XML 压缩文档上的数值更新方法*)

罗震霄[1] 和菊珍[1] 王晓玲[1] 艾丽君[2] 周傲英[1]

(复旦大学计算机科学与工程系 200433)[1] (上海宝信软件股份有限公司 上海 201203)[2]

摘 要 近年来,XML 已成为 Web 上信息交流和资源共享的主要载体。但 XML 自身的自冗余特性限制了它的普遍应用。目前,已经有研究成果提出了 XML 的压缩方法。压缩的 XML 文档能够有效利用存储空间,节省网络带宽。在实际应用中,经常需要对压缩存储的 XML 文档进行更新。对于大的压缩文档,如果先解压再更新,会消耗大量时间,因此,高效的更新方法应该避免解压缩文档,在压缩的 XML 文档上直接进行更新操作。本文针对压缩 XML 文档中的数值类型(包括整型和浮点型)数据,研究了在保持压缩状态条件下如何进行有效的数值更新,提出了基于 XPRESS 实现的 Naïve 数值更新方法,以及修改 XPRESS 编码方法实现的更为高效的 Pivot 数值更新方法。通过大量的实验证明,Pivot 数值更新方法不仅能够提供高效的更新处理,而且保持了 XPRESS 的高压缩率。

关键词 XML 压缩,XML 更新,查询处理

Efficient Numeric Update Method over Compressed XML

LUO Zhen-Xiao[1] HE Ju-Zhen[1] WANG Xiao-Ling[1] AI Li-Jun[2] ZHOU Ao-Ying[1]

(Department of Computer Science, Fudan University, Shanghai 200433)[1] (Shanghai Baosight Software Co., Ltd., Shanghai 201203)[2]

Abstract XML has become the de-facto standard for exchanging information on theWeb. However, XML data is recognized as verbose since its heavily repeated tags introduce significant redundancy. In order to save disk space and network bandwidth, a variety of compressing methods have been presented. Practically, Query and Update operations are two most frequently used operations. Efficient Update methods are required if there is a need to modify stored compressed XML data. In this paper, we focus on update problem of numeric data in compressed XML. Firstly, we make formal definition and classification of update types of numeric data. Secondly, we show major challenges and bottlenecks when dealing with the problem. Then, a naïve update method for compressed XML data using XPRESS approach is presented. In order to improve performance, a novel method - Pivot method is designed. Experiment results with DBLP data set show that the Pivot method achieves better performance yet not comprising on compression ratio.

Keywords XML compression, XML update, Query processing

1 问题描述

本文针对压缩 XML 文档中的数值类型(包括整型和浮点型)数据,研究了在保持压缩状态条件下如何进行有效的数值更新,提出了基于 XPRESS 实现的 Naïve 数值更新方法,以及修改 XPRESS 编码方法实现的更为高效的 Pivot 数值更新方法。

XPRESS 是目前 XML 压缩方法中同态压缩的代表,其压缩后的文档保留了原文档结构,对部分查询可以实现"避免解压缩"的直接处理,是一种灵活的压缩策略。因此,本文在 XPRESS 压缩方法的基础上,探讨数值更新问题。XPRESS 对 XML 文档中的数值应用了差值编码方法(differential encoding),这使得压缩文档可以在避免解压缩的条件下支持数值类型的匹配查询和区间查询。

针对上述编码方式,对压缩 XML 文档进行数值更新时,先从压缩文档读入数据,判断是否满足指定的更新条件。如果满足,则对当前数据进行更新并判断是否需要更新最小值以及其他元素。完成所有更新操所后将数据编码写回原压缩 XML 文档中。在判断更新条件时,我们采用 XPRESS 提供的查询处理方法[4,5]。从上述流程中,我们注意到两个问题:更新后原有最小值(下文简称为 min 值)是否仍为当前同路径下所有数值中的最小值;以及当最小值发生改变时怎样对整个压缩文档进行更新。这两个问题是数值更新的核心问题,在已有的工作中没有涉及。鉴于此,本文重点关注这两个问题,提供了相关的解决策略。

2 Naïve 数值更新方法

定义 1(XPath 的值谓词) 对于形如:

$Q=a_1/\cdots/a_i[predicate]\cdots/a_j//a_{j+1}\cdots/a_n$

的 XPath 查询,如果其中的"predicate"可表示为一个数值 value 的函数表达式:predicate=f(value)。

上述更新函数 $f(x)$ 是关于 x 的多项式,则称这样的 predicate 为 XPath 的值谓词(Value Predicate)。数值更新函数 $f(x)$ 可表示为:

$f(x)=C+Ax+Bx^2*g(x)$

其中,A、B、C 为常数,$g(x)$ 为关于 x 的多项式。包含值谓词的 *path* 表示为:$path=sPath\odot vPre$,其中,*sPath* 为 Simple Path, *vPre* 为 Value Predicate。

对于不带值谓词的数值更新,通过 XPRESS 查询处理找到了满足 *sPath* 的数值 *value*,针对更新函数 $f(x)$,高效的更新操作必须满足以下条件:

a)为避免最小值不稳定时间耗费,min 值位置须保持不变。即:

$\forall x_1<x_2: f(x_1)<f(x_2)$

这要求 $f(x)$ 为单调函数。

b)为避免差不稳定时间耗费,更新前后所有数据与 min

*)基金项目:Sybase 项目资助。**罗震霄** 本科生,主要研究方向为 XML 压缩与查询;**和菊珍** 硕士研究生;**王晓玲** 博士,讲师;**艾丽君** 工程师;**周傲英** 教授,博士生导师,研究方向:数据库系统、数据挖掘和流数据管理、WEB/XML 数据管理、P2P 计算和系统等。

的差必须保持不变。即：

$\forall x_1, x_2 : x_1 - x_2 = f(x_1) - f(x_2)$

$\therefore \forall x_1, x_2$：

$x_1 - x_2 = A(x_1 - x_2) + B(x_1^2 * g(x_1) - x_2^2 * g(x_2))$

$\therefore A=1, B=0$

这要求 $f(x)$为一次函数且其一次项系数为 1。

由于一次项系数为 1 的一次函数是单调函数，综合条件 a)、b)得：当且仅当 $B=0, A=1$ 时，不带值谓词的更新问题存在高效的方法。将 min 更新为 min+C 即可，无需更新其他数据。该更新的时间复杂度为 O(1)。

当 $B!=0$ 或 $A!=1$ 时，由于所有值相对 min 更新的幅度不同，无法用 O(1)的方法实现复杂度为 O(n)的更新，因此不存在高效的更新方法，必须对符合 *sPath* 的每一个数据进行更新。其时间复杂度为 O(n)。

对于带有值谓词的数值更新，在更新过程中不仅要考虑更新对象所处的路径，还要考虑更新对象本身是否满足更新的 Value Predicate 条件。对所有满足 path 路径条件的 *value*：

当 *value* 不是 min 值时，如果更新后的值不影响 min，即 f(value)≥min，对 value 的更新不会引起 min 的改变。其时间复杂度为 O(1)。

如果更新后的 value 影响当前的 min 值，即 f(value)< min 时，value 更新后将成为新的最小值，引起最小值的改变。其他所有数据必须根据 f(value)进行重新编码。其时间复杂度为 O(n)。

当 value 本身就是 min，并且该 value 更新后变小，即 f(min)<min 时，min 更新后仍然为最小值，但是 path 下的其他数值型数据 value 必须根据产生的新最小值 f(min)进行重新编码，其时间复杂度为 O(n)。

当 value 为 min，而且更新以后不会使 value 发生任何变化，即 f(min)=min 时，不需要进行任何更新操作。

但当更新后得到的值比 min 大，即 f(min)>min 时，必须遍历所有相同 path 下的 value 寻找新的最小值，再根据新的最小值进行重新编码，其时间复杂度为 O(n)。

3 Pivot 数值更新方法

从 Naïve 更新方法中，我们注意到更新效率由于受到编码方法本身的限制而难以提高。为此我们提出了 Pivot 更新方法：首先，计算出每一组特定路径下所有数值的中心点 pivot，对每个 pivot 都产生一个序号作为它的索引，通过序号可以对 pivot 直接读取。在压缩源文档时，将选取的 pivot 附加存储在压缩文件上，同一路径下所有数值都基于它们的 pivot 进行差值编码。在一段时间内，用一个固定的 pivot 值近似动态变化的数据中心点，这段时间内更新的数值仍然基于原 pivot 编码。当更新操作的次数达到指定阈值时，重新计算并选取新的 pivot。

为缩小编码范围，pivot 需要尽可能接近实际数据的中心点，分别利用数学中的平均值、Euclidian 距离[9]和 Chebychev 距离[9]选取一组数据的中心点 pivot：

1. MeanPivot：对一组数据$\{x_i\}$ ($i=1, 2, \cdots, n$)，假设 max 和 min 分别为最大值和最小值，则 *mean* 值为：

$$mean=\frac{1}{2}(\max+\min)$$

假设 $M(x_i)$为任意 x_i 与 *mean* 的距离，

$M(x_i)=|x_i - mean|$

当 $M(x_i)$最小时，所对应 x_i 的值称为 MeanPivot。

2. EuclidianPivot：对一组数据$\{x_i\}(i=1, 2, \cdots, n)$，令，$E(x_i)=\sqrt{\sum_{j=1}^{n}(x_i - x_j)^2}$，当 $E(x_i)$最小时，所对应 x_i 的值称为 EuclidianPivot。

3. ChebychevPivot：对一组数据$\{x_i\}(i=1, 2, \cdots, n)$，令，$C(x_i)=\sum_{j=1}^{n}|x_i - x_j|$，当 $C(x_i)$最小时，所对应 x_i 的值称为 ChebychevPivot。

Pivot 方法对非数值型数据的编码方式与 XPRESS 相同。对数值型(包括整型，浮点型)数据的编码方式如下所示：

Encode Model	Pivot Index	Data Value

其中，第一项 Encode Model 表示对不同长度的数据采用不同的编码策略，用 8 个位(bit)来标示数据值的编码方式。共有三种方式：

1)u8：正数和零 00000000，负数 00000001

2)u16：正数和零 00000010，负数 00000011

3)u32：正数和零 00000100，负数 00000101

其中，绝对值相同的数据编码方式相同，区别仅在于编码的最后一位，'0'表示正数和零，'1'表示负数。

根据 Encode Model 所表示的编码方式对数值进行基于 pivot 的差值编码。差值编码记录于第三项 Data Value 中。

第二项 Pivot Index 由两部分组成，分别为索引长度和索引值自身，如下所示：

Pivot Length	Index

因为索引是变长的，所以在 Pivot 索引编码中，第一项 Pivot Length 表示索引的长度，我们用 8 个位进行标示。

表 1 Pivot Length 对应 Index 编码方式

Pivot Length	Index 编码方式
00000001	u8
00000011	u16
00000101	u32

Index 的值表示该数据所对应 pivot 的序号。可以利用 Index 直接读取 pivot 值。

对于不带值谓词的数值更新，进行更新之前，通过 XPRESS 查询处理找到所有满足 *sPath* 的数据 value，针对更新函数 $f(x)$，当 $B=0, A=1$ 时，$f(x)$是一次项系数为 1 的一次函数。只要将 pivot 更新为 pivot+C 即可，无需更新其他值。该更新的时间复杂度为 O(1)。

当 $B!=0$ 或 $A!=1$ 时，不存在高效的更新方法，必须对符合 sPath 的每一个数据进行更新，因此只存在复杂度为 O(n)的更新算法。

对于带有值谓词的数值更新，进行更新之前，通过 XPRESS 查询处理找到所有路径表达式满足 sPath 且值满足 vPre 的数据 value，当更新操作的次数未达到指定阈值时，value 在更新后依然基于原 pivot 编码，并不改变 pivot 值。如果更新次数达到阈值，需要重新计算 pivot。每次更新的时间消耗指更新次数未达阈值的情况，其时间复杂度为 O(1)。

4 实验及分析

我们对所提出的 Naïve 方法和 Pivot 方法在不同大小的 DBLP[8]文档上进行了大量的更新实验以考查它们的效率。对于 Pivot 策略，根据 pivot 选取的不同，可分为 MPivot 方法、EPivot 方法和 ChPivot 方法。

(下转第 144 页)

总结 本文主要研究方位关系的层次表示与推理。以方位带为基础的方位关系层次表示存在弊端，将方位关系泛化后得到的关系不互斥，因此，本文提出了另一种方位关系的层次表示，即基于空间粒度的层次表示。论文介绍了区域覆盖层次结构，分别从点对象和矩形对象两方面来研究方位关系的层次表示与推理。实验表明，推理的结果符合人们的认知，我们提出的方法是可行的。

参 考 文 献

1 Hirtle S, Jonides J. Evidence of Hierarchies in Cognitive Maps. Memory and Cognition, 1985, 13(3):208～217
2 Guttman A. R-trees: a Dynamic Index Structure for Spatial Searching. In:Proc. of ACM SIGMOD Int Conf on Management of Data, 1984,14(2):47～57
3 Samet H. The Design and Analysis of Spatial Data Structures. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990
4 Car A, Frank A. General Principles of Hierarchical Spatial Reasoning: The Case of Wayfinding. In:the Proc. of the 6th Int Symposium on Spatial Data Handling, 1994. 646～664
5 Clementini E, Di Felice P. Topological Invariants for Lines. IEEE Trans on Knowledge and data Engineering, 1998,10(1):38～54
6 Guo Ping, Ye Lian, Fan Li. Qualitative spatial reasoning based on fusion combinative table. In:Proceedings of the 2005 international symposium on intelligence computation and applications, 2005. 691～696
7 Schlieder C. Reasoning about ordering. In:A. U. Frank and W. Kuhn, eds. Spatial Information Theory - A Theoretical Basis for GIS, Proc. COSIT'95, 1995, 988:341～349
8 Freksa C. Temporal Reasoning Based on Semi-Intervals. Artificial Intelligence, 1992,54(1-2):199～227
9 Peuquet D, Zhan C-X. An algorithm to determine the directional relationship between arbitrarily-shaped polygons in a plane. Pattern Recognition, 1987, 20(1):65～74

(上接第 107 页)

表 2 实验文档特征

	Doc1	Doc2	Doc3	Doc4	Doc5
大小 MB	1.42	7.14	12.5	16.6	20.4
元素个数	3701	17703	31260	41212	48540
最大深度	6	6	6	6	6
平均深度	2.90	2.90	2.90	2.90	2.90

所有的实验都在一台配置为 Intel Pentium 4，1.80GHz 处理器，256MB 内存的机器上执行，并采用 Java 语言实现。实验文档的特征如表 2 所示。根据 DBLP 元素的特点，我们针对路径"/dblp/inproceedings/year"下的数值进行更新，实验文档 Doc1—Doc5 在该路径下所有数值的最小值均为 1972。我们选用四类不同的更新操作如表 3 所示。

表 3 各类更新实例

	path	$f(x)$	种类
Up1	dblp/inproceedings/year[text()='2000']	$x-50$	1
Up2	dblp/inproceedings/year[text()='1972']	$x-2$	2
Up3	dblp/inproceedings/year[text()='1999']	$x+10$	3
Up4	dblp/inproceedings/year	$x+15$	4

对于第一类更新，Naïve 方法需要找到新的最小值并对整个文档重新进行编码，而 Pivot 方法只需对满足条件的数据进行更新。文档较小时，由于搜索整个文档所需要的时间开销有限，各种方法的更新效率差别不大。当文档大小超过 10MB 时，Pivot 方法的优势开始明显。随着数据集的增大，Naïve 方法与 Pivot 方法的差距逐渐增大。三种 Pivot 方法中 MeanPivot 为最优选择。

对于第二类更新，Naïve 方法需要对整个文档重新编码，Pivot 方法操作避免了差不稳定性，因此复杂度为 $O(1)$。随着文档增大，Naïve 方法与 Pivot 方法的差距逐渐增大。对 20.4MB 的文档，Pivot 方法的效率几乎为 Naïve 方法的两倍。在三种 Pivot 方法中，EuclidianPivot 为最优选择。

对于第三类更新，Naïve 方法和 Pivot 方法的操作相同，分别只更新 min 值和 pivot 值。四种方法的效率相当。

对于第四类更新，由于 Pivot 方法只要读取 pivot 值而无需读取整个文档，其效率远好过 Naïve 方法。为了衡量出 Pivot 方法相对 Naïve 方法的优势，我们采用 CompareCost 来表示 Pivot 方法相对 Naïve 方法所提高的效率。

$$CompareCost=\frac{t(Naiv)-t(Pivot)}{t(Naive)}$$

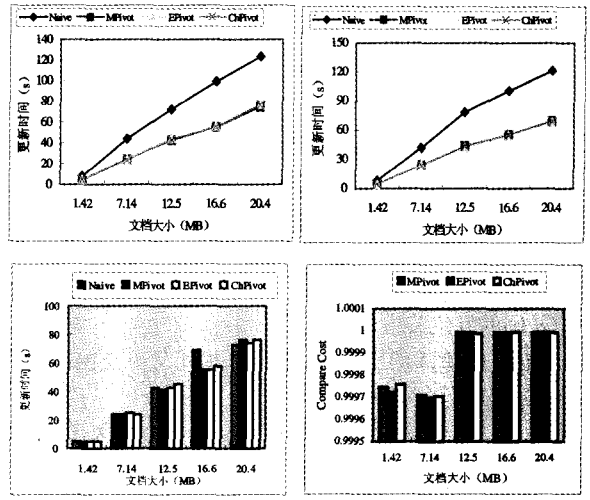

图 1 实验结果

结论及后继工作 本文探讨了压缩 XML 文档中数值类型数据的更新问题。提出了三种有效的数值更新算法，并且通过大量的实验证明：Pivot 数值更新方法的更新效率在各种情况下均好于 Naïve 更新方法，且其压缩率并没有受到影响。在遵循"避免解压缩"原则的条件下，我们将继续考虑压缩 XML 文档上非数值类型数据的更新问题及 XML 键约束对更新一致性的影响等问题。

参 考 文 献

1 Liefke H, Suciu D. XMILL: An Efficient Compressor for XML Data. In:SIGMOD Conference, 2000. 153～164
2 Tolani P M, Haritsa J R. XGRIND: A Query-Friendly XML Compressor. ICDE, 2002. 225～234
3 Min Jun-Ki, Park Myung-Jae, Chung Chin-Wan. XPRESS: A Queriable Compression for XML Data. In:SIGMOD Conference, 2003. 122～133
4 He Juzhen, Ng W, Wang Xiaoling, et al. An Efficient Co-operative Framework for Multi-query Processing over Compressed XML Data. DASFAA, 2006. 218～232
5 和菊珍，彭敦陆，王晓玲，周傲英．优化分布式环境中的多个 XML 查询．计算机科学，2005，32(增刊 A)
6 XML. http://www.xml.com/
7 XPath. http://www.w3.org/TR/xpath20/
8 DBLP. http://dblp.uni-trier.de/xml/
9 金路，等著．数学分析．复旦大学出版社