港口数据 MOLAP 中层次维的存储

潘明霞,王 宇
PAN Ming-xia,WANG Yu

大连理工大学 管理学院 信息管理系,辽宁 大连 116024
Department of Information Management,School of Management,Dalian University of Technology,Dalian,Liaoning 116024,China
E-mail:ychpmx@163.com

PAN Ming-xia,WANG Yu.Storage of hierarchical dimension in port data MOLAP.Computer Engineering and Applications,2010,46(3):211-214.

Abstract: With the decision requirements of port enterprises,a data cube is confirmed by analyzing the present port data. MOLAP is the application query based on data cube,and support hierarchical dimension is an important feature in MOLAP,usually,the hierarchical dimension is stored by the array,but the array storage cannot present the hierarchical feature of the dimension,and it makes the redundant data cell.For the lack of array storage,use the dimension storage tree to store the dimensional information,which can present the hierarchical information of the dimension,and eliminate the redundant data cell,so it facilitates the query and the update of the hierarchical dimension.Also,it can save the storage space and optimize the query time,because it is encoded by binary coding.

Key words: port;data cube;MOLAP;hierarchical dimension

摘 要:从港口企业面临的决策需求出发,分析港口现有数据来构建港口数据立方体。多维联机分析处理(MOLAP)是在数据立方体上进行的应用查询,支持维层次是 MOLAP 的一个重要特征,一般层次维是以数组形式进行存储的,但是数组存储不仅不能体现维的层次特征,还使得数据单元出现冗余。针对数组存储的不足,采用维层次存储树来保存层次维信息,体现了维的层次特性,消除了冗余数据,方便层次维的查询和更新,且各层维成员采用二进制编码方式,不仅节省了存储空间,还提高了查询效率。

关键词:港口;数据立方体;多维联机分析处理;层次维
DOI:10.3778/j.issn.1002-8331.2010.03.065 **文章编号:**1002-8331(2010)03-0211-04 **文献标识码:**A **中图分类号:**TP311;U691

1 引言

随着信息化在港口的普及,管理信息系统、电子港务平台、电子商务平台、电子口岸等技术已在港口得到了广泛应用。这些技术的应用在提高港口生产效率的同时也使得港口数据库中的数据每天高达数万条,这些数据分布在各个业务数据库中。面对这些纷繁复杂的数据,港口业务人员采取了数据大集中的措施来利用这些数据,但是它为管理人员提供的辅助决策信息只表现为数据表层的查询和统计,不能获得数据属性的内在关系和隐藏的信息,即淹没了的信息乃至知识,造成了资源的浪费。现有港口系统是典型的 OLTP 系统,它采用传统 OLTP 应用系统的设计方法,按照港口实际生产流程来设计和存储各个业务系统的数据库中的表,而这些数据往往呈现分散存储、数据结构不统一、数据冗余、数据不完整等特征,使得数据难以集成为统一的数据平台。为了充分利用这些重要的信息资源,来辅助港口管理人员决策,迫切需要建立一种新型系统,这种新型系统应该为分析任务而设计;提供全港综合而完整的数据概括;数据可靠且具有较强的一致性;能对当前的和历史的数据进行报表处理和联机分析处理;可以从不同角度按不同主题进行数据分析,为港口管理层提供决策依据。基于以上特点,建立港口生产数据仓库的思想便应运而生。数据仓库是面向主题的、集成的、稳定而随时间不断变化的历史数据的集合,它能够将分布在企业的各种数据进行再加工,从而形成一个综合的、清洁规范的、面向分析的统一平台。文献[1]提出了构建港口数据仓库的思想,文献[2-4]进一步对港口数据仓库构建进行了研究,该文在文献[2]的基础上,研究数据仓库上的一个重要应用——OLAP。目前 OLAP 主要有以关系表为存储结构的 ROLAP 和以多维立方体存储的 MOLAP 两种查询[5]。维是 OLAP 的查询角度,ROLAP 中维存储在维表中,通过维表的主键与事实表的外键连接,查询时需要多表连接,查询时间较长,影响查询效率。为了提高查询响应速度,文献[6-8]提出了利用位图索引技术,一定程度上提高了查询的速度,但是 ROLAP 利用的是成熟的关系数据库理论和技术,关系数据库主要是面向 OLTP 的,以至于 ROLAP 直接依赖于数据仓库,与之紧密结合,表示多维数据困难,数据的存储效率和访问的性能相对较差。而 MOLAP 事先将汇总数据计算好,保存在自己特定的数据库中,使得 OLAP 与数据仓库分离开,降低了耦合度。其中的维作为

作者简介:潘明霞(1981-),女,硕士研究生,研究方向为数据仓库,数据挖掘,决策支持等;王宇(1959-),男,博士,教授,研究方向为数据挖掘,决策支持,知识管理等。

收稿日期:2008-07-29 **修回日期:**2008-10-20

存储在数组单元中的度量值的坐标,查询时通过坐标可直接定位到数组的任何一个数据单元,因此访问速度较快。文献[9]采用了数组的方式来存储维信息，但是它忽略了维的层次关系,从而造成了数据单元的数据冗余,存储空间浪费,以至于查询效率不高。该文针对港口数据立方体维具有层次性的特点,研究港口数据 MOLAP 层次维模型及其存储实现,采用维层次存储树来存储维信息，其中每层维成员都以二进制进行编码,不仅节省了存储空间,还优化了维信息的查询和更新。

2　港口数据立方体

港口信息技术的广泛运用提高了港口生产效率,使得港口每天的交易数达到数万条，这些数据保存在各个数据库中,如客户管理、仓库管理、船舶管理、货物管理、车辆调度等数据库。这些数据在各自的数据库系统中有着各自的存储格式,数据分散、格式不一致、数据冗余等特点使得现有的数据不能得到合理的利用,造成了资源的浪费。为了充分利用这些数据资源来辅助管理决策,港口有必要将这些分散存储在各自数据库中的数据集中到一个统一的数据平台上,按照统一的格式存储,达到数据清洁规范,即利用现有的数据建立港口数据仓库来辅助决策。

数据仓库中的数据是多维的,通常用数据立方体来表示多维数据,数据立方体使得分析者能够从不同角度(维)来观察存储在数据仓库中的数据,或者对这些数据进行多维分析,立方体中的每个单元代表一个度量值。文章选取港口数据库中存储的数据,经过港口数据仓库模型的分析和构建,得到了经过数据预处理的清洁的数据。构建的数据仓库包含维(时间、货主、货物类型、航线、船舶)和度量(交易次数、货物重量、利润),其中,维有着各自的层次关系,时间(年＜季度＜月份),货主(区＜省份＜城市),航线(国家航线＜省份航线＜地区航线)。从中抽取出时间维、货主维和航线维,度量为交易次数构建三维数据立方体,表示为图 1,这里数据立方体实例采用了不同的数据粒度,时间维为季度,货主维为城市,航线维为地区航线。

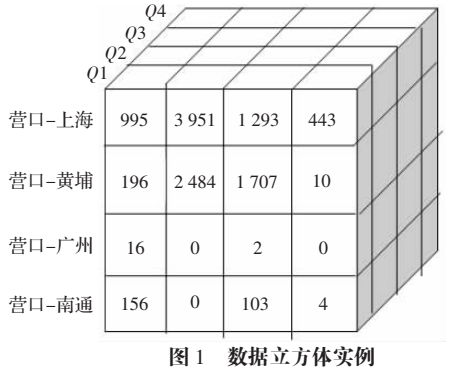


图 1　数据立方体实例

3　MOLAP 中层次维的编码

MOLAP 是数据立方体上的一个应用查询分析,它从多角度(多维)来察看和分析数据立方体中的数据。MOLAP 中的维有着各自的层次关系,它们之间的关系可以用维概念层次树来表示。维概念层次树是数据立方体中的每一个维的层次表示,它是数据归约表示的一种方式,可以呈现维的粒度。例如港口数据立方体中的货主维的维层次树如图 2 所示。

为了便于对数据立方体进行快速查询,通常需要对维属性进行编码,针对每棵维层次概念树,可以采用十进制编码、位图编码和二进制编码的方式。本章采用二进制编码方式先对维层次成员进行编码,接着采用位拼接法对维成员编码,二进制编码的优越性如 3.2 小节表 2 所示。

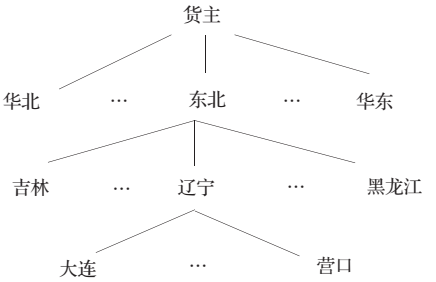


图 2　货主维的概念层次树

3.1　维层次成员编码

假设 L_j^i 为维 D_i 中的第 j 层属性,其值域为 $dom(L_j^i)=\{d_1^i, d_2^i,\cdots,d_i^i,\cdots,d_m^i\}$,则在维 D_i 中的维层次 L_j^i 各属性成员编码为:

$$Bit(L_j^i):dom(L_j^i)\rightarrow \{\langle b_{k-1},\cdots,b_i,\cdots,b_0\rangle | b_i\in\{0,1\}, i=0,1,\cdots,k-1\}$$

k 为维 D_i 的维层次 L_j^i 中成员二进制编码的位数，通常取值为 $k=\lceil \mathrm{lb}m\rceil$,其中 m 为 L_j^i 层中不同成员的最大个数,即该层中不同成员的最大个数。如货主维共有三个不同维层次{地区、省、城市},对于最高层次“地区”来说,不同成员的个数为 6,其编码位数为$\lceil \mathrm{lb}6\rceil$=3 位,同样对于“省”,其最大个数为 16,编码位数$\lceil \mathrm{lb}16\rceil$=4 位,“城市”的不同成员为 61 个,编码位数$\lceil \mathrm{lb}61\rceil$=6位。

3.2　维成员编码

D_i 维的维成员编码是由 D_i 维中所有维层次属性($L_i^1, L_i^2,\cdots, L_i^h$)的二进制编码按层次由高到低依次进行位拼接组合而成的混合码[8,10],即:

$$Bit(D_i)=(\cdots((Bit(L_i^1)<<BitNumber(L_i^2)|Bit(L_i^2)<<BitNumber(L_i^3)|Bit(L_i^3))\cdots)<<BitNumber(L_i^h)|Bit(L_i^h)$$

其中 $BitNumber(D_i)$ 且维 D_i 的各个成员编码总位数,是其所在 D_i 维中所有维层次成员的二进制编码的位数之和,即:

$$BitNumber(D_i)=\sum_{j=1}^{h} BitNumber(L_i^j)$$

结合图 2 的货主维层次树表示的维层次,得出货主维的层次维的二进制编码如表 1 所示。该维的总位数,即货主维编码总位数为 3+4+6=13 位,可以用 2 个字节存储,东北地区辽宁省大连市的维层次编码为 0101100100100,它是由 Bit(地区=东北) Bit(省=辽宁)和 Bit(城市=大连)组合而成,即:

表 1　货主维的二进制编码表

区层次编码表		省层次编码表		城市层次编码表		货主维编码表	
区	编码	省	编码	市	编码	地区	编码
…	…	…	…	…	…	…	…
华东	000	黑龙江	1010				
华北	001	吉林	1011	大连	100100	东北辽宁大连	0101100100100
东北	010	辽宁	1100	营口	110100	东北辽宁营口	0101100110100
…	…	…	…	…	…	…	…

Bit(东北地区辽宁省大连市)=(Bit(地区=东北)<<Bit-$Number$(省)|Bit(省=辽宁))<<$BitNumber$(城市)|Bit(城市=

表 2　维二进制编码与其他编码方式的比较

	航线维				货主维				时间维		
	国家	省份	地区	总计	区	省	市	总计	季度	月份	总计
维成员个数	46	65	153		6	16	61		4	12	
维成员总数	46	2 990	457 470	457 470	6	96	5 856	5 856	4	48	48
整数编码/byte	2	4	6	6	1	2	4	4	1	2	2
位图编码/bit	46	2 990	457 470	460 506	6	96	5 856	5 958	4	48	52
维层次编码/bit	6	7	8	21	3	4	6	13	2	4	6

大连)=(010<<4|1100)<<6|100100=0101100100100

从港口数据仓库中抽取出时间维、货主维和航线维，表 2 列出了由它们构成的维层次编码与整数编码、位图编码方式所需的 *Bit* 位的比较。由于维层次编码一般只需⌈lb*m*⌉个 *Bit*位，比位图编码方式所需 *m* 个 *Bit* 位节省了很多的空间，且随着维成员个数的增多，其节省的空间将呈指数增加，维层次编码也比整数编码数据平均压缩 3~4 倍。

4　MOLAP 中层次维的存储

对各维成员编码后，就要按照一定的数据结构将层次维进行存储。一般的维成员层次存储可用一维数组和二维数组来表示，但是数组存储不能体现层次关系，且会出现冗余数据单元现象。本章根据维层次成员之间的关系，采用维层次存储树来存储层次维信息，接着对维层次存储树进行查询和更新，并与二维数组存储进行性能比较。

4.1　维层次存储树

根据维层次之间的关系，构建维层次存储树来存储各维的信息，整个 DataCube 需要 *n* 棵(*n* 为维数)维层次存储树来进行存储，这 *n* 棵维层次存储树相互独立，每一棵树存储一个对应维的成员。这里所定义的维层次存储树的数据结构如下：

```
Typedef struct Treenode
{
  struct Treenode *parent;
  struct Treenode *children;
  struct Treenode *brother;
  varbinary node_bit;
  int node_flg;
}Treenode;
```

*parent 是指向父结点的指针；*children 是指向孩子结点的指针；*brother 是指向兄弟结点的指针；node_bit 是表示此结点的层次编码组合；*node_flg* 是标记位，当 *node_flg*=1 时，该结点为目录结点，当 *node_flg*=0 时，该结点为叶子结点。

根据图 2 所示的货主维的概念层次树，按照维层次存储树的结构构建出货主维的维层次存储树如图 3 所示，它包括地区、省份和城市三个层次。

4.2　维层次存储树的查询和更新

构建维层次存储树的目的是为了更方便地对维层次信息进行查询和更新，以下主要介绍在维层次存储树上进行的查询和更新操作。

4.2.1　查询

可以运用两种方法进行查询：一是从根结点开始从上而下进行层级查询，对于一个结点的编码，从根结点开始逐一进行前缀编码匹配，直到匹配成功为止；另一个是对每一层进行顺序查询，从指向本层次叶子结点的指针开始，逐一检索，找到目的结点，然后沿着维层次树向上，找到其祖先结点，进行编码拼接得到该结点编码。若知道一个结点的编码为0101100110100，从树的根结点开始，第一层匹配 010 为东北地区，第二层匹配 1100 为辽宁省，第三层匹配为 110100 为营口市，因此该结点为东北地区辽宁省营口市；同理，若要查询东北地区辽宁省营口市该结点的编码，可以从第三层的第一个结点开始，找到营口市在该层的编码 110100，接着沿着 parent 指针上溯到第二层辽宁省，得到辽宁省的编码 1100，接着上溯到东北地区得到其编码为 010，再进行编码拼接就可以得到目的结点的编码 0101100110100。

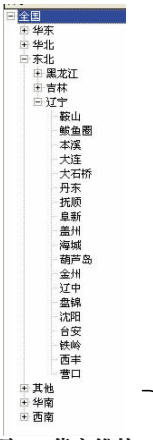


图 2　货主维的概念层次树

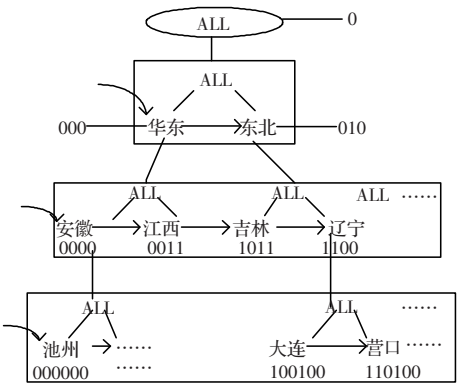


图 3　货主维层次存储树

4.2.2　更新

(1)维层次成员的更新：在维层次存储树中，可以方便地进行层次成员的更新。当增加一个结点时，只是找到其父结点，然后插入，同时在插入位置改变其兄弟结点的指针；与此类似，当删除一个结点时可以找到其父结点，然后删除该结点，同时改变在删除位置的兄弟结点的指针。

(2)维层次的更新：维层次的更新主要是维层次的增加删除。增加一个维层次相当于增加一棵层次树，在要进行插入的两个层次之间，新增加层次结点的孩子指针指向原来上层结点的孩子，同时把其父指针指向上层结点，最后原来上层结点的孩子指针指向新增加的层次结点；与此类似，删除一个层次相当于删除一棵层次树，把要删除层次结点的父结点的孩子指针指向它的孩子结点，同时把这些孩子结点的父指针指向其父结点，然后删除这些层次结点即可。

(3)维的更新：维的增加就是增加一棵维层次存储树，根据新增加的维进行构造一棵维层次存储树。删除维即把要删除维对应的维层次存储树删除。

4.3　性能分析

相对于二维数组，采用维层次存储树来存储维信息，在维存储空间和查询时间上有着较好的性能，消除了冗余数据，数据单元所需的存储空间减少，进而提高了查询效率。下面主要

从存储空间和查询时间两个方面来比较维层次存储树和二维数组的性能。

(1)维层次存储树与二维数组的存储空间的比较

假设维 D 最低层为 d_h,而最高层 ALL 不需存储,可用二维数组 $Array[|d_h|],[|D|-1]$表示这个维层次的成员结构。设 u_d 是层次 d 成员数组元素所占的存储单元大小,所需存储空间为:

$$|d_h|*(h-1)*\underset{d\in D,d\neq ALL}{MAX}(u_d)$$

货主维的二维数组存储方式如图 4 所示。

城市层	省份层	地区层
大连	辽宁	东北
营口	辽宁	东北
本溪	辽宁	东北
…	…	…
南京	江苏	华东
苏州	江苏	华东
常州	江苏	华东
…	…	…

图 4　货主维层次结构的二维数组存储方式

如图 3 所示的维层次存储树来存储维层次信息,去除了层次之间的冗余数据,节省了存储空间。假设维 D 有 h 层,最高层 ALL 不需要存储,则第 2 层到第 h 层所需存储空间为 $\sum_{i=2}^{h} d_i*MAX(u_{di})$,$u_d$ 是 u_{di} 中的最大值,则

$$\sum_{i=2}^{h} d_i*MAX(u_{di})<\sum_{i=2}^{h} d_i*MAX(u_d)$$

由于维层次成员呈逐层增多的趋势,很明显 $\sum_{i=2}^{h} d_i<|d_h|*(h-1)$,因此有:

$$\sum_{i=2}^{h} d_i*MAX(u_{di})<|d_h|*(h-1)*\underset{d\in D,d\neq ALL}{MAX}(u_d)$$

用维层次树存储数据虽然去除了层次之间的冗余数据,减少了数据值的存储空间,但是维层次存储树中包括三种指针(parent,children,brother),它们也占据一定的存储空间。设存储树有 h 层,每层结点个数为 d_i,则整棵树的指针个数为 $3\sum_{i=2}^{h} d_i-d_h-h$,在层数很大,每层结点又很多的情况下,指针占据的存储空间还是很可观的。

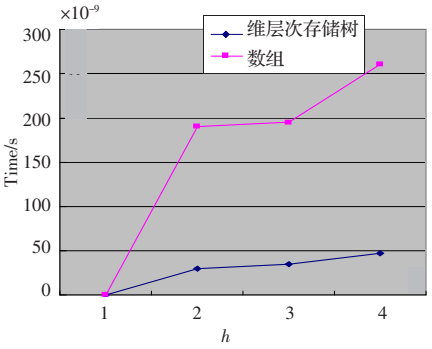
(2)维层次存储树与二维数组的查询时间的比较

图 5 显示了维层次存储树与二维数组的查询性能的比较。可以看出,多维多层数据以维层次树的方式存储数据不仅体现了数据之间的层次关系,去除了层次之间的数据冗余,而且数据的查询效率也得到了提高。

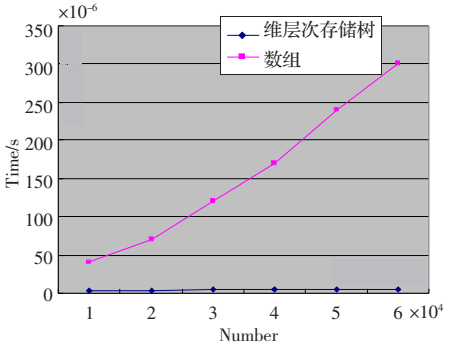
图 5(a)表示在数据量不变的条件下,查询时间随层数的变化情况,可以明显看出维层次存储树所需的查询时间小于二维数组所需的时间,且随着维层次的增加,查询所需时间的差距越明显;图 5(b)表示层数相同的情况下,查询时间随数据量的变化情况,随数据量的增加,由于维层次存储树中维的层次关系,查询是按层次索引查询,所以维层次存储树查询时间增加不是很明显,而二维数组的查询时间明显增加。

5　结论

文章针对采用数组来存储维信息的不足,结合港口立方体



(a)数据量不变的条件下,查询时间随层数的变化情况



(b)层数相同的情况下,查询时间随数据量的变化情况

图 5　维层次存储树与二维数组的查询性能比较

维具有层次的特点,采用了维层次存储树来存储维信息,对每个维进行索引,减少了查询时间,提高了查询效率。在今后的工作中,将进一步研究立方体中的度量值在立方体中的聚集存储问题,从而提高 MOLAP 的查询和更新效率。

参考文献:

[1] 施思明,杨红霞.数据仓库在港口企业的应用设想[J].中国港口,2004(10):39-40.

[2] 邢攸达,王宇,潘明霞.港口生产数据仓库设计与实现[J].计算机辅助工程,2007,16(4):84-88.

[3] 沈志荣.船舶代理业的数据仓库及 OLAP 建模技术[J].微计算机信息,2007,23(2-3):169-171.

[4] 段成华,刘娟.海运数据仓库的设计与实现[J].计算机辅助工程,2003(3):40-44.

[5] 王珊.数据仓库技术与联机分析处理[M].北京:科学出版社,1998.

[6] Chan C Y,Ioannidis Y E.Bitmap index design and evaluation[C]//Haas L M,Tiwary A.Proc of the ACM SIGMOD International Conference on Management of Data.New York:ACM Press,1998:355-366.

[7] Wu M C,Buchmann A P:Encoded bitmap indexing for data warehouses[C]//Proc of the 4th International Conference on Data Engineering (ICDE).Los Alamitos:IEEE Computer Society Press,1998:220-230.

[8] Wu M C.Query optimization for selections using bitmaps[C]//Delis A,Faloutsos C,Ghandeharizadeh S.Proc of the ACM SIGMOD International Conference on Management of Data.New York:ACM Press,1999:227-238.

[9] 陈志伟,李伟生.MOLAP 中层次维的存储[J].计算机工程,2002,28(7):116-117.

[10] 胡孔法,董逸生,陈崚.数据仓库中一种基于维层次编码的位图索引方法[J].东南大学学报:自然科学版,2005,35(2):171-177.