基于邻接矩阵的面向对象XML继承关系的研究

周 健[1,2],孙丽艳[1]
(1.安徽财经大学 信息工程学院,安徽 蚌埠 233010;
2.北京科技大学 信息工程学院,北京 100081)

摘 要:面向对象XML提供更加新颖的查询方式,使查询方式智能、效率高效。查询中需要从模式文档获取元素间继承、引用关系,现有方法需要不断遍历冗余、复杂的DOM树,该方法算法设计复杂,而且效率低下。文中利用邻接矩阵提供了一种新颖方法解决了复杂继承关系的判别问题,使得算法设计简单,并提高了效率。

关键词:面向对象XML;邻接矩阵;继承;引用
中图分类号:TP391 **文献标识码**:A **文章编号**:1673-629X(2008)10-0106-04

Study of Inheritance about Object-Oriented XML Based on Adjacency Matrix

ZHOU Jian[1,2], SUN Li-yan[1]
(1. Sch. of Info. Eng., Anhui University of Finance & Economics, Bengbu 233010, China;
2. Sch. of Info. Eng., Beijing University of Science and Technology, Beijing 100081, China)

Abstract: Object-oriented XML supplies a novel method of query that is more intelligence and efficiency. The new method needs to acquire the information about inheritance and reference. A new method is put forward to deal with the question of complex relationship about Inheritance and reference, the method improves efficiency, because DOM tree which is redudant does not use in the process of query again and again.

Key words: object-oriented XML; adjacency matrix; inheritance; reference

0 引 言

XML[1,2](Extended Markup Language)是由嵌套的标记元素构成的自描述标记语言,它正成为Internet上数据表示和交换的主要标准,针对XML数据的存储[3~5]和查询技术[6~11]已成为研究热点。面向对象的方法具有很强的建模能力。如何将面向对象的特征引入到XML中,用以提高XML的建模能力[12,13]和查询功能是一个重要的研究方向。

通过扩展模式语言如XML Schema[14]和DTD[15],使得XML充分支持面向对象概念,如继承[16~18]、引用[15,16,18],并提供更加新颖的查询方式,基于元素之间的继承和引用的面向对象查询方式。这些关系的判断来自扩展DTD或XML Schema的解析,但都不能令人满意。

主要有以下原因:

1)因DOM树中不但有关系信息也有数据,所以DOM[19]树对关系判断是冗余的;

2)关系判断需要在DOM树上不断遍历;

3)面向对象查询方式更多依赖一些间接的关系信息;

4)继承树的冗余与算法复杂性;

5)查询中需要得到继承关系的路径信息。

因此有必要构建一种新方法来解决上述问题。通过邻接矩阵获取间接关系,避免了DOM树的重复查询,可以有效提高查询效率并且改善了查询方法。

关系的获取需满足:

(1)能判断直接和间接的继承、引用关系;

(2)能够得到继承、引用关系的关系路径长度;

(3)能够得到继承、引用关系的关系路径的个数;

(4)能够判断元素层次之间是否有引用关系;

(5)有效提高元素间继承、引用关系判断效率;

(6)能判断由多态性而引起的间接引用关系。

收稿日期:2008-01-25
基金项目:安徽省科研计划项目资助(2007jq1084)
作者简介:周 健(1979-),男,安徽凤阳人,博士研究生,讲师,研究方向为XML数据库、网络安全;导师:张晓琳,博士,教授,研究方向为XML数据库、数据流、数据挖掘。

1　面向对象 XML

XML 是用于 Web 信息交换的层次数据,而 XML 文档是由一系列嵌套结构构成的,元素由其子元素构成。面向对象 XML 含有元素层次、继承、多重继承的信息。XML 文档中结构的信息由扩展模式语言来进行描述,包括为文档提供一个结构框架;为元素定义一个内容模型、数据类型和数据约束等定义。扩展模式语言描述了面向对象的一些重要概念,例如:元素继承层次、多重继承等信息。XML 数据是 XML 模式的一个实例,一个 XML 模式可以对应多个 XML 文件。遵循 XML 核心规则的文档是良构文档,遵循 XML 核心规则并符合指派给该文档的模式语言的文档是有效文档。

文中讨论的 XML 数据都假设是遵循扩展模式语言的良构 XML 文档[20]。

图 1 中给出一个大学信息 univ, person、student、teacher 和 TA 构成一个元素层次, course、gradCourse、underCourse 构成另一个元素层次。元素层次指在一个元素层次中,一个子元素从它的超元素继承元素和属性,并有自己定义的元素和属性。元素 person 有一个 ID 属性和一个子元素:@pid、name。元素 student 继承元素 person 的属性和子元素,并且有自己定义的属性和子元素:@sno、addr、taken,这些元素属性称为独有属性。元素 teacher 继承元素 person 的属性和子元素,并且有自己定义的属性和子元素:@tno、phone、taught。TA 是 student 和 teacher 的子元素,继承 person、student 和 teacher 的属性和子元素,是多重继承,没有独有属性。多态性是指一个元素具有外延性,由于多态性,persons 包含类 person, teacher、student 和 TA 的对象。引用是在不同元素层次下,元素和元素之间产生关系。如学生可以选择几门课程,一门课程可以被几个学生选择,关系产生在元素层次分别为 person 和 course。元素层次 course 和 person 一样,在这里不做详细介绍。

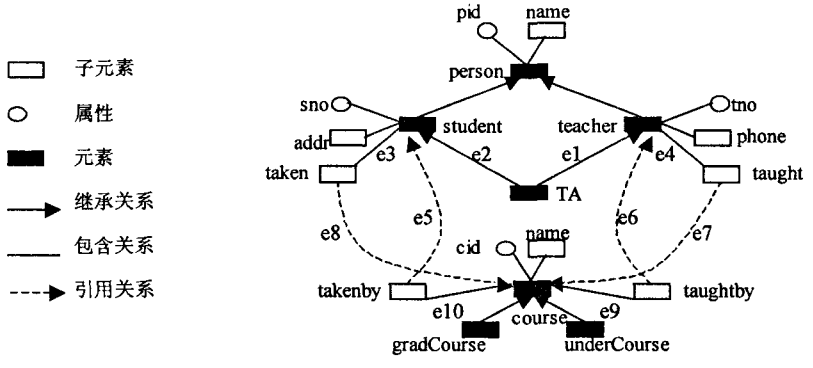

图 1　元素层次示例

2　面向对象 XML 查询方式

给出两个面向对象查询方式的特例,说明继承关系和引用关系判断的重要性。有时用户需要查询一个元素及其所有元素的信息,例如查询全部人的信息,包括:person, student, teacher 和 TA。这个查询能被表示为对四个元素的查询,但不简练并且性能差,为了解决这个问题,提出了包含元素的概念,利用元素之间的多重继承[17]。

查询语句 1:查询全部人的名字,包括 person、student、teacher 和 TA。

```
Querying
(http://www.univ.cn/univ.xml)
  /univ/person↓:(pid: $p, name: $n)
Constructing
(http://www.univ.cn/result.xml)
```

查询语句 2:查询教师"Alley Srivastava"教授的全部 course,包括 undercourse 和 gradeCourse。

```
Querying
(http://www.univ.cn/univ.xml)//TA: $a,
  &a/name: $n(="Alice Bumbulis"),
  &a/teacher/@(course↓)course: $c,
  (http://www.univ.cn/univ.xml)//course↓: $u,
    $u/cid: $d(= $c),
    $u/name: $b
Constructing
(http://www.univ.cn/result.xml)/univ/course: $b
```

在上面查询语句中,"$a/teaches/@(course↓)courses"得到 course 和它的子元素 underCourse 和 gradCourse 的全部 ID。从上文的查询看出,扩展后的面向对象查询方式需要在查询引擎中具有判断元素之间是否有继承和引用关系的机制。

3　面向对象 XML 的矩阵定义

定义(1):一个 $m \times m$ 阶的矩阵 $\boldsymbol{A}$,其元素 a_{ij},赋值为正整数和零。当节点 n_i 与节点 n_j 具有关系时,$a_{ij} > 0$,否则 $a_{ij} = 0$,称矩阵 $\boldsymbol{A}$ 为面向对象 XML 继承关系的邻接矩阵。矩阵的行标题(e_i)和列标题(d_j)为元素名。

定义(2):面向对象 XML 的初始邻接矩阵多次自乘的结果为 $\boldsymbol{A}^i(i \neq 0)$,其元素 a_{ij},赋值为正整数或零。当节点 n_i 与节点 n_j 具有间接关系时,$a_{ij} > 0$,否则 $a_{ij} = 0$,这个矩阵 $\boldsymbol{A}^i$ 就称之为面向对象 XML 继承关系的第 i 次邻接矩阵。

定义(3):在面向对象 XML 中,某个元素与另一个元素之间具有直接或间接

关系,若两点之间是连通的,则连接两点之间的点线集合为一条关系路径。关系路径可以是多条,关系路径可以交叉,关系路径不允许出现有向环。

定义(4):一个面向对象 XML 中,某个元素与另一个元素之间具有间接关系,关联两者之间的路径不只一条,具有 K 条关系路径,称两元素之间具有 K 条关系路径。

定义(5):一个面向对象 XML 中,某个元素与另一个元素之间具有直接或间接关系,且连接这两个元素之间的关系路径包含边数为 S,称 S 为两元素之间的关系长度,为整数。若 $S=1$,则两元素之间有直接关系,若 $S>1$ 则两元素有间接关系。

公式(1):$\boldsymbol{H}=\sum_{i=1}^{n-1}\boldsymbol{A}^i$ 为关系矩阵的和,n 为一个元素层次下元素个数,$\boldsymbol{A}$ 为初始邻接矩阵。

公式(2):$\boldsymbol{U}=\sum_{i=1}^{n-1}\boldsymbol{A}^I * m^i$ 为带有关系路径长度和关系路径数量信息的邻接矩阵的和,n 为一个元素层次下元素个数,$\boldsymbol{A}$ 为初始邻接矩阵,m 是代表路径长度信息的权位。

4 继承、引用关系判断

面向对象 XML 中的直接继承关系可以从初始邻接矩阵中直接获得,如图 2(a)中矩阵 $\boldsymbol{A}$ 中元素 a_{ij} 为非零值,表明对应的两个元素之间有直接继承关系。如果 a_{ij} 值为零,说明两元素之间没有直接继承关系,但不表明没有间接继承关系。

$$\boldsymbol{A}=\begin{pmatrix}0&1&1&0\\0&0&0&1\\0&0&0&1\\0&0&0&0\end{pmatrix}\quad \boldsymbol{A}^2=\begin{pmatrix}0&0&0&2\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix}\quad \boldsymbol{H}=\begin{pmatrix}0&1&1&2\\0&0&0&1\\0&0&0&1\\0&0&0&0\end{pmatrix}$$

(a)初始矩阵 (b)二阶矩阵 (c)多次矩阵和

图 2 面向对象 XML 邻接矩阵

依靠关联矩阵获取间接继承关系是不够的,因此需依靠关联矩阵的多次自乘结果。图 2(b)矩阵 $\boldsymbol{A}^2$ 为两个初始邻接矩阵的乘积,其中元素 $a_{14}=2$,说明元素 person 和 TA 之间有继承关系,但 $a_{12}=0$,$a_{13}=0$ 说明元素 person 和元素 teacher、student 之间的继承关系丢失了。保存元素之间的所有继承关系,需要使用公式(1)将邻接矩阵和邻接矩阵的所有自乘结果相加,图 2(c)中矩阵 $\boldsymbol{H}$ 为矩阵 $\boldsymbol{A}$ 和矩阵 $\boldsymbol{A}^2$ 的和,很容易地判断元素之间的直接和间接关系。

矩阵 $\boldsymbol{A}^i$ 的阶数 i 代表继承关系的路径长度,也就是关系长度,$\boldsymbol{A}^2$ 的阶数为 2,说明元素 person 和 TA 之间的关系长度为 2,$a_{14}=2$ 说明有 person 和 TA 之间的继承关系有两条不同的关系路径,图 1 中 person → student → TA、person → teacher → TA 为两条不同的关系路径,都说明了 person 和 TA 之间的间接继承关系。数值越大继承关系越复杂,即数值大于 1 存在多重继承。存在两个邻接矩阵,$\boldsymbol{A}_i$、$\boldsymbol{A}_j$($i\neq j$)如果在面向对象 XML 中存在两个不同元素,并在这两个矩阵中对应的 a_{ij} 不为零,或在一个矩阵中对应两个元素的关系 a_{ij} 的值大于 1,则说明两元素之间具有多个关系路径。但存在关系路径长度丢失问题,提出公式(2),公式(2)是公式(1)的改进,具有权值 m(可为 16、10、8、2 等),如矩阵 $\boldsymbol{A}^2$ 和 $\boldsymbol{A}^4$ 中 a_{14} 元素都不为零,设分别为(2、2),权值为 10,则矩阵 $\boldsymbol{H}$ 中 $a_{14}=2*10^2+2*10^4=20200$,不仅表明了元素之间存在继承关系,而且也包含了继承信息来自两个矩阵,继承关系的路径为 4 条。可以保存每个阶数下的路径个数,而矩阵阶数是路径长度。图 2(c)中矩阵 $\boldsymbol{H}$ 由公式(1)得到($\boldsymbol{H}=\boldsymbol{A}^1+\boldsymbol{A}^2$),非零元素表明对应的行和列元素具有继承关系。

引用关系发生在层次间,因此判断层次之间的引用关系,是把有向的面向对象 XML 的关系图变无向图,判断图中是否存在有环结构,如果有环结构则说明层次之间具有引用关系,如图 1 所示,将该图变为无向图,则图中有两个环(student,taken,takenby,course)和(teacher,taught,course,taught),环结构的存在说明图中具有引用关系。使用关联矩阵去除一行所得到的关联阵来判断,如果该关联阵中存在一个$(n-1)(n-1)$方阵是奇异的,即它的行列式值为零,则方阵所对应的边集中必存在环,因为形成环的边所对应的列矢量必线形相关,将使行列式值为零。非奇异的方阵所对应的边中不存在环,也就是不存在引用。

元素 teacher 和子元素 taught,元素 student 和子元素 takenby,子元素 taken 和元素 course,子元素 taught 和 course 存在直接的引用关系。由于元素存在多态性。所以元素 person 和元素 course 之间存在间接的引用关系。在间接的引用关系中存在着多个继承关系和引用关系。用一个邻接矩阵保存面向对象 XML 中所有直接引用关系。矩阵的横坐标为 taken、taught、takenby、taughtby,纵坐标为 teacher、stduent、course,从矩阵 $\boldsymbol{E}$ 中可以得到两个层次之间具有直接的继承关系,可通过矩阵的自乘,得到间接的层次关系。判断间接引用关系,还需要注意继承关系的作用,例如 person 和 course 之间没有直接的继承关系,但一个 student 具有多态性,因此 person 和 course 之间具有间接的引用关系,这样就需要首先判断他们之间是否有继承关系,再判断 student 和 course 之间的引用关系。在判断元素之间的引用关系时,通过矩阵 $\boldsymbol{E}$ 判断是否存在直接的引用关系,如不存在,再通过继承关系的矩阵判断是否

有继承元素,最后再搜索矩阵 E(如图 3 所示)。

	ta	student	teacher	taken	taught	takenby	taughtby	course
e1	-1	0	1	0	0	0	0	0
e2	-1	1	0	0	0	0	0	0
e3	0	1	0	-1	0	0	0	0
e4	0	0	1	0	-1	0	0	0
e5	0	-1	0	0	0	1	0	0
e6	0	0	-1	0	0	0	1	0
e7	0	0	0	0	1	0	0	-1
e8	0	0	0	1	0	0	0	-1
e9	0	0	0	0	0	0	-1	1
e10	0	0	0	0	-1	0	0	1

	ta	student	teacher	taken	taught	taken by	taughtby	course
e3	0	1	0	-1	0	0	0	0
e5	0	-1	0	0	0	1	0	0
e8	0	0	0	1	0	0	0	-1
e10	0	0	0	0	-1	0	0	1

图 3　层次关系的关联矩阵

5　测试结果

硬件环境为:CPU 为奔腾 2.0GHz、512M 内存、120G 硬盘的兼容机(2M 缓存)。软件环境为:操作系统 win2000(SP1)、Java 软件包 SDK1.4.0 版本、XML4J-bin.4.3.0。文档结构为两层的嵌套级别为 5 级的中等规模 XML 文档,共分成 10 组,第一个文档为 2.15M,共有两个层次 7 个类,每个类有 1000 个对象,对象大小为中等程度,嵌套级别为 1～4 级,此后每个文档的每个类每次增加 1000 个对象,处理的时间基本按照文档大小增加而线形增加。由于 XML 文档解析成 DOM 树的时间会随 XML 文档的大小而增加,因此文档越大,处理的时间会显著增加,在实验中分别使用邻接矩阵和不使用邻接矩阵建立分布和集中式存储。实验表明(见图 4)建立分布式存储模式中利用邻接矩阵优于使用 DOM,优化程度明显,优化程度取决于继承和引用的复杂程度,但对于集中式存储作用不大,因集中式存储建立较少依赖模式文档。

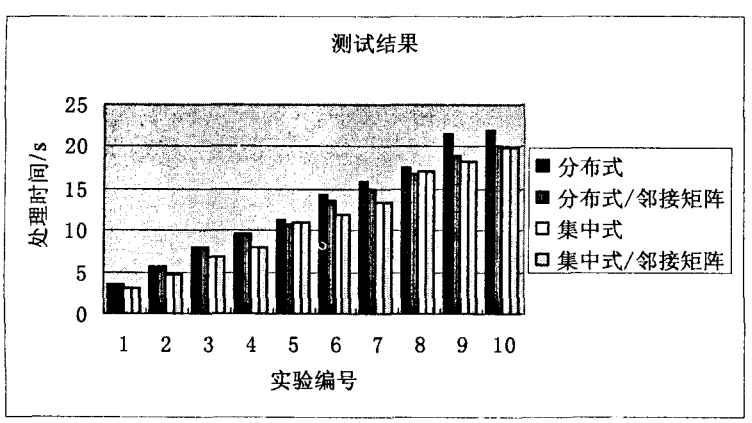

图 4　测试结果

6　结束语

使用关联矩阵判断面向对象 XML 中直接和间接的继承、引用关系,简化 DOM 树遍历,有效解决查询中关系判断问题,并用于存储模式的建立,实验结果也证明了该方法的有效性。该方法使查询语言具有更好的查询效率,查询功能更丰富,推动在线查询的人性化、智能化的趋势。在下一步工作中将研究矩阵方程的简化,以便于在较大模式文档中使用。

参考文献:

[1] DeRoses C J. XML pathlanguage(XPath) version1.0[EB/OL]. 2001-11. http://www.w3.org/tr/1999/RECx-Path19991116.

[2] Extensible Markup Language (XML) 1.0 (Third Edition) [EB/OL]. 2004-04. http://www.w3.org /TR/REC-xml/.

[3] Jagadish H V, Al-khalifa S, Chapman A, et al. A native XML database[J]. The International Journal on Very Large Data Bases, 2002, 11(4):274-291.

[4] Luo Cheng, Jiang Zhewei, Hou Wen-Chi, et al. A relational model for XML structural joins and their size estimations [R]. London:[s.n.], 2007.

[5] 孙　伟, 刘大昕. 一种 XML 代数及其查询优化方法[J]. 哈尔滨工程大学学报, 2007, 28(8):899-904.

[6] Chamberlin D, Florescu D, Robie J. XQuery: aquery languge for XML[EB/OL]. 2001-02. http://www.w3.org/tr/2001/wd xquery 20010215.

[7] Liu M, Ling T W. Towards declarative XML querying[C]// Proceedings of WISE 2002. Singapore: IEEE Computer society, 2002:127-138.

[8] Goldman R, Widom J. DataGuides: enabling query formulation and optimization semistructured database[C]// The 23rd VLDB Conf. Athens:[s.n.], 1997:436-445.

[9] Li Q, Moon B. Indexing and querying XML data for regular path expressions[C]// the 27th VLDB Conf. Roma: Morgan Kaufman, 2001: 361-370.

[10] Grus T. Accelerating xpath location steps[C]// Proc of the 21st ACM SIG MOD conf. Madison: [s.n.], 2002:109-120.

[11] 孔令波, 唐世渭, 杨冬青, 等. XML 数据的查询技术[J]. 软件学报, 2007, 18(6):1399-1418.

[12] Cooper B, Sample N, Franklin M J, et al. A fast

(下转第 112 页)

$\max(O_1^k * s,(1-O_1^k) * s)$, $O_2^k = O_1^k, O_3^k = O_1^k, \cdots, O_n^k = O_1^k$;设 E_i^k 为第 k 次迭代后各网络的最小误差,令 $W_1^k = W_i^k, \Delta W_1^k = \Delta W_i^k, \cdots, W_{i-1}^k = W_i^k, \Delta W_{i-1}^k = \Delta W_i^k, W_{i+1}^k = W_i^k, \Delta W_{i+1}^k = \Delta W_i^k, \cdots, W_n^k = W_i^k, \Delta W_n^k = \Delta W_i^k$($W_i^k$ 表示第 k 次迭代后网络 i 的权值), $O_1^k = O_i^k, \cdots, O_{i-1}^k = O_i^k, O_{i+1}^k = O_i^k, \cdots, O_n^k = O_i^k$。

(5)通过如下公式分别对各网络进行权值调整,并记下权值调整量 ΔW_i^{k+1}:

$$\Delta W_1^{k+1} = -\eta_1 \delta O_1^k + \alpha \Delta W_1^k$$

$$\Delta W_2^{k+1} = -\eta_2 \delta O_2^k + \alpha \Delta W_2^k$$

……

$$\Delta W_n^{k+1} = -\eta_n \delta O_n^k + \alpha \Delta W_n^k$$

令 $k = k + 1$,返回步骤(3)。

(6)若 E_i 为到目前为止网络的最小误差,则以网络 i 作为训练完成的网络,删除其它网络。训练结束。

4 仿真实验与结果

通过在 Matlab 中编程对本算法进行了测试。异或问题是典型的线性不可分问题,常用于网络算法的验证,因此文中用它进行仿真实验。构建三个相同结构的三层网络,输入层、隐含层、输出层神经单元个数分别为 2、2、1。输入为{0,0}、{0,1}、{1,0}、{1,1},对应的目标输出为 {0}、{1}、{1}、{0}。随机对权值进行初始化,网络的学习速率分别为 0.16、0.12、0.05,动量系数 α 取 0.91,误差精度为 1e-06,最大迭代次数为 500 次。由于权值为随机选取,因此对网络进行了多次训练,求得平均迭代数,实验结果如表 1 所示。

用带动量 BP 算法进行网络训练,当 η 取 0.2 及 0.12 时,误差曲面经常产生振荡且容易产生平坦区;当 η 取 0.05 时,收敛速度明显减慢。用基于并行策略的 BP 算法(PBBP)进行训练,η_1、η_2、η_3 分别取 0.2、0.12、0.05,误差曲面趋于平滑,很少有平坦区出现,收敛速度很快。结果表明,基于并行策略的 BP 算法能够有效加快网络的收敛速度,避免平坦区引起的迭代次数增加。

表 1 算法执行结果的对比

算法	η 值	平均迭代次数	误差
基于并行策略 BP 算法	(0.2,0.12,0.05)	33	4.77531e-07
带动量 BP 算法	0.2	154	6.36393e-07
带动量 BP 算法	0.12	110	9.51818e-07
带动量 BP 算法	0.05	207	7.60022e-07

5 结束语

基于并行策略的 BP 算法通过对多个采用不同学习速率的网络同时训练,在训练过程中取长补短,充分发挥了学习速率变化对加快网络学习速度的作用,但是也有一定的缺陷,即不同的问题采用多少个网络进行训练合适及学习速率的取值都没有理论依据,只能通过实验来确定。

参考文献:

[1] Rumelhart D E, Hinton G E, Williams R J. Learning representations by back - propagating errors[J]. Nature, 1986, 323(9):533-536.

[2] Rumelhart D E, McClelland J L. Parallel Distributed processing, Explorations in the Microstructure of Cognition, Vol. 1: Foundations[R]. Cambridge, MA: MIT press, 1986.

[3] Jacobs R A. Increased rates of convergence through learning rate adaptation[J]. Neural Networks, 1988(1):295-307.

[4] 黄德双. 神经网络模式识别系统理论[M]. 北京:电子工业出版社,1996.

[5] 刘希玉,刘 宏. 人工神经网络与微粒群优化[M]. 北京:北京邮电大学出版社,2006.

[6] 周志华,曹存根. 神经网络及其应用[M]. 北京:清华大学出版社,2004.

(上接第 109 页)

index for semi structured database[C]// The 27th VLDB Conf. Roma: Morgan Kaufmann, 2001:341-350.

[13] Zou Q, Liu S, Chu W. Ctree: a cmpact tree for indexing XML data[C]// the 6th ACM WI DM 04. New York: [s. n.], 2004:39-46.

[14] XML Schema 1.1 specification[EB/OL]. 2000-04. http://www.w3.org/XML/Schema.

[15] Java DTD Parser[CP/OL]. 2004-07. http://www.wutka.com/dtdparser.html.

[16] Wang G, Liu M. Extending XML Schema with Nonmonotonic Inheritance[C]// In Proceedings of 1st International Workshop on XML Schema and Data Management (ER Workshop XSDM'03). Chicago, Illinois, USA: [s. n.], 2003.

[17] 张晓琳,王国仁. 用继承扩展 XML-RL[J]. 小型微型计算机系统,2005,26(2):243-247.

[18] 张晓琳,谭跃生,周 健. 用继承扩展 XML Schema[J]. 计算机工程与应用,2006,42(4):179-182.

[19] Document Object Model (DOM Level 3 Core)[EB/OL]. 2002-04. http://www.w3.org/DOM/.

[20] 张晓琳,王国仁. 面向对象的 XML 数据的存储模式研究[J]. 小型微型计算机系统,2004,25:11-13.