可查询 XML 数据压缩技术概述①

童李文 杨良怀 龚卫华 古 辉 (浙江工业大学 计算机科学与技术学院 浙江 杭州 310032)

摘 要: 随着互联网技术的迅速发展，XML 已经成为 Web 上信息表示和数据交换的事实标准。XML 数据的冗余性影响了 XML 数据传输、查询处理等方面的效率，数据压缩是解决冗余的一种途径。介绍了典型的可查询 XML 压缩技术，阐述了各种压缩技术的优缺点，比较了各压缩技术的压缩率、压缩时间、支持查询的类型等；最后总结了可查询 XML 压缩技术的不足之处及其发展的趋势。

关键词: XML 查询；XML 压缩；可查询压缩

Survey of Queriable XML Data Compression Techniques

TONG Li-Wen, YANG Liang-Huai, GONG Wei-Hua, GU Hui

(School of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310032, China)

Abstract: With the rapid development of Internet, XML is the de facto standard for exchanging and presenting data on the Web. The redundancy of XML data affects XML data transmission, query and other aspects. Data compression is a way to resolve the problem. This paper compares various XML data compression techniques, whose cons and pros are compared and analyzed in terms of compression ratio, compression times, and query performance and so on. The survey is concluded by pointing out the future work to be done.

Keywords: XML query; XML compression; queriable compression

1 引言

可扩展标记语言(XML)已成为 Internet 上数据存储、交换的标准，XML 的自描述特性使得它允许用户根据数据描述甚至数据关系来制定标记，同时还允许用户开发特定领域的标记语言。这种格式化的数据一方面方便不同领域间的数据表示；另一方面却使得 XML 文档存在较大的数据冗余，导致 XML 数据存储、交换、管理等代价增加，妨碍了 XML 技术的有效应用。

根据对查询的支持程度，XML 数据压缩可以分为不可查询与可查询的数据压缩技术。前者主要是用来解决 XML 的数据冗余问题，以满足网络宽带和存储空间的需求，其压缩率都高于传统的数据压缩技术，但是不能够对压缩数据进行直接查询，查询之前必须完全解压缩文档，增加了系统的负担。因此，如何在没有完全解压缩的情况下进行随机存取和查询处理成了研究的一个热点。可查询的 XML 数据压缩技术的具有多方面优点：减少查询数据时对磁盘的访问次数，节省时间；减少数据处理时对内存的要求；减少对网络带宽的要求。

本文分析了当前可查询 XML 压缩技术，如 XGrind[1,2]、XPRESS[3]、XQZip[4]、XCQ[5,6]、XCpaqs[7]、XQueC[8]、XSeq[9]、XBzip[10]、QXT[11]；讨论了各压缩技术的优缺点，对各个压缩技术的性能进行比较，最后指出了当前可查询 XML 压缩技术存在的不足及其发展趋势。

2 非可查询压缩技术

XMill[12,13]是 Hartmut.Liefke 和 Dan.Suciu 于 2000 年提出的第一个 XML 专用压缩转换算法，主要目标是获取高压缩率，尽可能地减少 XML 文档的容量，提高空间利用率。包括以下几个步骤[14]:

1) 分离 XML 文档的结构和数据，即将标签、属

① 基金项目:国家高技术研究发展计划(863)(2007AA01Z153);浙江省基金(Y1090096,Y1080102)
收稿时间:2009-06-23

性和数据值分开;

2) 用字典编码的方式表示起始标签，用指定的字符替换结束标签，然后进行压缩;

3) 元素按照类型和路径组成字符串数据存入容器(信息块)进行初步编码;

4) 对每种类型的元素重复步骤 3。

最后，XMill 将进行初步编码的容器和结构数据重新整合后使用压缩工具 GZip 压缩，得到压缩文档。XMill 的优点是文档压缩比高。缺点是：1)它不支持对压缩文档的随机查询，仅当整个文档解压后方可查询;2)由于采用 GZip 进行最后的压缩处理，因此也继承了 GZip 的缺点，当文件容量比较小(小于 20K)时压缩效果很差。

在 XMill 之后又出现了众多非可查询的压缩技术，如 AXECHOP[15],XComp[16],XMLPPM[17],XWRT[18]。此类压缩技术特点是压缩率高，却不支持压缩查询，需要对压缩文档进行完全的解压之后才能进行查询等操作。而 XML 文档在应用当中最重要的仍然是查询处理问题，需要对大规模的压缩文档进行操纵(查询、更新)，可查询的 XML 压缩技术便应运而生。

3 可查询压缩技术

3.1 XGrind

XGrind 于 2002 年提出，它采用一种同构转换策略将 XML 文档转换成压缩形式。这种变换的优点在于保留了原始文档的句法结构和语义信息，这就意味着压缩文档也能使用 SAX 和 DOM 解释器来解析而无须预先解压缩。实际上，我们可以把经 XGrind 压缩的 XML 文档看成是原来的 XML 文档，只是其中的标签和数据被压缩编码替换了，包括以下几个主要步骤：

1) 利用 DTD 解析器解析文档的 DTD，为含有枚举类型的属性建立符号表;

2) 利用 XML 解析器对文档进行第一次扫描统计元素和不含枚举类型值的属性的信息;

3) 对文档进行第二次扫描，对枚举型的属性值采用 Log2K 对 K 个值进行映射压缩；对标签等结构数据进行简单的字符加代码替换;

4) 对一般的属性和数据利用静态的 Huffman 编码进行同态压缩。

XGrind 的优点是支持可查询压缩。缺点是：1)该方法只针对 XML 文档的冗余标签进行压缩，没有处理文档中的路径和数据重复问题，其压缩率远远低于 XMill 和 XMLPPM；2)无法对所有的复杂查询进行直接解析，对于包含范围谓词查询，必须对压缩文档进行对应部分的局部解压缩。

3.2 XPRESS

XPRESS 采用了与 XGrind 压缩算法相似的同构压缩思想，与原来 XML 文档无论在语义还是结构上都保持相同。但 XPRESS 采用了一种特殊而新颖的编码方式，即一种称为逆算术编码的编码方式，将文档树中的整个路径映射到一个实数区间中。步骤如下：

1) 第一遍扫描文档，利用分析器来计算每个不同的元素出现的频率次数，这个频率是作为逆算术编码的输入；同时，类型推导引擎用来推断元素数据值的类型，同时产生独立编码中的统计类型。

2) 第二遍扫描文档，利用编码器将每一条树路径进行逆算术编码;

3) 对数据进行类型独立的编码，最后形成压缩文档。

XPRESS 所采用逆算术编码的一个重要特性是一条树路径对应的实数区间会被该路径的后缀路径所对应的实数区间所包含，这一特性可以有效地提高查询的效率；数据类型自动推导功能避免了人工干预。对数值型数据的范围查询不用解压缩就能直接进行操作，而枚举型数据和字符型数据的范围查询则必须先执行部分解压缩。算法的缺点是：1)由于要对文档进行两次扫描，压缩的时间长；2)所采用的数据类型识别过于简单。

3.3 XQueC

与之前的压缩算法只在静态纯文本上操作不同的是，XQueC 将对 XML 文档的操作与数据库技术相结合，形成了一种自适应的压缩器[19]。与 XMill 算法类似，XQueC 也是基于将 XML 文档进行数据和内容分离的思想，然后对每个容器用算法进行压缩。算法包括以下步骤：

1) 将 XML 文档数据分离成：结构树、容器、结构概要，同时得到结构数据和字符信息等内容数据;

2) 对于结构，采用简单的二进制编码方法来对元素和属性名称进行编码;

3) 将所有路径相同，类型相同的数据放入同一个容器；然后对容器进行分组，采用不同的压缩算法;

4) 对文档中的数据既可以采用 ALM 算法，也可

以采用 Huffman 编码;

5) 最后将结构和数据两部分压缩的结果结合形成压缩文档。

XQueC 的优点是: 1)它使用各种结构信息, 如 DataGuide[20], 结构树及其它索引, 支持在压缩文档进行查询操作, 特别是 XQuery 查询, 而 XQuery 自身拥有的完整操作集使得在压缩文档的查询处理更加有效; 2)支持复杂的 XPath 查询: 如嵌套的谓词; 同时支持更复杂的操作, 如聚合、连接等。缺点是压缩过程中的为了保存结构信息以及指向单独压缩的数据项的信息, 要产生很多指针, 从而引起极大的空间开销。

3.4 XQzip

XQzip 与 XMill 一样采用将 XML 文档结构和内容数据分离后进行分别压缩的思想, 不同的是该算法引入一种称作结构索引树(SIT)的索引结构, 这种结构有利于查询操作中节点的定位, 支持对压缩文档的查询。它包括以下步骤:

1) 将 XML 文档数据分离, 得到结构数据和字符信息等内容数据;

2) 数据项用 Gzip 进行压缩存储到块中去, 利用 HashTable 对块数据进行访问;

3) 利用索引构造器来构造文档的结构索引树;

4) 所得到的块、HashTable、SIT 构成 XQzip 的压缩仓库, 形成压缩文档。

算法的特点是: 1)引入 SIT 去除 XML 文档中的重复结构来提高查询执行的效率; 2)压缩率接近 XMill, 比已有的其他可查询压缩系统高; 3)支持复杂的 XPath 查询: 如多重查询、深度内嵌的谓词、有混合的基于数值和基于结构的检索条件的谓词的检索。缺点是由于在压缩的过程中要构造索引树, 所以压缩消耗的时间长, 同时当文档比较大时, 构造的索引树要占用过多的内存。

3.5 XCQ

2006 年提出来的 XCQ 利用 DTD 中的信息来提高压缩处理的效率,采用 DTD 树和 SAX 事件流解析(DSP)算法。DSP 算法包括两个策略: 一个是基于路径划分分组(PPG)策略; 另一个是块统计签名(BSS)索引策略。前者的主要思想是将结构流与数据流相分离存储。与 XMill 不同的是: 结构流用的是 DTD 中的信息进行编码; 数据流是基于 DTD 中的路径, 而不是简单的名称。PPG 数据流中的数据块可以作为一个独立的单元被压缩和解压。这种分割策略使得用户通过对压缩文档中与查询条件相匹配的那块进行解压就可以得到查询的结果, 而不用全部解压。后者用来在 PPG 中帮助块检索。XCQ 对 XML 文档采用结构和内容数据相分离分别压缩处理的方式,但与 XMill 不同的是 XCQ 借助 DTD 得到文档的结构信息。它包括以下步骤:

1) 使用 DTD 解析器解析 DTD 文档, 生成 DTD 树;

2) 采用 SAX 解析器将相应的 XML 文档解析生成事件流, 传送给 DSP 模块;

3) 结合 DTD 树和 SAX 事件流, 形成数据流集, 数据流集中的每个数据块分别使用 Gzip 进行压缩;

4) 利用 BSS 模块为 PPG 中的每个数据块生成 BBS 索引;

5) 最后将 DSP 产生的结构流、BSS 模块产生的索引和 PPG 数据流形成压缩文档。

算法的优点是: 1)压缩率比较高, 接近 XMill; 2)引入 PPG 策略, 使得用户不用全部解压文档就可得到查询结果; 3)引入 BBS 索引策略用来在 PPG 中提高块检索效率。缺点是: 1)由于需要对文档的 DTD 进行解析生成 DTD 树, 所以压缩/解压的时间消耗更多; 2)只支持简单的精确和范围查询, 不支持更复杂的查询操作, 如聚合、连接等。

3.6 XCpaqs

XCpaqs 是李建中等人于 2004 年提出的 XML 压缩技术, 其压缩思想与 XGrind 相似, 不同的是 XCpaqs 对标签和路径分别编码、分别压缩, 对少量特殊类型的数据使用单独的压缩算法。它主要包括以下步骤:

1) 将 XML 文档结构和数据分离, 用(路径编码, 路径内容)的形式表示文档; 以路径为基本单位对结构进行压缩;

2) 对同类路径下的内容采取相同的方法进行压缩;

3) 通过数据类型识别器识别各种类型的数据, 再采用相应的压缩方法对数据进行压缩;

4) 最后将路径编码和数据编码结合形成压缩文档。

它的特点是: 1)它能部分地在内存中执行 XPath 查询, 支持复杂的和长的 XPath 查询; 2)在结构和内容进行分离时, 对标记和路径分别编码。缺点是不支

持更复杂的操作，如聚合、连接等。

3.7 XSeq

2005 年提出的 XSeq 与 XMill 思想类似，将文档结构和数据相分离，然后采用 Sequitur[21]算法进行压缩，Sequitur 算法中用产生式规则来表示输入序列中的重复结构，每一个重复串形成语法规则库中的一个规则，用一个非终结符来表示。然后每读入一个字符对规则库进行调整。最后的规则库里就包含了输入序列中的所有重复串，并得到原序列的一个层次结构表示。XSeq 主要步骤包括：

1) 用 SAX 解析 XML 文档，将结构和数据分离，存放到不同的容器中；

2) 再用 Sequitur 算法对各个容器进行压缩形成压缩文档。

算法的优点是：1)Sequitur 算法处理容器时生成结构容器和数据容器的索引，不用引入额外的索引结构，提高查询效率；2)查询时避免对不相干数据的扫描，对离散数据的查询效率高。3)支持复杂的 XPath 查询：如多重查询、嵌套的谓词检索。算法的缺点是：1)当文件容量非常大时，处理离散数据的效率低；2)由于采用了 Sequitur 算法，生成语法规则和数据索引消耗的时间过长；3)算法的压缩率低，与 GZIP 接近。

3.8 XBzip

2006 年提出来的 XBzip 引入 XBW[22]转换方法将 XML 文档标签树转换成等价的线性表示，即利用两个数组来表示 XML 文档树结构，第一个数组包含有以适当顺序排列的树标签；第二个数组中存储的是 XML 文档结构树的二进制编码。然后再利用 PPM 算法上面所得到的两个表格进行编码压缩。数据则是根据结构进行分类压缩。算法主要包括以下步骤：

1) 解析文档，利用 XBW 将文档转换成两个数组；

2) 再利用 PPM 算法上面所得到的两个数组进行编码压缩，得到压缩文档。

算法的优点是：1)压缩效果好，适合以中等速度对大数据集进行查询；2)XBW 转换方法简化了 XML 文档结构树，提高编码效率。缺点是：1)由于 XBW 转换消耗时间长，所以压缩/解压时间长；2)为了支持查询，需要存储索引信息，使得压缩文档的容量增大。3)只支持简单的范围查询、选择查询等。

3.9 QXT

2007 年提出的 QXT 结合 XWRT 来支持部分解压缩查询。将标签进行归类：字类、数字类、空格类、字符类等；然后采用适当的算法进行压缩，得到压缩文档。算法主要步骤：

1) 对 XML 文档进行第一次扫描，构造标签类的字典，并计算每一类的频率；

2) 第二次扫描文档，将数据依据路径存储到不同的容器中去；

3) 采用 LZ77[23]或者 LZMA[24]算法对容器进行压缩，得到压缩文档。

算法的优点是：1)压缩时候需要的内存比 XBzip 小很多；2)不需要额外的解析器、查询处理器等，避免兼容性问题的产生。缺点是：1)压缩率低，比 GZIP 略好，远低于 XMill；2)由于要对文档进行两次扫描，压缩消耗的时间多。3)只支持简单的精确、范围查询和多重查询。

4 可查询压缩技术性能比较

至此本文在以上两章中介绍了一些比较知名的 XML 压缩算法，重点介绍可查询压缩算法的工作步骤，同时指出了各个算法的优缺点。为了更深入的了解这些算法，我们对各压缩技术中的一些主要性能：压缩率、压缩时间、支持查询的类型等进行简单的比较。由于其中大部分可查询压缩技术的源代码没有公开，所以对查询性能的比较只能引用相关参考文献的描述。

从表 1 总结可以得到一些结论。

XGrind 的压缩率约为 XMill 的一半，因为需要对文档进行两次的扫描才能完成压缩。XGrind 的查询性能比 XMill 好 2 到 3 倍，XGrind、XPRESS 都支持 XQuery、XPath 等查询语言，是因为它们都采用了同型转换策略来保持原有的 XML 文档结构，但是它们只支持简单的数值数据和字符数据的查询，却不支持复杂的查询处理。XPRESS 压缩率比 XMill 好 80%左右，因为 XPRESS 采用了逆算术编码方案对 XML 文档中元素的树路径进行编码，压缩与解压时间比 XGrind 略好，同时由于逆算术编码的特性，使得 XPRESS 查询时间比 XMill 快了将近 3 倍。XQueC 的压缩率比 XPRESS 略逊，这是由于它需要对文档的信息进行预处理。XQzip 的压缩率可以与 XMill 相比拟，大约比 XGrind 好 16.7%。XQzip 的压缩和解压缩速度可与 XMill 和 GZIP 相比拟，XQZip 的查询性能约是 XGrind

表 1 支持查询的 XML 压缩技术性能比较

压缩技术	压缩率	压缩时间	底层压缩方案	解析器	模式	支持查询类型	随机访问	转换方式
XMill	Gzip 两倍	----	GZip、BZip	--	--	--	不支持	异构
XGrind	低于XMill	约 XMill 两倍	Huffman	SAX	有	Exact-match, Prefix-match, partial-match,	不支持	同态
XPRESS	高于 XGrind	约为 XMill 两倍	逆算术编码、Huffman	SAX	无	exact match, range queries exact match	不支持	同态
XQueC	接近 XPRESS	约为 XMill 两倍	ALM、Huffman	SAX	无	Exact-match, Prefix-match, fuzzy-match, Joins, nested query	支持	异构
XQZip	接近XMill	约为 XMill 两倍	GZip	SAX	无	Exact-match, Prefix-match, multi-predicate, deeply nested predicate,	不支持	异构
XCQ	接近XMill	约为 XMill 两倍	Bzip2、GZip	SAX	有	Exact-match, Prefix-match, multi-predicate, deeply nested predicate,	不支持	异构
XCpaqs	接近XMill	与 XMill 相近	Huffman	SAX	无	Exact-match, Prefix-match	不支持	异构
XSeq	接近GZip	比 XQzip 稍快	Sequitur	SAX	无	Exact-match, Prefix-match, multiple nested predicates	支持	同态
XBzip	接近GZip	约 XMill 两倍	XBW、PPMdi	DOM	无	Parent ,child, block of children, rank and select query	支持	异构
QXT	接近 XBzip	约 XMill 两倍	LZ77、LZMA	FSA	无	Exact-match, Multiple queries.	不支持	异构

的 13 倍。XCQ 也具有较好的查询性能。XSeq 采用基于语法的压缩算法 Sequitur，支持随机访问，压缩率与 GZIP 接近，查询时间比 XQzip 稍快，但是压缩的时间约是 XGrind 的 3 倍。TREECHOP 通过一次扫描完成对文档的压缩，适用于数据流的压缩，常用于网络文件传输，压缩/解压的速度比 XQZip 稍慢。XBzip 比 XGrind、XPRESS、XQzip 的压缩率与查询性能都很出色。QXT 压缩过程中占用的内存比 XBzip 小很多，但压缩率不高，低于 XMill，比 GZIP 好 20%~30%，压缩/解压缩的时间较长。

5 可查询压缩技术的发展趋势

尽管 XML 压缩技术已经取得了一些进展，但是仍然存在不足，特别是可查询 XML 压缩技术仍不成熟。在查询速度和压缩率上还没有取得比较满意的成果，有必要进一步研究可查询 XML 压缩技术。可查询 XML 压缩在以下方面有待进一步深入研究：

1) 支持快速导航操作：许多 XML 应用，如协作文档编辑系统，依赖有效的树遍历，使用诸如 DOM 标准接口。Halverson[25]等人论证了将导航和结构相结合对于查询计算是最有效的。因此，要求存储模型能支持 XML 文档进行有效的快速遍历。

2) 有效压缩：有效的压缩需要有高压缩率、较短的压缩与解压时间以及较少的内存消耗。

3) 支持插入和删除操作：已有不少的文章提出的算法支持有压缩查询，但是不能执行诸如节点插入、删除、更新操作。这些操作在许多的数据库应用中是很常见的操作。

4) 支持有效的连接操作：目前的查询优化技术过多的是使用结构连接，即用一个常量时间操作来决定两个节点之间的祖先-后代关系，这就要求任何通用的XML 存储模式都要支持在恒量时间内完成相应的操作。

5) 有效的查询索引结构：许多应用研究要求在它们的数据域中增加额外的、有效的索引。也就是数据库系统所使用的存储模式必须提供一个简单、有效、稳定的方式来引用它所存储的数据项。支持在压缩文档上进行查询、插入、删除、更新等操作。

6) 较少的人工干预、提供友好的用户接口：要想XML 压缩技术成为一种基本的、通用的压缩工具来应用，就必须减少在应用过程中的人工干预，同时具有友好的用户接口。

总之，由于 XML 数据表示比关系数据更加复杂和灵活，处理的代价也更大，因而有关 XML 数据压缩还有很多问题需要解决。国际上，这方面的研究也还是刚刚起步，特别是在支持复杂查询的 XML 压缩算法方面还有许多问题需要解决。

参考文献

1 Tolani P, Haritsa J. XGRIND: A Query-Friendly XML Compressor. Proc. of the 18th International Conference on Data Engineering, San Jose, CA, March 2002. New York: IEEE Computer Society, 2002.225－234.

2 Min J, Park M, Chung C. A compressor for effective archiving, retrieval and update of XML documents. ACM Transactions on Internet Technology, 2006,6(3): 223－258.

3 Min J, Park M, Chung C. XPRESS: a queriable compression for XML data. Halevy AY, Ives ZG, Doan A, et al. Proc. of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, Jun 2003. New York: ACM Press, 2003.122－133.

4 Cheng J, Wilfred Ng. XQzip: Querying Compressed XML Using Structural Indexing. Bertino E, Christodoulakis S, Plexousakis D, et al. LNCS 2992: Proc. of the 9th International Conference on Extending Database Technology, Heraklion, Greece, March 2004. Berlin, Germany: Springer-Verlag, 2004.219－236.

5 Ng W, Lam WY, Wood PT, Levene M. XCQ: a queriable XML compression system. Knowledge and Information Systems, 2006,10(4):421－452.

6 Lam W, Ng W, Wood P, Levene M. XCQ: XML compression and querying system. Proc. of the Twelfth International World Wide Web Conference, Budapest, Hungary, May 2003.

7 Wang H, Li J, Luo J, Zhenying He. XCpaqs: compression of XML document with XPath query support. Proc. of the 2004 International Conference on Information Technology: Coding and Computing, Las Vegas, Nevada, USA, April 2004. New York: IEEE Computer Society, 2004.354－358.

8 Arion A, Bonifati A, Manolescu I, Pugliese A. XQueC: Pushing Queries to Compressed XML Data. Johann CF, Peter CL, Serge A, et al. Proc. of the 29th International Conference on Very Large Data Bases, Berlin, Germany, 2003. San Francisco, CA: Morgan Kaufmann Publisher Inc, 2003.1065－1068.

9 Yongjing Lin Y, Zhang Y, Li Q, Yang J. Supporting efficient query processing on compressed XML files. Haddad H, Liebrock L M, Omicini A, et al. Proc. of the 2005 ACM Symposium on Applied Computing, Santa Fe, New Mexico, USA, March 2005. New York, USA: ACM Press, 2005.660－665.

10 Ferragina P, Luccio F, Manzini G, Muthukrishnan S. Compressing and searching XML data via two zips. Carr L, Roure DD, Iyengar A, et al. Proc. of the 15th International World Wide Web Conference, Edinburgh, Scotland, UK, May 2006. New York, USA: ACM Press, 2006.751－760.

11 Skibinski P, Swacha J. Combining efficient XML compression with query processing. Ioannidis YE, Novikov B, Rachev B, et al. LNCS 4690: Proc. of the 11th East-European Conference on Advances in Databases and Information Systems, Varna, Bulgaria, Oct 2007. Berlin, Germany: Springer-Verlag, 2007. 330－342.

12 Liefke H, Suciu D. XMill: an efficient compressor for XML data. Chen Weidong, Naughton J F, Bernstein

PA. Proc. of the 2000 ACM SIGMOD International Conference on Management of Data , May, Dallas, Texas, USA. New York: ACM Press, 2000.153 − 164.

13 Liefke H, Suciu D. An extensible compressor for XML data. SIGMOD Record, 2000,29(1):57 − 62.

14 魏裕凯.XML 压缩技术的研究与应用[硕士学位论文].武汉:华中科技大学, 2006.

15 Leighton G, Diamond J, Miilder T. AXECHOP: A grammar-based compressor for XML. Proc. of the 2005 IEEE Data Compression Conference, Snowbird, UT, USA, March 2005. New York: IEEE Computer Society, 2005.467 − 484.

16 Li W. XComp: An XML compression tool. [Master's thesis],Waterloo Canada:University of Waterloo, 2003.

17 Cheney J. Compressing XML with multiplexed hierarchical PPM models. In Proc. of IEEE Data Compression Conference(DCC 2001), Snowbird, UT, USA, March 2001. New York: IEEE Computer Society, 2001.163 − 172.

18 Skibinski P, Grabowski S, Swacha, J: Fast transform for effective XML compression. Proc. of the IXth International Conference CADSM 2007, Lviv, Ukraine,Publishing House of Lviv Politechnic National University, 2007.323 − 326.

19 孟晓峰,周龙骥,王珊.数据库技术发展趋势.软件学报, 2004,15(12):1822 − 1836.

20 Goldman R, Widom J. Dataguides: Enabling Query Formulation and Opeimization in Semistructured Databases. Proc. of VLDB, 1997.

21 Nevill-Manning CG,Witten I.H. Linear-time, incremental hierarchy inference for compression. Proc. of the Data Compression Conference (DCC). Snowbird, UT, USA, March 1997. New York: IEEE Computer Society, 1997.135 − 143.

22 Ferragina P, Luccio F, Manzini G, Muthukrishnan S. Structuring labeled trees for optimal succinctness, and beyond. IEEE Focs, 2005.184 − 193.

23 Ziv J. Lempel A. A universal algorithm for sequential data compression. IEEE Transactions on Information Theory, 1977,23(3):337 − 343.

24 7-zip compression utility, http://www.7-zip.org

25 Halverson A, Burger J, Galanis L, Kini A, Krishnamurthy R, Rao AN, Tian F, Viglas S, Wang Y, Jeffrey F. Naughton, DeWitt DJ. Mixed Mode XML Query Processing. Proc. of the 29th International Conference on Very Large Databases (VLDB), Berlin, Germany, 2003. San Francisco, CA: Morgan Kaufmann P