chang-Hao　CHENG Wei　SUN Yu-Fang
(Institute of Software, Chinese Academy of Sciences, Beijing 100080)

Abstract　With more and more XML application coming forth, the support for XML is becoming a very important part of the DBMS function. This paper brings forward XPostgreSQL, the new version PostgreSQL completely supporting XML, and introduces the strategies on the support for XML of the XPostgreSQL from three aspects, XML documents storage, the transformation from the XML documents to relational data and the transformation vice versa. In this process, many technologies and concepts such as the Map file, side table and XPathRewriter are first mentioned in the research on the XML support for PostgreSQL.

Keywords　XPostgreSQL, XML, PostgreSQL, XType, DBMS, XPath, MAP file, Side table, XPathRewriter

1　引言

自从W3C于1998年2月推出XML1.0版本[2]，因其具有的许多特独特的优点，受到了众多软件厂商的推崇，XML文档渐渐成为信息交换的标准格式。作为信息的主要存贮载体和管理工具，数据库管理系统(DBMS)对于XML的支持成为一种必然的趋势。现在流行的通用商业数据库管理系统(DBMS)，比如ORACLE 8i\9i，DB2，SQL Server都在不同的程度上增加了对于XML的支持。而作为开源(OPEN SOURCE)软件在数据库领域的最重要的代表的PostgreSQL[1]，对于XML的支持却非常有限。本文将从应用的角度，全面讨论其对XML支持的策略，提出一些全新的解决方案，并将实现该策略的PostgreSQL版本命名为XPostgreSQL。

2　XML与数据库

XML即Extensible Markup Language（可扩展的标记语言），它使用标记来标识一个大的信息体中的不同的信息元素，主要用来进行信息的传递。它是SGML(Standard Generalized Markup Language标准通用化标记语言)的子集，它的1.0版本由W3C于1998年2月推出[2]。

XML文件结构严谨，它通过标签的嵌套形成一个具有严格层次结构的文档结构树。

例1　下面是一个简单的XML文档，图1是其对应的结构树模型。

```
<Person>
 <name>John</name>
 <age>26</age>
 <address>
   <country>China</country>
   <Province>JiangSu</Province>
 </address>
</Person>
```

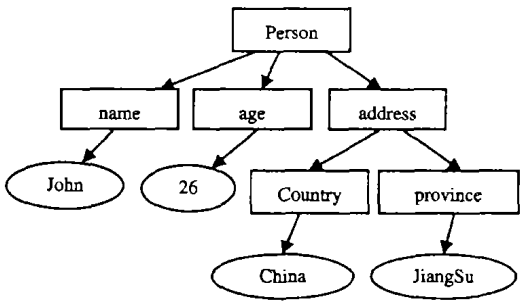

图1　XML文档树结构模型

XML文档大体上分为两种，一种是以数据为中心(data-centric)[7]，这种应用一般把XML文档当作一个数据传送器。它的数据一般都具有较好的格式，具有较小的粒度。另一种是以文档为中心(document-centric)[7]，主要是面向用户的，其数据一般不具有非常好的格式化结构，粒度较大。XML文档的分类并没有一个明确的标准，但是对XML文档进行分类，可以帮助选择XML文档与关系表之间模式映射的实现策略。

XML系统(包括XML文档、XML解析器等等)和数据库系统都可以用来存储和操作数据，因此在某种程度上来讲，

刘长浩　硕士研究生，研究方向为数据库管理系统为XML。程　伟　博士生，研究方向为数据库管理系统。孙玉芳　教授，博导，研究方向为系统软件、中文信息处理。

维普资讯 http://www.cqvip.com

二者可以相互替换。但是，XML 所具有的独立于平台的特性，以及由于其标签所占的空间造成的冗余和处理文本的低效率使得它适合进行数据的传输，而不是存储。而数据库系统，从其创建起初就具有的数据完整性、数据一致性约束等特点，使其非常适合于数据的存储与管理。因此 XML 和数据库被集合起来使用以获得更好的数据存储和传输性能。

数据库对于 XML 的支持的核心部分[7]包括数据库管理系统(DBMS)对 XML 文档的存储、关系数据到 XML 文档的转换、XML 文档到关系数据的转换。下面将从这三个方面介绍 XPostgreSQL 对 XML 支持的实现策略。

3 XML 文档在 XPostgreSQL 中的存储

XML 数据可以以三种方式存储在数据库中。第一种是以字符型大对象(CLOB)的方式存储在数据库关系表的某一列中。这种方式比较适用于对 XML 文档的修改和更新不是很频繁、每次请求都是提取整个文档的情况。它对于修改更新操作的花费较大，因为每一次 XML 文档的更新，都需要一次对大对象的写出写入磁盘操作。

第二种方法是把一个 XML 文档映射到多个关系表中，称之为对象关系映射存储(Object Relational Storage)。这样，就可以以操作关系表的方式来提取和更新 XML 文档中的数据而不会导致写出写入整个 XML 文档的磁盘操作。这种方法比较适用于对 XML 文档的更新比较频繁、每次请求的只是少量节点的情况。这种方法涉及到 XML 模式与数据库模式(Database Schemas)之间的映射[3]。

第三种方法是利用操作系统的文件系统来存储 XML 文档，即关系数据库的表中只保存这个文档的文件名和指向这个文件的指针。这种方法的优点是占用空间很少，易于查找。缺点是由于文档没有存储在 DBMS 中，数据库系统所具有的一致性、完整性约束以及安全性的检查就没有办法施用到这些文档上面来，因此有造成数据不一致性和其他安全方面的危险。

在 XPostgreSQL 中，提供了一个标准的数据类型 XType，用来存储和管理 XML 文档。为了很好地满足不同的需求，采用了第一和第二种方法来实现该数据类型，以供用户选择使用。为 XML 文档创建特定的数据类型满足了新的 SQL 标准中 XML 相关的部分 SQL/XML[4]所要求的在 SQL 中具有自己的 XML 数据类型的要求。

3.1 字符型大对象(CLOB)存储方法

XPostgreSQL 字符型大对象的存储方法既可以用来存储没有特定 XML Schema 的 XML 文档，也可以存储符合某个 XML Schema 的 XML 文档。这种方法可以很好地满足文档传输的应用。为了提高这种存储方法的查询效率，XPostgreSQL 提供了称作索引表(side-table)的索引方法。

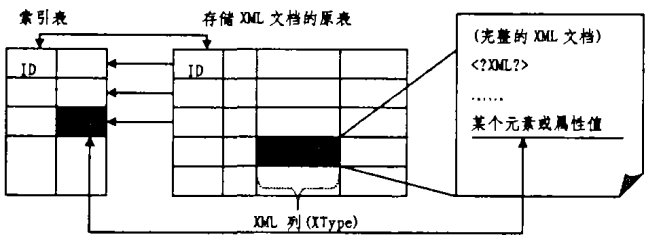

图2 CLOB 存储与索引表

在关系数据库中，索引都是建立在表的某一或多个列上的。当 XML 文档作为一个整体存储到数据库中时，按照传统的方法就没有办法在 XML 文档内部的数据上建立索引了。为了加快 XML 文档的查找速度，XPosgreSQL 提供了基于 XML 文档内部信息的索引策略。这是通过另外建立一个表(索引表(side table))来实现的，这个表在物理上独立存在。该表有两列，一列为唯一标识存储 XML 文档的表(以下就称之为原表)中一行的 ID，另一列为原表的 XML 列对象内部参加索引的某个元素或属性的值。索引表与原表中的行一一对应。它们之间的相互关系如图2所示。在使用的时候使用子查询(Sub Query)来指明查询的条件。

索引表只有两列，并且和原表的行一一对应。但是 XPostgreSQL 并没有把它合并到原表中，这是因为索引表是用户根据自己的需要而建立的，不同的用户可能需要在不同的节点上建立索引，同一个用户在不同的阶段也可能在不同的节点上建立索引。而原表一旦建立，再进行更改就需要花费巨大的代价，所以 XPostgreSQL 把索引表和原表分开来建立。实际上，用户可以通过视图(View)来把这两个表合起来使用。

可以看出，要使索引表能够正确地被使用，必须保证它与原表保持数据的一致性。即要求对原表中的数据进行更新操作(插入、更新、删除)时，索引表也要及时根据原表的变化更新自己的数据。在 XPostgreSQL 中这是通过创建在原表上的规则[1,5]来实现的，实际上在创建索引表的时候，内部实现包括两个部分：1)创建物理的索引表，根据原表中目前的元组初始化新创建的索引表的元组。2)在原表存储 XML 文档的列上创建若干规则(RULE)，使得对于原表的更新及时反映到索引表上。PostgreSQL 通过 CREATE RULE 为这种实现的机制[1,5]提供了可能。

3.2 对象关系存储

对于那些需要经常进行更新文档的应用，字符型大对象的存储方法的代价太大了，因为对文档内容的每一次更新都将导致整个 XML 文档的一次读出写入磁盘操作。XPostgreSQL 提供了另一种存储方法——基于 XPath 的对象关系存储来满足这种应用。对象关系存储把 XML 文档分拆成多个碎片存储在关系数据库的关系表中，这些关系表不仅存储 XML 文档中的数据，而且存储 XML 的结构信息，使得能够从这些关系表重构原始的 XML 文档。

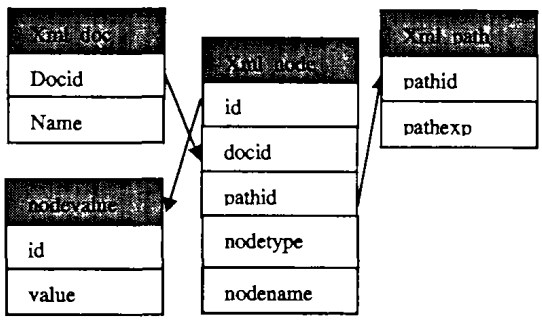

图3 基于 XPath 的 XML 内部存储

在 XPostgreSQL 中，分拆是基于 XPath 的，即把一个 XML 文档树以节点为单位分拆存储，对于每一个节点以其从根节点到该节点的 path 进行标识。对于文档树每一个节点，建立一个关系表，存储每一个节点的 path 以及对应的节点内容(或者以相关的其他表来存储)。这样，通过 XPath 对于文档某个内部节点的访问，就可以通过对 XPath 串和关系表中的 path 的值进行字符串的比较来实现。在 XPostgreSQL 中，

维普资讯 http://www.cqvip.com

为了节省存储空间以及操作方便，把存储节点值和节点表达式的部分抽取出来，使用了四个关系表来存储 XML 文档(图3)。这样，XML 文档的数据信息由表 nodevalue 存储，而结构信息由表 Xml_path 以及 Xml_node 存储。以后对于这些 XML 文档内部信息的 XPath 的操作，都通过 XPathRewriter 的重写，转化为对相应的表的 SQL 操作。

基于表的对象关系存储，其主要难点在于 XPathRewriter 的设计及实现，即由 XPath 请求到 SQL 请求的转化，Yoshikawa 和 amagasa 提出了一个转化的算法(9)，XPostgreSQL 将在这个算法的基础上完成 XPathRewriter 的设计及实现。

对于 XML 文档的导入，XPostgreSQL 对不同的数据源提供不同的 SQL 函数。*XTypeFromVarchar*()，*XTypeFromText*()，*XTypeFromFile*()三个函数实现了分别从类型 varchar，text 和操作系统中的文件导入 XType 类型数据的功能。这三个函数可以通过一个标志参数来决定使用那种存储方法。对于从存储在 XType 中的文档提取数据，XPostgreSQL 将提供两个函数 *extract*()和 *extractvalue*()来分别得到文档中的数据。

4 关系数据到 XML 数据的转化

XPostgreSQL 支持 XML/SQL 标准，提供了直接从扩展的 SQL 语言获得 XML 的功能。它是通过改造 select 语句，使得其输出结果是 XML 文档，这部分 SQL 语句称为 FOR XML 子句。嵌套使用改造后的 select 语句可以产生嵌套结构的 XML 文档。这种方法被称为基于 SQL(SQL-based)的方法。

FOR XML 子句的语法是：

FOR XML [*XMLSCH*]

如果指定了 *XMLSCH* 参数，结果中需要返回 XML 文档的模式信息。

FOR XML 子句首先在 DBMS 中创建了一个通用的表，它包含了最后生成的 XML 树的所有信息。这个表连续的若干行对应于 XML 树中的一个节点。请求语句必须提供建立这个通用表的所有信息。在这个表中除了各种元素和属性信息，还存在另外的两列(最前面的两列)，分别存储自己的和父节点的 Tag 值，各个元素通过这两个属性列，形成了一个具有鲜明层次性的 XML 文档树。另外，请求还必须提供这个通用表的每个列的列名。

一个带有 FOR XML 子句的完整的 SELECT 语句的语法如下：

SELECT *m*, *n* {,*colname* [AS *elmentname* ! *tagvalue* ! *subname* ! ELMENT]}
FOR XML [*XMLSCH*]

上面 SQL 语句中 AS 后面的语法元素指定列的性质共由四部分组成。其中第一部分指定了该列在生成的 XML 文档中的元素名；第二部分指定了该列的 Tag 值；第三部分指定了该列的列名(对应于最后的 XML 文档中该元素的属性名或子元素名)；第四部分是可选的，它可以用来指定生成的是属性还是子元素。

在通用表建立了以后，XPosgreSQL 根据前两列的标签值(*m*,*n*)建立具有层次结构的 XML 文档树。

FOR XML 子句可以使用户得到任何想要得的 XML 文档树。它可以指定某一列对应于 XML 文档中的属性或是子元素；可以指定元素、属性的名字；指定元素属性在 XML 文档树中的层次。

另外，XPostgreSQL 还提供一个功能模块以基于模板(template based)的方法实现从关系数据到 XML 数据的转换(图4)。模板实际上是一个嵌有 SQL 语句的 XML 文档，由用户提供，该功能模块处理该模板，在 SQL 语句的相应位置替换上 SQL 语句返回的数据，最后的输出就是用户所要求的 XML 文档。这种方法实现简单且直观，对于比较简单的应用可以很好的满足。

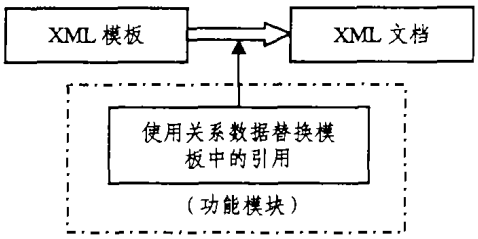

图4 基于模板的映射

5 使用 MAP 文件实现 XML 文档到关系数据的转化

XPostgreSQL 把一个 XML 文档中的数据通过一个 MAP 文件映射到数据库的一个或多个关系表中。这个涉及到 XML 模式和数据库模式间的映射，而 MAP 文件就是完成这样的功能。MAP 文件实际上也是一个 XML 文档，它指定了目标关系表的模式，并通过 XPath 指定 XML 文档中的节点于关系表中列的映射关系。关系表中的数据与 XML 文档中的数据并不是一一对应的，实际上，XPostgreSQL 只获取关系表模式所规定的列所需要的值。MAP 文件是一个四层结构的 XML 文档，其结构如下：根节点是〈MAP〉，根节点下有若干个〈table〉子节点，指定了目标关系表的名称，以及组成这个表的若干列。列有四个属性，分别为：列名、列的数据类型、XML 文档的一个 XPath、一个是否允许重复出现的标志。其中 XPath 说明了该列的数据的来源(对应的 XML 文档内部节点)。图示化的过程见图5。

XPostgreSQL 根据这个 MAP 文件所确定的信息生成 SQL 语句序列的脚本。这个脚本包含的语句序列有：

1) 根据 MAP 文件中的标签为〈table〉的元素及其子元素〈column〉而生成的创建表的命令。其中，列的名字和标签 column 的属性 name 的值相同，列的类型和标签 column 的属性 type 相同。

2) 向创建的关系表中插入值的 SQL 语句。这些值是由 XPath 所指定的 XML 文档中的元素或属性的值。

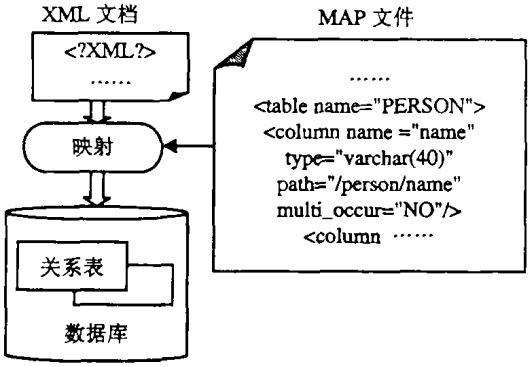

图5 通过 MAP 文件实现 XML 文档到关系数据的转换

最后，执行该脚本，就可以实现从 XML 文档到关系数据的转换了。

维普资讯 http://www.cqvip.com

6 XPath 的处理

XPath 是操作 XML 文档内部节点的 W3C 推荐标准，XPostgreSQL 所有涉及到 XML 文档内部信息的操作，比如使用 MAP 文件实现 XML 数据到关系数据的转化、索引表的实现以及一般的 XML 内部信息提取，都离不开 XPath 操作的支持。因此 XPath 操作的效率对于整个系统的性能有着重要的影响。

由于 XPostgreSQL 采用了两种完全不同方法存储 XML 文档，这也导致了对 XML 文档节点的 XPath 操作有两种完全不同策略。下面将简单说明 XPostgreSQL 在采取不同存储策略时 XPath 操作的实现方法。

对于以字符型大对象(CLOB)方式存储的 XML 文档，要对其内部节点进行操作，没有别的选择，必须以传统的方式(DOM 或是 SAX)对其进行解析，对于 XML 内容的任何操作都会导致一次整个 XML 文档的磁盘写出写入操作。目前，有许多基于 DOM 或 SAX 的 XML 解析函数库，XPostgreSQL 采用了基于 DOM 的快速 LibXML C 库。

对于以对象关系映射存储的 XML 文档，由于 XML 文档实际上是以碎片的形式存储在四个关系表中的，因此对于文档内容节点的 XPath 操作就可以直接通过 SQL 操作相应的关系表来进行，而不必解析整个 XML 文档。但是这就要求 XPostgreSQL 能够准确地知道某一个 XPath 所指定的 XML 文档节点存储在哪个关系表，以及这个 XPath 操作对应这个关系表的哪个 SQL 操作。在 XPostgreSQL 中，这个工作是由 XPathRewriter 来完成的，XPathRewriter 以分析器(Parser)输出的用户请求树作为输入，扫描其中的 XPath 相关操作，然后通过参考系统中四个关系表的信息，生成对这些关系表的相应的 SQL 操作语句，重写了原始的 SQL 请求。

重写以后的请求树像其它的用户请求一样被传递给重写器(Rewriter)，进行传统的规则重写优化，参见图6。因此它能够获得很好的执行效率。

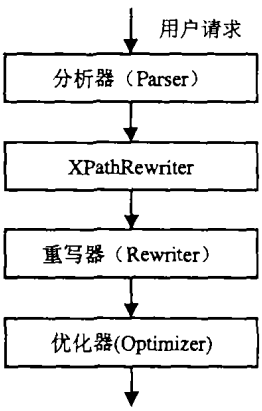

图6 XPathRewriter 在请求处理中的位置

总结 XPostgreSQL 是通过以下三个方面的支持，对 PostgreSQL 进行扩展而得到的支持 XML 的数据库管理系统(DBMS)：

1) PostgreSQL 对于使用 C，PERL，PL/SQL 语言编写的用户自定义函数的支持；

2) XMLsoft.org 提供的基于 C 的快速的 LibXML 库；

3) PostgreSQL 提供的用户自定义数据类型，用户自己定义规则等功能。

图7是其体系结构的简单描述。

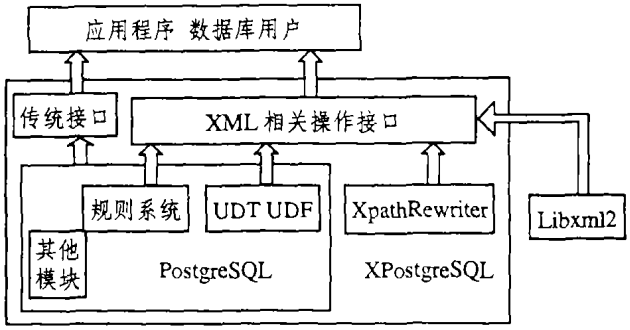

图7 XPostgreSQL 简单体系结构图

它提供了如下的功能：

1) 一个标准的数据类型 XType，用来存储和管理 XML 文档；

2) 两种不同的 XML 文档存储方法；

3) FOR XML SQL 子句实现直接从 SQL 语句获得 XML 文档；

4) 利用 MAP 文件实现从 XML 文档到关系表的转化；

5) 一系列函数和 SQL 操作符可以在 XType 类型的数据上进行 XML 操作；

6) 将标准的 W3C XML Schema 数据模型转化为数据库中的模式。

新的 SQL 标准 SQL\XML 主要新增加了两个方面的规范：使用 SQL 语句请求、访问 XML 内容的规范，以及使用 SQL 语句直接从关系表中获得 XML 的规范。XPostgreSQL 通过实现 XPath 操作的 SQL 函数和 FOR XML 子句较好地遵循了 SQL\XML。

参 考 文 献

1 PostgreSQL Development Group. PostgreSQL V-7.3.4 source codes. PostgreSQL website http:www.Postgresql.org

2 Bray T, Paoli J, Sperberg-McQueen C M, Maler E. Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation 6, Oct. 2000

3 info@xlnt-software.com Handling XML documents using traditional databases. July/Aug. 2002

4 Melton J, ed. XML-Related Specifications (SQL/XML). ISO-ANSI 2001-12-17

5 Stonebraker M, Hanson E, Potamianos S. The Postgres Rule Manager. EECS Department, University of California Berkeley, Ca., 94720

6 Berglund A, et al. XML Path Language (XPath) 2.0. W3C Working Draft 12, Nov. 2003

7 Bourret R. XML and Databases. http://www.rpbourret.com November, 2003

8 Drake M. Oracle XML DB. www.oracle.com 2004

9 Yoshikawa M, Mamagasa T. XRel: A Path-Based Approach to Storage and Retrieval of XML Documents Using Relational Databases. ACM Transactions on Internet Technology, 2001, 1(1)