第33卷 第18期 计 算 机 工 程 2007年9月
Vol.33 No.18 Computer Engineering September 2007
·软件技术与数据库· 文章编号：1000—3428(2007)18—0069—03 文献标识码：A 中图分类号：TP311

用于内存数据库的 Hash 索引的设计与实现

袁培森，皮德常
（南京航空航天大学信息科学与技术学院，南京 210016）

摘 要：电信领域已成为数据密集型行业，需要高性能的数据库系统作为支撑系统，基于磁盘的数据库系统不能满足“实时”、“近实时”访问数据库的需求，将数据库核心数据驻留在内存中，可以使用内存数据库来满足需求。Hash 索引是数据库系统中广泛使用的索引技术之一，它能够快速地访问数据，易于设计和实现。该文根据内存数据库的特点，为电信网管系统的内存数据库设计并实现了 Hash 索引。
关键词：内存数据库；Hash 索引；磁盘数据库系统

Design and Implementation of Hash Index Used in Main Memory Database

YUAN Pei-sen, PI De-chang
(College of Information Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016)

【Abstract】 Telecom has became a data-intensive industry, which enlists the support of high performance database. Disk-resident database(DRDB) can't satisfy the demand of real-time or nearly real-time performance, so to design a kind of new database system, in which "core data" reside in main memory is considered, is needed. Hash index technique is a kind of index technique, used in database system, which can quickly access data, easy to design and implement. According to the features of main memory database, this paper designs and implements a kind of Hash index used in telecom network's main memory database system.
【Key words】 MMDB; Hash index; disk-resident database(DRDB)

随着应用的发展，由于“I/O瓶颈”问题，基于磁盘的数据库系统(DRDBS)不能满足现代应用对数据库的实时性处理的要求。如电信网管系统、电话交换机的实时呼叫处理、移动通信HLR/VLR等系统，它们需要实时地对数据进行处理。基于磁盘的数据库系统(Oracle,SQL server等)，不能满足实时性需求。内存数据与磁盘数据在访问时间上相差 5 个数量级[1]，内存的速度具有明显的优势。内存容量增大，而价格却在不断地下降[2,3]。从 20 世纪 80 年代开始，数据库研究人员考虑把整个或者大部分数据库放在内存中。内存数据库(MMDB)是实时数据库研究的基础，并成为了研究的热点。

磁盘和内存在存储方式上有很大的差别，基于磁盘结构的数据库系统的数据结构、算法、查询方法、索引等，不适合用在内存数据库系统中。为了充分利用内存的优点，就要为 MMDB 设计新的数据结构、算法、查询方法、索引等。根据电信网管系统中内存数据库的特点，笔者设计了一个用于内存数据库的 Hash 索引，并在内存数据库系统中加以应用。

1 内存数据库和 Hash 索引

1.1 内存数据库

内存数据库就是把整个或部分数据库放在内存中。内存数据库的定义为：设有数据库系统DBS，DB为DBS中的数据库，DBM(t)为在时刻t，DB在内存的数据集，DBM$\subseteq$DB。TS为DBS中所有可能事务的集合，$AT(t)$为在时刻t处于活动状态的事务集，$AT(t)\subseteq TS$。$D_t(T)$为事务T在t时刻所操作的数据集，$D_t(T)\subseteq$DB。若任意时刻t，均有

$$\forall T\in AT(t),\ D_t(T)\subseteq DBM(t)$$

成立，则称DBS为一个MMDBS[1]、DB为MMDB。内存数据库实质就是将数据库的“工作版本”存放在内存中。

内存按字或字节存储，通过指针访问。磁盘是以数据块形式进行存储。由于内存和磁盘的存储方式的差别，因此内存数据库的数据结构的设计与DRDB差别很大。磁盘以块寻址，数据连续存放可以减少磁盘的寻道时间；内存是以字或字节形式存储，并通过指针进行访问，时间开销与存放方式基本无关。内存数据库一般是通过指针访问数据库中的记录[4~6]。

在DRDBS中，I/O是系统的瓶颈，算法优化的目标是减少I/O次数和增加磁盘利用率；在MMDB中，系统优化的目标是最大化CPU利用率和内存空间利用率。B树和B+树是磁盘数据库中广泛使用的索引技术，能最大化地减少I/O，但是空间的利用率仅为 60%左右[1]，其较低的空间利用率不适于内存数据库，需要设计适合内存数据库的索引。

1.2 Hash 技术

Hash技术在信息系统的数据存储与访问中占有重要的地位[7,8]。它把关键词直接映射为存储地址，达到快速寻址的目的，即Addr=$H(key)$，其中key为关键词；H为哈希函数。文献[4,5]讨论了几种常用的哈希函数：(1)除留余数法(Division Method)，$H(key)=key$ MOD p，p一般为质数；(2)随机数法(Random Method)，$H(key)$=random(key)，$random$为随机函数；(3)平方取中法(Midsquare Method)。

理想的情况是“一次定位到关键词对应的地址”，但Hash技术的缺点就是不同的关键词有可能冲突，即$key_1\neq key_2$，但是$H(key_1)=H(key_2)$，其中，key_1和key_2称为同义词。在Hash

作者简介：袁培森(1980－)，男，硕士，主研方向：数据库和数据挖掘；皮德常，博士、副教授
收稿日期：2006-12-28 **E-mail：**peiseny@yahoo.com.cn

技术中，冲突是不可避免的，只能减少冲突的概率[8]。处理冲突的方法分两大类[7]：开放地址法(open addressing)和独留链(separate chaining)。当出现冲突时，开放地址法通过试探性地增加一定的值，把冲突的关键词放到一个空闲位置；独留链法把冲突的关键词存储到一个线性链表中。

2 内存数据库 Hash 索引设计

2.1 内存数据库设计思想

电信网管数据库系统是电信底层支撑系统，它不仅要求数据库能够“实时”、“近实时”地处理数据，而且系统需有很高的稳定性，为了保证系统高稳定性，内存数据库表的容量是规划好的，表采用一次性静态进行连续分配。这样做的优点是方便数据在内外存间进行换入换出；也可以通过相对地址访问记录。表的记录存放在内存中，每条记录分配一个指针(4B 的记录号，称为伪指针)指向它。

使用指针的优点是：通过指针可以方便地获得记录的字段值；当更新记录时移动指针比移动记录的属性值快捷，并且降低了空间开销。当插入记录时，为此记录分配一个记录号，每张表的最大容量为 2^{32}。设表中的容量为C，则指针占(4*C)B的空间。

2.2 内存数据库 Hash 索引的设计

为了快速地定位数据库的记录，数据库系统设计了多种形式的索引，内存数据库也要建立索引。数据库中广泛使用的哈希索引有链接桶哈希(chained bucket hashing)、可扩展哈希(extendible hashing)、线性哈希(linear hashing)、修正的线性哈希(modified linear hashing)[5]，其中链接桶的哈希使用静态结构处理冲突，速度很快，但不适合动态环境。基于等值的比较，哈希技术能够快速地访问数据库，且易于实现，但不支持范围检索。网管数据库一般不涉及范围的检索，更新却非常频繁，因此，系统适合采用哈希索引。

系统为哈希索引建立 2 个空间：(1)模值空间 HashEntry，即哈希入口；(2)冲突空间 ConBulk，冲突空间存储冲突关键词的记录的指针。

哈希索引入口数据结构如下：

```
struct {
   unsigned long     Entry；
   unsigned long     Depth;
} HashEntry；
```

其中，Entry 为哈希入口的记录号；Depth 为冲突链的深度。

在哈希索引中，哈希函数最常用除留余数法为

H(*key*)=*key* MOD *p*

文献[7]说明了此方法是哈希函数中最好的。根据经验，p=(C/4)*2+13 比较好，既节省了的内存空间，又减少了冲突的概率。

2.3 冲突的处理

索引的冲突处理是实现的关键，为每一张表的索引分配与其表容量 C 相等的冲突空间，由 ConBulk 指向关键词冲突的记录，根据指针(记录号)链接成一条静态链。例如记录号为 7、5、3 的记录关键词冲突，ConBulk6、4、2 处的位置分别存储 5、3、0，其中 0 表示静态链尾，这样就把冲突的关键词链接起来。由于表的容量是规划好的，并且指针的空间很小，因此可一次分配 C 大小的冲突空间，这样哈希索引不必动态地分裂。冲突桶的大小为 C 大小的指针空间，每一个指针为 4 倍，每张表的索引占的总空间为

*p*sizeof(HashEntry) +*C*4

2.4 Hash 索引的算法

2.4.1 插入索引项算法

向表中插入记录时，为此记录分配一个记录号，在索引中插入索引项算法如下：

输入 pIDX /*指向索引的指针*/

TupKey /*记录关键词*/

TupNo /*待插入的记录号*/

步骤如下：

(1) 由关键词 TupKey 和哈希函数生成哈希值，记为 h=Hash(TupKey)；在冲突桶中 TupNo 对应的位置记为 CurPos；

(2)判断 h 处入口冲突深度是否为 0，是则将冲突桶中 CurPos 处置 0，深度值为 1；否则将模值空间的 h 处入口的记录号放在冲突空间的 CurPos 处，深度加 1；

(3)将 TupNo 放在模值空间中 h 的位置，即哈希入口处；

(4)成功返回 TRUE；失败返回 FALSE。

2.4.2 删除索引项

索引项删除算法如下：

输入 pIDX, TupKey。TupNo/*待删除的记录号*/

步骤如下：

(1) 由关键词 TupKey 和哈希函数生成哈希值，记为 h=Hash(TupKey)；模值空间 h 处哈希入口记录号记为 DelPos；

(2)考虑待删除的记录是否等于入口处的记录。是，则把 TupNo 在冲突桶中对应的记录号放在“h 在模值空间的入口处”，TupNo 对应的冲突空间位置置 0，冲突链深度减 1；否，则遍历冲突链：

```
While(DelPos!=TupNo)
{ Ptr=&ConBulk[DelPos-1];
  DelPos=ConBulk[DelPos-1]; }
```

(3)找到待删除的记录号，在冲突链中的位置为 DelPos；把后面的链上*Ptr=ConBulk[DelPos-1] ;把记录号对应的冲突链位置置为 0；冲突链深度减 1；

(4)成功返回 TRUE，失败返回 FALSE。

2.4.3 记录查找算法

利用索引在数据库查找记录算法如下：

输入 pIDX，TupKey。

输出 TupNo /*查找到的记录号*/。

步骤如下：

(1) 根据关键词 TupKey 和哈希函数，求得哈希值 h=Hash(TupKey)；

(2)根据哈希值获得 h 处的哈希入口记录号 TupNo，遍历冲突链，即

```
while(TupNo)
{ 根据记录号，获得该记录的关键词 tempKey；
  比较 tempKey 和 TupKey 是否相等？
  若相等，则返回 TupNo；否则 TupNo=ConBulk[TupleNo-1];}
```

2.4.4 Hash 索引的插入和删除的示例

图 1 给出了哈希索引的结构和插入删除记录示例。

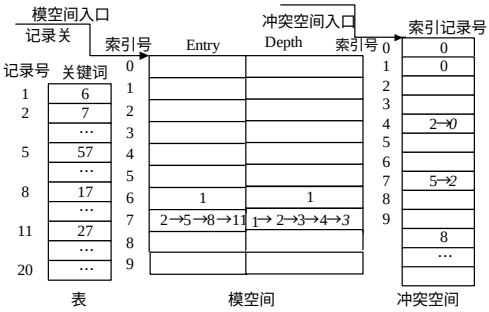

图 1 Hash 索引结构和操作示例

设 p=10，按记录号 1、2、5、8、11、20 的顺序插入索引，在冲突空间 ConBulk 的 10、7、4 单元链成了一条静态链，然后将记录号为 5 的记录删除。图中用箭头指示指针变化的过程，斜体表示删除记录后变化的结果。

2.4.5　算法的分析

由于哈希方法仍是等值的比较，因此可以把平均查找长度作为衡量哈希查找效率的度量，文献[7]定义了填装因子(load factor)填装因子：α =哈希表中填入记录数/哈希表长度，当 α 在 0.9 左右时，查找时间几乎是恒定的[6]。在此系统中假设哈希函数是均匀的，在索引项的插入时，哈希入口的记录平均比较一次，即$p=((C/4)*2+13)$次，$(C-p)$的记录查找 2 次，因此，均匀分布时的平均比较次数为$(p+(C-p)*2)/C\approx 3/2$。删除索引项和查找记录时的平均查找长度与插入相同。即使在最坏的情况下，即哈希函数极度倾斜，在内存中遍历的速度也很快。

表 1 显示了在内存中顺序遍历指针的实验结果。实验环境是 Pentium 4 2.0GHz，内存 512MB，在 Windows XP 下采用 VC6.0 编程实验。从实验结果可知，在内存中，遍历从 10 万~500 万个指针的时间基本是线性的，空间开销不大。因此，在内存数据库中，采用指针指向表的记录在空间上比较合理。即使在冲突链严重冲突的情况下，也可满足时间上的要求。

表 1　采用指针在数据表上的遍历时间

记录条数/$\times 10^4$	1	10	50	100	500
遍历空间/ms	0.0	15.0	16.0	31.0	157.0
所点空间/KB	39	390	1 953	3 906	19 531

3　总结

本文研究了内存数据库的索引特点和设计方法，设计了一种适于内存数据库的哈希索引，使用指针指向记录，利用静态链处理冲突，即节省空间，又满足时间上的要求。笔者分析了在冲突情况下平均的查找长度，研究了 Hash 索引项的插入、删除及利用 Hash 索引查找记录算法，这些都是对内存数据库索引设计的有益探索。

参考文献

1 刘云生. 现代数据库技术[M]. 北京: 国防工业出版社, 2001.

2 Bohannon P, Lieuwen D, Rastogi R, et al. The Architecture of the Dali Main-memory Storage Manager[J]. Multimedia Tools and Application, 1997, 4(2): 115-151.

3 杨武军，张继荣. 内存数据库技术综述[J]. 西安邮电学院学报, 2005, 10(3): 96-99.

4 Molina H G, Salem K. Main Memory Database Systems: An Overview[J]. IEEE Transactions on Knowledge and Data Engineering, 1992, 4(6): 509-516.

5 Lehman T J, Carey M J. A Study of Index Structures for Main Memory Database Management Systems[C]//Proceedings of the 12th International Conference on Very Large Databases, Kyoto. 1986-08: 294-303.

6 Yen L, Bastani F B. Parallel Hashing: Collision Resolution Strategies and Performance[J]. Journal of Parallel and Distributed Computing, 1995, 31(2): 190-198.

7 Owolabi O. Empirical Studies of Some Hashing Functions[J]. Information and Software Technology, 2003, 45(2): 109-112.

8 Luo Wenbin. Hashing via Finite Field[J]. Information Sciences, 2006, 176(17): 2553-2566.

9 阳国贵, 王　升. 主存数据库系统与技术[J]. 软件学报, 1994, 5(3): 22-28.

(上接第 68 页)

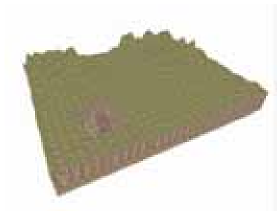

图 2　填挖方分析

图 3　虚拟漫游

4　结论

将可视化技术作为一件数据挖掘工具，可以利用可视化技术丰富的图形表达能力与高度的交互机制，充分调动用户的主观能动性，融入用户的知识与经验，真正实现探索性数据分析。尤其是面向空间数据，通过可视化技术可以有效地表达与分析复杂的空间实体与过程，帮助用户更好地了解其内在结构与本质特征。目前，有关这方面的研究才刚刚起步，如何将可视化与空间数据挖掘有效集成还有待进一步探索。

参考文献

1 刘湘南, 黄　方, 王　平, 等. GIS 空间分析原理与方法[M]. 北京: 科学出版社, 2005.

2 Jiawei H, Kamber M. 数据挖掘: 概念与技术[M]. 北京: 机械工业出版社, 2001.

3 唐泽圣, 孙延奎, 邓俊辉[J]. 科学计算可视化理论与应用研究进展[J]. 清华大学学报, 2001, 41(4/5): 199-202.

4 McCormick B H, DeFanti T A, Brown M D. Visualization in Scientific Computing[J]. Computer Graphics, 1987, 21(6): 973-982.

5 邸凯昌. 空间数据发掘与知识发现[M]. 武汉: 武汉大学出版社, 2001.

6 贾泽露, 刘耀林, 张　彤. 可视化交互空间数据挖掘技术的探讨[J]. 测绘科学, 2004, 25(9): 34-37.

7 Keim D A. Information Visualization and Visual Data Mining[J]. IEEE Transactions on Visualization and Computer Graphics, 2002, 8(1): 1-8.

8 Oliveira F, Levkowitz H. From Visual Data Exploration to Visual Data Mining: A Survey[J]. IEEE Transactions on Visualization and Computer Graphics, 2003, 9(3): 378-384.

9 Maceachren A M, Wachowicz M, Edsall R, et al. Constructing Knowledge from Multivariate Spatiotemporal Data: Integrating Geographic Visualization with Knowledge Discovery in Databases (KDD)[J]. International Journal of Geographic Information Science, 1999, 13(4): 311-334.

10 芮小平, 张彦敏. 空间信息可视化挖掘研究[J]. 测绘科学, 2005, 30(2): 64-66.

11 龚建华, 林　晖. 虚拟地理环境——在线虚拟现实的地理学透视[M]. 北京: 高等教育出版社, 2001.

12 王占刚. 基于改进三棱柱的三维地质模型及其可视化研究[D]. 北京: 中国矿业大学, 2003.