

# 一种对象级的 VCAS 查询和评分策略

缪丰羽, 林宏康

(宁德师范学院 计算机与信息工程系, 福建 宁德 352100)

**摘要:** XML 文档可以有多种查询方式, 对于略知文档结构的用户可以使用模糊的内容和结构查询, 即 VCAS 查询。提出了一种对象级的 VCAS 查询和评分策略, 其特点是将 XML 文档看做是交叉的许多对象树, 查询的匹配都是以对象树 ISO 或者对象树组 IRO 作为基本单位。这种方法使得 VCAS 的查询可以直接过滤掉许多没有意义的匹配结果, 提高查询的效率。对现有的评分方式做了改进, 挖掘了更多的内容约束来影响匹配结果的评分, 更能接近用户的查询意图。实验证明了文中提出的查询和评分策略的有效性。

**关键词:** XML; 模糊的内容和结构; 对象级; 查询; 评分

中图分类号: TP311.131

文献标识码: A

文章编号: 1673-629X(2013)03-0112-05

doi: 10.3969/j.issn.1673-629X.2013.03.029

## An Object-level Query and Scoring Strategy for VCAS

MIAO Feng-yu, LIN Hong-kang

(Department of Computer and Information Engineering, Ningde Normal University, Ningde 352100, China)

**Abstract:** There are several ways to query the XML documents. For the users who have a smattering of the structure of the documents, can use the vague content and structure query, named VCAS query. In this paper, an approach for VCAS query and scoring based on an object-level XML data model was proposed. The XML documents were regarded as many intersect object trees in the approach, and the matches of the query were ISO or IRO. This approach could filter many meaningless matches and improves query efficiency. In this paper, the existing scoring methods were also improved. More content constraints were mined for scoring, and get more close to the user's intention. The experiments demonstrate the effectiveness of the proposed methods.

**Key words:** XML; VCAS; object-level; query; score

## 0 引言

目前在 XML 数据查询领域研究的主要有两大类查询<sup>[1]</sup>: 一类是只包含内容信息的查询即 CO (content only) 查询, 也称为关键字查询, 另一类是既包含内容信息也包含结构信息的查询, 称为 CAS (content and structure) 查询。根据查询与结果的匹配程度, CAS 又可以分为严格的内容与结构 (SCAS) 查询和模糊的内容与结构 (VCAS) 查询。由于很多时候用户对 XML 文档内部的结构并不十分了解, 经常会给出一些不准确的查询条件, 因此, VCAS 查询相比于 SCAS 查询更具有实际的应用价值。然而, 由于 VCAS 查询中结构信息的表述通常不够准确, 因此 VCAS 查询具有一定的难度。

文中在文献[2,3]的启发下, 深入分析了目前在 VCAS 查询领域的几种典型的研究方法, 在文献[4]的基础上, 结合一种对象级的 XML 数据模型<sup>[5]</sup>, 提出了一种对象级的 VCAS 查询和评分策略, 对文献[4]的查询算法和评分公式分别进行了有效的改进。实验结果表明, 文中提出的这种对象级的查询和评分算法相比于文献[4]在查询效率上有着明显的提高。

## 1 相关研究

目前对于 VCAS 查询的研究, 主要集中在以下两个方面: 如何处理查询中的结构约束以及如何对查询结果进行评分排序。传统的信息检索技术在处理查询的内容模糊上已经取得了宝贵的经验<sup>[6]</sup>, 但是对于处理 VCAS 查询中的结构模糊, 方法还是比较有限。

关于如何处理 VCAS 查询中的结构约束, 现有的方法主要有:

1. 完全忽略结构约束, 将 VCAS 查询简化成 CO 查询<sup>[7]</sup>;
2. 对查询中的结构约束进行严格匹配, 即将

收稿日期: 2012-06-26; 修回日期: 2012-09-29

基金项目: 福建省自然科学基金 (2011J01357); 宁德师范学院科研资助项目 (2009Y033)

作者简介: 缪丰羽 (1983-), 女, 福建福安人, 讲师, 硕士, 主要研究方向为 XML 数据库技术。

VCAS 查询转化成 SCAS 查询;

3. 将 VCAS 中的结构约束进行适当的改写,即结构松弛,用松弛后的结构约束进行查询<sup>[8]</sup>;

4. 查询主要是根据内容约束来进行,结构约束仅作为结果评分的一部分依据<sup>[4]</sup>。

前两种方法完全没有发挥 VCAS 的优势,第三种方法虽然在一定程度上解决了查询结构不准确的问题,但却不能保证改写后的结构约束与原查询相关。第四种方法是目前为止相对理想的一种思路,但文中的算法只考虑到了查询的质量,查询效率不高。

关于 VCAS 查询结果的评分,主要有以下几种方法:

1. 在基于结构松弛的处理方式中,文献[8]提出了3种评分方法:小枝评分(twig score)、路径评分(path score)和二进制评分(binary score),这几种评分方式都完全依赖于结构松弛的方式和松弛后的结构约束,可能导致查询结果与用户查询意图不相关。

2. 文献[9]将 VCAS 查询分解成一个 CO 查询和 SCAS 查询,分别对查询结果进行评分,然后对得分进行综合。由于这种方法需要结构约束严格匹配,因此无法处理结构条件表达不准确时的查询结果。

3. 文献[10]提出了带权的 tf(weight term frequency)和 ief(inverted element frequency)计算公式,在此基础上计算一个谓词 about 的得分。这种方法的缺点在于标签权值由人工指定,不够客观,而且也不适用于 VCAS 查询。

4. 文献[4]中的评分策略综合考虑了内容和结构两部分的信息与查询结果的匹配程度,可以较为全面地对结果进行评分,但是这种评分方法并没有考虑到前期的查询结果是否具有实际意义,可能花费了很多时间去评估一些完全没有意义的匹配,并且在评分的过程中对于内容约束部分的挖掘还不够深入。

文中提出的对象级的查询和评分算法在内容查询的阶段就已经过滤掉没有意义的匹配结果,使得最后的查询结果评分更加的快速和高效,同时还改进了文献[4]中的评分公式,可以更加有效地对结果进行评分排序。

## 2 对象级 XML 数据模型

XML 数据是用来表示现实世界中的各种信息,XML 查询需要返回的也是有实际意义的结果,因此

可以考虑将 XML 数据同现实世界中描述信息的方式联系起来。在现实世界中,通常是将客观事物描述成一个个的对象,这些对象有自己的属性,并且相互之间有联系。当用户输入一个查询时,他所需要的也是一个有实际意义的对象或是一些有关联的对象。因此,搜索引擎需要找出一些对应于现实世界中对象的数据片段。在理想的情况下,一个实用的查询结果应该满足两个要求:

1. 返回有意义的结果,即返回的结果子树应该是对象级的;

2. 结果应该同查询有关,即同用户所关心的内容有关联。

现有的各种查询结果评分排序都是基于 XML 结节点级,而用户的查询其实是在对象级上的。因此文献[5]提出了一种对象级的 XML 数据模型。在这个模型中,XML 文档被看做一个对象树的集合,每个现实世界中的对象 O(包含其属性)被封装在一个对象树中,这个对象树的根结点就是 O 的代表结点;两个对象树通过一条结点内部的边或者引用边相连。查询中的匹配也是按照对象级来进行。当查询的匹配结果为单个对象时,可以称之为 ISO(Interested Single Object)<sup>[5]</sup>,当查询结果为若干相关联的对象时,可以称之为 IRO(Interested Related Object)<sup>[5]</sup>,其中 IRO 还可以细分为 IRO 对和 IRO 组<sup>[5]</sup>。对于用户的查询,对象树模型提供了一种更为精确的匹配,在返回的结果中自然过滤掉了无意义的匹配。因此,对于有意义的查询而言,这种方法可以有效地返回与用户意图相关的结果。同时,对象级的 XML 数据模型中还考虑到了引用边,并且通过区分引用边和结点内部的边来获取比图模型更加有效的结果匹配。

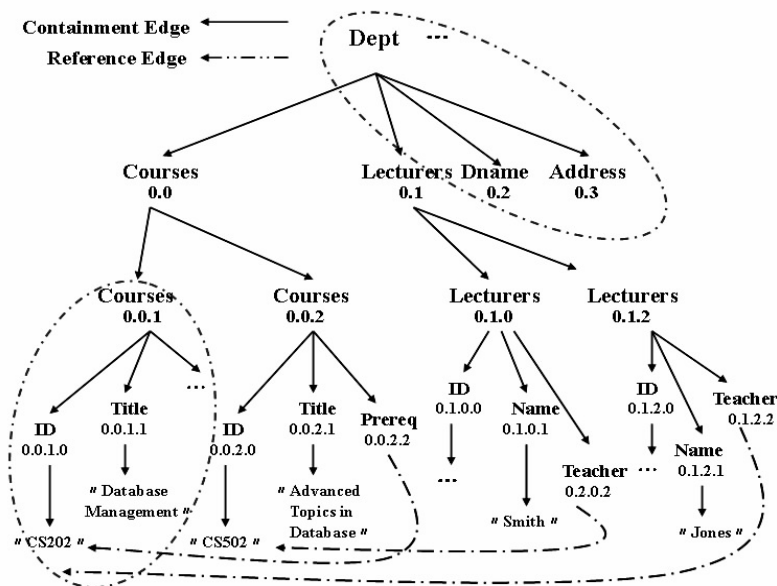


图1 XML 文档片段 (Dewey 编码)

例如,在图 1 中所示的 XML 文档片段中,一共有 5 个对象树(2 个 course,2 个 lecturer 和 1 个 dept),被虚线圈起来的部分分别是一棵 course 对象树和一棵 dept 对象树,dept 下面的结点 courses 和 lecturers 都是连接结点,用来连接 dept 对象、course 对象和 lecturer 对象。对于图 1 的一个查询“database, management”,LCA 返回的是根结点分别为 title0011 和 course00 的两棵子树,这两棵子树并没有什么实际的意义,然而,ISO 返回的则是一棵以 course001 为根结点的对象树,这棵对象树表示一门名为 database management 的课程。再如考虑一个查询“CS502, lecturer”,ISO 不能找到任何有质量的答案,因为没有任何一个对象能符合用户的要求,用户需要的是多个对象。然而,有一个结点 lecturer010 名为“smith”教授课程“cs502”(通过一个 ID 引用),这种情况下,根结点为 lecturer010 的对象树和根结点为 course002 的对象树组成 IRO 对返回。又如,对于图 1 中的查询“Jones, Smith, Database”,四个对象树 course001, course002, lecturer010 和 lecturer012 组成一个 IRO 组,其中 lecturer012 “Jones”教 course001 这门课, lecturer010 “smith”教授“database” course002 这门课。

### 3 对象级的 VCAS 查询和评分

通常使用的 CAS 查询语言源于 XPath,但又在其基础上进行了相应的限制和扩展。

一个 CAS 查询可以写成如下形式:  $path_1 [abouts_1] // \dots // path_n [abouts_n]$ , 其中每个路径  $path_1, \dots, path_n$  都是只含“//”的结点序列,谓词 abouts 是函数 about 的布尔组合。一个函数 about 的形式为 about (path, content), 其中 path 为路径, content 为内容约束,也就是若干个关键字。其含义为: path 所指示的上下文应该与 content 所指示的内容相关。

文中对于一个 CAS 查询的处理分为三个步骤:

1. 首先将其按照内容约束分解为若干片段;
2. 然后根据对象级的 XML 数据模型对查询片段进行处理,并对每个片段的匹配结果进行评分;
3. 综合每个片段的得分,确定整个 CAS 查询的匹配结果得分,根据得分情况最终确定查询结果。

#### 3.1 查询分解

对于一个 VCAS 查询,可以根据函数 about 的数目将其分解为若干个查询片段。每个查询片段是一个二元组  $\langle context, content \rangle$ , 其中 content 为内容约束,是一个关键字列表, context 是 content 所在的查询上下文。

定义 1<sup>[4]</sup> 关键字的查询上下文。给定 CAS 查询模式 p, k 是 p 的一个内容约束中的关键字,则从 p 的根节点到 k 所在叶节点的路径称为 k 的查询上下文。

关键字 k 的上下文就是指 k 所在的环境,一个关键字可以具有多个上下文。查询上下文其实就是查询的结构约束信息。

#### 3.2 查询片段处理和评分

对于一个查询片段  $\langle context, content \rangle$ , 可以从内容部分出发,先找到与 content 相匹配的文档片段,然后对这些文档片段进行结构约束和内容约束两部分的评估。

在对象级的 XML 数据模型下,其实就是利用 content 部分找到匹配的 ISO 或者 IRO, 由于 ISO 或者 IRO 为对象树,因此相比于普通的 XML 文档片段更加具有实际的意义。所得到的 ISO 和 IRO 比仅利用 content 进行匹配得到的文档片段在数量上精简了很多,也更接近于用户的查询目标。

下面给出根据 content 查找匹配的 ISO 或者 IRO 的算法:

Input: content: KW [m]; KW Inverted List: IL [m]; Connection table: CT; upper limit: L, L' for IRO pair and group

Output: Ranked Object List: RL

其中: IL 用来保存关键字的倒排表; CT 类似于图的邻接表,用来查找每个对象有意义的 n 跳连接; L 和 L' 用来表示 IRO 对和 IRO 组中的对象连接跳数的上限; RL 用来表示排好序的对象列表。

##### ● KWsearch

1. RL = ISO\_Result = IRO\_Result = {};
2. HT = a hash table from object to its rank;
3. ILs = the shortest inverted list in the IL[m];
4. for each object o in ILs do
5. Ko = getKeywords (IL, o);
6. If (Ko == KW)
7. initscore (o, Ko, KW, HT);
8. ISO\_Result.add(o);
9. else if (Ko  $\neq \emptyset$ )
10. IRO\_Pair = getIROPairs (IL, o, o, CT, L);
11. IRO\_Group = getIROGroups (IL, o, o, CT, L', Ko);
12. RL = ISO\_Result  $\cup$  IRO\_Pair  $\cup$  IRO\_Group;

##### ● initscore(o, Ko, KW, HT)

1. if (o is ISO and o not in HT)
2. HT.put (o.id, computeISOScore (o, Q, KW));

##### 3. else if (o not in HT)

4. computeTF \* IOF;

##### ● computescore (o, oList)

//需要同时考虑 o 的内容约束和结构约束

1. foreach object  $o' \in oList$
2.  $Ko' = \text{getKeywords}(IL, o')$ ;
3. if(  $Ko' == KW$  )
4.  $\text{initscore}(o', Ko, KW, HT)$ ;
5.  $\text{ISO\_Result.add}(o')$ ;
6. else if(  $Ko' \neq \emptyset$  and (  $Ko' \cup Ko == KW$  ) )
7.  $\text{initscore}(o', Ko', KW, HT)$ ;
8.  $\text{IRO\_Pair.add}(o, o')$ ;
9.  $\text{initscore}(o, Ko, KW, HT)$ ;
10.  $\text{updateIROscore}(o, o', oList, HT)$ ;
- $\text{computeISOscore}(o, Q, KW)$
1. update the ISOscore of  $o$  based on 公式 1、2;
2. put the updated (  $o$ , ISOscore ) into HT;
- $\text{updateIROscore}(o, o', oList, HT)$
1. update the IROscore of  $o$  based on 公式 1、3 ~ 5;
2. put the updated (  $o$ , IROscore ) into HT;

程序的主干部分在算法 KWsearch 中。其主要思想是扫描最短的关键字倒排表 ILs, 检查列表中的对象和它们的连接对象, 然后计算 ISO 和 IRO 评分。对于 ILs 中的每个对象树  $o$ , 通过调用函数  $\text{getKeywords}()$  找出包含在  $o$  中的关键字。如果  $o$  包含所有查询关键字, 则  $o$  为一个 ISO 对象, 然后通过调用  $\text{initscore}()$  计算  $o$  的 ISO 得分, 最后将  $o$  和它的得分一起存储到哈希表 HT 中。如果  $o$  只包含一部分关键字, 则将  $o$  做为一个 IRO 对象, 通过调用函数  $\text{getIROPairs}()$  和  $\text{getIROGroups}()$  找出其所有的 IRO 对和 IRO 组。

当  $o$  为一个 ISO 对象时, 函数  $\text{initscore}()$  根据公式 1 和公式 2 计算 ISOscore, 否则函数  $\text{initscore}()$  计算其  $TF * IOF$  相似性。

函数  $\text{computescore}()$  用来计算并更新对象  $o'$  在  $oList$  中的得分, 每个  $o'$  和  $o$  组成一个 IRO 对。对于每个这样的  $o'$ , 函数探查所有  $o'$  的倒排列表来检查两种情况:

1. 如果  $o'$  是一个包含所有查询关键字的 ISO 对象, 则根据公式 1 和公式 2 计算其 ISOscore 并添加到 ISO\_Result 中。
2. 如果  $o$  和  $o'$  都是 IRO 对象, 则它们的  $TF * IOF$  相似性进行初始化 (如果还没有初始化), 且它们的 IROscore 也被相应地更新。

在排序的过程中同时考虑到  $o$  的内容约束和结构约束, 其中公式 1 是根据 VCAS 中的结构约束部分对查询片段的结果进行的评分, 公式 2 ~ 5 是根据内容约束部分对匹配结果进行的评分, 具体参数可以参考文献[4]和文献[5]。文献[4]中对于 VCAS 的内容约束部分讨论的较为简单, 文中的算法是基于对象级的 XML 数据模型, 因此也将对象级模型中对 ISO 和 IRO

的内容约束引入文中的评分算法中, 结合 VCAS 的结构约束部分, 对 VCAS 查询片段进行评分, 可以使得匹配结果的得分更加接近用户的查询意图。

算法中用到的公式如下:

$$\text{sim}_{\text{level}}(v_q, v_d) = \text{sim}_{\text{label}}(v_q, v_d) \times (1 - \frac{|\text{level}(v_d) - \text{level}(v_q)|}{\max(\text{level}(q), \text{level}(d))}) \quad (1)$$

$$\text{ISOScore}(o, Q) = \rho(o, Q) * (c(o, Q) + s(o, Q)) \quad (2)$$

$$\text{Bonus}(o, Q) = w_1 * \text{BS}_{\text{IRO}_P}(o, Q) + w_2 * \text{BS}_{\text{IRO}_G}(o, Q) \quad (3)$$

$$\text{BS}_{\text{IRO}_G}(o, Q) = \sum_{\forall o' | (o, o') \in \text{IRO}_{P_{\text{G}}}(Q, L)} \rho(o', Q) \quad (4)$$

$$\text{BS}_{\text{IRO}_G}(o, Q) = \frac{\sum \forall_g \in \text{IROG}_{\text{roup}}(Q, L') |_{o \in g} \text{BF}(o, Q, g)}{|\text{IRO}_G_{\text{roup}}(o, Q)|} \quad (5)$$

### 3.3 查询模式评分

给定 VCAS 查询模式  $p$ , 对应的查询片段为  $p_1, p_2, \dots, p_n$ , 在某个 ISO 或者 IRO 上的得分分别为  $\text{score}_1, \dots, \text{score}_n$ , 则  $p$  在 ISO 或者 IRO (在公式中用  $o$  来表示) 上的综合得分为:

$$\text{score}(p, o) = \frac{1}{n} \sum_{i=1}^n \text{score}_i \quad (6)$$

综上三个步骤, 可以尽可能精确地得到一个 VCAS 查询的有意义的匹配结果集, 并对这些匹配结果进行评分排序, 从而提供给用户作为查询结果的参考。

## 4 实验评价

实验的硬件平台: AMD Athlon(tm) II x4 640 processor 3.00GHZ, 3.25 GB RAM。

软件平台: Windows XP professional 2002 sp3, 所有代码用 JAVA 语言实现。

为了便于同文献[4]中的 XScore 进行比较, 文中的实验采用了和文献[4]相同的测试数据: DBLP 数据集(230M)和合成数据集(150M), 其中合成数据集是根据原始的 DBLP 数据集进行变换得到的, 与 DBLP 描述的内容相同, 只是结构不同, 这样可以测试不同查询在异构数据中的表现。使用的 VCAS 查询也同文献[4]中相同。如下所示:

Q1://article[ about(. //author/name, wang) ]

Q2://author[ about(. //article/title, xml) ]

Q3://article[ about(. //title, XML) ] [ about(. //author, wang) ]

Q4://author[ about(. //title, XML) ] [ about(. //name, wang) ]



Q5://article[ about ( journal , ACM ) ][ about ( .//title,XML ) ]//author[ about ( . , wang ) ]

Q6://journal[ about ( .//title, ACM ) ]//article [ about ( .//title,XML ) ][ about ( .//author, wang ) ]

Q7://author [ about ( . , wang ) ]//journal [ about ( .//title,ACM ) ]//article[ about ( .//title,XML ) ]

Q8://dblp//author [ about ( .//article//title, xml ) ]//name[ about ( lastname, wang ) ]

Q9://dblp//journal//article[ about ( title, xml ) ]

文中所提出的算法用 OLScore 表示。此外,还比较了文献[11]中 Twig Score 和 Binary Score 两种方法。

4.1 查全率和查准率

查全率是指返回的相关结点在所有相关结点中的比例,查准率是指返回的相关结点在返回的所有结点中的比例,其中所有相关节点由标准 XML 查询来实现。由于文中提出的策略主要还是参考文献[4]所提出来的思路,因此,OLScore 的查全率和查准率同 XScore 接近,也能够达到很好的检索性能。实验数据显示:Twig Score 方法在查询模式与文档结构差别较大时,查全率不佳;在某些情况下,查准率也比较低。Binary Score 的查询质量总体比较粗糙,如图 2~5 所示。

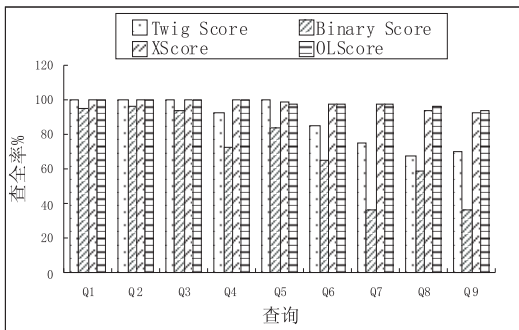


图 2 DBLP 上的查全率比较

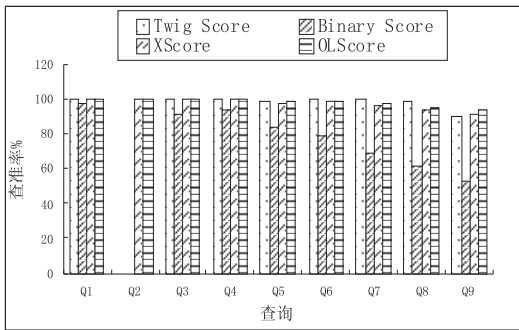


图 3 DBLP 上的查准率比较

4.2 查询和排序的响应时间

文献[4]提出的方法中只关注查询的质量,没有提到查询的效率,文中除了比较查全率、查准率之外,还比较了查询和排序的响应时间。实验结果显示:OLScore 的查询和排序的响应时间比 Twig Score 和 XScore 都要小得多, Binary Score 的响应时间则介于

OLScore 和 XScore 之间,如图 6 所示。

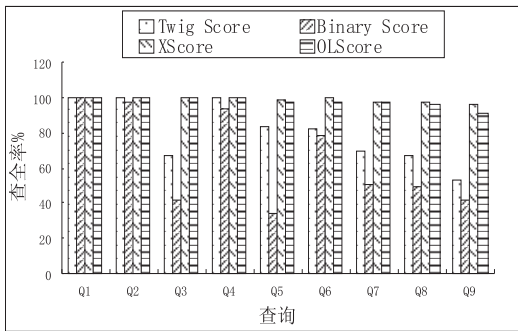


图 4 合成数据集上的查全率比较

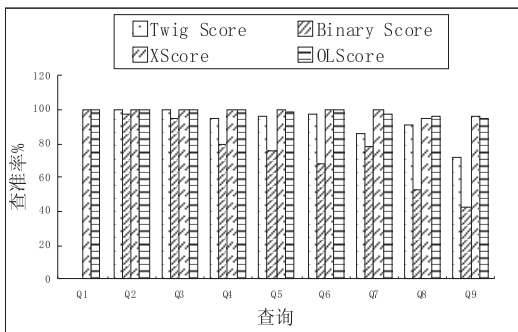


图 5 合成数据集上的查准率比较

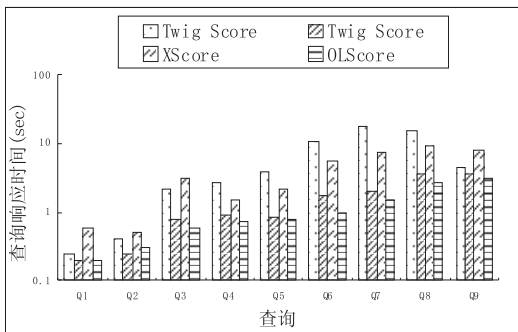


图 6 查询响应时间比较

5 结束语

文中研究了关于模糊的 XML 内容与结构查询 (VCAS),提出了一种对象级的 VCAS 查询和评分策略,这种策略基于一种对象级的 XML 数据模型,可以在查询过程中有效地过滤没有意义的文档片段,使算法在保证高查全率和查准率的前提下,进一步提高查询和排序的时间效率。在匹配结果评分排序方面,结合已有的评分方法,再增加了一些对内容方面的约束,使得评分更能体现用户的查询意图,更有助于匹配结果的排序。实验结果表明,提出的这种查询和评分策略能够取得较为优越的性能。

下一步,准备对文中提出的算法进行优化,并进一步考虑语义相关性<sup>[12]</sup>,使文中的算法能够适应异构的 XML 文档集。

另外大部分数据库都支持 Unicode 编码方式,可以在驱动的 URL 中加入编码信息,例如,连接 MySQL 数据库:

```
jdbc:mysql://localhost/test? useUnicode=true&character Encoding=UTF-8
```

#### 2.4 Java 程序与文件流的乱码问题

Java 中读写字节流文件使用 FileInputStream/FileOutputStream,读取字符流文件使用 FileReader/FileWriter,可以避免字节与字符之间的转换<sup>[12]</sup>。但以上几个类只能使用系统默认的编码方式,假如文件流本身的编码方式和系统编码方式不一致,也会出现乱码问题。此时,可以使用基于字符的类 Input Stream Reader/Output Stream Writer,优点是可以在构造函数中指定文件流的编码类型,Input Stream Reader(Input Stream in, Charset cs) 和 Output Stream Writer(Output Stream out, Charset cs)。

### 3 结束语

在上述分析中,给出了 Java 中乱码问题产生的原因,Java 中常见的几类乱码问题以及对应的解决方法。在实际的 Web 应用中,还会涉及到 Web 服务器,应用服务器以及 JDBC 数据库驱动等等,但只要根据字符编码及转换的原理,在程序的入口和出口采用统一的编码和解码方式,即可以解决 Java 乱码问题。

#### 参考文献:

- [1] Gosling J, Joy B, Steele G. The Java Language Specification [M]. MA: Addison-Wesley Publishing Company, 1996.
- [2] Lindholm T, Yellin F. The Java Virtual Machine Specification [M]. MA: Addison-Wesley Publishing Company, 1996.
- [3] 王子君, 范学峰, 张志浩. Java 编码问题研究与应用[J]. 计算机工程, 2002, 28(3): 242-245.
- [4] 冀振燕, 程 虎, 梅 嘉. Java 语言国际化的设计与实现[J]. 软件学报, 2000, 11(11): 1541-1546.
- [5] 冯金辉, 朱良森. Java 编码中文问题研究及解决方案[J]. 计算机系统应用, 2005(11): 79-81.
- [6] 许 晖, 李涓子. J2EE 系统国际化问题的解决方案[J]. 计算机工程, 2005, 31(18): 79-81.
- [7] 曹 莉, 赵文静. Java 中文处理研究[J]. 计算机技术与发展, 2006, 16(5): 100-103.
- [8] 金恩华, 徐良贤. J2EE Web 应用中汉字编码的研究[J]. 计算机应用与软件, 2005(6): 55-56.
- [9] 包竹苇, 李 森, 张 建. Java 网络传输中字符编码问题的研究[J]. 计算机工程与应用, 2007(4): 93-95.
- [10] 彭利民, 孙素云. JSP 和 Servlet 网络编程设计中汉字编码的研究[J]. 计算机与现代化, 2006(3): 43-45.
- [11] 刘 冰. Java 编程中中文问题的产生及其解决方案[J]. 现代计算机, 2010(3): 105-107.
- [12] 刘长生, 谢 强, 丁 林. Java 应用中的汉字乱码问题分析[J]. 计算机技术与发展, 2006, 16(1): 158-161.

(上接第 116 页)

#### 参考文献:

- [1] Govert N, Kazai G. Overview of the initiative for the evaluation of XML retrieval (INEX) 2002 [C]//Proc of the 1st Workshop of the Initiative for the Evaluation of XML Retrieval (INEX). Schloss Dagstuhl, Germany; European Research Consortium for Informatics and Mathematics, 2002: 1-17.
- [2] 周 健, 孙丽艳. 面向对象 XML 的存储模式的研究[J]. 计算机技术与发展, 2009, 19(3): 114-117.
- [3] 况 旭, 刘 波. XML 的面向对象语言特性[J]. 计算机技术与发展, 2010, 20(1): 54-57.
- [4] 刘喜平, 万常选, 刘德喜. 有效的 XML 模糊内容与结构检索和计分[J]. 计算机研究与发展, 2010, 47(6): 1070-1078.
- [5] Bao Zhifeng, Lu Jiaheng, Ling Tok Wang, et al. An Effective Object-level XML Keyword Search [J]. Computer Science, 2010, 5981: 93-109.
- [6] 万常选, 鲁 远. 基于权重查询词的 XML 结构查询扩展[J]. 软件学报, 2008, 19(10): 2611-2619.
- [7] Sigurbjornsson B, Kamps J, de Rijke M. The University of Am-

- sterdam at INEX 2004 [C]//Proc of the 3rd Workshop of the Initiative for the Evaluation of XML Retrieval (INEX). Berlin: Springer, 2004: 104-109.
- [8] Amer-Yahia S, Lakshmanan L V S, Pandit S. FlexPath: Flexible structure and full-text querying for XML [C]//Proc of the ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2004: 83-94.
- [9] Liu S, Chu W W, Shahinian R. Vague content and structure (VCAS) retrieval for document centric XML collection [C]//Proc of Int Workshop on the Web and Databases (WebDB). New York: ACM, 2005: 79-84.
- [10] Liu S, Zou Q, Chu W W. Configurable indexing and ranking for XML information retrieval [C]//Proc of the 27th Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval. New York: ACM, 2004: 88-95.
- [11] Amer-Yahia S, Koudas N, Marian A, et al. Structure and content scoring for XML [C]//Proc of Int Conf on Very Large Data Bases. New York: ACM, 2005: 361-372.
- [12] 黄 瑞, 史忠植. 一种新的 Web 异构语义信息检索方法[J]. 计算机研究与发展, 2008, 45(8): 1338-1345.

# 一种对象级的VCAS查询和评分策略

作者: [缪丰羽, 林宏康](#)  
作者单位: [宁德师范学院 计算机与信息工程系, 福建 宁德352100](#)  
刊名: [计算机技术与发展](#)  
英文刊名: [Computer Technology and Development](#)  
年, 卷(期): 2013(3)

本文链接: [http://d.g.wanfangdata.com.cn/Periodical\\_wjfz201303031.aspx](http://d.g.wanfangdata.com.cn/Periodical_wjfz201303031.aspx)