

不确定 XML 序列匹配关键技术研究

张晓琳, 王 鹏

(内蒙古科技大学 信息工程学院, 内蒙古 包头 014010)

摘 要: 不确定 XML 小枝模式匹配的关键技术包括不确定 XML 文档格式, 互斥节点识别, 实例树概率计算和概率阈值过滤。目前, 不确定 XML 二元结构连接和整体匹配方法无法充分利用算法的特点解决关键技术问题, 针对这一不足, 将序列匹配应用到不确定 XML 小枝模式匹配, 并对不确定 XML 序列匹配关键技术问题进行分析 and 证明, 理论分析和实验结果表明序列匹配方法应用到不确定 XML 小枝模式匹配是有效的。

关键词: 不确定 XML; 小枝模式匹配; 序列匹配; 概率阈值过滤

中图分类号: TP392

文献标识码: A

文章编号: 1000-7180(2014)04-0123-05

Research on Key Technologies of Uncertain XML Sequence-Based Matching

ZHANG Xiao-lin, WANG Peng

(School of Information Engineering, Inner Mongolia University of Science and Technology, Baotou 014010, China)

Abstract: Key technologies of uncertain XML twig pattern matching include uncertain XML document format, recognition of exclusive distribution nodes, calculation of the probabilities of witness tree and probabilistic threshold filtering. At present, binary structure join and holistic matching over uncertain XML can't make full use of the characteristics of algorithms to solve key technology issues. In order to overcome this limitation, this paper proposes an uncertain XML twig pattern matching method based on sequence-based matching. We also analyze and prove key technologies of uncertain XML sequence-based matching. The theory research and the results of experiment show the effectiveness of this method.

Key words: uncertain XML; twig pattern matching; sequence-based matching; probabilistic threshold filtering

1 引言

从 1980 年代末开始, 针对概率数据库的研究工作就从未间断, 这类研究工作主要将不确定性引入到关系数据模型中去, 取得了较大研究进展^[1]。但关系数据库是结构化的, 并不适合描述复杂的不确定性数据, 因此利用 XML 来管理不确定性数据已经成为当前的趋势。

不确定 XML 文档与普通 XML 文档相比, 具有概率信息, 需要额外处理包括不确定 XML 文档格式, 互斥节点识别, 实例树概率计算和概率阈值过滤

在内的关键技术问题。将二元结构连接应用到不确定 XML 查询的方法^[2]会产生过多的中间结果而影响查询效率。将整体匹配应用到不确定 XML 查询的方法^[3-5]由于算法内聚而不方便进行实例树概率计算和概率阈值过滤。序列匹配是将查询转化为序列匹配来处理, 可以避免繁复的结构连接操作。

本文将基于序列匹配的 LCS-TRIM 算法^[6]应用到不确定 XML 小枝模式匹配, 并对不确定 XML 序列匹配的关键技术问题进行分析 and 证明, 所做的具体工作如下: (1) 提出一种不确定 XML 文档格式, 对应的 p-文档模型属于非 DHF (Distributional

收稿日期: 2013-05-30; 修回日期: 2013-07-17

基金项目: 国家自然科学基金项目(61163015); 内蒙古自然科学基金项目(2013MS0909)

Hierarchy Free)^[7]; (2) 提出一种互斥节点识别编码 (REDN 编码); (3) 提出一种实例树概率计算方法; (4) 提出三次概率阈值过滤策略。

2 关键技术问题解决方案

2.1 不确定 XML 文档格式

目前不确定 XML 文档还没有符合 p-文档模型的统一格式。只有文献[8]提出一种不确定 XML 文档格式,但其逻辑结构与 p-文档模型还有差异。本文通过对普通 XML 文档的 DTD 进行扩展,提出一种符合 p-文档模型的不确定 XML 文档格式。DTD 扩展规则如下:

(1) 定义概率分布节点: $\langle ! \text{ELEMENT Dist} (\text{Dist}^*, (\text{element } 1 | \text{element } 2 | \text{element } 3 \dots | \text{element } n-1) +) \rangle$ 。此处 Dist 节点下可以有 0 或多个 Dist 节点说明新提出的不确定 XML 文档格式对应的 p-文档模型属于非 DHF。

(2) 为 Dist 节点定义类型(type)属性: $\langle ! \text{ATTLIST Dist type} (\text{ind} | \text{mux}) " \text{ind} " \rangle$, 默认值为 ind。

(3) 为所有节点定义概率属性: $\langle ! \text{ATTLIST element Prob CDATA "1.0"} \rangle$, 默认值为 1.0。

(4) 对于非叶子元素 $\langle ! \text{ELEMENT curElement} (\text{prev-def}) \rangle$ 转换为 $\langle ! \text{ELEMENT curElement} ((\text{prev-def}) | \text{Dist}) \rangle$, 其中 prev-def 是当前元素原始的定义。

文献[8]中不确定 XML 文档截取 28~35 行如图 1(a)所示,根据 DTD 扩展规则(1)~(4),图 1(a)所示不确定 XML 文档可以用图 1(b)所示新的不确定 XML 文档格式表示。

```

28.      <age>
29.      <Dist type="mutually-exclusive">
30.      <Val Prob=".2">54</Val>
31.      <Val Prob=".35">55</Val>
32.      <Val Prob=".1">56</Val>
33.      <Val Prob=".15">77</Val>
34.      </Dist>
35.    </age>

```

(a) 已有不确定 XML 文档格式

```

<Dist type="mux">
  <age Prob="0.2">54</age>
  <age Prob="0.35">55</age>
  <age Prob="0.1">56</age>
  <age Prob="0.15">77</age>
</Dist>

```

(a) 新不确定 XML 文档格式

图 1 不确定 XML 文档片段

如图 1(b)所示,新的不确定 XML 文档格式是

将元素与文本值看做一个整体,例如元素 age 与文本 54,55,56,77。图 1(a)所示不确定 XML 文档需要 6 个元素(1 个 age,1 个 Dist,4 个 Val),而图 1(b)所示不确定 XML 文档只需 5 个元素(1 个 Dist,4 个 age)就可以存储同样的不确定性数据。

2.2 互斥节点的识别

具有互斥关系的节点是指同一个 mux 分布节点下不同分支的节点,在查询结果实例树中互斥节点不能同时存在,需要提前识别出来,可以用一个编码记录文档中每一个普通节点的互斥信息。

定义 1 (互斥节点识别编码, Labeling Scheme for Recognition of Exclusive Distribution Nodes, 记为 REDN 编码) REDN 编码的数据结构是一个浮点数数组,数组中每一个元素由两部分组成,整数部分为 mux 节点在文档树中的后序遍历编号,小数部分为节点所在子树根节点在文档树中的后序遍历编号。

定理 1 当且仅当任意两个节点的 REDN 编码中存在这样两个浮点数,节点为互斥关系:整数部分相同,小数部分不同。

证明 如图 2(a)所示,节点 B 可以代表一个节点,还可以代表集合 B, $B = \{\text{以节点 B 为根的子树中所有节点}\}$, 节点 C 同理。因为集合 B 中任意一个节点与集合 C 中任意一个节点都具有互斥关系,所以可以把集合 B 抽象为一个节点 B,把集合 C 抽象为一个节点 C,那么就可以用节点 B 和节点 C 来标记子树中的所有节点。(证毕)

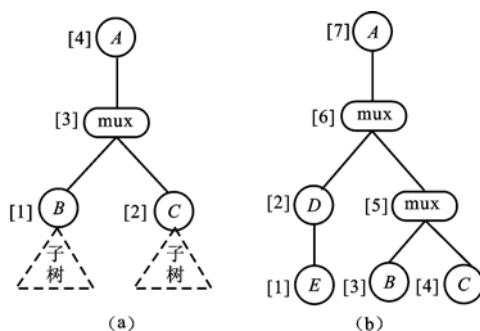


图 2 互斥关系节点示意

如图 2(b)所示,节点 B 的 REDN 编码为 {6.5, 5.3}, 节点 C 的 REDN 编码为 {6.5, 5.4}, 两个节点的 REDN 编码中有这样两个浮点数: 5.3 和 5.4, 整数部分相同为 5, 小数部分不同, 分别为 0.3 和 0.4, 因此互斥节点 B 和 C 可以借助 REDN 编码来识别。

2.3 实例树概率计算

基于序列匹配的 LCS-TRIM 算法在结构过滤阶段需要遍历实例树中所有节点,这是 LCS-TRIM 算法自身特点决定的,而不是为了计算实例树的概率而去额外遍历所有的节点,因此序列匹配可以将查询过程与实例树概率计算过程巧妙地结合到一起。

算法:实例树概率计算

Input:代表实例树的 n 元组

Output:实例树概率值

1. $p = \text{the Eprob of pattern tree'root}$ //实例树概率初始值为实例树根节点的存在概率
2. if (node not in Memo[])//Memo 为备忘数组
3. $p = p * \text{node.Cprob}$ //Cprob 为条件概率
4. Memo[] $\leftarrow$ node
5. else
6. do others
7. end if
8. return p //返回实例树概率值

这里需要用一个备忘数组 Memo 记录已参加过概率计算的节点,避免重复计算. 因为序列匹配方法中实例树概率计算是与 LCS-TRIM 算法结构过滤同时进行的,所以如图 3 所示会有两种情况导致重复计算:图 3 中 (a) 为模式树, (b) 为匹配到的实例树,对节点 A, C 进行结构过滤时会计算节点 D 的条件概率 0.3, 对节点 A, B 进行结构过滤时同样会计算节点 D 的条件概率 0.3. 这样节点 D 会被计算两次,从而导致实例树概率计算错误,因此需要采用备忘的方式设置一个数组存储已经参加过概率计算的节点,避免重复计算. 以此类推,图 3(c) 和图 3(d) 的情况也可以利用备忘数组解决。

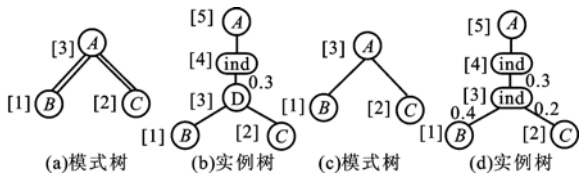


图3 导致重复计算的情况

2.4 概率阈值过滤

概率阈值是查询时给出的一个概率值,意义是要求查询结果实例树的概率值都大于等于给出的概率阈值,这一点可以利用概率阈值过滤策略来实现. 在 LCS-TRIM 算法中可以加入三次概率阈值过滤

来确保查询的结果符合概率阈值,同时又能提高查询的效率。

定义 2 (第一次概率阈值过滤, Probabilistic Threshold Filtering I, 记为 PTF I) 在 LCS-TRIM 算法子序列匹配过程中只选取存在概率大于等于概率阈值的节点。

定义 3 (第二次概率阈值过滤, Probabilistic Threshold Filtering II, 记为 PTF II) 在 LCS-TRIM 算法结构过滤过程中只选取根节点的存在概率与非根节点的条件概率乘积大于等于概率阈值的 n 元组。

定义 4 (第三次概率阈值过滤, Probabilistic Threshold Filtering III, 记为 PTF III) 在 LCS-TRIM 算法结构过滤过程中每当匹配到一个条件概率不等于 1 的节点, 就将新计算得到的当前实例树概率值与概率阈值进行比较, 一旦小于概率阈值, 立即终止结构过滤。

定理 2 PTF I, PTF II, PTF III 是有效的。

证明 如图 4 所示, 设集合 $A = \{\text{能够被 PTF I 过滤的 } n \text{ 元组}\}$, 集合 $B = \{\text{能够被 PTF II 过滤的 } n \text{ 元组}\}$, 集合 $C = \{\text{能够被 PTF III 过滤的 } n \text{ 元组}\}$, $A \cap B = \text{I}$, $A \cap C = \text{II}$, $B \cap C = \text{III}$. Eprob 为节点的存在概率, Cprob 为节点的条件概率, threshold 为概率阈值, 大括号联立的不等式之间为“或”的关系。

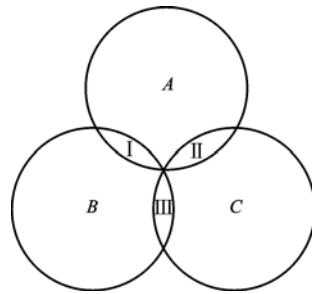


图4 三次概率阈值过滤的关系

$A \cap B = \text{I}$ 和 $A \cap C = \text{II}$ 的情况如图 5(a) 所示, 这时 n 元组被 PTF I 过滤掉的条件是式(1), n 元组被 PTF II 过滤掉的条件是式(2)。

$$\begin{cases} R. \text{Eprob} < \text{threshold} \\ n_1. \text{Eprob} < \text{threshold} \\ n_2. \text{Eprob} < \text{threshold} \end{cases} \quad (1)$$

$$R. \text{Eprob} * n_1. \text{Cprob} * n_2. \text{Cprob} < \text{threshold} \quad (2)$$

式(2)左边乘以 $n_3. \text{Cprob}$ 得到 $n_2. \text{Eprob}$, $n_2. \text{Eprob} < \text{threshold}$ 是 PTF I 中的一个条件(式

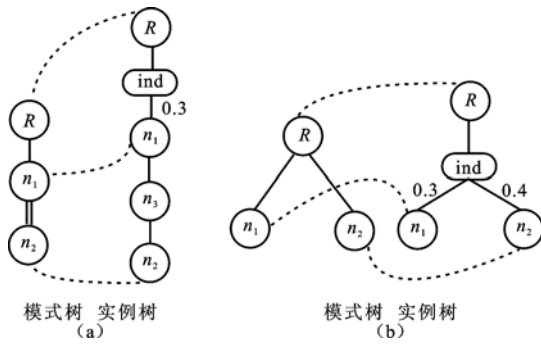


图5 三次概率阈值过滤冗余情况

(1). n 元组被 PTF III 过滤掉的条件是式(3).

$$\begin{cases} R.Eprob * n_1.Cprob < threshold \\ R.Eprob * n_1.Cprob * n_3.Cprob < threshold \\ R.Eprob * n_1.Cprob * n_3.Cprob * n_2.Cprob < threshold \\ Cprob < threshold \end{cases} \quad (3)$$

而 $R.Eprob * n_1.Cprob = n_1.Eprob, R.Eprob * n_1.Cprob * n_3.Cprob = n_2.Eprob, R.Eprob * n_1.Cprob * n_3.Cprob * n_2.Cprob = n_2.Eprob$, $n_1.Eprob < threshold$ 和 $n_2.Eprob < threshold$ 是 PTF I 中的条件(式(1)). 所以当 n 元组 $\in I \cup II$ 时, 无需进行 PTF II 和 PTF III. 因为在这种情况下 PTF I 已经包含 PTF II 和 PTF III, PTF II 和 PTF III 成为冗余操作.

$B \cap C = III$ 的情况如图 5(b) 所示. 这时 n 元组被 PTF II 过滤掉的条件是式(4), n 元组被 PTF III 过滤掉的条件是式(5).

$$R.Eprob * n_1.Cprob * n_2.Cprob < threshold \quad (4)$$

$$\begin{cases} R.Eprob * n_2.Cprob < threshold \\ R.Eprob * n_2.Cprob * n_1.Cprob < threshold \end{cases} \quad (5)$$

式中 $R.Eprob * n_2.Cprob$ 是 PTF II 条件的子集(式(4)), $R.Eprob * n_2.Cprob * n_1.Cprob < threshold$ 是 PTF II 的条件(式(4)), 所以当 n 元组 $\in III$ 时, 无需进行 PTF III. 因为在这种情况下 PTF II 已经包含 PTF III, PTF III 成为冗余操作.

综上所述, PTF II 和 PTF III 只有在某些特殊的情况下(I、II、III)才会变成冗余操作, 否则三次概率阈值过滤都能发挥各自的作用. 由此可见, 在查询中加入三次概率阈值过滤是有效的.(证毕)

3 实验

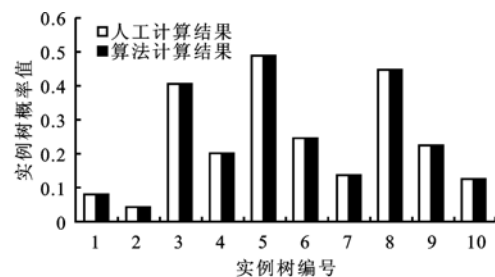
实验中的程序采用 Java 语言编写, 实验环境为 PC 机, 实验所采用的数据集是在 SIGMOD Record 数据集^[9]结构基础上随机加入概率分布节点以及概

率值人工合成的不确定数据集. 原始的 SIGMOD Record 数据集具有元素 11 526 个, 属性 3 737 个, 文档最大深度为 6, 平均深度 5.141 07. 实验所用的查询如表 1 所示. 实验中将基于序列匹配的 LCS-TRIM 算法应用到不确定 XML 小枝模式匹配, 对实例树概率计算的准确性, 概率阈值过滤的准确性以及概率阈值过滤的效果进行验证.

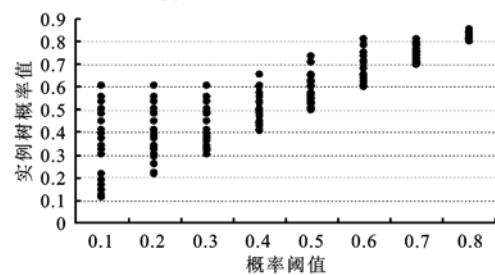
使用查询语句 Q4, 设置概率阈值为 0, 选取大小为 174 kB 的文档, 随机选取 10 棵实例树人工计算其概率值, 然后与算法计算的结果进行对比. 实验结果如图 6(a) 所示, 算法计算的结果与人工计算的结果完全一致, 实验结果表明在结构过滤同时进行实例树概率的计算是有效的.

表1 实验所用查询

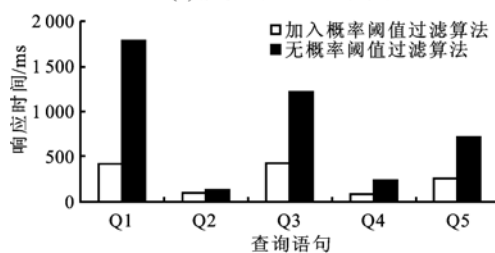
查询编号	查询表达式
Q1	SigmodRecord//articles//author
Q2	SigmodRecord[volume]/number
Q3	SigmodRecord[number]//title
Q4	SigmodRecord[volume=14]//articles
Q5	//articles//title



(a) 实例树概率计算的准确性



(b) 概率阈值过滤的准确性



(c) 概率阈值过滤的效果

图6 实验结果

使用查询语句 Q3,分别设置概率阈值为 0.1~0.8,选取大小为 174 kB 的文档,随机选取 20 棵实例树由算法计算其概率值.实验结果如图 6(b)所示,不同概率阈值对应的代表实例树概率值的点均投射在大于等于概率阈值的区域中,实验结果表明概率阈值过滤是有效的.

使用查询语句 Q1~Q5,设置概率阈值为 0.5,由于原始 SIGMOD Record 数据集的大小只有 467 kB,所以通过对数据集进行叠加扩大到适合实验的大小 60.7 MB.实验结果如图 6(c)所示,加入三次概率阈值过滤算法的响应时间要远远小于无概率阈值过滤的算法,实验结果表明概率阈值过滤能够提高算法的查询效率.

4 结束语

在存储同等不确定性数据的条件下,新提出的不确定 XML 文档格式与原有的不确定 XML 文档格式相比更简洁.另外,从理论上证明了 REDN 编码,实例树概率计算方法和三次概率阈值过滤策略的有效性.通过对不确定 XML 序列匹配的关键技术研究发现,序列匹配方法中的 LCS-TRIM 算法与不确定 XML 小枝模式匹配具有很好的契合度.

参考文献:

- [1] 郝忠孝. 不完全信息下 XML 数据库基础[M]. 北京: 科学出版社, 2011.
- [2] Yun J H, Chung C W. Efficient probabilistic XML query processing using an extended labeling scheme and a lightweight index[J]. Information Processing & Management, 2012, 48(6): 1181-1202.
- [3] 李亚文. 概率 XML 文档中 Holistic Twig 查询处理算

法的研究与实现[D]. 沈阳: 东北大学, 2009.

- [4] 刘潘. 概率 XML 文档中 Twig 查询处理算法的研究与实现[D]. 沈阳: 东北大学, 2010.
- [5] 张晓琳, 吕庆, 刘立新, 等. 一种高效的连续不确定 XML 小枝模式匹配算法[J]. 计算机应用研究, 2013, 30(2): 364-366.
- [6] Tatikonda S, Parthasarathy S, Goyder M. LCS-TRIM: dynamic programming meets XML indexing and querying [C]// Koch C, Gehrke J, Garofalakis M N, et al. Proceedings of the 33rd International Conference on Very Large Data Bases. New York: ACM Press, 2007: 63-74.
- [7] Kimelfeld B, Sagiv Y. Modeling and querying probabilistic XML data [J]. ACM SIGMOD Record, 2009, 37(4): 69-77.
- [8] Nierman A, Jagadish H V. ProTDB: Probabilistic data in XML [C]// Proceedings of the 28th International Conference on Very Large Data Bases. San Francisco: Morgan Kaufmann, 2002: 646-657.
- [9] Merialdo P. ACM SIGMOD Record: XML version [EB/OL]. (1999-12-07) [2013-06-06]. <http://www.dia.uniroma3.it/Araneus/Sigmod/>.

作者简介:



张晓琳 女, (1966—), 博士, 教授. 研究方向为数据库理论与技术.

王 鹏 男, (1987—), 硕士研究生. 研究方向为不确定 XML 索引和查询处理.