XML 关键字查询处理研究

周军锋[1),2)] 孟小峰[3)]

[1)](燕山大学信息科学与工程学院 河北 秦皇岛 066004)

[2)](河北省计算机虚拟技术与系统集成重点实验室 河北 秦皇岛 066004)

[3)](中国人民大学信息学院 北京 100872)

摘 要 关键字查询作为一种有效的信息检索手段,一直以来都是 XML 数据管理领域研究的热点问题,每年均有大量最新研究成果出现在各种顶级会议和期刊上.针对众多国内外研究者在 XML 关键字查询领域所作出的创新性工作,该文以 XML 关键字查询处理系统为框架来组织现有工作,重点分析和比较了查询生成、语义定义、排序机制、查询算法及结果展示等 5 个关键技术点所涉及的代表性工作的特点,并结合最新的应用需求从有效性和高效性的角度归纳出 XML 关键字查询技术后续研究面临的问题和挑战.

关键词 可扩展标记语言;关键字查询;查询生成;查询语义;排序机制;结果展示

中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2012.02459

Keyword Search on XML Data: A Survey

ZHOU Jun-Feng[1),2)] MENG Xiao-Feng[3)]

[1)](*School of Information Science and Engineering, Yanshan University, Qinhuangdao, Hebei* 066004)

[2)](*Key Laboratory for Computer Virtual Technology and System Integration of Hebei Province, Qinhuangdao, Hebei* 066004)

[3)](*School of Information, Renmin University of China, Beijing* 100872)

Abstract As an effective way of extracting desired information, keyword search on XML data has been a hot research issue along with the ever increasing of XML-based applications, and large volumes of research work appear on various top-level conferences and journals each year. In this paper, we organize existing works according to the architecture of an XML keyword search system, and give an in depth analysis to the representative techniques from five aspects, i.e., query generation, query semantics, ranking mechanism, query processing algorithms and result presentation. Finally, according to the requirements of latest applications, we conclude open research problems according to the inherent requirement of effectiveness and efficiency.

Keywords XML; keyword search; query generation; query semantics; ranking mechanism; result presentation

1 引 言

可扩展标记语言 XML(eXtensible Markup Language)的出现,使以网络应用为代表的一大批新型应用得以迅速发展.在当前各种网络应用中,如电子商务、电子政务、金融、出版、科学数据和各种资源的数字化等,XML 扮演着极其重要的角色,它已经成为数据交换事实上的标准、SOA 架构的基石.

高速增长的 XML 数据管理市场在国际范围内引起了各大厂商的激烈竞争.IBM 公司在其新推出的 DB2 9 版本中,直接把对 XML 的支持作为其新

收稿日期:2010-12-01;最终修改稿收到日期:2011-08-15.本课题得到国家核高基重大专项(2010ZX01042-002-003)、国家自然科学基金(61073060、61070055、91024032)、中国人民大学科学研究基金(10XNI018)资助.**周军锋**,男,1977 年生,博士,副教授,主要研究方向为 XML 数据库.E-mail: zhoujf@ysu.edu.cn. **孟小峰**,男,1964 年生,教授,博士生导师,主要研究领域为 Web 数据集成、XML 数据库、移动对象管理等.

产品的最大卖点;Oracle 和微软也同时宣称他们的产品可以实现高性能 XML 数据的存储与查询,使现有应用更好地与 XML 共存.除此之外,不同行业也都加紧制定本领域特定的标记语言,如用于对数学符号进行编码的 MathML、W3C 发布的 XHTML、化学领域的标记语言 CML、W3C 推荐的用于描述二维图形的标记语言 SVG、金融界的 XBRL(可扩展商业报告语言,XML 的一个子集)等等,实践中正在源源不断产生 XML 数据并推动着 XML 数据管理技术不断向前发展.如何有效地管理和利用快速增长的 XML 数据,一直以来都是数据库研究领域中的一个热点问题.

广义来说,用户可以使用两种查询机制,即结构化查询语言(如 XPath 和 XQuery),或者一组关键字从 XML 数据中获取所需的信息.

使用结构化查询语言可以精确表达用户的查询意图,但其正确使用的前提条件是用户了解所用的查询语言和所查询的 XML 文档的结构信息.即使对专业用户来说,了解复杂的 XQuery 语法和 XML 文档结构都是一件极其困难的事情,更不用说面向的普通用户,这极大限制了 XML 数据库系统的可用性.此外,数据间关系的对称性和 XML 文档结构的不对称性之间的矛盾以及 XML 文档结构随着企业业务的发展不断演变的现象将导致数据以不同的物理形式进行组织,会进一步加重用户了解文档结构和书写结构化查询表达式的负担,削弱数据库系统的可用性.

与此对应,关键字查询无需用户学习复杂查询语言和了解复杂文档结构的问题,从而能轻易查询 XML 数据,可有效增强 XML 数据库的可用性.尽管关键字查询技术在信息检索领域取得了巨大成功,XML 数据区别于普通文本的内在层次结构使得传统的关键字检索技术不再适用于 XML 数据.

针对这一问题,国内外的企业界和学术界都投入了大量人力物力进行技术攻关,相关研究成果最近几年频繁出现在数据库领域各种顶级会议和期刊上,但是基于 XML 数据的关键字查询问题仍没有得到很好的解决,在查询生成、语义定义、排序机制、查询算法以及结果展示方面依然存在很多问题,这些问题严重制约了 XML 应用的推广和普及.

本文基于这一背景,根据典型 XML 关键字查询处理系统①所涉及的关键操作及其作用,从查询生成、语义定义、排序机制、查询算法、结果展示及原型系统等六方面系统、深入地分析和比较现有工作的特点,并结合最新的应用归纳出 XML 关键字查询领域后续研究面临的关键问题和挑战.

本文第 2 节介绍 XML 关键字查询处理系统的架构,并总结出 XML 关键字查询处理技术涉及的 5 个方面的关键问题,以便读者对 XML 关键字处理技术有一个整体概念;第 3 节～第 7 节对各技术点的研究现状进行深入的分析和比较;第 8 节介绍现有的 XML 关键字查询原型系统;最后在第 9 节讨论未来一段时间可能的研究点.

2 XML 关键字查询处理系统

2.1 查询接口

已有关键字查询接口可以分为两大类:(1)纯文本输入.Google、百度等各种搜索引擎均采用这种输入方式.(2)可以指定属性和属性值的查询方式,这类似于很多网站提供的高级搜索功能.在 XML 关键字查询领域,XSEarch[1]查询的基本形式为 $Q=\{t_1,t_2,\cdots,t_m\}$,其中每个 $t_i=\langle l;k\rangle(1\leq i\leq m)$ 是一个二元组,l 为 XML 数据中元素的标签名,k 是用户指定的关键字,t_i 可以只由 l 或者 k 构成,t_i 可以通过符号"+"限定为在查询结果中必须出现,否则 t_i 可以不出现在结果中.此外,文献[2]在 XQuery 中嵌入的关键字查询方式也采用这种形式.

2.2 系统工作机制

图 1 为典型 XML 关键字查询系统结构图,如图所示,查询处理器负责查询生成、查询执行和结果

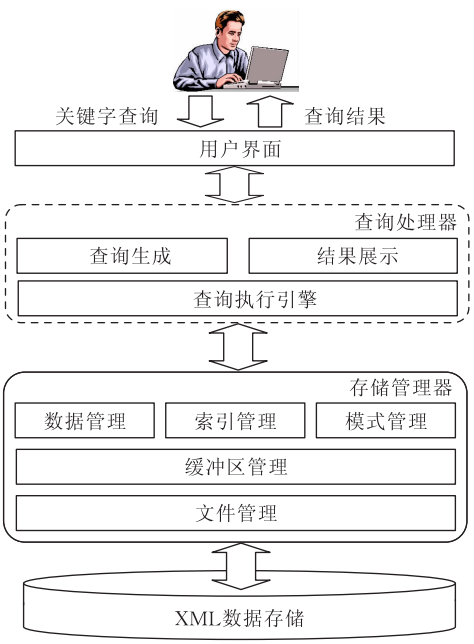


图 1 基于 XML 数据的关键字查询系统结构图

① 这里的"典型 XML 关键字查询系统"指去除不同系统的特色功能模块后,具备一般系统功能的关键字查询系统.

展示功能，存储管理器负责数据、索引和模式信息的管理，并基于缓冲区管理模块和文件管理模块调度数据库中存储的数据.

用户在进行关键字查询时，系统会涉及到以下的处理步骤：(1) 在用户键入关键字的同时，查询生成模块会调用查询清洗功能对输入的关键字进行容错处理，并识别可能的短语和剔除无意义的关键字，同时调用查询提示功能根据历史查询的统计信息为用户实时推荐其它相关的关键字以协助用户构造查询. (2) 查询执行引擎根据给定的查询语义和排序机制确定执行算法，然后调用存储管理器的相应模块获取满足条件的 Top-K 查询结果. (3) 结果展示模块将符合查询语义的结果按照其与用户查询的相关程度，依照特定展示策略依次呈现.

2.3　关键技术问题

基于 XML 数据的关键字查询技术涉及 5 个方面的关键问题：

(1) 查询生成，即根据用户输入的关键字构造合适的查询. 查询生成的目的是为了向查询引擎提交尽可能准确表达用户自身查询意图的一组关键字.

(2) 查询语义，即对于给定的关键字查询，定义什么样的结果是有意义结果的问题. 有效的查询语义可以为用户返回更多符合其查询意图的结果.

(3) 排序机制. 对于得到的结果，排序机制将计算其与用户查询的相关程度，并按照分值进行排序，将最相关的结果最先呈现给用户.

(4) 高效算法. 针对给定的查询，高效算法可以提升系统的响应速度、快速求解符合语义的结果.

(5) 结果展示，即对于得到的查询结果，以何种方式呈现给用户的问题. 有效的结果展示方式有助于用户快速定位所需信息、提高系统的可用性.

需要说明的是. 图 1 所示的存储管理器中涉及的技术问题可以借鉴关系数据管理领域的成熟技术和产品进行参考. 除了以上 5 个方面的问题，与 XML 关键字查询处理相关的研究工作还包括：结构化查询和关键字查询相结合的查询方式，如 TeXQuery[3]、FleXPath[4]、GalaTex[5] 及文献[2]的结构等；XML 数据流上的关键字查询；不确定性 XML 数据上的关键字查询；数据源的选择等. 限于篇幅，本文对这些问题不予讨论. 虽然应用领域不同，但其中关键字查询处理部分和本文所述的问题一致. 在本文的讨论中，我们将以上 5 个方面的核心技术问题纳入 XML 关键字查询系统来进行分类阐述. 这些研究点的组织如图 2 所示，其中关键问题(2)、(3)、(4)都属于查询执行引擎模块，在图中如虚线矩形所示. 后续内容将按照图 2 的分类方式逐一阐述.

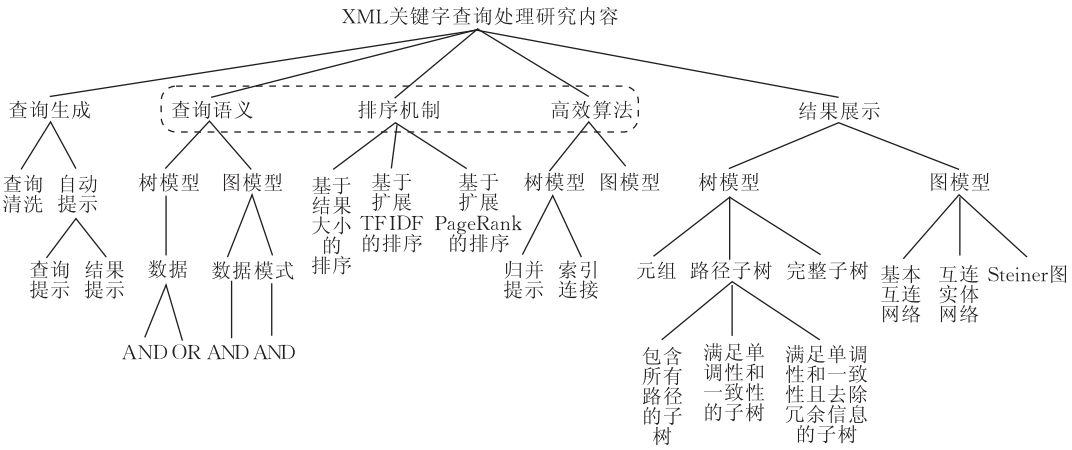


图 2　本文涉及的 XML 关键字查询处理关键技术内容

3　查询生成

3.1　研究动机

实际中，用户输入习惯的不同或拼写错误会导致查询中包含不相关或错误的单词. 例如，用户想找和 Allison Ross 相关的信息，但却错误的输入了“Alison Rose”；再如，用户输入的关键词可能是被找内容的同义词. 这些问题的出现在所难免，需要查询生成模块调用其自身的查询清洗功能来协助用户修正输入中存在的问题来改进查询结果的质量.

另外，当用户对所查询的内容不熟悉时，为用户自动提示与当前输入关键字相关的信息有助于用户快速构建查询. 根据提示信息的类型，可以分为查询提示和结果提示两大类. 查询提示通过将用户的实时输入和预先定义的字符串目录(内容来自被查询数据的统计信息和历史查询信息)进行匹配，预测用户未来可能输入的内容，是一种快速编辑文本内容

和构建查询的技术.结果提示通过在数据库中快速定位与输入查询匹配的结果,为用户实时反馈与当前输入关键字相关的查询结果.当用户对所查询的内容不熟悉时,这种方式可以改善传统信息检索系统先提交关键字再给出查询结果所造成的重复提交查询的问题.

3.2 研究现状

3.2.1 查询清洗

文献[6]主要研究计算语言领域的短语识别问题,文献[7-8]关注于信息提取方面的短语识别问题,这些方法基于给定的文本,通过训练一个语言概率模型来推断分隔符的位置.由于针对的不是查询,这些方法识别出来的短语在数据库中可能并不存在.另外,以上方法对短语中的单词是顺序敏感的,而灵活的关键字查询系统应该支持与顺序无关的短语查询.

文献[9-10]主要研究基于关系数据的关键字清洗策略,所解决的问题包括识别查询中无意义的关键字、修正错误拼写的关键字、识别有效的短语以及从文本中提取有效的关键字查询.

3.2.2 查询提示

在查询提示方面,已有的很多工具软件,如Visual Studio、Google以及桌面搜索等都提供了该功能.文献[11-17]研究了查询提示问题,其中文献[11]针对有限内存提出了单字符预测的问题并给出了相应的解决方案;文献[13]专门针对文本数据进行查询提示;文献[12]提出了一种短语预测算法来协助用户快速构建符合自身语义的查询;文献[14]系统地研究了数据库关键字查询提示的容错处理问题,即当用户由于不了解被查询内容而输入了错误的或者语义近似的单词时,如何为用户实时推荐近似匹配;文献[15]在文献[14]的基础上,将容错处理技术应用到输入URL数据时的实时提示.文献[16]进一步考虑了单词间的语义信息,为用户实时推荐与输入的单词相关的主题词,来协助用户构建面向某个主题的查询;文献[17]也是为用户推荐相关主题信息,和文献[16]相比,区别在于主题的推荐不是实时的,而是对Top-K查询结果进行概括得到的.

3.2.3 结果提示

文献[18-22]研究结果提示问题.其中文献[18-19]分别基于XML数据和关系数据为用户提供与输入同步的实时匹配结果,二者的共同点是关键字的匹配方式都是精确匹配.文献[20-21]研究了在用户输入不准确的前提下,基于相似性匹配技术为用户实时展示近似查询结果的问题.文献[22]在查询表格中键入关键字,实时给出相关查询结果,与其他结果提示方法相比,该方法可以为不同属性指定关键字,通过增加约束条件来提高查询的准确性.

图3展示了现有查询生成技术的研究进展情况,结合前面的阐述可以看出:现有技术大多关注文本数据和关系数据;随着研究的深入,尤其是实时结果提示问题,目前可以支持模糊匹配,对用户来说,查询变得更加容易,但对XML数据来说,仅涉及文献[18]的工作.

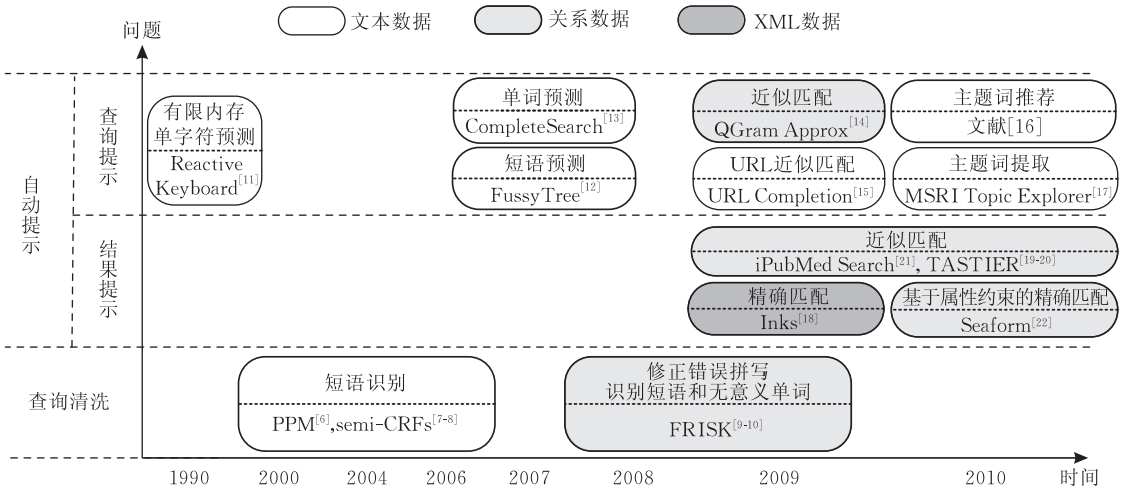


图3 查询生成技术分类比较

4 查询语义

本文划分语义类型的标准有3个:(1)数据模型,即定义语义时是否考虑元素间ID/IDREF及XLink信息,分为树模型和图模型两类.(2)依据对象,即定义语义时依据模式信息进行定义还是直接在数据上进行定义.(3)关键字关系,即关键字之间

需要满足 AND 还是 OR 关系. 如图 2 所示，现有的 XML 关键字查询语义可以分为 4 类，分别是：(1) 基于树模型和 XML 数据的 AND 语义. (2) 基于树模型和 XML 数据的 OR 语义. (3) 基于图模型和模式信息的 AND 语义. (4) 基于图模型和 XML 数据的 AND 语义.

4.1　基于树模型的 XML 关键字查询语义

在基于树模型的关键字查询语义中，现有工作都是基于数据进行定义的. 最低公共祖先 (Lowest Common Ancestor，LCA) 是所有语义的基础. 对于给定的查询 $Q=\{k_1,k_2,\cdots,k_m\}$ 而言，以 LCA 为根的子树是包含这些关键字对应结点的最小语义片段.

单纯基于LCA语义的工作主要涉及文献[23-24]，这两个工作都着眼于推断用户的查询意图，然后将以 LCA 为根的子树作为结果返回. 文献[23]从查询本身着手，首先基于统计方法判断用户的查询意图，即用户想找什么类型的结点①. 文献[24]基于文献[23]的工作，进一步考虑了多个结点之间及多个值之间的关键字和结构信息来推断用户的查询意图. 除此以外，其它语义都在 LCA 的基础上进行了不同程度的限制，下面分别进行阐述.

4.1.1　基于树模型和 XML 数据的 AND 语义

目前大多数基于树模型的 XML 关键字查询语义都属于这一类，共有 5 种代表性语义，分别是 ELCA、SLCA、MLCA、XSEarch 及 VLCA. 其区别可以简单阐述为：ELCA 和 SLCA 仅考虑结果的结构信息，其它 3 种语义进一步考虑结果中的元素名称.

ELCA 语义[25-26]. 其定义可以简单阐述为：如果一个 LCA 结点 v 是 ELCA 结点，则 v 满足在以 v 为根的子树中去除所有以其它 LCA 结点为根的子树后，v 仍然包含所有关键字. 例如，对于图 4 所示的文档和查询 $Q_1=\{k_1,k_2\}$ 来说，满足条件的 ELCA 结点为 r,a_1,d_3,a_5，注意 a_2 之所以不是 ELCA 的原因在于 b_3 是 k_1 和 k_2 的 LCA 结点，当移去以 b_3 为根的子树后，以 a_2 为根的子树仅包含关键字 k_2，因此 a_2 不是 k_1 和 k_2 的 ELCA 结点.

SLCA 语义[27]. 这种语义认为既然以某个结点为根的子树包含所有的关键字，那么它的祖先元素就不应该作为返回结果. 例如，对于图 4 所示的文档和查询 $Q_1=\{k_1,k_2\}$ 来说，满足条件的 SLCA 结点为 a_1,d_3,a_5. 文献[28]基于元素类型提出了一种类似语义 MLCA，虽然 SLCA 不考虑元素类型，但对于图 4 的 D_1 和 Q_1 来说，SLCA 和 MLCA 语义返回的子树根结点集合是一样的，这里不再单独说明.

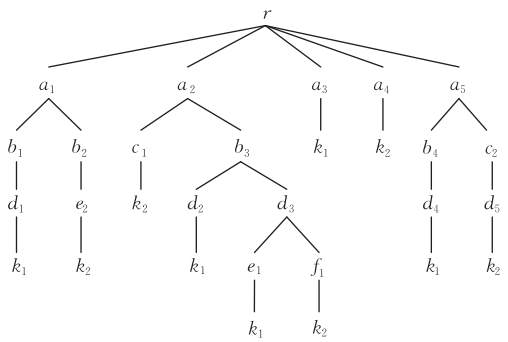


图 4　XML 文档 D_1 (元素名字中的字母表示其标签，数字用于区别有相同标签的不同结点)

XSEarch[1] **和 VLCA**[29] **语义**. 这两种语义在 LCA 的基础上，将不同关键字到 LCA 路径上存在同名结点的结果排除在外.

假设 T 表示给定 XML 文档中的一个子树，n_1 和 n_2 是 T 中的两个元素结点，元素名称指该元素在文档对应的 DTD 中的标签名 (英文称为 tag 或 label)，LCA(n_1,n_2) 表示 n_1 和 n_2 的 LCA 结点，$T_{|n1,n2}$ 表示从 LCA(n_1,n_2) 到 n_1 和 n_2 的两条路径.

定义 1 (互连关系). 如果 n_1 和 n_2 满足以下条件之一，则称 n_1 和 n_2 满足互连关系：

(1) $T_{|n1,n2}$ 中不存在有相同名称的不同结点.

(2) 有相同名称的结点只能是 n_1 和 n_2.

基于互连关系，文献[1]提出两种语义，一种是 *all* 语义，一种是 *star* 语义. 对于给定的查询 $Q=\{k_1,k_2,\cdots,k_m\}$ 来说，假设包含这 m 个关键字的元素结点是 $n_1,n_2,\cdots,n_m$，*all* 语义规定这 m 个结点中的任意两个之间必须满足互联关系，而 *star* 语义需要首先从 m 个结点中选定一个，其它 $m-1$ 个结点只需要和被选定的结点具有互联关系即可. 本文后续提到 XSEarch 语义时，均指 *all* 语义. 从这个意义来说，XSEarch 和 VLCA 语义相同，区别在于文献[29]中基于 VLCA 语义提出的 CVLCA 语义.

例 1.　对于图 4 所示的 XML 文档 D_1 和关键字查询 $Q_1=\{k_1,k_2\}$，假设 S_X 表示满足 X 语义的结点集合，则根据以上介绍，满足条件的 LCA 语义的结点集合 $S_{\mathrm{LCA}}=\{r,a_1,a_2,b_3,d_3,a_5\}$，满足条件的 SLCA 和 MLCA 结点集合 $S_{\mathrm{SLCA}}=S_{\mathrm{MLCA}}=\{a_1,d_3,a_5\}$，满足 XSEarch 和 VLCA 条件的结点集合 $S_{\mathrm{XSEarch}}=S_{\mathrm{VLCA}}=\{r,a_2,d_3,a_5\}$，满足 ELCA 语义的结点集合是 $S_{\mathrm{ELCA}}=\{r,a_1,d_3,a_5\}$. 可以看出，所有语义都返回了 d_3 和 a_5，但又存在和其它语义不同的

① 结点类型指在文档中从某个结点到根结点的路径.

返回结果,其关系如图 5 所示.

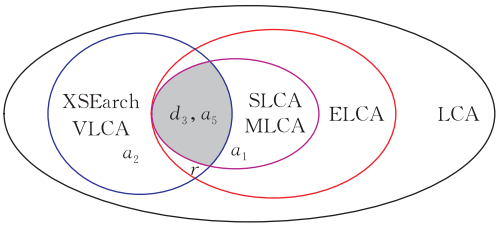


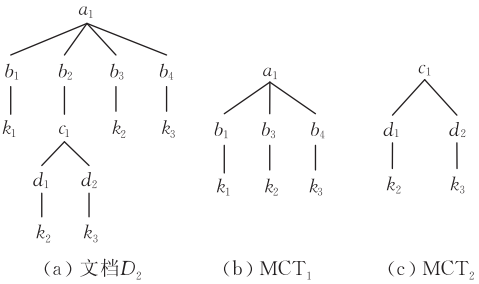
图 5 基于 $Q_1=\{k_1,k_2\}$和文档 D_1的查询语义关系示意

4.1.2 基于树模型和 XML 数据的 OR 语义

MCT 语义[30]. 对于给定的文档 D 和查询 $Q=\{k_1,k_2,\cdots,k_m\}$来说,MCT(Minimal-Cost Tree)的定义基于 4 个概念:① 关键字 $k_i(1\leq i\leq m)$的内容结点(content nodes)指直接包含 k_i的元素结点 n_i. ② k_i的近似内容结点(quasi content nodes)指 n_i的所有祖先结点. ③ k_i和结点 n(n 是 k_i的内容结点或近似内容结点)的关键结点(pivotal node)指在 n 的后代中,所有 k_i的内容结点中离 n 最近的内容结点. ④ 对于 k_i和 n 来说,其关键结点 n_i的关键路径(pivotal path)指从结点 n 到 n_i的路径. 最小代价树 MCT 的定义如下.

定义 2(最小代价树 MCT). 对于给定的查询 $Q=\{k_1,k_2,\cdots,k_m\}$和文档 D 中的结点 n,Q 的关键字和 n 对应的所有关键结点的关键路径构成了最小代价树 MCT.

例 2. 对于图 6 所示的 XML 文档 D_2和关键字查询 $Q_2=\{k_1,k_2,k_3\}$来说,k_1的内容结点是 b_1;k_2的内容结点是 d_1和 b_3;k_3的内容结点是 d_2和 b_4. k_1的近似内容结点是 a_1;k_2的近似内容结点是 a_1,b_2,b_3,c_1,d_1;k_3的近似内容结点是 a_1,b_2,b_4,c_1,d_2. 对于 k_1和 a_1来说,其关键结点是 b_1;对于 k_2和 a_1来说,其关键结点是 b_3,不是 d_1;同样,对于 k_3和 a_1来说,其关键结点是 b_4. 对于 k_1和 a_1来说,其关键路径是 $a_1\rightarrow b_1$;对于 k_2和 a_1来说,其关键路径是 $a_1\rightarrow b_3$;对于 k_3和 a_1来说,其关键路径是 $a_1\rightarrow b_4$. 基于以上结果,对于 Q_2和 a_1来说,其 MCT 如图 6 中 MCT_1所示;对于 Q_2和 c_1来说,其 MCT 如图中 MCT_2所示.



(a) 文档D_2 (b) MCT_1 (c) MCT_2

图 6 XML 文档 D_2及 MCT 示例

4.2 基于图模型的 XML 关键字查询语义

在基于图模型的关键字查询中,已有方法的语义主要借鉴了关系数据库中的关键字查询思想[31],即对于给定的一组关键字,返回包含所有关键字的最小连通网络(Minimal Total Node Network, MTNN). 现有方法都要求关键字之间是 AND 关系.

4.2.1 基于图模型和模式信息的 AND 语义

文献[32]基于模式图定义数据的最小连通网络. 文献[2,33]提出的关键字查询语义也是基于模式图进行定义的,其查询的关键字是由"label: keyword"中的 label 组成,语义为查询中所有 label 在模式图中对应的连通网络,因此这两个文献的关键字查询中,每个关键字其实都是模式中的元素. 文献[34]针对已有语义存在的问题,通过对元素进行类别标注,提出使用 MCN 语义来识别有意义的结果.

4.2.2 基于图模型和 XML 数据的 AND 语义

文献[35]研究了有向图上的关键字查询问题,并提出了一种双层索引来加速有向图上关键字查询处理的效率. 文献[36]提出了一种查询异构文本、XML 及关系数据的总体框架,对 XML 数据来说,其基本语义是半径为 r 的 Steiner 图.

表 1 中"结点名称"列表示该语义在定义时是否

表 1 XML 关键字查询语义

方法	数据模型		语义定义基础		关键字之间关系		其它因素		
	树	图	模式	数据	AND	OR	结构	结点名称	结点类别
XRank[25-26]	√			◉	√		●		
XKeyword[32]		◉	√		√		●		
XSEarch[1]	√			◉		◉	●	◆	
MLCA[28]	√			◉	√		●	◆	
SLCA[27]	√			◉	√		●		
Interconnection[33]		◉	√		√		●		
MSP[2]		◉	√		√		●	◆	■
BLINKS[35]		◉		◉	√		●		
VLCA[29]	√			◉	√		●	◆	
EASE[36]		◉		◉	√		●		

（续　表）

方法	数据模型		语义定义基础		关键字之间关系		其它因素		
	树	图	模式	数据	AND	OR	结构	结点名称	结点类别
XReal[23]	√			✓	√		●		
SAIL[30]	√			✓		✓	●	◆	
MCN[34]		✓	√		√		●		■
CLCA[37]	√			✓		✓	●	◆	
ISO&IRO[38]		✓		✓	√		●		■

考虑了结点标签名的影响；“结点类别”列指语义定义时是否考虑了结点所属类别的影响，结点类别包括实体、属性和连接结点. 可以看出随着研究的深入，不同工作在定义语义时考虑的角度和因素都有所不同. 总体而言早期的语义大多单纯从 XML 文档结构的角度来定义语义，而最近提出的语义考虑了更多的信息，如结点名称和类别，因而返回结果更加贴近用户的实际需求，更适用于以数据为中心的文档.

5　排序机制

对于得到的结果，排序机制将计算其与查询的相关程度，并给每个结果相应的分值，最后按照分值进行排序，将评分最高的结果最先返回给用户. 已有方法所采用的排序机制大致可以分为 3 类：(1) 以结果大小为标准的方法. (2) 以 PageRank 为基础进行扩展的方法. (3) 以 TFIDF 为基础进行扩展的方法，下面进行分类阐述.

5.1　基于结果大小的排序方法

在该类方法中，文献[32]基于模式图，首先根据给定的关键字查询得到包含所有关键字的最小候选网络(Minimal Total Node Network，MTNN)，MTNN 类似于 XQuery 语言对应的结构化查询表达式. 这样，每个关键字查询可能对应多个 MTNN，每个 MTNN 的评分是其包含的边的数目. 因此，如果一个 MTNN 对应多个返回结果，则这些结果的评分相同.

5.2　基于扩展 PageRank 的排序方法

该类方法的典型代表是 XRank[25]. 在计算元素的全局重要性方面，XRank 考虑了 3 种链接信息：(1) 元素之间的超链接. (2) 祖先结点到后代结点的链接. (3) 后代结点到祖先结点的链接. 在计算结果和查询间相关性方面，XRank 考虑了 3 种因素：(1) 结果专一性(Result Specificity)，可以简单理解为结果大小. (2) 关键字之间的接近性(Keyword Proximity)，包含关键字间物理距离的远近和关键字间语义距离的远近. (3) 超链接可知性(Hyperlink Awareness). 对于满足文献[25]中提出的 ELCA 语义[26]的结果 v 来说，v 和给定查询 $Q=\{k_1,k_2,\cdots,k_n\}$ 的相关程度用下面的式(1)来计算.

$$R(v,Q)=\Big(\sum_{1\le i\le n}\hat{r}(v,k_i)\Big)\times p(v,k_1,k_2,\cdots,k_n) \tag{1}$$

式(1)中等式左边的 $R(v,Q)$ 表示 v 和 Q 的相关度，等式右边由两部分的乘积组成：第 1 部分表示 v 与所有关键字的相关度之和，主要考虑了结果的专一性和超链接问题；第 2 部分的 $p(v,k_1,k_2,\cdots,k_n)$ 用于衡量关键字之间的接近度. 函数 p 的取值介于 0～1 之间，文献[25]的方法中，p 的取值与 v 中包含所有关键字的最小文本片段的长度成反比. 下面我们重点介绍式(1)中等式右边的和式部分.

该和式表示 v 与所有关键字的相关度之和，对于多次出现 k_i 的情况，文献[25]中取所有 $r(v,k_i)$ 的最大值. 文献[25]中 $r(v,k_i)$ 的计算如式(2)所示.

$$r(v,k_i)=e(v)\times decay^{t-1} \tag{2}$$

式(2)中的 $decay^{t-1}$ 用于衡量结果的专一性，其中，t 表示从 $v\sim k_i$ 的路径长度；$e(v)$ 表示 v 的全局重要性. 其计算方法如式(3)～(5)所示.

$$e(v)=\frac{1-d}{N_e}+d\times\sum_{(u,v)\in E}\frac{e(u)}{N_h(u)+N_c(u)+1} \tag{3}$$

式(3)将 XML 文档中的元素当成信息检索领域的文档来计算元素 v 的全局重要性. 式(3)中，(u,v) 表示 XML 文档中的一条边；E 表示边的集合，由 3 种边构成：超链接边 E_H(包括 ID/IDREF 和 XLink)、包含边 E_C 及反向包含边；N_e 表示文档中元素的总数；$N_c(u)$ 表示 u 的孩子结点数；$N_h(u)$ 表示从 u 出发的超链接数. 式(3)对 3 种边不加区分，而实际中包含边和超链接边是相互独立的，式(4)考虑了两种边的区别.

$$e(v)=\frac{1-d_1-d_2}{N_e}+d_1\times\sum_{(u,v)\in E_H}\frac{e(u)}{N_h(u)}+$$

$$d_2 \times \sum_{(u,v)\in E_C \cup E_C^{-1}} \frac{e(u)}{N_c(u)+1} \tag{4}$$

虽然式(4)对包含边和超链接边进行了区分,但没有对正向包含边和反向包含边进行区分,这在实际中并不合理.例如,对于图 7 所示的结点 n 来说,其重要性需要考虑其孩子结点 m 的重要性.由于 n 只有一个孩子 m,因此 m 可以向上传递的重要性应该等于其自身重要性,而不应该和式(4)一样,将向上传递的重要性除以 $N_c(u)+1$.修正后的计算方法对正向和反向包含边进行了区分,如式(5)所示.

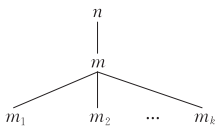


图 7 结点贡献度示意图

$$e(v)=\frac{1-d_1-d_2-d_3}{N_e}+d_1\times\sum_{(u,v)\in E_H}\frac{e(u)}{N_h(u)}+$$
$$d_2\times\sum_{(u,v)\in E_C}\frac{e(u)}{N_c(u)}+d_3\times\sum_{(u,v)\in E_C^{-1}}e(u) \tag{5}$$

综上,式(1)中的 p 说明文献[25]的方法考虑了关键字之间的接近度,式(2)中的 $decay$ 表明该方法考虑了结果专一性,而式(3)~(5)则说明该方法考虑了不同的超链接信息.

5.3 基于扩展 TFIDF 的排序方法

绝大部分已有的排序方法都是基于 TFIDF 进行扩展,这里介绍两个代表性工作:(1)结合向量空间模型和 TFIDF 的排序方法.(2)直接扩展 TFIDF 的排序方法.下面分别阐述.

5.3.1 结合向量空间模型和 TFIDF 的排序方法

XSEarch[1] 在对结果进行排序时,既考虑了单词的出现频率,也考虑了结果的大小及结构信息.其计算方法如式(6)所示.

$$score(q,N)=\frac{sim(Q,N)^{\alpha}}{size(N)^{\beta}}\times(1+\gamma\times AD(N)) \tag{6}$$

式(6)中等号左部表示查询 Q 与查询结果 N 的相关度评分,右部的 $size(N)$ 表示关系树 N 中的结点个数;$AD(N)$ 表示 N 中具有祖先后代关系的结点对的个数;$sim(Q,N)$ 表示基于向量空间模型计算得到的查询 Q 和 N 的相似度;β、α 及 γ 表示为 3 种度量因素设定的权重.从式(6)可以看出,在两个结果 $sim(Q,N)$ 相同的情况下,结点越多,其评分越小;结果中的祖先后代结点对越多,结果评分越大.对图 8 中的两个结果 N_1 和 N_2 来说,当 $sim(Q,N_1)=sim(Q,N_2)$,$size(N_1)=size(N_2)$ 且 $AD(N_1)=AD(N_2)$ 时,该语义得出的相关性评分 $score(Q,N_1)=score(Q,N_2)$,这种情况不一定和实际情况相符.

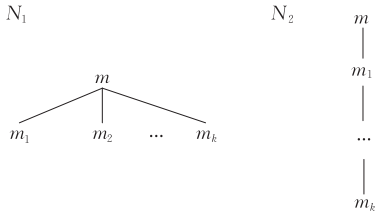


图 8 结果评分问题示例

$sim(Q,N)$ 的计算基于向量空间模型,如式(7)所示,其中 $\boldsymbol{t}_i$ 表示查询 $Q=\{t_1,t_2,\cdots,t_m\}$ 中的 t_i[①] 对应的向量,$\boldsymbol{n}_j$ 表示结果 N 中结点 n_j 对应的向量.

$$sim(Q,N)=\sum_{1\le i\le m,i\le j\le n}\boldsymbol{t}_i\cdot\boldsymbol{n}_j \tag{7}$$

假设文档 D 中包含 L_1 个关键字,D 对应的 DTD 中有 L_2 个标签,则向量 $\boldsymbol{t}_i$ 和 $\boldsymbol{n}_j$ 的大小是 $L_1\times L_2$,即对于每个关键字和标签构成的二元组 (k,l) 而言,向量中都有一个数字 $nbr(k,l)$ 和其对应.

对于查询 Q 中的每个 t_i 而言,其对应的向量 $\boldsymbol{t}_i$ 中的数字取值分为 3 种情况:(1) $t=$“$:k_i$”.(2) $t=$“$l_i:$”.(3) $t=$“$l_i:k_i$”.对于第(1)种情况,对于所有包含 k_i 的结点,假设其结点名为 l,则相应的 $nbr(k_i,l)$ 等于 1,否则为 0.对于第(2)种情况,所有 (k,l) 对中标签等于 l_i 的 $nbr(k,l_i)$ 取值为 1,否则为 0.第(3)种情况仅有 $nbr(k_i,l_i)$ 取值为 1,其它为 0.

对于 D 中的结点 n 而言,假设其包含的叶子结点集为 N_{leaf},则 n 对应的向量中的每个数字的取值如式(8)所示.

$$nbr_n(k,l)=\begin{cases}\sum_{n_l\in N_{\text{leaf}}} w(k,n_l), & label(n)=l\\ 0, & \text{其它}\end{cases} \tag{8}$$

式(8)中的 $w(k,n_l)$ 的计算采用 TFIDF 思想计算,$w(k,n_l)=tf(k,n_l)\times ilf(k)$[②],$tf(k,n_l)$ 和 $ilf(k)$ 的计算见式(9)和式(10).公式中 $occ(k,n_l)$ 表示 n_l 中 k 出现的次数,$words(n_l)$ 表示 n_l 包含的单词集合,M 是 XML 文档中的结点集合.

$$tf(k,n_l)=\frac{occ(k,n_l)}{\max\{occ(k',n_l)\,|\,k'\in words(n_l)\}} \tag{9}$$

$$ilf(k)=\log\left(1+\frac{|M|}{|\{n_l\in M\,|\,k\in words(n_l)\}|}\right) \tag{10}$$

① 参见文献[1],t_i 的组成方式分为 3 种:$l_i:k_i$;$l_i:$;$:k_i$.

② ilf 的英文全称是 inverse leaf frequency.

5.3.2 直接扩展 TFIDF 的排序方法

SAIL[30]采用基于 TFIDF 的思想来计算结果和查询的相关性. 和 XSEarch 不同，SAIL 考虑了更多的排序因素. 例如，XSEarch 使用 TFIDF 计算的是叶子结点和关键字的相关性，而 SAIL 计算的是任意结点和关键字的相关性，并通过路径长短来区分高度不同的结果的相关性；XSEarch 直接用 $tf(k,n_l)\times ilf(k)$ 来计算相关性，而 SAIL 考虑了更多的规范化因子，如对 tf 和 idf 取对数，并除以文档长度. 简单地说，SAIL 通过 TFIDF 考虑了单词的频率信息，通过路径长度对结果的高度进行区分，通过单词之间的紧凑度对结果的宽度进行区分. 文献[37]的排序思想和 SAIL 类似，通过结构距离和单词的统计信息来对结果和查询的相关性进行计算.

XReal[23]和前述工作的区别是语义定义，即通过统计的方法首先确定应该返回什么类型的结果，然后再对该类型的结果进行排序. 其结果排序思想同样基于 TFIDF，考虑的排序因素包括关键字的频率、关键字出现的顺序、关键字在查询结果中的位置(即结果的紧凑性)及结果的大小(高度)等. 文献[24]基于文献[23]的工作，考虑了多个结点之间及多个值之间的关键字和结构信息来推断用户的查询意图. 但文献[24]排序的对象不是查询结果，而是返回结果类型.

5.4 其它方法

文献[35]基于图模型数据，提出了一种结合结构大小、PageRank 以及 TFIDF 这 3 种排序思想的混合排序机制. 假定查询 $Q=\{k_1,k_2,\cdots,k_m\}$，结果 $T=\langle r,\{n_1,\cdots,n_m\}\rangle$，其中 r 是根结点，n_1 到 n_m 是包含关键字的 m 个结点. 则 T 和 Q 的相关性评分 $S(T)$ 的计算方法如式(11)所示.

式(11)中等号右边第 1 部分表示根结点的重要性，用基于 PageRank 思想的方法计算，第 2 部分表示每个包含关键字的结点相对于关键字的重要性，用基于 TFIDF 的方法计算，第 3 部分用于衡量查询结果的紧凑性，考虑的因素包括从根结点到包含关键字的结点之间的路径长度以及图中结点的出入度等.

$$S(T)=(\alpha\overline{S}_r(T)+\beta\overline{S}_n(T)+\gamma\overline{S}_p(T))^{-1} \quad (11)$$

除了以上介绍的代表性工作以外，还有大量工作都是基于 TFIDF 思想进行排序，如对图数据查询结果进行排序的 EASE[36]以及文献[39]提出的可配置参数的排序方法等. 总体来看，除了 XRank，大多数方法在进行结果排序时，都是改进 TFIDF 的计算方法，通过综合考虑结果大小、关键字的出现频率等信息，使之更能反映查询和结果的相关性. 随着研究的深入，越来越多影响结果排序的因素被考虑在内，如表 2 所示. 从实际应用的角度来看，排序效果的好坏除了受所考虑因素多少的影响之外，不同因素所占权重也在很大程度上影响排序结果的准确性. XReal 是最近提出的方法，通过统计获取关键字的分布信息，进而利用这些信息推断用户的查询意图，可以在很大程度上提高系统的推荐准确性.

表 2　XML 关键字结果排序策略(高度表示根结点到叶子的路径长度，宽度表示关键字之间的距离)

方法	数据模型		排序策略考虑的因素							
			结果本身特征					PageRank	TFIDF	向量空间模型
	树	图	结果高度	结果宽度	AD 对数目	边数目	关键字频率			
XKeyword[32]		√				■				
XRank[25]	✓		□	●				▲		
XSEarch[1]	✓				◇		△		○	◆
Fragment[40]	✓						△		○	◆
RSV[39]	✓		□				△		○	
SAIL[30]	✓		□	●			△		○	
CLCA[37]	✓		□	●			△		○	
XReal[23]	✓		□	●			△		○	
XBridge[24]	✓		□	●			△		○	
EASE[36]		√	□	●			△		○	
ISO&IRO[38]		√	□	●			△		○	
BLINKS[35]		√	□			■	△	▲	○	

6 高效算法

6.1 基于树模型的查询算法

基于树模型的查询语义都是基于 LCA 提出的，如何高效计算两个结点的 LCA 是其中的核心问题. 为此，大多数方法需要使用路径编码. 本节我们首先介绍已有方法所使用的编码方案，然后介绍两类典型算法：(1) 基于归并连接(Merge-Join)的方法. (2) 基于索引连接(Index-Join)的方法.

6.1.1 路径编码

Dewey 编码[41]方案为每个元素结点设定的编码由该元素的父结点编码和该元素的局部顺序值组成. 如图 9 中每个结点右边第一行所示,对一个元素结点 e 来说,Dewey 编码是由“.”分隔的数字串,其中最后一个数字表示 e 的局部顺序值,该值表示 e 在其父结点的所有孩子中自左至右的顺序,最后一个数字之前的数字串表示 e 的父结点的编码.

图 9 展示了一个 XML 文档 D_3 及其结点编码,左上角是 D_3 的 DTD. 图中每个结点右边第 1 行的编码是通常使用的 Dewey 编码[41],第 2 行编码是文献[42]提出的扩展 Dewey(Extended Dewey)编码,第 3 行带下划线的编码是文献[43]提出的 JDewey 编码.

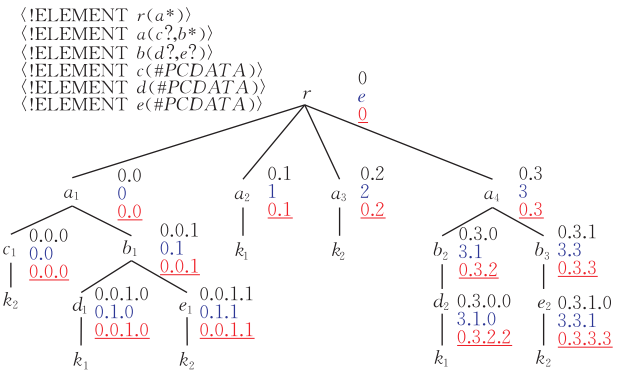


图 9 XML 文档 D_3 及路径编码示例

6.1.2 查询处理思路

为了处理给定的关键字查询 $Q=\{k_1,k_2,\cdots,k_m\}$,现有方法需要为每个关键字 k_i 构建包含元素编码的倒排表 L_i. 假设给定查询为 $Q_1=\{k_1,k_2\}$,则基于 D_3 的 Dewey 编码倒排表内容为 $L_1=\{0.0.1.0,0.2,0.3.0.0\}$,$L_2=\{0.0.0,0.0.1.1,0.1,0.3.0.0\}$. 已有方法使用 B^+ 树存储倒排表中的编码.

查询处理的基本思路可以概括为以下两步:

(1) 定位每个关键字对应的倒排表 L_i;

(2) 在 m 个倒排表中定位满足语义要求的结果.

现有方法在进行第 2 步处理时,本质上是采用基于连接(join)的方法,通过计算多个 Dewey 编码的公共前缀来找到包含所有关键字的结点. 现有方法通过改编关系数据库中的归并连接(Merge-Join)和索引连接(Index-Join)来处理所有的 Dewey 编码. 归并连接算法以从小到大的顺序扫描 m 个倒排表中的所有编码,通过使用栈来计算多个编码对应的包含所有关键字的最长公共前缀. 和归并连接不同,索引连接算法的提出基于以下观察:假设 v 是包含某个关键字的结点,w 是距离 v 最近的包含其它关键字的结点,u 是 v 和 w 的 LCA,则任何包含 v 的 ELCA 或者 SLCA 结点都不可能是 u 的后代结点. 因此索引连接算法通过使用 v 的编码在其它关键字对应的倒排表中查找距离其最近的编码,从而可以避免处理很多无用编码,进而提升算法性能. 以下内容中,我们将从这两个方面介绍几种典型算法.

6.1.3 归并连接算法

DIL[25]算法. DIL 算法中,关键字 k_i 对应的倒排表中每个元素是一个三元组$\langle DID,ER,PL\rangle$,其中 DID 表示直接包含 k_i 的结点 n_i 的 Dewey 编码,ER 表示根据式(5)得到的 n_i 的全局重要性,PL 表示 k_i 在 n_i 中的相对位置列表,所有三元组按照 DID 升序为序,简单起见,后续的例子仅包含 DID 和 ER,不包含 PL. 除倒排表外,DIL 算法还需要维护两个数据结构:栈和堆. 栈用于求解满足语义的结果,堆用于保存 Top-K 结果. DIL 的求解过程可以简单描述为:以从小到大的顺序一次性扫描关键字对应的倒排表中的所有元素,每当碰到一个满足条件的结果,将其放入结果堆中,等所有元素处理完毕,输出堆中的结果即为 Top-K 查询结果.

假设给定查询 $Q_1=\{k_1,k_2\}$,其倒排表中的内容如图 10 中 L_1 和 L_2 所示,图 10 栈中每个元素是 $(n+2)$元组$\langle N,ER[1],ER[2],ContainsAll\rangle$,这里 n 为给定查询中关键字的个数,对于 Q_1 来说是四元组,其中 N 表示 Dewey 编码的一个分量,所有栈元素中的 N 自栈底到栈顶表示一个 Dewey 编码 DID,$ER[1]$ 表示 k_1 对应的编码评分(Element Rank),$ER[2]$表示 k_2 对应的编码的评分,$ContainsAll$ 表示该编码对应的结点是否包含所有关键字.

DIL 首先处理$\langle 0.0.0,85\rangle$,状态如图 10(a)所示. 接下来处理$\langle 0.0.1.0,32\rangle$,状态如图 10(b)所示,注意从(a)~(b)的过程中,首先将图 10(a)中的栈顶元素弹出,然后将 85 根据衰减规则得到 77 后向下传递给 0.0. 由于 0.0 是 0.0.1.0 的前缀,因此不再继续弹出栈顶元素,然后将 0.0.1.0 中的 1 和 0 对应的元素压入栈顶,得到图 10(b)的状态. 然后处理$\langle 0.0.1.1,93\rangle$,和上一步类似,首先弹出(b)中的栈顶元素,然后将其衰减值 28 传递给栈顶元素,最后将 0.0.1.1 中最后一个 1 对应的元素压入栈顶,状态如图 10(c)所示. 接下来处理$\langle 0.2,21\rangle$,当弹出图 10(c)中的栈顶元素后,状态如图 10(d)所

示，表示 0.0.1 对应的结点 b_1 包含了所有关键字，因此是一个满足条件的结果，随后 b_1 被放入结果堆中. 需要注意的是当发现 0.0.1 满足条件后，其 $ER[i]$ 中的值不再向下传递. 后续的处理类似，这里不再赘述.

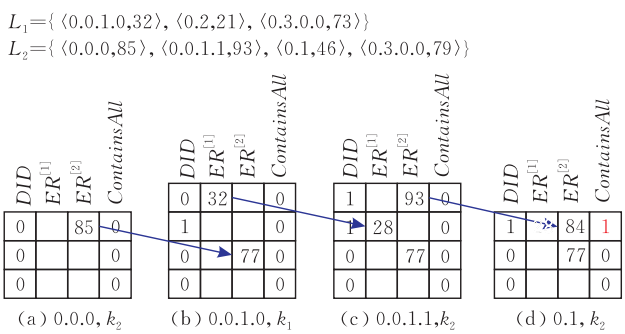


图 10　DIL 算法查询处理示意图

需要注意的是，该算法虽然在文献[25]中用来求解 ELCA 结点，但同样可用于求解 SLCA 结点. 文献[26]直接处理最短的倒排表，首先生成一组 ELCA 候选结果，然后检查候选结果中满足条件的结果，从而可以避免 DIL 算法扫描倒排表中所有元素的问题. 文献[44]进一步通过引入 Hash 计数的方法降低了算法的时间复杂度.

6.1.4　索引连接算法

IL 算法[27]. 对于求解满足 SLCA 语义的结果，文献[27]注意到 SLCA 的个数最多等于包含元素最少的倒排表的长度，例如给定两个关键字 k_1、k_2 以及包含 k_1 的结点 v，计算包含 k_1 和 k_2 的 SLCA 结点时，无需扫描 k_2 对应的倒排表 L_2 中的所有元素. 相反，只需要从 L_2 中找小于 v 编码的最大编码和大于 v 编码的最小编码，然后确定哪个编码能与 v 得到 SLCA 结点即可，因此文献[27]将包含结点 Dewey 编码的倒排表用 B^+ 树进行组织，通过索引查找的方式加速算法的执行. 显然，该方法只有在至少一个倒排表很短时才比较高效.

IMS 算法[45]. 当查询包含多个关键字时，IL 算法首先从两个倒排表中取两个结点计算其 LCA，然后将该结果与第 3 个倒排表中的结点进行比较，这种方法即使在结果很少的情况下，也可能在求解的过程中产生大量不必要的中间结果. 例如，对于图 11 所示的文档 D_4 而言，查询 $\{a,b\}$ 对应的 SLCA 集合是 $\{x_1,x_2,\cdots,x_{10}\}$，由于 a 对应的倒排表小于 b 对应的倒排表，已有方法会枚举每个 a 来计算可能的 SLCA 集合. 很明显，这种方法会造成很多冗余计算. 例如，$a_1 \sim a_{100}$ 之间的所有结点和 b_1 计算得到的 SLCA 结果都是 x_1. 基于以上观察，文献[45]提出了一种计算 SLCA 结果的高效算法 MS(Multiway-SLCA). MS 在计算 SLCA 集合时，每一步并非计算两个结点的 SLCA，而是基于锚点(Anchor Node)，每一步从所有关键字倒排表中取一个结点计算该组结点的 LCA，然后比较前一步得到的 LCA 和当前 LCA 的关系来确定前者是否为 SLCA.

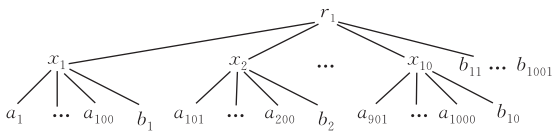


图 11　XML 文档 D_4[45]

基于 JDewey 的算法[43]. 文献[43]提出了一种 JDewey 编码，用来高效计算 ELCA 和 SLCA 结果. 如图 9 中带下划线的编码所示，每层[①]结点的 JDewey 编码在该层的分量具有唯一性，即层数和一个数字可以唯一确定一个结点，而 Dewey 编码需要所有数字分量组合起来才能唯一确定一个结点. JDewey 编码的另一个特性是：对于两个同层结点 v_1 和 v_2 来说，如果 v_2 的该层分量大于 v_1 在该层的分量，则 v_2 的所有孩子结点对应的层分量大于 v_1 的所有孩子对应的层分量.

基于 JDewey 编码，求结点间的 LCA 时，无需进行最长公共前缀的匹配操作. 对于给定的两个结点 v_1、v_2 及其 JDewey 编码 l_1 和 l_2，$l(i)$ 表示编码 l 中第 i 层的分量. 如果存在一个最大的 i 值使得 $l_1(i)=l_2(i)$，则可以确定第 i 层的结点 N 是 v_1 和 v_2 的 LCA. 由于 JDewey 编码通过层数和一个分量可以唯一确定一个结点，因此，可将 JDewey 编码按照不同层次分别进行存储，这和列存储的概念相同，可以在查询处理时提升系统的性能.

6.1.5　其它方法

针对 XSEarch[1] 的结果中除了直接包含关键字的结点名字可以相同外，其它结点对不能出现同名的情况，XSEarch 使用路径索引来解决该问题. 文献[29]针对 XSEarch 计算互连关系效率低的问题，提出 MDC 编码来快速推断给定结点的祖先结点名称，针对图 9 中的文档，其每个结点的 MDC 编码如结点右侧第 2 行部分所示.

除了得到所有满足条件的结果，实际中，还有很多工作着眼于得到和给定查询最相关的前 Top-K 个结果. 从 6.1.3 节的介绍可知，DIL 算法需要等到

① 层的概念和树中层的概念相同，根结点是第 1 层，依此类推.

所有元素都处理完毕后才能得到 Top-K 结果. 为了高效处理 Top-K 查询,文献[25]进一步提出了 RDIL 算法,该算法和 DIL 的区别在于:倒排表中元素的排序方式不再是按照编码大小排序,而是按照元素的评分值排序. 同时还将每个关键字对应的倒排表用一个独立的 B^+ 树进行组织,用来快速定位结点. 基于该数据结构,RDIL 使用扩展的 TA 算法来快速求解 Top-K 结果. 虽然 RDIL 在某些情况下可以快速得到所需的结果,但当所有的倒排表大小相近时,RDIL 仍需多次查找倒排表,从而造成较大的 CPU 和磁盘 I/O 开销. 基于此,文献[25]进一步提出了结合 DIL 和 RDIL 的混合式查询算法 HDIL,用以最大程度发挥 DIL 和 RDIL 的优势,避免其极端情况下的低效问题.

对于 MLCA 语义,文献[28]使用区间编码和栈,通过一遍扫描倒排表中的元素来计算满足 MLCA 语义的结果.

以上讨论的方法都是基于路径编码进行处理,文献[46]通过物化视图的方式高效地计算基于 SLCA 语义的关键字查询结果.

6.2 基于图模型的查询算法

6.2.1 基于模式图的方法

XKeyword[32]首先根据模式信息对 XML 文档进行分割,然后存入关系数据库并构建索引. 查询处理时,首先根据给定的关键字从模式图上得到所有的候选网络,即结构化查询表达式,然后基于关系查询引擎得到满足条件的结果. 文献[2]也是基于模式图,首先得到给定关键字对应的一组结构化查询表达式,然后分别处理得到最终的结果. 文献[34]考虑到文献[32]在求解候选网络时代价高的问题,提出了一种比模式图更紧凑的实体图来得到给定关键字对应的结构化查询,然后基于实体图提出了实体路径索引和部分路径索引来加速查询处理.

6.2.2 基于数据图的方法

文献[35]提出了一种双层索引来加速有向图上关键字查询处理的效率,其索引的对象包括基于关键字的倒排表以及从结点到关键字的最短路径. 文献[36]提出的倒排索引和传统的索引不同,该索引的 key 是关键字对,而被索引的对象是包含该关键字对的最大 r 半径图(r-radius graphs)以及相应的评分. 文献[38]维护了两个索引,分别是基于关键字的倒排索引和对象间的链接表,用于存储有逻辑链接关系的对象对.

6.3 性能比较

假设给定文档的最大深度为 d,给定的查询为 $Q=\{k_1,k_2,\cdots,k_m\}$,k_i 对应的倒排表为 L_i,文档中的结点数量为 n. 不失一般性,假定 L_1 最短,L_m 最长. 现有方法在求解满足条件的结果时,需要两种基本操作:(1) 比较两个编码的大小,代价为 $O(d)$. (2) 在 B^+ 树中查找匹配编码,代价为 $O(d\log|L_m|)$.

如表 3 所示,由于 DIL 算法需要扫描所有编码,每次扫描一个编码的代价为 $O(dm)$,其时间复杂度为 $O(md\sum_{i=1}^{m}|L_i|)$. IL、IS、IMS 及 JDewey[43] 的时间复杂度均为 $O(md|L_1|\log|L_m|)$. 对于 ELCA语义来说,HC 算法通过使用 Hash 计数器来降低算法的时间复杂度,是一种空间换时间的方法.

表 3 XML 关键字查询算法特点比较

语义	算法	数据模型		编码策略	数据结构			时间复杂度	空间复杂度
		树	图		倒排索引	栈	Hash 表		
ELCA	DIL[25]	√		①	△	◆		$O(md\sum_{i=1}^{m}\|L_m\|)$	$O(d)$
	IS[26]	√		①	△	◆	●	$O(md\|L_1\|\log\|L_m\|)$	$O(d)$
	HC[44]	√		①	△	◆	●	$O(md\|L_1\|)$	$O(n)$
	JDewey[43]	√		③	△			$O(md\|L_1\|\log\|L_m\|)$	$O(1)$
SLCA	IL[27]	√		①	△	◆		$O(md\|L_1\|\log\|L_m\|)$	$O(\|L_1\|)$
	IMS[45]	√		①	△			$O(md\|L_1\|\log\|L_m\|)$	$O(m)$
	JDewey[43]	√		③	△			$O(md\|L_1\|\log\|L_m\|)$	$O(1)$
MLCA	MLCAS[28]	√		④	△	◆		$O(d\sum_{i\in[1,m]}\|L_i\|+\prod_{i\in[1,m]}\|L_i\|)$	$O(d)$
VLCA	VLCAStack[29]	√		②	△	◆		$O(d\sum_{i\in[1,m]}\|L_i\|+md\log(md)k)$	$O(md)$

注:其中 k 表示结果数量. 由于倒排索引一般都用 Hash 表组织,这里不再单独列出. Hash 表一列的值表示对应文献中 Hash 还用于其它方面. 编码策略一列中的数字代表的含义为:① Dewey. ② MDC. ③ JDewey. ④ 区间编码,其它参数见 6.3 节的描述.

需要注意的是,这些算法的效率和所用的数据集并无直接的关系,而是和被处理的倒排表中关键字的分布情况相关. 为了验证这些方法的特点,我们实现了所有 SLCA 和 ELCA 相关算法,并在文献[47]的实验部分进行了详细比较,感兴趣的读者可以参阅文献[47]获取详细的比较和分析. 这里仅给出不同方法的特点供读者参考.

从实现的角度看,DIL 的性能和需要处理的编

码数量成正比，多数情况下性能较低；IL 相对简单，且在多数情况下可以获得较好的性能；IMS 在通常情况下可以跳过更多的元素，但当查询结果较多时，其性能反而不如 IL 高效；JDewey[43] 虽然连接操作的代价较低，其最大问题在于当从较低层次发现一个结果后，从高层倒排表中删除祖先结点的代价很高，且层次越高，执行删除操作的代价越高，当查询结果多时，执行删除操作的代价会极大影响系统的整体效率.

7　结果展示

7.1　基于树模型的查询结果展示

7.1.1　基于元组的结果展示方式

该方式仅返回和关键字相关的信息. 对于给定的关键字查询 $Q=\{k_1,k_2,\cdots,k_n\}$ 来说，XSEarch[1] 的查询结果是一个 n 元组 $N=\langle v_1,v_2,\cdots,v_n\rangle$，其中 $n_i(1\leq i\leq n)$ 是直接包含 k_i 的结点且 N 中的结点满足 all 语义①. VLCA[29] 和 XSEarch[1] 的区别在于返回结果中多了一个根结点，即对于查询 Q 来说，返回结果的形式为 $(r;l(v_1):c_1,l(v_2):c_2,\cdots,l(v_n):c_n)$，其中 r 是 $v_1,v_2,\cdots,v_n$ 的 VLCA 结点；c_i 是 v_i 包含的值结点.

7.1.2　基于完整子树的结果展示方式

该方式展示以某个结点为根的整个子树. 如 XRank[25] 和 XKSearch[27] 的返回结果分别是 ELCA 和 SLCA 结点. 在此基础上，XRank 还支持用户查看以 ELCA 为根的子树内容. XReal[23] 首先判断哪种类型的结点和用户的查询相符，然后返回以该类型结点为根的子树.

7.1.3　基于路径子树的结果展示方式

该方式返回的是从 LCA 结点到关键字的路径所构成的子树. 和第 2 种结果展示方式相比，该方式去除了子树中不包含关键字的结点. 在此基础上，现有方法根据路径子树中结点所包含的关键字集合的包含关系又进一步分为 3 种结果展示方式：(1) 包含所有路径的子树. (2) 满足单调性和一致性的子树. (3) 满足单调性和一致性且去除冗余信息的子树.

包含所有路径的子树. 文献[48-49]将 XML 文档中的元素分为实体结点、属性结点及连接结点，基于此，提出了推断关键字角色的规则，即相对于 XQuery 表达式来说，该关键字相当于 where 子句中的谓词还是 return 语句中的返回表达式，进而提出了推断用户查询意图的规则，最后基于实体展示结果. 例如，对查询 $Q_3=\{\text{Mike}\}$ 来说，图 12 中包含“Mike”的最低实体是编码为 0.0 的 author，文献[48]为用户展示的结果如图 13 的 r_1 所示. 再如对查询 $Q_4=\{\text{Mike,XML}\}$ 来说，文献[48]为用户展示的结果如图 13 的 r_2 所示. 注意，两个结果中都有指向未展示内容的链接，如图 13 中的“+”结点所示，用户可以通过点击链接浏览其内容.

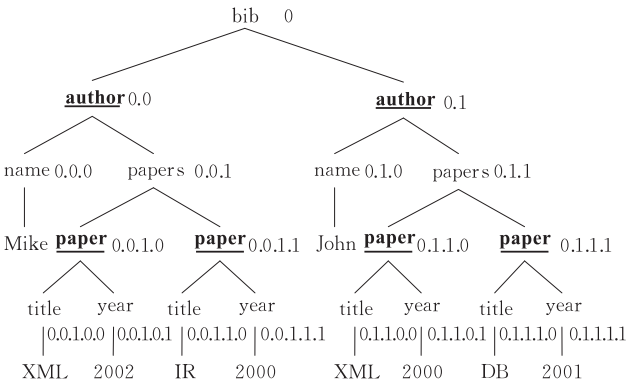


图 12　XML 文档 D_5，带下划线的结点是实体

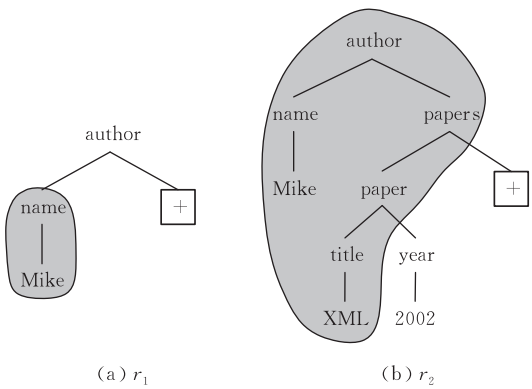


图 13　查询结果展示示例

该类工作中，SAIL[30] 返回的最小代价树 MCT 由某个结点 n 和它所对应的所有关键结点的关键路径构成. 文献[37]返回的紧凑连接树 CCTree (Compact Connected Tree)由 CLCA 结点 n 和从 n 到和其对应的关键字的路径构成.

满足单调性和一致性的子树. 文献[50]通过分析基于文本数据的关键字检索的特点，提出 XML 关键字查询技术应该满足的两个重要属性，即单调性(Monotonicity)和一致性(Consistency). 基于单调性和一致性的约束，文献[50]在展示结果时，仅为用户呈现从 SLCA 结点到各个相关匹配(定义见文献[50])构成的子树，对于查询 Q_3 和 Q_4 来说，返回的结果如图 13 中的阴影部分所示.

① 为了简单起见，这里的查询没有使用 XSEarch 中“label:keyword”的形式，且假设使用 all 语义.

满足单调性和一致性且去除冗余信息的子树. 虽然文献[50]提出的方法满足单调性和一致性的要求,文献[51]通过分析发现,基于文献[50]的方法得到的结果中,仍然存在冗余数据.文献[51]针对文献[50]可能丢失有用结果且在返回的结果中可能包含无用信息的问题,基于有效贡献者(Valid Contributor)的概念来判断以 LCA 为根的子树中哪些是有用结点,并在删除无用结点后返回从 LCA 结点到各个关键字构成的子树,该方法的返回结果也满足一致性和单调性的要求.

7.2 基于图模型的查询结果展示

基于图模型的结果展示工作中,基本语义是包含所有关键字的互连网络.根据对结果中结点和边的处理方式,可以分为 3 类工作:(1)基本互连网络.(2)互连实体网络.(3)Steiner 图.

基本互连网络. XKeyword[32]是该类工作的典型代表,其返回结果中每对结点间最多有一条边相连,展示时,根据结果的结构相似性对同构结果进行聚类,以便用户快速浏览所有可能的结果并定位自身感兴趣的结果集.

互连实体网络. 文献[34]的返回结果是基于实体语义的连接网络,表现在语义上是实体间的关系,表现在结构上,和文献[32]相比,除了某些实体结点外,其它结点被删除后,该结果不再连通.

Steiner 图. 为了最大限度保留结点间的多种连接方式,文献[36]返回的结果中任意两个结点间保留了其在数据图中的所有边,以便用户理解结果语义.

7.3 其它结果展示方式

文献[52-53]提出了对 XML 关键字查询结果进行区分的方法,该方法在查询结果过大时,可以对结果进行合理的概括,用以协助用户理解某个查询结果的特点.与此相对应,文献[54]提出了一种对查询结果间的差异性进行量化的机制来减轻用户主观判断结果相关性的负担.文献[55-56]通过使用相关反馈的方式来对查询结果进行过滤.

8 相关系统

研究者通过开发原型系统,对所提出的想法进行验证,极大促进了新技术的发展.下面我们根据已有原型系统的特点进行分类阐述.

8.1 查询生成相关的系统

FRISK[10]基于文献[9]提出的识别查询中无意义的关键字、修正错误拼写的关键字、识别有效的短语以及从文本中提取有效关键字的查询清洗技术,为用户展示高质量的查询清洗结果.

文献[15]为用户实时推荐近似的 URL,其功能通过 Firefox 的插件来展示.Yahoo! Finance① 支持模糊匹配的功能,根据用户的输入实时推荐相关的单词和短语来协助用户构造查询.

Tastier[19-20]和 InteractiveSearch[21]② 基于关系数据,为用户实时推荐与输入关键字匹配的结果,同时该系统支持模糊匹配的功能.Seaform[22]在查询表格中键入关键字,实时给出精确匹配的查询结果.与 Tastier 和 InteractiveSearch 相比,Seaform 可以根据不同属性指定关键字,增加了约束条件,提高了实时推荐的准确性,同时 Seaform 还可以对结果按照特定属性进行聚类展示,以方便用户浏览.INK[18]基于 XML 数据,可以针对关键字进行准确匹配,实时展示查询结果.

8.2 查询处理相关的系统

8.2.1 基于树模型的系统

XSEarch[1]主要展示"label:keyword"方式的查询接口和基于 TFIDF 的思想计算 Top-K 查询结果.

XRank[25]为用户展示基于 PageRank 思想排序后满足 ELCA 语义的查询结果.

EXTRUCT[57]展示基于文献[58]提出的 NTC(Normalized Total Correlation)和 NTPC(Normalized Term Presence Correlation)来提升查询结果的准确率和召回率.

XReal[59]主要展示如何推断用户的查询意图.针对用户提交的关键字查询,系统首先推断用户的查询意图并返回相应的结果.用户可以从中选择符合自身查询意图的结果进行反馈,之后 XReal 根据用户选定结果的特点为用户返回高质量的结果.同时,XReal 允许用户配置排序参数.

Timber[60]是最早开发的原型系统之一,目前支持嵌入 MLCA[28]和 MSP 语义[2]的 XQuery 查询.

TopX[61]是一个基于文本和半结构化数据的 Top-K 查询引擎,可以有效地支持基于结构和内容的模糊查询,并提供反馈机制来对查询结果进行过滤.

8.2.2 基于图模型的系统

XKeyword[32]针对图模型数据,基于关系存储,为用户展示聚簇后的查询结果.EASE[62]基于图模型数据,展示了如何以统一的方式查询异构文本、关

① http://finance.yahoo.com/
② http://dblp.ics.uci.edu/

系以及 XML 数据的问题，其查询语义和排序策略来自于文献[36].

8.3　结果展示相关的原型系统

XSeek[63]通过推断用户的查询意图，基于实体来展示结果. 目前，XSeek 支持文献[50]提出的结果生成策略.

TargetSearch[64]能自动识别查询目标并组合出相应的查询结果，这些查询结果满足文中提出的原子性和完整性(atomicity and intactness)要求.

eXtract[65]展示了如何根据文献[52]提出的结果特征抽取策略对查询结果进行概括，以方便用户理解返回结果的意义.

XSACT[66]主要基于文献[54]提出的结果比较策略为用户展现不同结果的差异性.

8.4　其它相关系统

虽然和 XML 关键字查询相关的技术已经得到了深入的研究，并产生了一大批具有代表性的原型系统，但都没有进入实际应用领域. 目前工业界广泛使用的数据库系统，如 Oracle，DB2 及 SQL Server 等，在对关键字查询的支持方面，都停留在对基于关系数据的关键字查询的处理，均不支持基于 XML 数据的关键字查询，一种简单的方法是通过支持 XQuery 的 contains 函数实现，如“//chapter[contains(.,'XML')]”，但这种方式本质上是文本检索，没有考虑 XML 数据的内在结构信息.

在基于 XML 数据的开源数据库方面，Timber 和 TopX 如前所述. 其它开源系统，如 BaseX①、eXist②及 dbXML③等对于 XML 关键字查询的支持都是通过支持 XQuery 的 contains 函数来实现，并非真正意义上的 XML 关键字查询处理系统.

9　研究展望

通过以上内容的分析可知，自从 XML 出现以来，针对 XML 关键字查询技术的研究就一直没有停止过，本文所述工作仅仅是其中的一部分，还有很多未提及的工作，如 INEX Workshop 上的工作、各大会议的 Workshop 等. 国内的研究[15-16,18-23,29-30,34,36-38,62,67]也一直没有间断. 这些工作为 XML 数据管理技术的研究与发展做出了重大的贡献，极大地增强了用户查询和利用 XML 数据的能力，但是就 XML 关键字查询而言，存在的问题依然很多. 从当前技术发展和现实应用的角度来看，有效性和高效性是 XML 关键字查询处理系统最终追求的目标，两者缺一不可.

有效性也指易用性，体现在如下几点：(1) 查询输入时，系统应该能够为用户反馈和当前输入信息相关的信息，以便用户轻松构造查询. (2) 每个返回结果本身所包含的语义信息可以解释其包含的关键字之间的关系，以便用户能够轻易理解返回结果所传达的信息. (3) 系统能够通过用户提交的关键字信息推断用户的查询意图，以便为用户返回满足其查询意图的结果. (4) 当返回结果和用户的查询意图存在偏差时，系统应该提供可行的方式对返回结果进行汇总、过滤或者依照不同的标准对结果进行聚类. 有效性的缺失意味着系统可能无法协助用户构造查询、返回结果不符合用户的查询意图或不可解释、没有相应的对结果进行归纳和概括的功能，这将直接造成系统用户的流失，影响系统的应用效果.

高效性指查询处理的效率. 对于 XML 关键字查询处理系统而言，高效性和有效性同等重要，这一点对基于 Web 服务环境中的应用而言尤其重要，这时系统需要响应大量用户的查询请求并提供实时反馈. 高效性缺失的直接后果是系统无法真正应用于实际中服务用户. 下面进行具体阐述.

9.1　有效性

9.1.1　有效的查询生成技术

问题：如何确定“相关”词汇和短语. 文献[68]为用户推荐的关键词仅限于在查询结果中经常出现的高频词. 实际应用中，用户需要的可能恰恰是具有代表性的低频词. 如何发现具有代表性的词汇，而不单单是用出现频率的高低来衡量相关性是要解决的关键问题，其核心是如何划定相关性计算的范围. 如果将没有意义的查询结果中包含的信息也考虑在内，则相关性计算的效果难以保证. 同时，XML 数据的层次结构特点将为排除不相关信息带来新的挑战.

9.1.2　有效的查询语义

问题 1：如何确定“实体”. 众所周知，关系数据库中关系的基本组成单位是元组，因此很自然的，基于关系数据库的关键字查询语义是以元组为基本单位的连通网络. 一个元组代表一个现实世界中的实体对象，可以表达完整的语义信息，而关系数据库中的元组和 XML 文档中的元素并非等价的语义单元. 已有的各种 XML 关键字查询语义以元素作为基本语义单元，这导致很多情况下，无法解释基于已有语义的 XML 关键字查询结果. 相反，如果可以有

① http://www.inf.uni-konstanz.de/dbis/basex/
② http://exist.sourceforge.net
③ http://sourceforge.net/projects/dbXML-core/

效获取实体信息,并返回基于实体或者实体间关系的结果,则可有效提升现有语义的表达能力.目前为止,还没有一种方法能够在不用人工干预的情况下准确识别实体信息.

问题2:如何扩展XML关键字查询的表达能力.目前的XSD支持数据类型的定义,当XML文档的XSD可用时,如果能在关键字查询中选择性的附加谓词信息,可有效提升关键字查询的语义表达能力.例如,从DBLP数据库中"找Mike发表于2002年以后的文章".同时,当数据库中大部分数据在某个属性上具有相同的属性值,仅有少部分数据具有区别于大多数数据的属性值时,用户需要使用不等于语义.例如,用户可能想了解NBA球员中非美国籍的球员都有谁;或者想了解在某个单位,除了汉族,谁是少数民族等等.如果没有简单谓词的协助,已有的关键字查询语义无法准确表达以上的查询需求.这种附加约束条件的匹配操作依赖于相关属性的类型和语义.

9.1.3 有效的排序机制

已有排序方法在进行结果排序时,考虑了:(1)元素之间的超链接;(2)祖先结点到后代结点的链接;(3)后代结点到祖先结点的链接;(4)单词的统计信息;(5)结果的结构信息.虽然这些工作取得了很大进展,实际的应用效果仍有值得改进的空间.

问题1:考虑结点权重的排序机制.基于关系数据的评测表明,结点权重会在很大程度上影响结果的排序质量[69].目前仅有文献[67]讨论了如何对XML数据中的结点名称设定权重.未来的排序机制应能体现权重对于结果排序的影响.

问题2:个性化的排序机制.数据是客观存在的,而用户的需求是个性化的.已有工作很少根据用户需求来提高检索结果的质量.虽然信息检索领域有很多成功经验可以借鉴,例如,通过分析查询日志和用户的点击流,系统可以返回个性化的查询结果;迄今为止,在XML关键字查询结果的排序问题上,支持个性化排序机制的工作还没有,其难点在于进行个性化排序操作时,考虑哪些影响排序结果的因素,这些因素的权重如何确定,如何评定结果排序的质量等.

问题3:不确定性XML数据的结果排序.大规模使用精确数据的时代已经过去,大多数应用都面临如何有效处理不确定性数据的困扰.数据的不确定性要求能够对结果进行有效的排序,支持Top-K查询是处理这些数据的本质需求.虽然Top-K查询处理机制已经得到了深入的研究,但对于不确定XML数据所增加的额外选项来说,研究还远远不够.

9.1.4 有效的结果展示功能

动态facet search指对结果不按事先设定的属性进行聚类,而是将结果根据有代表性的特征动态组合,以便用户快速定位感兴趣的结果.系统在展示结果时,提供动态facet search功能可有效协助用户从大量返回结果中定位所需信息,增强系统的易用性.

已有的研究主要集中在对查询结果以某种方式进行聚类或者提供facet search的能力.这些方法所依赖的聚类或者facet search的属性都是预先设定好的.很多时候,用户依然无法有效定位所需信息.以DBLP为例,如果用户想了解和XML关键字查询相关的文章,可在DBLP提供的facet search接口(http://dblp.l3s.de/?q=&newQuery=yes&resTableName=query_resultGGjnfG)中输入"XML keyword search"进行查询,结果显示,只有32个结果(注:查询时间为2010年11月18日),这显然远远少于DBLP中收录的有关XML关键字查询的文章数量.为了得到更多的结果,再次输入"XML keyword",返回结果只有37个;再次放宽查询条件,输入"XML",发现有1734个结果.显然,要找到所需的文章,用户不得不通过某个属性浏览所有返回结果才能找到和XML关键字查询相关的所有文章.

9.2 高效性

9.2.1 可扩展的高效算法

高效算法的重要性等同于有效的排序机制,有效的排序机制可以返回用户想要的结果,但高效算法是保证系统可用性的关键,扩展性是系统无法回避的,文献[69]的实验结果表明,面对大规模数据集时,已有基于关系数据的系统在扩展性方面表现不够好,这是至今为止还没有原型系统进入应用领域的重要原因.虽然还没有工作对基于XML数据的关键字查询处理算法的性能进行系统的比较和分析,我们认为,在XML关键字查询处理领域,同样存在类似的问题,尤其是面向Web服务环境中的应用.

9.2.2 高效的实时结果提示算法

为了实时显示和输入关键字相关的结果,研究者已经提出了很多基于磁盘的索引结构和算法[13,70-72].为了获得更好的交互速度,需要在服务器的内存中缓存大量索引和临时查询结果,当数据集增大的时候,内存需求量会急剧增加.和直接返回最终查询结果相比,实时结果提示对算法的高效性提出了更高的要求.目前仅有文献[18]研究了基于

ELCA 语义的结果提示功能，但提示的是单个 ELCA 结点，并非具有语义表达能力的数据片段，原因在于已有的数据片段构造方法非常耗时，不能满足在用户输入的同时进行实时结果提示的要求.

9.3 标准化评测体系

信息检索技术的发展已经向我们证明了标准化评价的重要性，这一点可以从 TREC① 出现以来，查询有效性在 6 年内提高了一倍②来得到直接证明.

INEX③ 针对基于内容(content-based)的 XML 检索构建了标准化的评测手段，对基于数据(data-based)的 XML 检索而言，还没有相应的组织来从事标准化评测体系的构建工作. 文献[69]基于相同的数据集和查询对已有的技术进行了测试，结果表明基于相同数据集和查询得到的评测结果比原文章中给出的数据差很多. 这可能是因为作者没有在自己的文章中说明测试时的一些调优手段和参数. 同样 TREC 和 INEX 的评测结果也印证了这一事实，该问题的存在使得读者无法准确比较不同技术的特点. 这可能导致在基于数据的 XML 检索领域，很难取得和信息检索领域以及基于内容的 XML 检索领域匹配的实用成果. 据我们了解，本文提到的原型系统都还没有进入实际应用领域，其中一个潜在的问题是每个系统的评测效果均来自研究者主观认定的评价标准和测试平台.

早在 2002 年，文献[73]就曾提到，针对某类特征的数据设计的检索方法，不应该在其它特征的数据上进行性能比对. 正如文献[74]所述，为了评价基于不同结构化数据的关键字查询处理策略的有效性，急需该领域的研究团体一起来构建全面、综合性的标准化评测机制，包括构建标准化的测试数据集和查询.

参 考 文 献

[1] Cohen S, Mamou J, Kanza Y, Sagiv Y. XSEarch: A semantic search engine for XML//Freytag J C eds. Proceedings of the 29th International Conference on Very Large Data Bases (VLDB). USA. Morgan Kaufmann Press, 2003: 45-56

[2] Yu C, Jagadish H V. Querying complex structured databases//Koch C eds. Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB). University of Vienna. ACM Press, 2007: 1010-1021

[3] Amer Y S, Botev C, Shanmugasundaram J. TeXQuery: A full-text search extension to XQuery//Feldman S I, Retsky M eds. Proceedings of the 13th International Conference on World Wide Web (WWW). Manhattan. ACM Press, 2004: 583-594

[4] Amer Y S, Lakshmanan L V, Pandit S. FleXPath: Flexible structure and full-text querying for XML//Weikum G, König AC, Deßloch S eds. Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD). Paris. ACM Press, 2004: 83-94

[5] Curtmola E, Amer Y S, Brown P, Fernàndez M. GalaTex: A conformant implementation of the XQuery FullText language//Florescu D, Pirahesh H eds. Proceedings of the 2nd International Workshop on XQuery Implementation, Experience, and Perspectives (XIME-P). Baltimore. ACM Press, 2005: 1024-1025

[6] Teahan W J, Wen Y, McNab R J, Witten I H. A compression-based algorithm for Chinese word segmentation. Computational Linguistics, 2000, 26(3): 375-393

[7] Sarawagi S, Cohen W. Semi-Markov conditional random fields for information extraction//Saul L, Weiss Y, Bottou L eds. Proceedings of the 18th Advances in Neural Information Processing Systems (NIPS). Vancouver. MIT Press, 2004: 1-8

[8] Mansuri I R, Sarawagi S. Integrating unstructured data into relational databases//Liu L, Reuter A, Whang K Y, Zhang J eds. Proceedings of the 22nd International Conference on Data Engineering (ICDE). Atlanta. IEEE Computer Society, 2006: 29

[9] Pu K Q, Yu X. Keyword query cleaning. Proceedings of the VLDB Endowment, 2008, 1(1): 909-920

[10] Pu K Q, Yu X. FRISK: Keyword query cleaning and processing in action//Ioannidis Y eds. Proceedings of the 25th International Conference on Data Engineering (ICDE). Shanghai. IEEE Press, 2009: 1531-1534

[11] Darragh J J, Witten I H, James M L. The reactive keyboard: A predictive typing aid. IEEE Computer, 1990, 23(11): 41-49

[12] Arnab N, Jagadish H V. Effective phrase prediction//Koch C eds. Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB). University of Vienna. ACM Press, 2007: 219-230

[13] Bast H, Weber I. The completesarch engine: Interactive, efficient, and towards IR&-DB integration//Hellerstein J eds. Proceedings of the 3rd Biennial Conference on Innovative Data Systems Research (CIDR). Asilomar: www.crdrdb.org, 2007: 88-95

[14] Chaudhuri S, Kaushik R. Extending autocompletion to tolerate errors//Cetintemel C eds. Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD). Providence. ACM Press, 2009: 707-718

[15] Wang J, Li G, Feng J. Automatic URL completion and prediction using fuzzy type-ahead search//Allan J eds. Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Boston. ACM Press, 2009: 634-635

① The Text REtreival Conference, http://trec.nist.gov/
② http://trec.nist.gov/overview.html
③ INitiative for the Evaluation of XML Retrieval, http://inex.is.informatik.uni-duisburg.de/

[16] Fan J, Wu H, Li G, Zhou L. Suggesting topic-based query terms as you type//Han W S eds. Advances in Web Technologies and Applications, Proceedings of the 12th Asia-Pacific Web Conference (APWeb). Busan. IEEE Computer Society, 2010: 61-67

[17] Raju S, Kumar S, Udupa R. Suggesting related topics in web search//Crestani F eds. Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR). Geneva. ACM Press, 2010: 705

[18] Li G, Feng J, Zhou L. Interactive search in XML data//Quemada J eds. Proceedings of the 18th International Conference on World Wide Web (WWW). Madrid. ACM Press, 2009: 1063-1064

[19] Li G, Ji S, Li C, Feng J. Efficient type-ahead search on relational data: A tastier approach//Cetintemel C eds. Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD). Providence. ACM Press, 2009: 695-706

[20] Li G, Ji S, Li C, Wang J, Feng J. Efficient fuzzy type-ahead search in tastier//Li F eds. Proceedings of the 26th International Conference on Data Engineering (ICDE). California. IEEE Press, 2010: 1105-1108

[21] Ji S, Li G, Li C, Feng J. Efficient interactive fuzzy keyword search//Quemada J eds. Proceedings of the 18th International Conference on World Wide Web (WWW). Madrid. ACM Press, 2009: 371-380

[22] Wu H, Li G, Li C, Zhou L. Seaform: Search-as-you-type in forms. Proceedings of the VLDB Endowment, 2010, 3(2): 1565-1568

[23] Bao Z, Ling T W, Chen B, Lu J. Effective XML keyword search with relevance oriented ranking//Ioannidis Y eds. Proceedings of the 25th International Conference on Data Engineering(ICDE). Shanghai. IEEE Press, 2009: 517-528

[24] Li J, Liu C, Zhou R, Wang W. Suggestion of promising result types for XML keyword search//Manolescu I eds. Proceedings of the 13th International Conference on Extending Database Technology (EDBT). Lausanne. ACM Press, 2010: 561-572

[25] Guo L, Shao F, Botev C, Shanmugasundaram J. XRank: Ranked keyword search over XML documents//Halevy A Y eds. Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data. California. ACM Press, 2003: 16-27

[26] Xu Y, Papakonstantinou Y. Efficient LCA based keyword search in XML data//Kemper A eds. Proceedings of the 11th International Conference on Extending Database Technology (EDBT). Nantes. ACM Press, 2008: 535-546

[27] Xu Y, Papakonstantinou Y. Efficient keyword search for smallest LCAs in XML databases//Özcan F eds. Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD). Baltimore. ACM Press, 2005: 537-538

[28] Li Y, Yu C, Jagadish H V. Schema-free XQuery//Nascimento M A eds. Proceedings of the 30th International Conference on Very Large Data Bases (VLDB). Toronto. Morgan Kaufmann Press, 2004: 72-83

[29] Li G, Feng J, Wang J, Zhou L. Effective keyword search for valuable LCAs over XML documents//Silva M J eds. Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM). Lisbon. ACM Press, 2007: 31-40

[30] Li G, Li C, Feng J, Zhou L. SAIL: Structure-aware indexing for effective and progressive top-k keyword search over XML documents. Information Sciences, 2009, 179(21): 3745-3762

[31] Hristidis V, Papakonstantinou Y. Discover: Keyword search in relational databases//Bernstein Phil eds. Proceedings of the 28th International Conference on Very Large Data Bases (VLDB). Hong Kong, China. Morgan Kaufmann Press, 2002: 670-681

[32] Hristidis V, Papakonstantinou Y, Balmin A. Keyword proximity search on XML graphs//Dayal U eds. Proceedings of the 19th International Conference on Data Engineering (ICDE). Bangalore. IEEE Computer Society, 2003: 367-378

[33] Cohen S, Kanza Y, Kimelfeld B, Sagiv Y. Interconnection semantics for keyword search in XML//Herzog O eds. Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management(CIKM). Bremen. ACM Press, 2005: 389-396

[34] Zhou J, Bao Z, Ling T W, Meng X. MCN: A new semantics towards effective XML keyword search//Zhou X F eds. Proceedings of the 14th International Conference on Database Systems for Advanced Applications (DASFAA). Brisbane. Springer, 2009: 511-526

[35] He H, Wang H, Yang J, Yu P S. BLINKS: Ranked keyword searches on graphs//Chan C Y eds. Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD). Beijing. ACM Press, 2007: 305-316

[36] Li G, Ooi B C, Feng J, Wang J, Zhou L. EASE: An effective 3-in-1 keyword search method for unstructured, semi-structured and structured data//Wang J T eds. Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD). Vancouver. ACM Press, 2008: 903-914

[37] Feng J, Li G, Wang J, Zhou L. Finding and ranking compact connected trees for effective keyword proximity search in XML documents. Information Systems, 2010, 35(2): 186-203

[38] Bao Z, Lu J, Ling T W, Xu L, Wu H. An effective object-level XML keyword search//Kitagawa H eds. Proceedings of the 15th International Conference on Database Systems for Advanced Applications (DASFAA). Tsukuba. Springer, 2010: 93-109

[39] Liu S, Zou Q, Chu W. Configurable indexing and ranking for XML information retrieval//Sanderson M eds. Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR). Sheffield. ACM Press, 2004: 88-95

[40] Carmel D, Maarek Y S, Mandelbrod M, Mass Y, Soffer A. Searching XML documents via XML fragments//Callan J P eds. Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR). Toronto. ACM Press, 2003: 151-158

[41] Tatarinov I, Viglas S, Beyer K S, Shanmugasundaram J, Shekita E J, Zhang C. Storing and querying ordered XML using a relational database system//Franklin M J eds. Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data. Madison. ACM Press, 2002: 204-215

[42] Lu J, Ling T W, Chan C Y, Chen T. From region encoding to extended Dewey: On efficient processing of XML Twig pattern matching//Böhm K eds. Proceedings of the 31st International Conference on Very Large Data Bases (VLDB). Trondheim. ACM Press, 2005: 193-204

[43] Chen L J, Papakonstantinou Y. Supporting top-k keyword search in XML databases//Li F eds. Proceedings of the 26th International Conference on Data Engineering (ICDE). California. IEEE Press, 2010: 689-700

[44] Zhou R, Liu C, Li J. Fast elca computation for keyword queries on XML data//Manolescu I eds. Proceedings of the 13th International Conference on Extending Database Technology (EDBT). Lausanne. ACM Press, 2010: 549-560

[45] Sun C, Chan C Y, Goenka A K. Multiway slca-based keyword search in XML data//Williamson C L eds. Proceedings of the 16th International Conference on World Wide Web (WWW). Banff. ACM Press, 2007: 1043-1052

[46] Liu Z, Chen Y. Answering keyword queries on XML using materialized vews//Buchmann A eds. Proceedings of the 24th International Conference on Data Engineering (ICDE). Cancún. IEEE Press, 2008: 1501-1503

[47] Zhou J, Bao Z, Wang W, Ling T W. Fast SLCA and ELCA computation for XML keyword query based on set intersection operation//Proceedings of the 28th International Conference on Data Engineering. Washington, USA, 2012: 905-916

[48] Liu Z, Chen Y. Identifying meaningful return information for XML keyword search//Chan C Y eds. Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD). Beijing. ACM Press, 2007: 329-340

[49] Liu Z, Chen Y. Return specification inference and result clustering for keyword search on XML. ACM Transactions on Database Systems, 2010, 35(2): 1-47

[50] Liu Z, Chen Y. Reasoning and identifying relevant matches for XML keyword search. Proceedings of the VLDB Endowment, 2008, 1(1): 921-932

[51] Kong L, Gilleron R, Lemay A. Retrieving meaningful relaxed tightest fragments for XML keyword search//Kersten M L eds. Proceedings of the 12th International Conference on Extending Database Technology (EDBT). Saint Petersburg. ACM Press, 2009: 815-826

[52] Huang Y, Liu Z, Chen Y. Query biased snippet generation in XML search//Wang J, Li T eds. Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD). Vancouver. ACM Press, 2008: 315-326

[53] Liu Z, Huang Y, Chen Y. Improving XML search by generating and utilizing informative result snippets. ACM Transactions on Database Systems, 2010, 35(3): 1-43

[54] Liu Z, Sun P, Chen Y. Structured search result differentiation. Proceedings of the VLDB Endowment, 2009, 2(1): 313-324

[55] Schenkel R, Theobald M. Structural feedback for keyword-based XML retrieval//Lalmas M eds. Proceedings of the 28th European Conference on IR Research (ECIR). London. Springer, 2006: 326-337

[56] Pan H, Schenkel R, Weikum G. Fine-grained relevance feedback for XML retrieval//Myaeng S H eds. Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR). Singapore. ACM Press, 2008: 887

[57] Termehchy A, Winslett M. EXTRUCT: Using deep structural information in XML keyword search. Proceedings of the VLDB Endowment, 2010, 3(2): 1593-1596

[58] Termehchy A, Winslett M. Keyword search for data-centric XML collections with long text fields//Manolescu I eds. Proceedings of the 13th International Conference on Extending Database Technology (EDBT). Lausanne. ACM Press, 2010: 537-548

[59] Bao Z, Lu J, Ling T W. XReal: An interactive XML keyword searching//Huang J eds. Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM). Toronto. ACM Press, 2010: 1933-1934

[60] Jagadish H V, Al-Khalifa S, Chapman A et al. Timber: A native XML database. The VLDB Journal, 2002, 11(4): 274-291

[61] Theobald M, Bast H, Majumdar D, Schenkel R, Weikum G. TopX: Efficient and versatile top-k query processing for semistructured data. The VLDB Journal, 2008, 17(1): 81-115

[62] Li G, Feng J, Wang J, Zhou L. An effective and versatile keyword search engine on heterogenous data sources. Proceedings of the VLDB Endowment, 2008, 1(2): 1452-1455

[63] Liu Z, Walker J, Chen Y. XSeek: A semantic XML search engine using keywords//Koch C eds. Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB). University of Vienna. ACM Press, 2007: 1330-1333

[64] Liu Z, Cai Y, Chen Y. TargetSearch: A ranking friendly XML keyword search engine//Li F eds. Proceedings of the 26th International Conference on Data Engineering (ICDE). California. IEEE Press, 2010: 1101-1104

[65] Huang Y, Liu Z, Chen Y. eXtract: A snippet generation system for XML search. Proceedings of the VLDB Endowment, 2008, 1(2): 1392-1395

[66] Liu Z, Natarajan S, Sun P, Booher S, Meehan T, Winkler R, Chen Y. XSACT: A comparison tool for structured search results. Proceedings of the VLDB Endowment, 2010, 3(2): 1581-1584

[67] Liu D, Wan C, Chen L, Liu X. Automatically weighting tags in XML collection//Huang J eds. Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM). Toronto. ACM Press, 2010: 1289-1292

[68] Tao Y, Yu J X. Finding frequent co-occurring terms in relational keyword search//Kersten M L eds. Proceedings of the 12th International Conference on Extending Database Technology (EDBT). Saint Petersburg. ACM Press, 2009: 839-850

[69] Coffman J, Weaver A C. A framework for evaluating database keyword search strategies//Huang J eds. Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM). Toronto. ACM Press, 2010: 729-738

[70] Bast H, Chitea A, Suchanek F M, Weber I. ESTER: Efficient search on text, entities, and relations//Kraaij W eds. Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR). Amsterdam. ACM Press, 2007: 671-678

[71] Bast H, Mortensen C W, Weber I. Output-sensitive autocompletion search//Crestani F eds. Proceedings of the 13th International Conference on String Processing and Information Retrieval (SPIRE). Glasgow. Springer, 2006: 150-162

[72] Bast H, Weber I. Type less, find more: Fast autocompletion search with a succinct index//Efthimiadis E N eds. Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR). Seattle. ACM Press, 2006: 364-371

[73] Voorhees E M. The philosophy of information retrieval evaluation//Peters C eds. Proceedings of the 3rd Workshop of the Cross-Language Evaluation Forum (CLEF). Rome. Springer, 2002: 355-370

[74] Chen Y, Wang W, Liu Z, Lin X. Keyword search on structured and semi-structured data//Cetintemel C eds. Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD). Providence. ACM Press, 2009: 1005-1010

ZHOU Jun-Feng, born in 1977, Ph. D., associate professor. His research interests include XML structured query processing and XML keyword search.

MENG Xiao-Feng, born in 1964, Ph. D., professor, Ph. D. supervisor. His research interests include Web data management, Cloud data management, mobile data management, XML data management, flash-aware DBMS, privacy protection.

Background

The appearance of XML is the inherent requirement of Web-based applications, such as e-business, e-government, banking business, press industry, etc, where XML plays an important role, it actually becomes a de facto standard for information representation and exchange over the Internet.

With the expanding of application fields, huge volumes of data are organized or exported in XML format. How to effectively utilize XML data has been a hot research issue along with the ever increasing of XML-based applications, and many research work appear on various top-level conferences and journals each year. Empowering users to access databases using simple keywords can relieve users from understanding the complex query language and possibly fast evolving data schemas.

In this paper, we give an overview of the state-of-the-art techniques for supporting keyword search on XML data, including query generation, query semantics, ranking mechanism, query processing algorithms and result presentation. We organize existing works according to the architecture of an XML keyword search system, and give an in depth analysis to the representative techniques. Finally, according to the inherent requirement of effectiveness and efficiency of an XML keyword search system, we discuss applications that are built upon keyword search, such as query generation, query semantics, ranking mechanisms and result presentation, and identify the challenges and opportunities of future research to advance this field.

This research was partially supported by the grants from National Science and Technology Major Project (No. 2010ZX01042-002-003), the National Natural Science Foundation of China (Nos. 61073060, 61070055, 91024032), the Fundamental Research Funds for the Central Universities, the Research Funds of Renmin University of China (10XNI018).