基于 XML 文档检索的存储研究与实现

Research on the XML document storage and implement

朱艳红　（石家庄邮电职业技术学院 计算机系 石家庄 050021）
何东彬　（石家庄学院 计算机系 石家庄 050035）

摘要：因特网诞生以来，网络信息资源急剧增长，如何利用数据检索技术有效的发现和使用资源成为急待解决的问题。对如何更好的检索 XML 等结构文档，已经作了大量的研究，提出了许多有效的查询方法，但目前对结构文档的索引结构及存储的研究却比较少，因此本文探讨了 XML 文档检索模型存储结构的主要技术问题，并提出了对置入表进行压缩编码的一种基于数据库存储的方案。

关键词：XML　检索　索引　存储

1　引言

经过近些年的发展，扩展标记语言(XML)正在成为英特网上数据表示、交换和集成的标准。XML 是一种结构化文档，物理形式上有着与其所表达的内容相对应的组织结构和层次关系，XML 文档的语义信息存在于文档的文本与结构之中。

传统信息检索返回信息的最小单位是整个文档。但许多时候，用户关心的不是全文，而只是文档中的某一个部分。我们大概都有类似的经历，面对过于简单的查询手段和返回来的无穷尽的检索结果，不得不再次依靠人力在返回的整篇文档中寻找关注信息，有可能在发现结果前就已经失去耐心。如何改变我们所面临的“富数据，穷信息”的窘境，就要求使用一种有效的方式对结构化文档的内容和层次结构进行检索，并以一种恰当的方式返回不同粒度的结果，即整篇文档中用户感兴趣的那一部分，而非全部。目前，对结构文档的检索已经做了许多工作，许多检索方法非常有效，但对结构文档的索引结构及存储的研究却比较少[1]，然而，这一部分非常重要，对检索性能的影响很大，因此本文着重解决这方面的问题。

2　信息粒度

对于返回信息的粒度问题，[2]提出一种逻辑文档的概念，文档树的每一棵子树都是一个逻辑文档，文档的类型就是子树根的类型，所以根节点相同的所有子树属于同一类，即同一类型的逻辑文档。由于逻辑文档的存在，因此可提供不同粒度的查询，所以逻辑文档成为检索结果所能返回信息的最小单位。那么在对文档结构索引的阶段，应该包含文本节点对于逻辑文档的归属信息。

3　索引模型

主流的全文索引模型有倒排索引(Inverted index)、署名文件、位图、Pat 数组。目前来说，倒排索引模型实现相对简单、查询速度快、容易支持同义词查询，应用比较广泛和成熟，被大部分商用搜索引擎所采用，因此本文选用倒排模型。

倒排索引是受到书目索引的启发而派生出来的，它由一系列“关键字－指针”对组成。关键字实际上是索引的查找键，包括文本集中出现的所有词项(停用词除外)。指针指向的是该词项在文本集中出现的所属文档、频率、位置。

4　存储方式

倒排索引的存储可以选择使用文件系统管理，也可以结合数据库管理系统来存储索引的内容。选择不同的存储方式会带来效率及实现上的差异。两者各有其优缺点。

使用文件系统管理方式，相比数据库存储在空间的使用上占有一定优势，同时也带来灵活性，存储时可

以获得更好的优化性能。而且利用文件系统构建存储的自实现方案，可以充分利用数据本身的结构及访问特点，通过对数据 Cache 的仔细设计及存储位置的合理安排可以获取更高的访问效率[3]。

但倒排文件的置入表放在硬盘上，数据文件很大，当检索高频词汇的置入表时将会占用大量 I/O 带宽和内存空间，对系统效率产生很严重的影响。另外，不同关键词项的置入表长度差别很大，分布不均匀，大部分词项出现次数较少，置入表长度只为 1 或 2，但某些词项的出现频率则异常的高，会造成该词项的置入表过长，从而使得系统的整体性能受到损害。[3] 虽然通过混合方法策略以及使置入表可调增长的算法可以在一定程度上解决，但整体说来，使用这种存储方式在实现上难度较高，需要处理更多的细节，增加了开发时间，且维护性较差。

比较而言，使用成熟的关系数据库管理系统，可以极大的简化系统实现，并发控制、数据恢复、事务处理等都可以借助于 DBMS 完成，从而减少了开发工作量和开发难度。另外，对于个别词项的置入表过长的问题可以将其拆分为多条记录存储。同时，通过研究置入表的数据结构特点，对其进行压缩[3]。因此，基于关系数据库的存储方式索引具有以下优点：效率比较高，容易维护，对多用户的并发访问、存取控制、备份和错误处理都提供了良好的支持[4]，实现灵活，开发时间短，维护简单。

但是数据库存储方式可能带来网络流量的压力及 SQL 语句查询的开销，在性能上无法达到最优，也不够灵活。

选择文件系统还是数据库，要根据实际情况来确定。综合考虑各方面因素，本文采用第二种方案，并在此基础上改进，例如尽可能使用存储过程提高效率，减少数据传输量等等，以改善系统性能，使其在某种条件下达到最优。

5　索引构造

具有相同结构的 XML 文档易于存储在关系数据库中并且检索也很方便，目前已经有许多这方面的研究（例如美国 Pennsylvania 大学的 STORED，加拿大 Tomoto 大学的 ToXin 等）。但是，对于诸多的非标准化的结构文档来说，也有检索的价值和必要。因此本文讨论的是，基于不同结构标准的具有普适性的检索方法所需要的存储结构。当用户对 XML 文档检索时，不需要了解文档的结构信息，只需直接输入查询关键词或附加部分结构信息的关键词就可以了。但是，系统应尽可能使用户看到返回查询结果文档的结构信息，以助于进行二次精确查询。在进行索引结构的设计时，考虑到文档的结构信息单独保存，可以存在一定的冗余情况；在索引优化阶段，可以对结构相似或完全相同的文档进行合并存储。

对结构化文档索引并建立倒排文件，文[1]给出了 5 种形式：ANWR（Inverted Index for All Nodes with Replication），ALWR（Inverted Index for All Levels with Replication），LNON（Inverted Index for Leaf Nodes Only），ANOR（Inverted Index for All Nodes without Replication），RNON（Inverted Index for Root Node Only）。综合其在索引空间需求、实测磁盘存取数据时间和平均磁盘读取时间的各项指标，认为 LNON 的综合优势比较明显，本文依此构建索引的存储结构，并给出一个完整的设计方案。

6　索引空间压缩

对于检索系统，最大限度的提高用户的感受，其中检索响应速度是一项非常重要的指标。对于一般的平面文件的索引来说，倒排文件的大小可能会达到原文件大小的 300%[3]，甚至更多，对于要同时保存结构信息的 XML 检索来说，将更加庞大。所以，如何减少索引数据量和数据压缩以提高查询质量就显得更为重要。

按[2]中提出的检索方法，对文档结构信息进行保存的空间需求更少，其保存的信息包括：当前节点在树中的先序遍历号（preOrder），最右孩子的先序遍历号，以当前节点为根子树的最大频率。在检索时，根据这些信息就可以完整还原出一棵文档树，检索的同时动态计算权重，以使系统性能在检索速度和存储空间需求平衡之上达到最优。

为了减小索引文件的大小，Lucene[5] 对索引数据使用了压缩技术。首先，对词典文件中的关键词进行了压缩，关键词压缩为＜前缀长度，后缀＞，例如：当前词为“中国人民”，上一个词为“中国”，那么“中国人民”压缩为＜2，人民＞。其次是对数字的压缩，对于有序

数字序列，数字只保存与上一个值的差值，这样可以减小数字的长度，从而减少保存该数字需要的字节数。例如：当前文档序号是18891，上一文章号是19007，在不压缩的情况下，要用3个字节才能保存，压缩后只需保存116，使用一个字节就可以了。

[4]提出了一种数据编码方式，将整数值按照"变量格式化长度"编码机制存储，可以有效的降低存储空间需求，具体办法就是将每个byte的最高bit定义为是否存在附加的数据，低7位保存实际数据。如表1所示。

表 1　整形数存储空间需求

Min value	max value	bits encoded	bytes required
0	127	7	1
128	16383	14	2
16384	2097151	21	3
2097152	268435455	28	4

7　索引存储结构

倒排文档索引通常使用B树进行存储和更新[2]，但是，一个B树存储实现非常困难，而且大多数计算机语言并不提供支持。实现一个高效易用的，能够处理并发访问和事务处理及错误恢复的算法就更加困难。幸运的是，关系数据库通常是使用B树来进行索引的[4]，因此本文在基于数据库系统存储索引结构的基础之上，给出了一种新的有效的索引存储设计。另外，由于使用支持事务的SQL数据库，检索的代价很小，返回的数据量也很少。

数据库中应该建立相应的关键字索引(字典数据)，大多数关系数据库都会按索引关键字建立一个B树索引。对于所有文本经过分词后的词项，都存储在关键词索引表中，这样在数据量大的关键字置入表中可以有效减少冗余，并进一步对整数进行压缩。

表 2　关键词索引表

column name	type	bytes	description
term	varchar	≤100	索引关键词
termCode	integer	4	索引关键词编码
docCount	integer	4	共有多少个逻辑文档包含此词
termCount	integer	4	(可选)这个词在文档集合中共出现次数

对于文档中元素的标签来说，会出现大量的重复，因此需要单独建立一个标签字典表。另外，将关键词和标签的字符数据数字化，将字符的比较转化为数字的比较，使得检索速度更快，返回数据也较少。

表 3　元素标签索引表

column name	type	bytes	description
tagName	varchar	≤100	元素标签名
tagCode	integer	4	元素标签编码

表 4　文档结构索引表

column name	type	bytes	description
docId	varchar	4	文档编号
flags	tinyint	1	说明记录数，并且表明当前是第几条记录
block	varbinary	≤2040byte	储存内容为：先序号、最右孩子序号、当前子树的最大词频

表 5　文档索引表

column name	type	bytes	description
docId	integer	4	文档编号
docName	varchar	50	文档名称
position	varchar	60	文档保存物理位置

表 6　叶节点索引表

column name	type	bytes	description
termCode	integer	4	索引关键词编码
firstDocId	integer	4	所在的第一个文档号
firstLDocNum	integer	4	firstDoc的直接逻辑文档的个数
docFlag	tinyint	1	指明记录数，并表明当前是第几条记录
block	varbinary	≤510bytes	对文本数据编码存储

叶节点索引表中的docFlag说明属于当前termCode的记录数及记录顺序：docFlag值=0说明符合条件的记录只有一条；(当前记录)docFlag值>128则当前记录是符合条件的第一条记录，当前byte的后

7 位保存符合记录条数。docFlag 值<128，则当前 byte 保存的值为记录的顺序号。

在[4]中指出一般情况下关键词的倒排文档列表的单条记录的列表长度不会超过 255 字节，由于本文逻辑文档概念的提出，增加了元素直接父节点的先序号内容，因此单条记录列表长度增加倍数应该等于直接父节点的个数，我们给出一个保守的估计值 2，所以 block 字段长度定义为 510bytes，(随着实验结果可以调整)其字段内存储十六进制编码，用以保存文档号、直接逻辑文档的个数(直接父节点，例如个数为 X)、叶节点的先序号、父节点的先序号(直接逻辑文档)、频率、位置信息列表；…… ；此叶节点在第 X 逻辑文档中的先序号，其父节点的先序号，频率、位置信息列表。

具体格式如下：

firstDoc. LeafNode. preOrder

firstDoc. LeafNode. parentPreOrder(存储该叶节点的先序号，也是该节点的类型)

firstDoc. LeafNode. freq(e. g. freq = N1)

firstDoc. LeafNode. pos1

…… ……

firstDoc. LeafNode. posN1

secondDoc. Id(第二个文档的文档号，保存相对值，即相对于 firstDocId 的差值)

secondDoc. CurrentLDocNum (第二个文档的直接逻辑文档的个数，即所属父节点的个数)

secondDoc. LeafNode. preOrder

secondDoc. LeafNode. parentPreOrder

secondDoc. LeafNode. freq(e. g. freq = N2)

secondDoc. LeafNode. pos1

…… ……

secondDoc. LeafNode. posN2

…… ……(直到第 X 个逻辑文档的叶节点为止)

thirdDoc…… ……

…… ……

在 block 字段中，将整形数进行压缩存储，对有序的文档号进行差值保存，即相对于 firstDocId 的每一个后续文档，其文档号值保存的是与前一个文档号的差值，同时所有整数值使用压缩编码方式存储。

8 结束语

基于上述工作，我们使用 Java 开发了基于 XML 文件检索系统软件 XMLsearcher 的原型系统，数据库使用 SQLServer2000，将文档文件以操作系统文档形式保存，而索引信息则保存到数据库中。实践证明，系统在文档集合规模不大的情况下(小于 200M)，索引表中的数据量与原文件基本持平，并且具有较好的检索性能。

对 block 的编码还可以有更复杂的方法，比如说将同一词项在文档树中的不同节点聚合，然后加入其在 block 中的位置信息，以利于快速查找，但是在索引时会增加索引时间和耗费更多的存储空间。如何对 block 编码进行优化，以提高检索的性能，将在今后进一步工作中继续研究。

参考文献

1 Yong Kyu Lee, Seong - Joon Yoo, Kyoungro Yoon Index Structures for Structured Documents ACM, 0-89791-830-4/96/03 1996.

2 R. Bayer and E. McCreight. Organization and Maintenance of Large Ordered Indexes. Acta Informatica,1:173-189,1972.

3 Christos Faloutsos, Douglas W. Oard A survey of information retrieval and filtering methods Computer Science Technical Report Series; Vol. CS-TR-3514 1995.

4 Steve Putz. Using a Relational Database for an Inverted Text Index. In Xerox Palo Alto Research Center, Technical Report SSL-91-20, Xerox PARC, January 1991.

5 ERIK HATCHER OTIS GOSPODNETIC Lucene in Action[M] Manning Publications Co. 2005.