·软件技术与数据库· 文章编号：1000—3428(2012)14—0035—03 文献标识码：A 中图分类号：TP311

一种基于B+树的混合索引结构

长孙妮妮[a]，张毅坤[a]，华灯鑫[b]，邹子夏[a]，陈 浩[a,b]

(西安理工大学 a. 计算机科学与工程学院；b. 机械与精密仪器工程学院，西安 710048)

摘 要：针对文件中存在字符属性和数值属性特点的数据，提出一种基于B+树的2层混合索引结构。在索引创建过程中，将文件中的数据根据其类型进行分类，第1层是为数据属性建立B+树索引结构，第2层是根据不同的数据类型建立不同的索引结构。实验结果表明，该索引结构能实现跨数据范围的检索，提高索引的创建速度和空间利用率。

关键词：倒排索引；B+树；混合索引；时间开销；空间利用率；查询效率

Hybrid Index Structure Based on B+ Tree

ZHANGSUN Ni-ni[a], ZHANG Yi-kun[a], HUA Deng-xin[b], ZOU Zi-xia[a], CHEN Hao[a,b]

(a. School of Computer Science and Engineering; b. School of Mechanical and Precision Instrument Engineering, Xi'an University of Technology, Xi'an 710048, China)

【Abstract】 Considering the numerical attribute data's cross-data range retrieval needs for the files that have the property of character and number, a 2-layer hybrid index structure based on B+ tree is proposed. That data in the files is classified according to their types in the process of creating index, the first layer sets up a B+ tree index structure for the attribute of the data and the second layer constructs different index structure according to the different types of data. Experimental results show that the hybrid index structure not only adapts to the above retrieval needs, but also effectively improves the index creation speed and the space utilization.

【Key words】 inverted index; B+ tree; hybrid index; time overhead; space utilization; query efficiency

DOI: 10.3969/j.issn.1000-3428.2012.14.010

1 概述

在分布式网络环境中，随着互联网络的高速发展各个网络结点之间共享数据已经是大势所趋。而为网络结点的数据建立合理有效的索引结构是解决多点、多区域网络结点共享数据的前提。网络结点间快速检索的必要条件是，需要在每个单点建立一种快速高效的索引结构。

索引技术是现代信息检索、搜索应用、数据挖掘的关键技术之一[1]，而倒排索引和B+树索引是目前应用较广泛的2种索引技术。其中，倒排索引技术具有实现相对简单、查询速度快以及易于支持同义词查询等扩展功能的优点，被广泛应用于大规模文档集的检索与其他各种信息的检索[2]。另外，B+树是B树的一种变体，它适合随机和顺序处理的应用环境[3]，而且它在数据库系统、文件系统建立索引[4-5]等方面有广泛的应用。它理论上可以对任何数据类型的属性建立索引，同时能保持与数据文件大小相适应的索引层次。查找具有稳定的I/O开销，对索引更新具有较好的支持，能承受各种大量的工作负载[6]。

然而，国内外研究学者对倒排索引的研究主要是针对其创建过程中创建速度较慢以及查询时过多的I/O操作导致时间和空间效率低下[7-8]等问题展开工作。但是，在面对跨数据范围检索要求时显得束手无策。传统B+树是以插入法建立的，频繁的数据插入使B+树在结点分裂时需要花费大量的时间。

本文针对存在字符属性和数值属性特点的文件能完成数值属性数据的跨数据范围检索，提出一种基于B+树的混合索引结构。

2 混合索引结构设计思想及创建过程

2.1 混合索引结构设计思想

针对建立索引的对象包含2类数据，即字符型数据和数值型数据，本文在设计索引结构时采用分层的思想。即在该索引结构的第1层为数据所属的属性建立索引，第2层是针对第1层数据属性所对应的属性值建立索引。此时，如果该属性值属于字符型数据就对其建立倒排索引，如果该属性值属于数值型数据就对其建立B+树索引。这样，与传统的B+树结构对所有不同类型的数据都建立B+树索引不同。该索引结构对所有字符型数据没有建立B+树索引，由结点分裂带来的时间开销问题就得以避免，同时在结点分裂过程中产生的临时结点所占用额外的存储空间也相对减少。因此，该索引结构提高了索引的创建速度和空间利用率。倒排索引针对字符型数据的检索具有良好的性能，但它又不能完成数值型数据跨数据范围检索的需求。而B+树由于叶子结点的有序性保证了它对数值型数据检索时具有优势。所以，本文的混合索引结构采用分层和针对不同类型的数据建立不同索引结构的思想来完成跨数据范围检索的需求，即所有的数值型数据都在第2层建立B+树索引，当索引结构建立完成后需要跨数据范围检索时，该任务就会由第2层的B+树索引部分完成。混

基金项目：国家“973”计划基金资助项目(2009CB426302)

作者简介：长孙妮妮(1988－)，女，硕士研究生，主研方向：数据库索引技术，软件工程；张毅坤、华灯鑫，教授、博士；邹子夏，硕士研究生；陈 浩，博士研究生

收稿日期：2011-11-08 **E-mail**：ykzhang163@163.com

合索引结构利用并继承了这2种结构的优点，同时又摒弃了这2种结构的不足。在达到高效索引创建速度和空间效率的同时又能完成跨数据范围检索的需求。混合索引结构如图1所示。

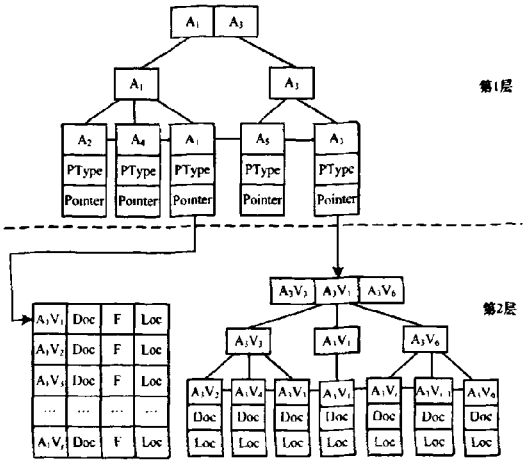

图1 混合索引结构

在图1中第1层是为建立索引对象所属属性建立B+树索引结构，在该层B+树索引结构中所有的非叶子结点存储的是具体的属性，而所有的叶子结点包含三部分信息 <A_i,Ptype, Pointer>：

(1)A_i是建立索引的数据的属性，$i\in[1,n]$，n为所有属性个数。

(2)Ptype为指向第2层结构的指针类型 PType∈{B+tree, Inverted _index}。

(3)Pointer为指向第2层B+树根结点或倒排表头的指针，即根据属性的不同类型该指针指向不同的索引结构。

第2层是与第1层属性对应属性值的倒排表或者B+树索引结构，倒排表的索引结构包含四部分信息 <A_iV_j,Doc,F, Loc>：

(1)A_iV_j为第i个属性的第j个属性值，$i\in[1,n_1]$、$j\in[1,m]$，n_1为字符属性的个数，m为第i个属性包含的属性值的个数。

(2)Doc为该属性值所在的文件编号，每个文件编号唯一。

(3)F为属性值在文件中出现的频率。

(4)Loc为该属性值所在的文件位置信息。

第2层的B+树索引结构的非叶子结点只包含属性值，叶子结点都是有序的，且每个叶子结点包含三部分信息 <A_RV_S, Doc,Loc>：

(1)A_RV_S为第R个属性的第S个属性值，$R\in[1,n_2]$、$S\in[1,p]$，n_2为文件中数值属性的个数，p为第R个属性的属性值的个数。

(2)Doc为该属性值所在的文件编号，每个文件编号唯一。

(3)Loc为该属性值所在文件的位置信息。

2.2 混合索引结构的创建过程

混合索引结构的创建过程如下：

(1)分析需要建立索引的文件，如果是文件的一个新属性，则在该混合索引结构的第1层为该属性建立一个新的结点。

(2)根据该属性的属性值类型确定第2层的索引结构。如果是数值类型的属性值，则对该属性值建立B+树索引；如果是字符类型的属性值对该属性值建立倒排索引。

(3)重复步骤(1)，如果当前属性与之前建立索引的属性相同，则在混合索引的第1层不增加新的属性结点，只把该属性的属性值添加到对应的第2层的索引结构中。

(4)如果当前的属性与之前建立索引的属性不同，那么在该混合索引结构的第1层增加新的属性结点，然后根据步骤(2)判断对该属性值需要建立何种类型的索引结构。重复上述步骤，直到所有的数据建立完索引为止。

图2是以西安理工大学激光雷达大气遥测研究中心激光雷达监测站点所监测的激光雷达数据文件为例，建立的激光雷达监测数据的索引结构。图3是对图2中数据的具体说明。

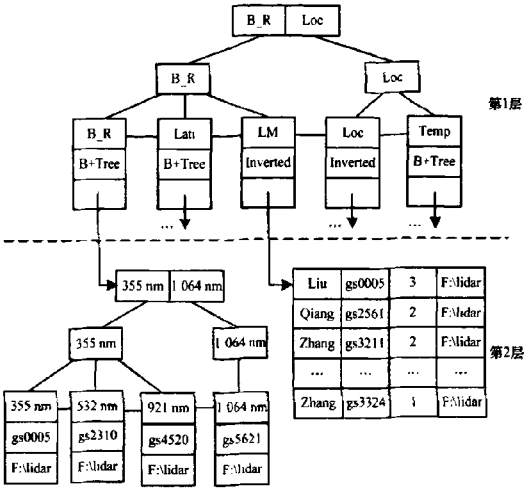

图2 激光雷达监测数据索引结构

B_R	Lati	LM	Loc	Temp
Backscatter Ratio	Latitude	Lab Manager	Location	Temperature
反相散射率	纬度	实验负责人	测量地点	温度

图3 激光雷达监测数据的特征说明

3 混合索引结构性能分析与比较

定义 设n_1为数值属性的个数，n_2为每个数值属性包含的属性值的平均个数，n_3为字符属性的个数，n_4为每个字符属性包含的属性值的平均个数。设k为索引结构的第1层B+树的阶，m为索引结构的第2层B+树的阶，N为所有的属性值的个数，$N=n_1\times n_2+n_3\times n_4$。

3.1 创建混合索引的空间性能分析

本文混合索引结构的第1层是B+树索引结构，其高度为$\log_k(n_1+n_3)$。假设B+树的所有非叶子结点都有k个子结点，此时第1层B+树有FB_{Num}个结点占用存储空间：

$$FB_{Num}=k^0+k^1+\cdots+k^{\log_k(n_1+n_3)}=\frac{k\times k^{\log_k(n_1+n_3)}-1}{k-1}=\frac{k\times(n_1+n_3)-1}{k-1} \quad (1)$$

在第2层的倒排索引结构中，则有$SI_{Num}=n_4$个结点占用存储空间。第2层B+树的高度为$\log_m n_2$，假设B+树的所有非叶子结点都有m个子结点。此时，B+树有SB_{Num}个结点占用存储空间：

$$SB_{Num}=m^0+m^1+\cdots+m^{\log_m n_2}=\frac{m\times m^{\log_m n_2}-1}{m-1}=\frac{m\times n_2-1}{m-1} \quad (2)$$

由式(2)可以得到本文提出的混合索引结构有HB_{Num}个结点占用存储空间：

$$HB_{\text{Num}} = n_1 \times SB_{\text{Num}} + n_3 \times SI_{\text{Num}} + FB_{\text{Num}} = n_1 \times \frac{m \times n_2 - 1}{m-1} + n_3 \times n_4 + \frac{k \times (n_1 + n_3) - 1}{k-1} \quad (3)$$

传统的B+树是为所有属性值建立索引，所以有 TB_{Num} 个结点占用存储空间：

$$TB_{\text{Num}} = \frac{m \times N - 1}{m-1} + \frac{k \times (n_1 + n_3) - 1}{k-1} = \frac{m \times n_1 \times n_2 + (m-1) \times n_3 \times n_4 + n_3 \times n_4 - 1}{m-1} + FB_{\text{Num}} = n_1 \times SB_{\text{Num}} + n_3 \times SI_{\text{Num}} + FB_{\text{Num}} + \frac{n_1 + n_3 \times n_4 - 1}{m-1} \quad (4)$$

由上述推算过程明显可知式(4)大于式(3)，而且由式(4)减去式(3)可知，在创建索引的过程中与传统 B+树索引结构相比，本文的混合索引结构有 $\frac{n_1 + n_3 \times n_4 - 1}{m-1}$ 个结点不占用存储空间。而且随着字符属性个数 $n_3 \times n_4$ 的增加，传统的 B+树索引结构所占用的存储空间也会随之变大，本文索引结构表现出来的优势更加明显。

图 4 是本文索引结构和传统 B+树结构创建索引占用的存储空间比较。

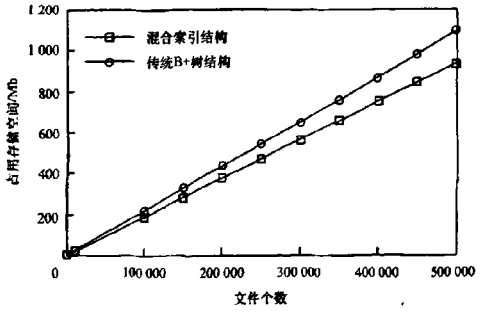

图 4 创建索引占用的存储空间比较

图 4 所示的实验结果验证了上述推算过程的正确性，可以看出，激光雷达文件数少于 200 000 时，2 种索引结构所占用的存储空间差距不太明显，当文件数目在 200 000 到 500 000 时，2 种索引结构所占用的存储空间差距明显变大。即随着文件数目的增加，字符属性个数也随着相对的增加，那么本文混合索引结构的空间性能明显优于传统 B+树结构。

3.2 创建混合索引的时间性能分析

传统的 B+树是以插入法建立的，当结点填充满时 B+树的结点必须分裂。分裂所花费的时间开销极大影响了 B+树的创建效率。在 B+树的创建过程中结点分裂的个数越少，创建索引花费的时间越少。混合索引结构第 1 层 B+树的高度为 $\log_k(n_1+n_3)$，假设 B+树所有的非叶子结点都有 k 个子结点。此时第 1 层 B+树有 FB_{div} 个结点需要分裂：

$$FB_{\text{div}} = k^0 + k^1 + \cdots + k^{\log_k(n_1+n_3)-1} = \frac{\frac{n_1+n_3}{k} \times k - 1}{k-1} = \frac{n_1+n_3-1}{k-1} \quad (5)$$

混合索引第 2 层 B+树的高度为 $\log_m n_2$，假设 B+树所有的非叶子结点都有 m 个子结点。此时 B+树有 SB_{div} 个结点需要分裂：

$$SB_{\text{div}} = m^0 + m^1 + \cdots + m^{\log_m n_2 - 1} = \frac{\frac{n_2}{m} \times m - 1}{m-1} = \frac{n_2 - 1}{m-1} \quad (6)$$

本文混合索引结构中结点分裂个数共有：

$$HB_{\text{div}} = FB_{\text{div}} + SB_{\text{div}} = \frac{n_2 - 1}{m-1} + \frac{n_1 + n_3 - 1}{k-1} \quad (7)$$

传统的 B+树是为所有的属性值建立索引，结点分裂个数为 TB_{div} 个。即：

$$TB_{\text{div}} = \frac{N-1}{m-1} + \frac{n_1+n_3-1}{k-1} = \frac{n_1 \times n_2 + n_3 \times n_4 - 1}{m-1} + \frac{n_1+n_3-1}{k-1} \quad (8)$$

$$TB_{\text{div}} - HB_{\text{div}} = \frac{(n_1-1) \times n_2 + (n_3 \times n_4)}{m-1} \quad (9)$$

由上述推算过程可以看出，式(8)大于式(7)，而且由式(9)可以看出，传统的 B+树索引结构需要分裂的结点个数比与本文混合索引结构多 $\frac{(n_1-1) \times n_2 + (n_3 \times n_4)}{m-1}$ 个。即随着字符类型数据个数和数值类型数据个数的增加，传统 B+树结构需要分裂的结点也随之增加，那么这些分裂结点在索引结构创建过程中所花费的时间也会随之增加。本文混合索引在创建时间上所表现出来的优越性更加明显。

图 5 是本文索引结构和传统 B+树索引结构创建索引时间比较。

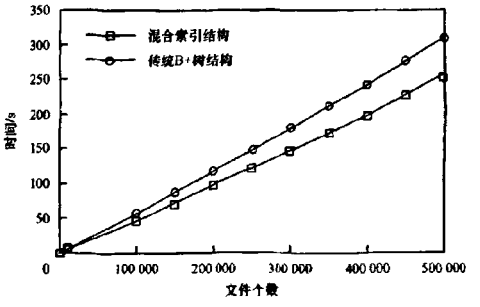

图 5 创建索引时间比较

图 5 实验结果验证了上述推算过程的正确性，可以看出激光雷达文件数少于 200 000 时，2 种索引结构在创建索引时所花费的时间差距不太明显，当文件数目在 200 000 到 500 000 时，2 种索引结构所花费的时间差距明显变大。即随着激光雷达文件数目的增加，字符型数据个数和数值型数据个数也随着增加，那么本文混合索引结构的时间效率明显优于传统 B+树结构。

3.3 混合索引结构的查找效率分析

在该混合索引结构中如果查找数值属性时，其查找的时间复杂度约为：

$$\log_k n_1 + \log_m(n_1 \times n_2) \quad (10)$$

如果查找字符属性时，则查找的时间复杂度约为：

$$\log_k n_3 + \text{lb}(n_1 \times n_4) \quad (11)$$

该混合索引结构的查找时间复杂度介于式(10)和式(11)之间。传统的 B+树无论是查找字符属性数据还是查找数值属性的数据，其查找的时间复杂度都约为：

$$\log_k(n_1+n_3) + \log_m(n_1 \times n_2 + n_3 \times n_4) \quad (12)$$

由式(10)和式(12)可知，在查找数值属性的数据时，传统的 B+树的查找效率明显低于混合索引结构的查询效率。由式(11)和式(12)可知，两者的大小与变量的取值范围有关。当 m=2 时，式(12)大于式(11)，即查找字符属性的数据时，传统 B+树的查找效率明显低于混合索引结构的查询效率。

由此分析可知，混合索引结构的查询效率不低于传统的 B+树结构。

因此，由上述三方面的分析可以看出，本文提出的混合索引结构在创建时间效率和空间效率上明显优于传统 B+树结构，在查找效率上不低于传统 B+树索引结构。

(下转第 40 页)

虚拟机配置为 2 个虚拟 CPU、1 GB 内存；所有物理机的控制网为千兆网，数据网使用 HBA 光纤卡与共享存储装置连接。

应用表 1 提供的典型参数，结合探测到的各物理机的负载情况，通过本文公式(CPU 虚拟度与内存虚拟度分别通过式(1)和式(3)获得，各利用率指标为实测结果)计算得出如表 3 所示的数据(CPU 利用率为 15 min 以内，用指数函数作为影响因子的加权平均值)。

表 3 利用本文算法计算的各指标

计算机	CPU 虚拟度	内存虚拟度	CPU 利用率/(%)	内存利用率/(%)
1	1.08	0.54	25	69
2	0.80	0.40	20	55
3	1.25	0.54	28	70
4	1.08	0.54	30	68
5	0.50	0.50	21	66
7	0.58	0.58	17	72
集群整体	0.91	0.53	28	65

表 3 表示使用本文算法所达到实验效果的内在原因，表 2 是系统的外部表现形式。计算机 1~计算机 6 所承载的虚拟机很容易理解，并与前述算法对应起来。关键是计算机 4 只承载了 15 个虚拟机，这体现了本文算法的优点之一，即在首次为待开启虚拟机分配资源时，为快速响应大量的并发请求，使用比其虚拟度或负载高的计算机，但计算机 1~计算机 3 的虚拟度与负载均在容纳范围内，即物理机为虚拟机提供的资源足以满足业务需要，所以，系统认为暂时不必发生虚拟机迁移这样的重量级调整情况。

5 结束语

当前，云计算技术正以前所未有的速度高速发展，资源调度作为云计算操作系统的重要组成部分，还处于起步阶段。显然基本的自适应调度只能为业务正确地分配合适的资源提供基本保障，还有诸多的技术与方法可以对调度的成效做深层次优化，如利用业务资源的互补关系对虚拟资源进行互补性调度，利用相同或依赖性强业务对局部配置要求较高的特性进行业务聚集性调度，对物理资源的硬件健康度进行评估和预测以进行健康性调度等。今后将围绕历史数据挖掘、传感芯片控制、行业业务特点等问题与技术展开深入研究，以提高资源使用效率与系统智能化水平。

参考文献

[1] 杜 海，陈 榕. 基于完全虚拟化的进程监控方法[J]. 计算机工程，2009，35(8)：88-90.

[2] 郭本俊，王 鹏，陈高云，等. 基于 MPI 的云计算模型[J]. 计算机工程，2009，35(24)：84-86.

[3] 库劳里斯. 分布式系统概念与设计[M]. 北京：机械工业出版社，2004.

[4] Calheiros R N, Ranjan R, Cesar A F, et al. A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services[R]. New York, USA: Cornell University, Tech. Rep.: GRIDS-TR-2009-1, 2009.

[5] 宋远骏，杨孝宗. 多机多任务实时系统云调度策略[J]. 计算机学报，2000，23(10)：1107-1110.

[6] 陈 俊，陈孝威. 基于移动 IPv4/IPv6 演进云计算框架设计[J]. 计算机应用研究，2011，28(6)：2321-2323.

[7] 李建锋，彭 舰. 云计算环境下基于改进遗传算法的任务调度算法[J]. 计算机应用，2011，31(1)：184-186.

[8] 刘进军，赵生慧. 面向云计算的多虚拟机管理模型的设计[J]. 计算机应用，2011，31(5)：1417-1419.

编辑 陆燕菲

(上接第 37 页)

4 结束语

本文提出的基于 B+树的混合索引结构，在激光雷达数据文件建立索引时其时间效率和空间效率取得了明显的效果而且查找效率也不低于传统 B+树结构。即在单个激光雷达站点上，为每个站点的激光雷达数据能达到快速创建速度的同时还能节省存储空间，同时也能快速地查询数据。然而，解决单点数据快速建立索引的问题是解决多点、多区域网络结点之间数据共享的前提。将单点建立索引的方法有效地应用于分布式网络环境下的多个站点中是下一步研究的重点。

参考文献

[1] 郭瑞杰，程学旗，许洪波，等. 一种基于动态平衡树的在线索引快速构建方法[J]. 计算机研究与发展，2008，45(10)：1769-1775.

[2] 刘小珠，彭智勇，陈 旭. 高效的随机访问分块倒排文件自索引技术[J]. 计算机学报，2010，33(6)：977-987.

[3] 胡玉乐，孙 莉，王 梅. RB+树——一种列存储数据的树型索引结构[J]. 计算机研究与发展，2010，47(增刊)：335-342.

[4] 吴恒山，徐晓军，桂 浩. 基于改进 B+树索引的结构连接算法[J]. 计算机工程，2005，31(16)：86-88.

[5] 卢 萍，陈进才. 一种基于对象存储的文件系统的设计[J]. 计算机科学，2008，35(10)：131-133.

[6] Cui Yu, Bailey J, Montefusco J, et al. Enhancing the B+ Tree by Dynamic Node Popularity Caching[J]. Information Processing Letters, 2010, 10(7): 268-273.

[7] 刘小珠，孙 莎，曾 承，等. 基于缓存的倒排索引机制研究[J]. 计算机研究与发展，2007，44(增刊)：153-158.

[8] Liu Xiaozhu. Efficient Maintenance Scheme of Inverted Index for Large-scale Full-text Retrieval[C]//Proc. of International Conf. on Future Computer and Communication. Wuhan, China: [s. n.], 2010: 565-570.

编辑 陆燕菲