S_Schema 及其关系映射方法

游中胜
(重庆师范大学数学与计算机学院　重庆 400047)

摘　要　本文根据 W3C 最新提出的 XML Schema 规范，提出了一种等价于 XML Schema 的数据模型 S_Schema，并实现了 S_Schema 到关系模式的生成算法和 XML 文档到关系数据库的加载算法。实验证明，S_Schema 方法在数据转储过程中的信息保持、映射后的查询更新操作等方面的综合性能要优于文本、Xparent 方法。
关键词　S_Schema (Steady Schema)，XML，XML Schema，XML 存储，XML 查询

S_Schema and it's Relation-Based Mapping Method

YOU Zhong-Sheng
(Dept. of Mathematics and Computer Science, Chongqing Normal University, Chongqing 400047)

Abstract　According to the XML Schema specification published recently by W3C, this paper presents a data model named S_Schema that is equivalent to XML Schema. At the same time, the mapping algorithm from S_Schema to relational Schema and the loading algorithm from XML document to relational database are completed in this paper. By experiment, it proves that S_Schema mapping is better than other methods such as TEXT and Xparent on information maintaining and data querying and updating.
Keywords　S_Schema (Steady Schema), XML, XML Schema, XML Storage, XML Query

1　引言

随着 Internet 上越来越多的数据用 XML(eXtensible Markup Language)[1]文档表示，XML 文档的存储和查询成为人们日益关心的问题。用关系数据库对 XML 文档进行存储不但可以充分利用现有关系数据库的丰富资源和管理经验，而且可以充分利用 XML 的优点发展 Internet 应用。但是由于 XML 数据和关系型数据在组织上的差异，在存储转换过程中，可能会造成有用信息的丢失，而且转换过程本身也会增加系统开销影响数据库的效率，增加复杂性。研究 XML 数据在关系数据库中的有效存储成为 XML 研究中的一个热点和难点。

2　基于关系的 XML 存储方法

根据存储时是否使用 XML 模式(DTD 或 XML Schema)，基于关系的 XML 存储可以分为结构映射方法和模型映射方法两种类型。

2.1　模型映射方法

模型映射方法是一种无模式映射方法，它用固定的关系模式来存放任何格式的 XML 数据，而不考虑 XML 文档的模式，其本质是存储 XML 文档本身的结构信息。在模型映射方法中，XML 文档被看作为由元素和属性等节点组成的有向有序的树或图的结构，关系模式就相当于一个模板，XML 在关系数据库中的存储按数据库提供的模板来组织数据。模型映射方法主要有 EDGE 方法[2]和 XParent 方法[3]等。

EDGE 方法[2]是较早研究边映射的一种方法，D. Florescu 和 D. Kossman 提出了一种基于边的映射，并命名为 Edge。Edge 的思想是：将整个 XML 文档图中的每条边存储在关系表(Edge Table)中，每条边由源节点和目标节点的 Id 号唯一确定，每条边同时还包含目标节点的名称、值和类型信息，ordinal 字段指出了子元素兄弟节点间的顺序信息。根元素节点源节点 Id 号为 0。由于 Edge 方法将所有 XML 数据都用 Edge 表存放，操作方法简单，但在 Edge 中每条边都是单独的管理，所以在用户进行查询操作时就需要执行大量的链接操作以形成路径。

综合 Edge 方法的优缺点，H. Jiang 等人提出了 XParent 方法[3]，XParent 是一个由四个关系表组成的数据库模式，四个关系表 LabelPath(ID, Len, Path)、DataPath(Pid, Cid)、Element(PathID, Did, Ordinal)、Data(PathID, Did, Ordinal, Value)、分别用来存储边路径、数据路径、XML 文档中的元素和数据。XParent 将边路径和数据路径明确地分开，分别用 LabelPath 和 DataPath 两个不同的表进行存储。相对于 Edge 方法，XParent 对所有不同的边和路径都进行了存储，因此 XParent 在查询时可以根据路径很容易地找到元素的信息，因而其在查询效率方面要优于 EDGE 方法。

模型映射方法不但存储了 XML 文档本身的数据信息，同时还存放了其相应的模式信息，对某些应用来说提供了一定的灵活性。然而在模型映射方法中，每个数据项都要重复其模式信息，因此严重增加了系统的磁盘开销；而且由于需要处理更多重复的模式信息，因而也增加了系统的时间成本。

2.2　结构映射方法

结构映射方法也称为有模式映射方法，即在进行关系数据库的 XML 存储时，先根据 XML 模式(或挖掘出 XML 文档中固有的模式信息)生成相应的关系模式，然后再根据生成的关系模式对 XML 文档进行解析分解并将它存放于相应的数据表中。结构映射方法的代表作主要有 STORED 方法[4]和 DTD 方法[5]。

STORED 方法[4]是较早用关系数据库来存储 XML 数据最有意义的尝试。STORED 结合关系数据库技术和半结构化数据处理技术来实现对半结构化数据的管理。它采用数据挖掘算法从 XML 文档中提取有代表性而且置信度大于预先给定阈值的 DTD，根据提取出的 DTD 自动生成关系模式，对

游中胜　硕士，讲师，研究方向：遗传算法、随机过程、数据挖掘。

于个别不符合所提取 DTD 结构的 XML 数据，STORED 将它们另存在“OVERFLOW”表中。由于 STORED 用关系表和“OVERFLOW”共同存储数据，对于那些结构变化较大的文档，“OVERFLOW”将变得很庞大，不但耗费大量的数据空间，而且查询操作也需要经常徘徊于关系表和 OVERFLOW 之间，效率不高。

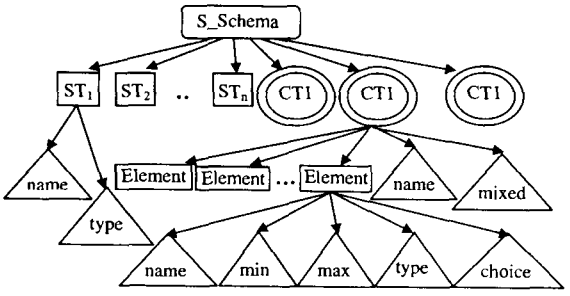
DTD 方法[5]是 XML 领域比较有影响的一个研究，DTD 方法根据 XML 模式信息(DTD)来生成相应的关系模式。首先根据 XML 文档的 DTD 定义建立一个有向 DTD 图，图中的每一节点代表一个 XML 元素、XML 属性或表示元素间关系的符号，“?”用来表示可选元素即值可为空的元素，“*”用来表示集合值子元素，即那些在其父元素下能够多次出现的元素，一旦建立好了 DTD 图，就可以遍历它从而构建需要的关系模式。

3 S_Schema 数据模型及其映射方法

现有的结构映射方法的研究都是根据 XML 文档的 DTD (Document Type Definition，文档类型定义)转化为关系模式来对 XML 数据进行存储的。DTD 是近年来 XML 技术领域所使用的最广泛的一种模式。但是，DTD 并不能完全满足 XML 自动化处理的要求，不能很好地实现应用程序不同模块间的相互协调，缺乏对文档结构、属性、数据类型等约束的足够描述等等。根据 W3C 最新提出的 XML Schema 规范，本文提出了一种全新的数据模型 S_Schema，S_Schema 是 XML Schema 的一种等价形式，并由 XML Schema 变换而来。

3.1 S_Schema 数据模型

S_Schema(Steady Schema)将 XML Schema 的层次结构变换为相对固定(Steady)的二维平面结构，并通过元素间的相互引用将元素间的嵌套、递归等关系隐藏其中。S_Schema 同样是一个 XML 文档，遵循 XML 的所有语法规则，可以用 DOM 来表示 XML 文档的结构(Document Object Model，文档对象模型)。在 S_Schema 的 DOM 树表示中，我们用圆角矩形框表示根元素，用双线椭圆表示 S_Schema 中的复杂元素，矩形框表示 Element 元素(简单元素或复杂元素)，三角形表示属性元素，则按 DOM 树的生成思想，S_Schema 可用图 1 所示的 DOM 树来表示，图 2 为 S_Schema 的一个实例。



其中 ST-SimpleType CT-ComplexType

图 1 S_Schema 的 DOM 树表示

由图 1 可知，S_Schema 为一个包含了三层元素的具有固定结构的模式：

第一层是整个文档的根元素，用标签 S_Schema 来表示；

第二层由简单类型元素申明和复杂类型元素申明两部分组成。其中简单类型元素申明主要列举出 XML Schema 中的简单类型元素并保留其名字属性(name)和类型属性(type)等信息，简单类型元素直接用标签 element 表示；复杂类型元素申明列举出 XML Schema 中所出现的所有复杂元素，包括本地复杂元素和全局复杂元素，复杂元素申明的用 ComplexType 标签表示，其中含有该元素的名称属性(name)和 mixed 属性(mixed 标示该复杂类型元素是否为混合元素)；

第三层为复杂元素内部所引用的元素，所有复杂元素所引用的元素用标签 element 表示，各元素在 S_Schema 中的位置与其在 XML Schema 中的位置一致(即在生成 S_Schema 时应保留各元素间的顺序信息)。并分别标出以下属性：

name：所引用元素的名称；

type：所引用元素的类型，如果所引用元素是复杂类型用 complex 表示，如果是简单类型则直接标出其相应的简单类型，如 string、integer 等；

min，max：该元素出现的最小次数和最大次数，unbounded 表示无穷大；

choice：choice 元素中的子元素均提取出来作为 choice 父元素的子元素处理，同样根据分配律 $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$，将 choice 的修饰属性下放为对其子元素进行修饰，而且由于 choice 元素的子元素也可能是 choice 元素，即存在 choice 的嵌套结构，用 0、1、2…等数字来表示 choice 属性的值，其中 0 表示最外层 choice 下的子元素一级，1 表示所嵌套的 choice 元素的子元素一级，依此类推。

由于在 S_Schema 中包含了所有复杂类型元素的基本信息，原来的 XML Schema 中元素间复杂的层次嵌套关系也得以保持，各简单元素的类型及引用次数等基本信息都没有丢失。

简单类型元素申明

复杂类型元素申明

```
<S_Schema>
  <element name="PERSONA" type="string" />
  <element name="SUBHEAD" type="string" />
  <ComplexType name="EPILOGUE">
    <element name="SPEECH" type="complex" min="1"
max="unbounded" choice="true" />
    <element name="SUBTITLE" min="0" max="unbounded"
type="string" />
    <element name="TITLE" max="unbounded" min="1"
type="string" />
    <element name="STAGEDIR" min="1" type="string"
choice="true" max="unbounded" />
  </ComplexType>
  .....
</S_Schema>
```

图 2 S_Schema 实例

3.2 S_Schema 方法

S_Schema 方法主要包括 S_Schema 生成、关系模式生成、XML 文档加载、XML 查询到 SQL 查询的转换和实验五个阶段，如图 3 所示。

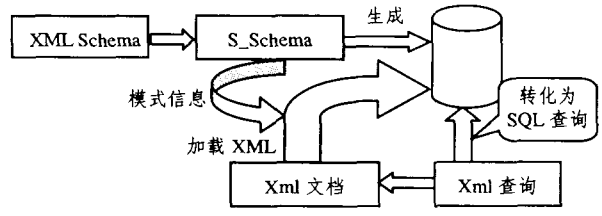


图 3 S_Schema 方法实验设计示意图

4 实验结果和分析

本文实现了从 S_Schema 到关系模式的生成算法和 XML 文档的加载算法，从实验上对 S_Schema 方法和文本方

法及 Xparent 方法进行了比较。在本文中主要考虑了映射后的查询效率问题。本文数据来源 http://metalab.unc.edu/bosak/xml/eg/shaks200.zip。实验的具体软、硬件环境分别如下：

硬件环境：CⅢ 1.2G Hz CPU；384M SDRAM 内存。软件环境：操作系统 Microsoft Windows 2000 Advanced Server (5.0, Build 2195)；开发平台 Microsoft Aspx..net Version 1.0.3705；数据库服务器 Microsoft SQL Server 2000 Enterprise Edition。

4.1 S_Schema 方法的可恢复性

映射后的文档能否完全恢复关键是看从 XML 到数据库的转换过程中是否存在信息丢失的问题，是否完整地保存了 XML 文档的全部信息。S_Schema 方法以 XML Schema 为研究对象，针对 XML 存储的几个难点问题都做了一些相应的处理：

1. S_Schema 文件是 XML Schema 的等价形式，因而能够完整地保存 XML 文档的结构信息和元素的基本信息；

2. 对于 XML 多值元素间的顺序问题，由于在将 XML 文档存储到数据库时，每条记录都有其相应的 ID 号，又由于 XML 文档的解析采用的是 SAX 编程接口，SAX 采用的是只向前式的读取方法，因此根据数据库中元素的 ID 号即可推知多值元素间的顺序；

3. 对于文档中的混合内容元素，本文将标签内的所有内容作为一个整体进行处理，即拆分了 XML 文档，又保存了其完整性，利于 XML 文档的恢复；

4. 对于文档中的一些细节信息，如数据类型、字段长度等，XML Schema 也提供了很好的支持。

因此，从理论上讲，S_Schema 方法能够完整地保持 XML 文档的结构信息和数据信息，可以实现 XML 文档的完全复原。

4.2 S_Schema 方法和其它方法的比较

本文实验选用的查询充分考虑到了 XML 查询单表查询、多表连接等不同的方面，能够有效地验证 XML 的查询。实验所选用的查询主要有(Xpath)：

查询 1 //PLAY/ACT

查 询 2 //PLAY/ACT/SCENE/SPEECH/LINE/STAGEDIR

查询 3 //PLAY/ACT/ SCENE /TITLE

查询 4 //PLAY/ACT[2]

查询 5 //PLAY/ACT/SCENE/SPEECH[SPEAKER = "PISANIO"]

表 1 S_Schema 实验数据(数据均以毫秒(ms)为单位)

查询	查询 1	查询 2	查询 3	查询 4	查询 5
文本方法	16293.4288	310.4464	300.432	4856.984	370.5328
Xparent	620	287	50	2560	80
S_Schema	351	140	10	180	30

表 1 列出了实验得出的各种方法在不同操作情况下的操作性能，由表 1 可以看出：相对于其它两种方法，文本存储方法在数据查询和数据操作等方面效率都明显低于其他方法。虽然采用有效的索引机制可在一定的程度上改善文本存储方法的查询或更新效率问题，但在文本存储方法中，数据更新或查询操作最大的开销在于 XML 文档的装载和转储时间，而且装载和转储时间随着文档内容的增加而显著增加，转储比装载的开销更大。所以可以断定，在计算机内存比较小或者 XML 文档比较大的情况下，采用文本存储方法进行文档的存储并进行数据更新、查询操作，估计是无法忍受的。

按 Xparent 方法进行 XML 文档的存储，其查询更新数据的性能显著要高于文本方法，这主要是因为 Xparent 方法采用关系数据库进行数据的存储，关系数据库在进行数据的存取时只需根据需要处理与它相关的一些数据，而不像文本方法那样需要将整个文档全部加载到内存，从查询处理的基本原理上来说，关系数据库存储方法就要优于文本存储方法。而且关系数据库有效的索引、存储管理等查询机制更有效地优化了其查询处理等方面的性能，因此其对数据的装载和转储性能均要明显高于文本存储方法。

S_Schema 方法表现出的查询性能大大地高于其他两种方法，其主要原因主要有以下几点：其一，S_Schema 方法不但存储了文档的数据信息，而且保存了其模式信息，数据的查询操作可以根据模式信息来进行，能够做到有章可循，而不需要像 Xparent 方法一样只能逐步由边构造出路径信息进行一步一步渐近式的搜索；其二，S_Schema 方法根据模式信息进行文档的拆分，因而文档在数据库中的存储保持了一定的语义信息，存储在同一个表中的数据具有一定程度的关联性，而 Xparent 方法对文档的拆分根据的是 XML 文档本身的结构信息，丝毫没有考虑到 XML 文档的语义信息，因而其在数据库中的存储可以说是碎末状的(fragment)。所以 Xparent 方法在数据查询及更新时要进行更多的连接操作，其查询操作性能自然要低于 S_Schema 方法。

结论 根据 W3C 最近发布的 XML Schema 标准，本文提出了一种等价于 XML Schema 的对 XML 模式进行描述的数据模型 S_Schema，并实现了 S_Schema 到关系模式的自动生成算法及 XML 文档到数据库的加载算法(本文称之为 S_Schema 方法)，实验证明，和其它现存的映射方法相比，S_Schema 方法具有以下一些特点：

1)在 XML 到关系数据库的存储过程中，能够保持文档的完整性，能够从关系数据库中完全恢复文档；

2)支持含有混合类型元素文档和含有递归元素文档的存储和查询；

3)数据查询及更新操作效率优于文本方法和 Xparent 存储方法。

XML Schema 的发布，使得 XML 文档结构更能得到准确有效的定义，为 XML 文档的存储与恢复提供了更为有利的条件。本文提出的基于 S_Schema 的映射方法，不但操作简单，而且也考虑到了 XML Schema 映射中的多值元素、递归元素和可选元素的映射等难点问题，为做进一步的研究提供了方便。

参 考 文 献

1 Extensible Markup Language (XML) 1.0 (Second Edition)[R]. W3C Recommendation 6 October 2000. http://www.w3.org/TR/2000/REC-xml-20001006

2 Florescu D, Kossman D. Storing and Querying XML Data using a RDBMS. [J]. IEEE Data Engineering Bulletin, 1999,22(3):27~34

3 Jiang H, Lu H, Wang W, Yu J X. Path materialization revisited: An efficient storage model for XML data. [C]. In:Proc. of ADC, 2002

4 Deutsch A, Fernandez M, Suciu D. Storing semistructured data with STORED. [C] In:Proc. of the ACM SIGMOD Conf. 1999

5 Shanmugasundaram J, Tufte K, He G, et al. Relational database for querying xml documents: limitations and opportunities [C]. In:Proc. of the 25th VLDB Conf., 1999