数据仓库中大表数据的一种快速选择方法①

王　勇，戴牡红

(湖南大学 信息科学与工程学院，长沙 410082)

摘　要：传统的数据仓库优化技术在大表的选择上只能实现降低查询处理时间或减少存储以及维护开销中的一个，无法达到同时优化的效果。在传统的优化技术上，利用水平分区和位图索引方法的特点，提出了 HP→BIs 方法。实验结果表明，通过该方法可以同时降低查询处理时间和减少存储以及维护成本。

关键词：数据仓库；决策支持系统；位图索引；水平分区；性能优化

A Fast Selection Method at Large Table of Data in Data Warehouse

WANG Yong, DAI Mu-Hong

(Department of Information Science and Engineering, Hunan University, Changsha 410082, China)

Abstract: Traditional selecting optimization structures at large table in data warehouse can only either shorten the time of query processing, or reduce the storage and maintenance overhead. It can't achieve a combined effect at the same time. Based on the traditional optimization techniques, this paper proposes a new approach, namely, HP→BIs, by exploiting their similarities of horizontal partitioning and bitmap indexes. Experimental results show that with this approach, we can significantly shorten the time of query processing, reduce the storage and maintenance cost.

Key words: data warehouse; decision support system; bitmap indexes; horizontal partitioning; optimization

现在，企业的数据仓库中积累了大量的原始数据，而这些数据大都存储在几个主表（这些主表由于数据量巨大并且表字段多，本文将称之为大表）中，这些大表在日常查询中被经常涉及到[1]。为了快速有效的从这些大表中查询出有用信息，人们提出了许多优化技术。这些技术主要分为两大类：

1. 冗余结构，如：物化视图、索引及垂直分区等；

2. 非冗余结构，如：水平分区和并行处理等[2-8]。

但是单独使用这些优化技术只能实现降低查询处理时间或减少存储以及维护开销中的一个，无法达到同时优化的效果。为提高查询大表数据的效率，针对水平分区和位图索引方法的特点，我们提出了 HP→BIs 方法。该方法是通过特定方式将水平分区和位图索引两者组合起来使用。实验结果表明，HP→BIs 方法可以明显的降低查询处理时间和减少存储以及维护成本。

1　水平分区及位图索引概述

1.1 水平分区概述

数据库的水平分区是一种处理大表的优化技术。它是对表的行进行分区，通过这样的方式不同分组里面的物理列分割的数据集得以组合，从而进行个体分割（单分区）或集体分割（一个或多个分区）。水平分区可以将这些大表分割成便于管理的小单元，且每一分区可进一步划分为更小的单元。用'分而治之'的方法来处理无限膨胀的大表，给大表在物理一级的可管理性。并且所有在表中定义的列在每个数据集中都能找到，所以依然保持着表的特性[2]。将大表分割成较小的分区具有如下优点：①增强可用性;②维护方便；③均衡 I/O；④改善查询性能；⑤分区是对用户透明的[3-5]。

1.2 位图索引概述

位图索引是一种新兴的索引结构，是一种二进制编码方法，其主要在数据仓库和决策支持系统中用到，

① 基金项目:国家自然科学基金(61070194)
收稿时间:2011-01-24;收到修改稿时间:2011-03-07

它可以显著地提高性能和节省存储空间，因此得到广泛的应用。通常情况下，索引都要耗费比较大的存储空间，而位图索引采用了压缩技术实现磁盘空间缩减。

位图索引的基本原理是在索引中使用位图而不是列值。它主要针对大量相同值的列而创建(例如：类别,操作员,部门 ID,库房 ID 等)。位图索引是用一组{0,1}位串来表示数据表中的某一列属性值，其{0,1}位串的位数为该列属性值不同取值的个数。

但是位图索引的选择比较复杂。假如表 T 有 n 个属性，那么所有的索引组合是 $K=(2^{2^n}-1)$种。如果 n=5，则 $K=2^{32}-1$[5-9]。

2 大表数据快速选择方法的实现

2.1 HP→BIs 方法提出的背景

湖南某高尔夫有限公司作为为高尔夫运动者提供打球场地预定的一家高尔夫球业务公司，会员的中介费用成为公司的主要的收入来源。在过去，由于传统的高尔夫运营模式，即“贵族化”路线，使得企业的会员只是很少的一部分白领阶层。随着社会的发展，公司的业务范围越来越大，公司在客户关系管理方面面临着巨大问题。同时，随着公司业务的拓展，公司数据库储存的客户数据也急剧的增加，公司决策层需要从多方面对客户数据进行分析，从而发掘更多的潜在客户。为给领导决策和业务人员技术分析提供多角度、极具参考价值的第一手辅助信息，该公司设计、开发了 GOLF 决策支持系统。该系统从异构数据源中抽取、转换和装载数据，经过预处理加工后，构建成一个涵盖预订、产品、话务、结算等业务的数据仓库，然后基于相关业务分析指标和多维度视角对数据仓库进行数据挖掘和深度分析。然而数据并不代表有用的信息，如何快速有效的从这些大表中进行信息查询变得尤为重要。

如上所述，虽然水平分区（HP）和位图索引（BI）有许多优点，但是也有不利的方面。对 GOLF 决策支持系统而言，增加位图索引及维护会产生大量的系统开销——而不必要的索引更会降低系统性能。因此，必须合理、正确的使用索引。水平分区虽然可以将这些大表分割成便于管理的小单元，但是有些表的数据会在很短的时间内迅猛地增长。所以，单独使用水平分区，有时候也难以对这样的庞大的数据集进行有效的管理[3,5,7,9]。

综上所述，在 GOLF 决策支持系统中，单独使用水平分区或位图索引，都无法满足需求。通过阅读文献以及实际操作，并结合水平分区和位图索引的特点，我们提出了 HP→BIs 方法。

现在绝大多数数据库都支持水平分区和位图索引。HP→BIs 方法首先将数据库中的大表分为几个子分区，然后再在这些子分区上建立合适的位图索引。架构图如图 1 所示。

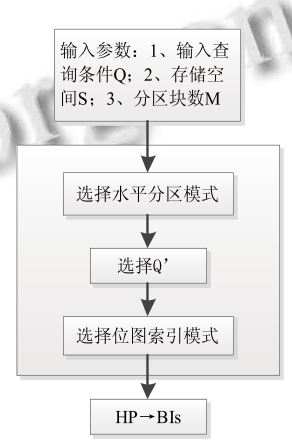


图 1 HP→BIs 方法总体框图

2.3 HP→BIs 方法的描述及实现

2.3.1 HP→BIs 方法描述

HP→BIs 方法可以分为如下步骤：

假设数据仓库中有一个大表 T，维表集 $D=\{D_1, D_2,\ldots,D_m\}$以及一组常见的查询 $Q=\{Q_1,Q_2,\ldots,Q_n\}$。并且每个查询 $Q_i(1<=i<=n)$有查询权值，参数 S 表示所需磁盘空间，参数 W 代表水平分区的分区块数。

①求出大表的属性集 DA，并且水平分区和位图索引共享这个集合；并求出集合 DA 的最小完备集 CMDA，CMDA⊆DA；

②针对查询集合 $Q=\{Q_1,Q_2,\ldots,Q_n\}$，分区块数 W 以及 CMDA，使用遗传算法，对大表 T 进行水平分区，得到集合 HFSET⊆CMDA；

③对每个查询 Q_i， 求出它的权值 $R(Q_i)$，且

$$R(Q_j)=\frac{C[Q_j,FS]}{C[Q_j,\phi]} \quad (1)$$

其中，$C[Q_j, FS]$代表在分区后执行查询 Q_j 的代价，$C[Q_j,\Phi]$代表在未分区表中执行查询 Q_j 的代价。 对于 DBA 设定的 λ 值，如果 $R(Q_j)<\lambda$，则表明 FASET 分区

对该查询有利。否则，对该查询没有帮助。求出所有查询 Q'={Q'1, Q'2,…,Q'm}，Q'i 代表所有 R(Qj)>λ 的查询条件。

④确定位图索引属性：基于查询 Q'={Q'_1, Q'_2,…,Q'_m}，求出合适的索引属性集 IA。属性集 IA 可能和 FASET 重复，所以我们需进一步计算，得到最终的位图索引属性集 BISET=IA-HFSET。

2.3.2 水平分区的选择

为了更好地将大表数据进行分区，我们采用贪心算法，该算法描述如下：

Generate initial population ;
Perform selection step;
WHILE stopping criterion not met DO
Perform crossover step;
Perform mutation step;
Perform selection step ;
END WHILE[10]

2.3.3 位图索引的选择

为了更好的选择表中的属性列来建立位图索引，我们采用遗传算法，该算法描述如下[11]：

INPUTS:
查询集合：Q = {Q_1,Q_2 ,…,Q_n}；
S：表示所需的存储空间；
BISET：表示执行位图索引的属性集合；
BIm: 表示在属性 Am 上执行位图索引；
Size(BIm)表示 BIm 的存储开销；
C[Q',HP]:表示使用水平分区执行查询 Qi 的系统开销；
OUTPUT: $Config_{finale}$: 最终选定的 BIs 方法.
BEGIN
$Config_{finale} = BI_{min}$;
S:= S-Size(BI_{min});
BISET := BISET-A_{min}; (A_{min} 表示用于定义 BI_{min} 的属性)
WHILE (Size($Config_{finale}$) <= S) DO
FOR each $A_m \in$ BISET DO
IF(C[Q',($Config_{finale}$[BI_m))]<C[Q',HP])
AND (($Size(Config_{finale}[BI_m)$ <= S))
THEN
$Config_{finale}$:= $Config_{finale}$ U BI_m;
Size($Config_{finale}$) :=
Size($Config_{finale}$)+Size(BI_m);
BISET:= BISET –A_m;
END

3 实验结果及分析

为了评估 HP→BIs 方法的性能，我们将以湖南某公司 GOLF 决策支持系统的数据为基础，比较以下四种方法：没有进行任何优化、使用水平分区、使用位图索引以及使用 HP→BIs 方法。

数据集：在 GOLF 决策支持系统中，有一个主表(预定基本信息表 BOOK((BookNO,CardNO,CrsID, BookType, OperTime 等属性)，1378 万条记录),4 个维表{预定商品信息表 BookItem ((BookItemID, BookNO, PriceID, ItemName, ItemNum 等属性)，3754 条记录)，预定嘉宾信息表 BookGuest ((Guest- ID, BookNO, GuestName, Mobile, CardNO 等属性)，8946 条记录)；预定结算信息表 BookSettle ((Settle-ID, BookNO, SettleType, SettleTime, SettleStatus 等属性),649 条记录)，预定处理过程表 BookDeal ((DealID, BookNO, DealMan 等属性),253 条记录)}。

查询：我们将执行 50 个查询语句，其中 15 个统计运算 count( )，15 个求和运算 sum( )，10 个求平均值运算 avg( )，5 个求最大值运算 max( ),5 个求最小值运算 min( )。

测试环境：

① 微机配置如下：Intel(R) Core(TM)2 Duo CPU 2.53GHz 微机(2G 内存、160GB 硬盘)；

② 操作系统为 Windows XP SP3；

③ 数据库为 Oracle 11G。

3.1 实验结果及分析

首先，我们进行单独使用水平分区和位图索引技术的性能比较。此时水平分区块数 W =60（根据 2.3.2 所述的贪心算法，求出以每月为单位的水平分区，即：5 年*12 个月=60）。实验结果如图 2 所示：

图 2 所示的是对于磁盘空间从 40 到 400M 这个范围内，执行上述的 50 个查询，分别使用水平分区和位图索引方法的性能对比。我们可以很明显的看出，当给定的磁盘空间在 40 到 150 M 之间的时候，使用水平分区的性能明显优于使用位图索引。但是磁盘空间越来越大时，特别是对于其中的 15 个关于统计方面的查询，使用位图索引的优势就立即显示出来了。这是因

为使用位图索引后，有关统计方面的查询就不再需要去直接访问主表了，从而大大的提高了查询速度。并且，我们可以从本实验中得出，如果大部分查询都与统计相关，那么 DBA 完全可以单独使用位图索引方法。

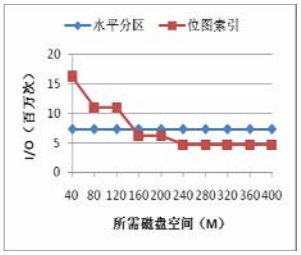


图 2

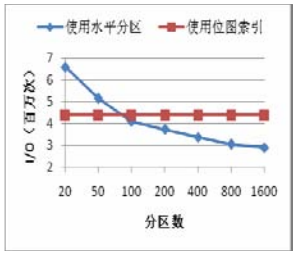


图 3

其次，我们计算分区数 W 的变化对水平分区以及位图索引性能的影响，此时存储空间 S=400M。实验结果如图 3 所示。实验结果表明，随着分区块数的增大，水平分区的性能越来越高。

第三，我们将分别比较以下四种方法（没有进行任何优化、使用水平分区、使用位图索引以及使用 HP→BIs 方法），如图 4 所示。

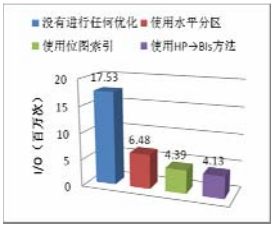


图 4

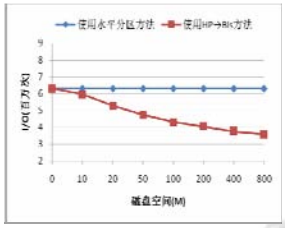


图 5

由图 4 可以看出，采用 HP→BIs 方法，其展现的 I/O 性能要远远胜于其中的两种方法（没有进行任何优化、水平分区方法），略高于位图索引方法。实验结果同时显示，采用 HP→BIs 方法所需磁盘空间也比单独使用位图索引方法少很多。单独使用位图索引需要磁盘空间 117M，而使用 HP→BIs 方法只需要 68M，降低了 41.9%。

最后，当磁盘空间在 0-800M 在个区域内变化时，我们比较 HP→BIs 方法与水平分区方法(W=60)的 I/O 性能。实验结果如图 5 所示，随着磁盘空间的增大，HP→BIs 方法所展现的性能越来越优于使用水平分区方法。

4 结语

本文主要讨论了数据仓库中大表数据的一种快速查询方法。由于每种数据库优化技术都有各自的优缺点,所以为了达到更好的优化效果,必须通过某些特殊方式把两种或多种优化技术结合起来使用。本文中的 HP→BIs 方法便是通过利用水平分区和位图索引方法的特性结合起来使用，从而达到了降低查询处理时间和减少存储以及维护成本。

在接下来的工作中,我们将继续研究传统的优化技术（如：视图、分区、索引等）的特性，我们相信通过特定的算法将它们结合起来，能更好的提升数据仓库中大表的查询速度等性能。

参考文献

1 Power DJ, Sharda R. Decision Support Systems. Berlin: Springer Berlin Heidelberg, 2008.1539–1548.

2 刘博.Oracle 数据库性能调整与优化.大连:大连理工大学, 2007.

3 Bellatreche L, Missaoui R, Necir H, Drias H. Selection and pruning algorithms for bitmap index selection problem using data mining. The 9th International Conference on Data Warehousing and Knowledge Discovery. Regensburg, 2007. Heidelberg: Springer Berlin, 2007. 221–230.

4 Agrawal S, Narasayya V, Yang B. Integrating vertical and horizontal partitioning into automated physical database design. Proc. of the ACM International Conference on Management of Data, Paris, New York: ACM, 2004. 359–370.

5 Oracle Data Sheet.Oracle partitioning.White Paper: http://www.oracle.com/technology/products/bi/db/11g/, 2007.

6 Azefack S, Aouiche K, Darmont J. Dynamic index selection in data warehouses. 4th International Conference on Data Warehousing and Knowledge Discovery. Regensburg, 2007. Heidelberg: Springer Berlin, 2007. 64–73.

7 张闯,周丽娟,高志新,褚金凤.Oracle 10g 索引技术在数据仓库中的应用.计算机应用,2007,26(1):35–37.

8 万怀宇,黄厚宽.位图索引及其在数据仓库中的应用研究.计算机应用,2006,15(12):31–33.

9 Agrawal S, Chaudhuri S, Narasayya V. Automated selection

(下转第 229 页)

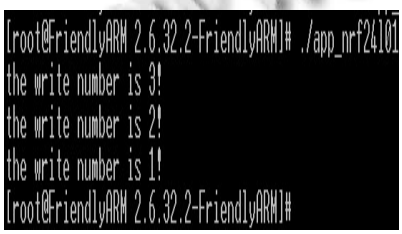
阻塞后，写入下一部分数据

```
    .....
}
close( nrf24l01_fd); }
```

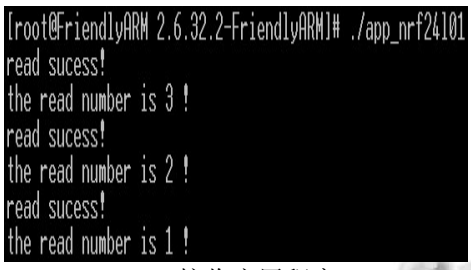
巡检方式 :

```
main（）
{   int i,nrf24l01_fd，receive[32];
nrf24l01_fd=open("/dev/nrf24l01",O_RDWR);     //以可读可写的方式打开设备文件
   while(1)
{ i=ioctl(nrf24l01_fd,0,0); //调用 ioctl 函数查询标志寄存器状态
if（i==3）
read(nrf24l01_fd,receive,1); //寄存器状态表明接收到有效数据则调用 read 函数
.....}
close( nrf24l01_fd); }
```

以上程序在 linux 平台上的实验效果截图如图 1 所示：



(a) 发送应用程序



(b) 接收应用程序

图 1 实验效果截图

以发送“321”这个数字队列为例，从图 1 可知发送和接收程序均较好地实现了功能。

5 总结

本文结合 linux 系统字符驱动的基本框架，详细论述了开发无线通讯芯片 nRF24L01 驱动的过程和方法，对开发其他类型的字符驱动也具有一定的借鉴价值。

本文的创新点是在驱动中采用了中断和巡检两种方法来处理无线数据的收发情况，以适应于不同的应用环境，并得到了验证。目前该驱动已成功地运用在本实验中心对气体传感器信号的无线监测项目当中。

参考文献

1 时志云,盖建平.新型高速无线射频器件 nRF24L01 及其应用.国外电子元器件,2007,(8):42–44.

2 陈莉君.深入理解 Linux 内核源代码.北京:人民邮电出版社,2002.

3 李俊.嵌入式 Linux 设备驱动开发详解.北京:人民邮电出版社,2008.

4 李桦,高飞,等.嵌入式 Linux 设备驱动程序研究.微计算机信息,2010,5(2):68–70.

5 Rubini A, Corbet J. Linux Device Drivers. 2nd Ed. O’Reilly Media, Inc.June. 2001.

(上接第 237 页)

of materialized views and indexes for SQL databases. Proceedings of 26th International Conference on Very Large Data Bases, Cairo, 2000. San Francisco: Morgan Kaufmann Publishers Inc, 2000. 496–505.

10 Ladjel B, Kamel B. An evolutionary approach to schema partitioning selection in a data warehouse. Proc. of 7th International Conference on Data Warehousing and Knowledge Discovery. Copenhagen, 2005. Heidelberg: Springer Berlin, 2005. 115–125.

11 Bellatreche M, Drias N. Selection and pruning algorithms for bitmap index selection problem using data mining. 9th International Conference on Data Warehousing and Knowledge Discovery, Regensburg, 2007. 221–230.