一种实时更新索引结构的设计与实现

The Design and Realization of a Real－time Updating Index

王智强　刘建毅　（北京邮电大学智能科学技术研究中心　100087）

摘要：在搜索引擎的信息检索中，索引性能的优劣是影响检索质量的一个重要因素。本文针对面向主题搜索引擎内容覆盖范围窄、更新速度快的特点，设计了一种实时更新的索引结构，该方案的倒排索引结构打破了传统索引单一结构的形式，由主倒排索引、附加倒排索引和删除文件列表组成，很好的解决了索引的更新问题，实验结果显示该索引结构具有良好的性能。

关键词：主题搜索引擎　倒排索引　实时更新

1　引言

本文描述的信息检索系统采用倒排索引数据结构，尽管在建立时需要付出一定的代价，但是极大的提高了查询的效率。本文针对面向主题搜索引擎要求更新速度快的特点，提出了一种实时更新的动态索引结构，在基本不影响查询效率的前提下，大大提高了索引的更新时间。

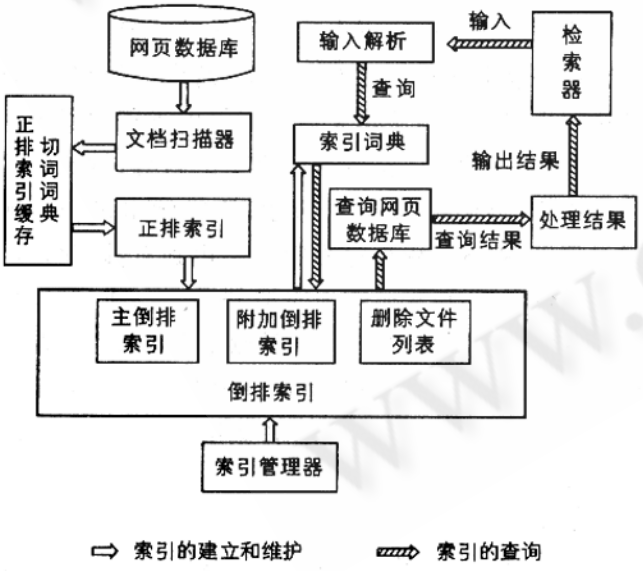

图 1　索引模块的体系结构

2　索引模块的体系结构和数据结构

本系统采用目前广泛使用的倒排索引模型实现，其体系结构如图 1 所示，索引模块主要由六个部分组成：文档扫描器、切词词典、正排索引、倒排索引、索引词典和索引管理器。

2.1　文档扫描器

该部分完成的任务是根据切词词典的内容，对从网页数据库中读取的文档进行扫描，得到索引词汇的出现次数、位置信息等统计信息。

2.2　切词词典

在中文索引系统中根据索引基本单位的不同存在两种基本的倒排索引结构：基于字的倒排索引和基于词的倒排索引。字表法把源文档中每一个字的位置信息记录到索引单元中，词典对每个不同的字符都保留一个指针，记录该字符在索引中的存储位置。而词表法把词作为处理和存储的基本单位。

字表法和词表法各有其有缺点。字表法通用性强，实现和维护相对容易，但是检索时有可能断章取义，将完全不相关的文档也检索出来，在索引中存储的信息相对较多。词表法的优点是其索引数据库可以组织的比较小，检索处理的速度比较快，而且能够根据词义进行概念检索，实现同义词、反义词的处理，但缺点是需要对源文档进行词的切分，词典中需要存储大量的词汇。

本系统的开发是针对特定主题的信息检索系统，因此采集来的文档包含的内容相对较为集中，切词准确率高，因此本系统采用基于词的倒排索引。

切词词典结构如图 2 所示，它本身有两项功能：存

储切词词汇、缓存文档扫描得到的词汇信息。每个词汇包含一个链表，用来记录文档扫描得到的词汇位置，扫描结束后将里面存储的内容按照正排索引文件的格式加以存储。文档在入库之前就完成了汉语编码的转换，因此后端处理汉字的编码均为 GB2321。

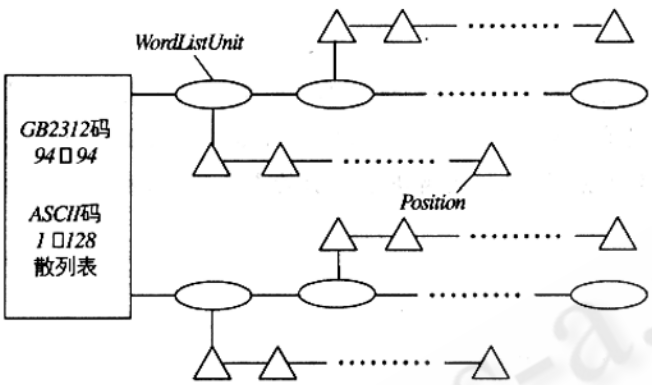

图 2　切词词典数据结构

每个 WordListUnit 结构中包括：词汇（string）、词汇编号（int）、在一篇文档中出现次数（int），位置链表（list）、是否是停止词（bool）。

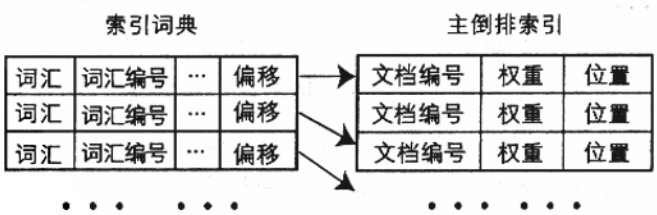

图 3　主倒排索引数据结构

2.3 正排索引

正排索引以文档编号（DocID）为键值，每个文档组成一个正排索引文件，按文档编号命名，每个文件包括索引头和索引体两部分：

文件头：文档编号（int）；文档标题和 Meta；文档中出现的索引词汇数（int m_nDicWordNum）；文档中出现的词汇总数（int m_nDicTolWordNum）；

索引体：词汇编号（int）、出现次数（int）、在文档中的位置（int）；

2.4 倒排索引

为了实现索引的实时更新，本系统与传统的倒排索引设计有很大的不同，它由三部分组成：主倒排索引、附加倒排索引和删除文件列表。倒排索引和附加倒排索引都是按照词汇编号排序，存储项包含文档编号、权重和词汇位置。删除文件列表中存储数据库中已被删除，但尚未从索引中删除的文档对应的编号。在部分更新时主倒排索引内容并不发生变化，只对附加倒排索引和删除文件列表进行更新，大大节省了更新时间，从而满足实时的要求。

2.5 索引词典

每个词汇的索引信息存储在 IndexWordUnit 结构中，每个 IndexWordUnit 中包含：词汇（string）、词汇编号（int）、词汇在倒排索引中的偏移量（int），包含该词汇的文档数目（int）、主索引中是否有该词汇（bool）、附加倒排索引中是否有该词汇（bool）。

2.6 索引管理器

索引管理是模块的重要组成部分，它管理索引的并发控制，存取控制，事务处理以及异常处理。

3 索引的建立、查询和更新

3.1 索引的初始建立

建立索引时，主要经过三个步骤：文档扫描、建立正排索引和建立倒排索引。首先文档扫描器从网页数据库中读取采集器搜索来的文档，进行处理，在处理过程中得到的词汇相关信息存储到正排索引缓存中；扫描结束后把正排索引缓存中的内容，按照词汇编号排序、存储，形成正排索引，正排索引是一个中间结果，但在索引更新过程中需要用到它，因此对其进行存储；对正排索引按词的 ID 进行重新排序，就得到了基本的倒排索引结构，存储到主倒排索引中就完成了文档的索引。索引建立时所有的内容都存储在主倒排索引中，不涉及附加倒排索引和删除文件列表的操作。

3.2 索引的查询

对用户的输入首先进行解析，得到查询词。把查询词提交给索引词典，如果索引标志位为 False，说明索引中没有该词汇，返回空集。如果为主倒排索引标志为 True 则读取该词汇的编号、偏移量和包含词汇文档数目（主倒排索引中），通过这些得到词汇的相关信息。之后再用词汇编号读取附加倒排索引中的内容，两个结果相加为最终的查询结果。所有查询词的结果

组成索引集，根据索引词在文档中的存储位置以及删除文件列表，对索引集进行过滤，形成结果集。根据文档信息的重要程度以及与查询词的相关性对结果排序得到最终的查询结果。

3.3 索引的更新

为了使索引满足搜索引擎实时更新索引的要求，本系统设计了独特的倒排索引结构，整个倒排索引由三个部分组成：主倒排索引、附加倒排索引和删除文件列表。绝大部分索引都存储在主倒排索引中，它不支持索引的插入和删除，要进行主倒排索引的更新需要对倒排索引重建；附加倒排索引存储容量较小，采用链表结构，支持文档的实时插入，具有良好的更新性能，当有文档插入时，主倒排索引并不更新，只需要把新加入的内容存储到附加倒排索引中即可。

传统索引模型更新性能差的主要原因是其索引文件存放连续，这样再要把新加入的文件集添加入索引时必须要打乱原来的索引，导致索引需要重新排序，造成更新性能的下降，这主要是数组数据结构的缺点。本系统在索引中采用链表构建的附加倒排索引，提高了索引的更新性能。附加倒排索引结构如图 4 所示。每个词汇对应一条索引链，InvWordIndex 为链的根节点，其数据结构包括：词汇编号，附加倒排索引中包含该词汇的文档数，InvWordNode 链表，链表中是否存储有内容。链中的每个节点 InvWordNode 对应一篇出现该词汇的文档，其数据结构包括：文档编号，词汇在文档中的权重，该词汇在文档中出现的次数，出现位置链表。当有新文档加入附加倒排索引时，如果该词汇出现在文档中，首先申请一个空白的 InvWordNode 结构，把正排索引文件中的内容写入其中，然后将 InvWordNode 加入到链表末端，这样就完成了文档的插入。因为并不涉及到 I/O 的操作，和数据的移动因此速度很快。

索引的更新包括文档插入、删除和修改，一般对文档的修改比较少见，特别是对网页文档的修改，同时对文档的修改也可以看成是先删除再插入，因此本系统只处理插入和删除操作。针对倒排索引的特殊结构，本系统的索引更新方式有两种：实时更新（real – time update）和全面更新（full update）。当有文档插入时索引启动实时更新程序，只把文档索引插入到附加倒排索引中，主倒排索引并不发生变化。当附加倒排索引的大小超过一个阈值后，系统会自动触发全面更新，对主倒排索引和附加倒排索引进行合并，重新构建主倒排索引，把删除文件列表中的文档从索引中删除，清空附加倒排索引和删除文件列表。

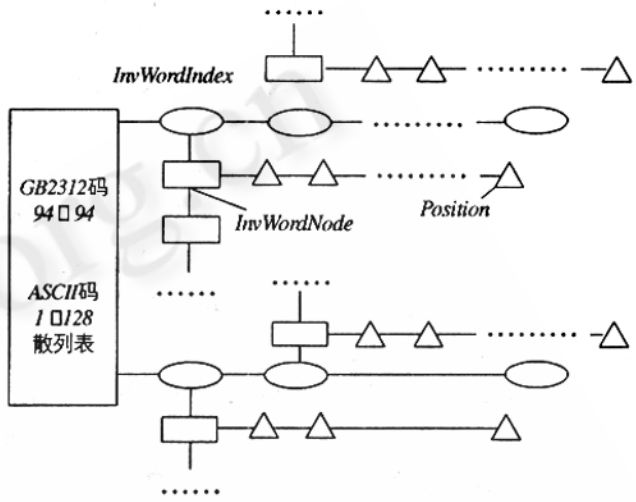

图 4　附加倒排索引数据结构

因为词汇索引存储在主倒排索引和附加倒排索引两部分，这就给文档的删除带来了很大的困难，为了解决这一问题，本系统在倒排索引结构中加入删除文件列表，当有文档需要删除时，并不需要知道该文档内容存储在索引的那一部分，也不需要直接到索引中删除，而是将文档编号记录到删除文件列表中，这样大大节省了删除时间。检索时在返回结果前对结果集进行检查，把其中已经删除的过滤即可。

采取这种索引结构具有很多独特的优点：由于附加倒排索引较小，容易更新；采用删除文件列表避免了直接对索引进行删除操作；同时针对附加倒排索引和主倒排索引的特点采用不同的实现方法，可进一步提高索引的性能。

4 性能分析

测试环境：Intel P4 2.6G CPU，512M DDR，Microsoft Windows XP 操作系统。测试的数据：100M 网页文件。实验结果如表 1。

表 1 索引初始建立时间

名称	系统初始化	建立正排索引	建立倒排索引	总计
耗时(秒)	5	93	104	202

通过对程序的跟踪发现,程序的大部分时间消耗在 I/O 操作上。建立正排索引的时间包括:从数据库中读取网页、解析和扫描文档、将结果写入正排索引文件,其中大部分时间消耗在后两项处理工作上。本系统的词典采用北大现代语法信息词典,该词典的总词汇数为 68200 个(包含单汉字),实验结果显示 100M 网页数据中包含 25820 个词汇,由于词汇数目很多,因此建立倒排索引绝大时间大部分消耗在重组每个词汇的信息建立倒排索引上。

如果把数据集作为需要更新的文档,更新索引所需要的时间见表 2。在更新前索引为空的条件下,进行全面更新需要 104 秒,而部分更新只需要 8 秒。如果更新前索引中存储有 100M 网页文件的索引(与更新数据不同),则全面更新需要 165 秒,而部分更新消耗的时间只与更新网页的大小有关,与以前索引中存储多少数据无关,仍只需要 8 秒。从表中数据不难看出部分更新的速度远远大于全面更新的速度。在实际操作中部分更新可以经常进行,并不需要积累到 100M 数据再更新,所以部分更新的速度会更快。系统的实际运行显示,完全能够满足索引实时更新的要求。

表 2 索引更新时间

名称	全面更新时间(秒)	部分更新时间(秒)
索引为空	104	8
索引不为空	165	8

5 结论

本文研究了索引的各种实现方法,根据面向主题搜索引擎的特点设计并实现了一种实时更新的索引结构,其倒排索引由主倒排索引、附加倒排索引和删除文件列表三部分组成,独特的结构很好的解决了索引的更新问题。通过系统的实际运行显示,该结构完全满足索引实时更新的要求。

参考文献

1 M. Araujo, G. Navarro, and N. Ziviani. Large text searching allowing error. In Pro. WSP'97, pages 2 – 20, Valparaiso, Chile, 1997. Carleton University Press.

2 Gonnet, G. H. et al. New indices for text: PAT trees and PAT arrays, Information Retrieval: Data Structure and Algorithms (Frakes, W. B. and Baeza – Yates, R. A. (eds.)), Prentice – Hall, New Jersey, pp. 66 – 82, 1992.

3 C. Faloutsos, S. Christodoulakis. Description and performance analysis of signature file methods. ACM TOIS, 5(3):237 – 257, 1987.

4 C. Faloutsos and R. Chan. Fast text access methods for optical and large magnetic disks: design and performance comparison. In Proc. Of VLDB'88, pages 280 – 293, Los Angeles, CA, USA, 1988.

5 刘学文、陶晓鹏、于玉、胡运发,一种全新的全文索引模型——后继数组模型,软件学报,Vol. 13 No. 1, 2002。

6 Ricardo Baeza – Yates. Modern Information Retrieval. New York: ACM Press, 1999:191 – 198.