

# 一种 OLAP 应用系统的设计和实现

周 龙, 郑 诚

(安徽大学 计算机学院, 安徽 合肥 230039)

**摘 要:**通过对数据仓库和 OLAP 概念及体系结构的分析,描述了一种 OLAP 应用系统的设计方案,并介绍了它的具体实现方法。基于数据仓库的查询,一般都是及时特定查询,要在严格的响应时间内执行复杂的查询,遍历百万上亿的记录,同时进行可能很复杂的搜索、连接和汇总的操作。查询的数据吞吐量和响应时间是判断数据仓库性能的重点。CUBE 的计算是 OLAP 及时查询的基础,提高查询的速度需要对 OLAP 进行预先的计算。文中系统比较了一些计算立方体的算法,并运用到具体的系统当中。

**关键词:**数据仓库;OLAP;多维数据立方体

**中图分类号:**TP311.13

**文献标识码:**A

**文章编号:**1673-629X(2006)06-0101-03

## Design and Implementation of One OLAP System

ZHOU Long, ZHENG Cheng

(Coll. of Computer, Anhui Univ., Hefei 230039, China)

**Abstract:** By analyzing the concept and system frame of data warehouse and OLAP, describe one method for design and implementation of OLAP. In general, queries based on data warehouse are required to be with fast response time. The query data throughput and response time are very important to data warehouse's capability. The computation for cube is the basic of ad hoc query. In order to improve the relevant queries, must pre-compute the cube for OLAP. In this paper, compared some algorithms of computing for cube, and put it into the idiographic system.

**Key words:** data warehouse; OLAP; cube

### 0 引 言

数据仓库(Data Warehouse)是一种语义上一致的数据存储空间,它充当支持数据模型的物理实现,并存放企业战略决策所需信息<sup>[1]</sup>。数据仓库的精髓在于针对联机分析处理(OLAP)提出了一种综合的解决方案,但与很多技术不同的是,它主要是一种概念,在此概念指导下完成系统的构造。为了提高数据的面向分析的性能,发展起了 OLAP 技术,可以提供给用户从多角度,方便地查询、分析数据,给决策者提供支持。

文中在分析数据仓库和 OLAP 概念、特征、体系结构的基础上,提出一套实现联机分析处理(OLAP)的完整解决方案,并探讨了对数据立方体的聚集计算的一些算法。

## 1 数据仓库的概念和结构体系

### 1.1 数据仓库的概念

作为数据管理,传统的联机事务处理系统(On-Line Transaction Processing, 简称 OLTP)主要用于事务处理,但

随着企业信息系统产生大量数据,人们对 OLTP 的分析处理能力不满意。如何提高分析处理能力成为企业决策管理者所面临的一个重要课题。

数据仓库技术就是在这样的背景下发展起来的。数据仓库概念创始人 W. H. Inmon 在《Building the Data Warehouse》一书中对数据仓库的定义是:“数据仓库是支持管理决策过程的、面向主题的、集成的、随时间变化的持久的数据集”<sup>[2]</sup>。构建数据仓库的过程是根据预先设计好的逻辑模式,从分布在企业内部各处的 OLTP 数据库中提取数据并对其经过必要的变换,最终形成全企业统一模式数据的过程。

### 1.2 三层数据仓库结构

(1)底层:仓库数据库服务器。

该层处于数据仓库的底层,实际上它是一个关系型的数据库系统,使用网间连接程序从数据库中提取数据。

(2)中间层:OLAP 服务器。

(3)顶层:前端工具,包括查询和报表工具、分析工具和数据挖掘工具。

## 2 OLAP

### 2.1 概 念

根据 OLAP Council 公布的白皮书(OCWP)对 OLAP

收稿日期:2005-10-02

作者简介:周 龙(1981-),男,安徽黄山人,硕士研究生,研究方向为数据挖掘与知识发现;郑 诚,副教授,硕导,研究方向为数据挖掘与知识发现。

的定义<sup>[3]</sup>: OLAP 是可以分析专家、管理人员获得对信息数据有深刻认识的工具。它可以访问各种可能的数据信息,并且保证访问过程的迅捷性、访问数据的一致性和访问手段的交互性。

OLAP 中变换过的数据以数值的形式存储在数据仓库中,因此,从这个角度看,OLAP 和数据仓库是互补的。数据管理的工作由数据仓库完成,而 OLAP 负责数据的分析处理工作。

OLAP 对数据库数据提供一个多维的视图。无论这些数据在物理上是怎么存储的,从用户的角度看,数据都被看成是多维立方体。数据立方体的每个单元中存放了一个数值(measure)。

## 2.2 多维数据模型上的 OLAP 操作

数据仓库需要简明的面向主题的模式,便于联机数据分析<sup>[1]</sup>。而在数据仓库模型中用的最多的是多维数据模型,常见的有星型模型、雪花模型和事实星座模型等。在多维的数据模型中,数据是以多维的形式组织的,便于用户从多角度去观察数据。基于多维模型之上,可以做一些 OLAP 典型的操作,像上卷(roll-up)、下卷(drill-down)、切片和切块、转轴等。

## 2.3 实现多维数据模型的方法

根据数据存储的格式不同,可以将 OLAP 分为 3 种<sup>[4]</sup>:ROLAP(Relational Online Analytical Processing)关系型 OLAP;MOLAP(Multi Dimensional OLAP)多维 OLAP 和 HOLAP(Hybrid OLAP)混合型 OLAP。

ROLAP 以二维关系表为核心表达多维概念,通过将多维结构划分为两类表:维表和事实表,可以使关系型结构能较好地适应多维数据的表示和存储。MOLAP 方案是以多维方式来组织数据,以多维方式来存储数据。

# 3 应用实例

## 3.1 系统结构

文中的 OLAP 系统由 3 层结构构成(如图 1 所示):

- (1)数据仓库服务层;
- (2)Brain Insight Analysis Services(BIAS)服务器;
- (3)查询客户端。

Brain Insight Analysis Services(BIAS)服务器的功能在这个模型中是最重要的,因为它负责实时的查询。

### 3.1.1 数据仓库服务层

该系统底层是数据仓库服务层,是建立在一个或多个 DBMS 源之上。由于 OLAP 面向的是分析和管理人员,所以数据要求是综合数据,以便决策者从更高层次,更全面的角度来分析数据。因此数据仓库成为 OLAP 分析的主要数据源。它可以通过索引技术,像位图索引、连接索引等,支持快速的聚集。

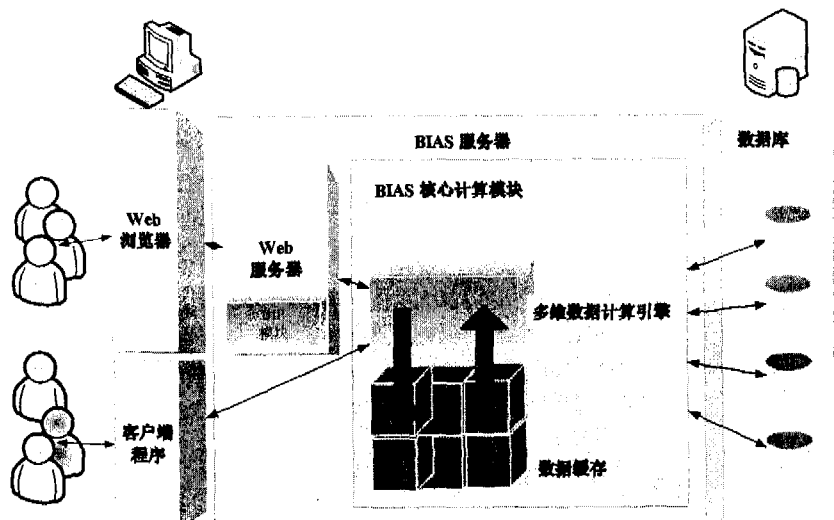


图 1 BIAS 服务器系统结构图

### 3.1.2 Brain Insight Analysis Services(BIAS)服务器

BIAS 服务器是用于联机分析处理(OLAP)的中层服务器,它包括一个计算分析服务器,可以用于对要分析的多维数据模型的构造和计算;提供针对多维数据集的基于浏览器的快速查询和分析访问,并且将数据仓库或者数据库中的数据组织成包含预先计算聚合数据的多维数据集,以便为复杂的分析、查询提供快速的解决方案。

### 3.1.3 查询客户端

支持 DSS 查询,允许用户从数据立方体中浏览数据的缓存(像切片和切块)。当一个查询被提交给客户端,如果符合数据立方体中的条件,查询就会被传到底层,返回一个数据集。

## 3.2 立方体的存储和查询问题

由于数据立方体的存储是预先计算好的,所以它查询起来很快。假如数据立方体引擎可以回答所有的查询,那么 OLAP 系统可以支持所有的实时查询。但是这里存在一个矛盾的地方,由于存储和维护数据立方体需要磁盘空间和 CPU 的计算。因此在存储空间和响应时间之间需要做一个权衡,在整个立方体集中,选择性地物化一个合适的子集。

## 3.3 立方体的计算

立方体计算的算法有以下几种:

1)基于矩阵数组的算法(Gray)。K 维立方体存储在 k 维数组中<sup>[5]</sup>。优点:简单、查询方便。缺点:受内存大小限制。针对该缺点,文献[5]中给出了 Array-Chunking 优化算法,在该算法中数据立方体可以分块导入内存,数据立方体大小可以不受内存大小的限制。

2)PIPESORT 算法<sup>[6]</sup>。给出了一个完整的数据立方体计算优化算法。它通过遍历节点网络,根据选择节点的最小父节点来计算该节点的原则,画出节点网络的计算路径,然后按顺序计算每条路径上的节点。该算法的缺点是:算法复杂度受维度个数影响太大。

3)OVERLAP 算法<sup>[7]</sup>。该算法通过交叠计算数据立

方体,通过部分排序减少排序步骤的方法来最小化磁盘读取次数。

4) Partitioned - Cube & Memory - Cube 算法<sup>[8]</sup>。其实是两个针对不同情况两个算法。如果内存不满足数据立方体大小,则调用 Partitioned - Cube 算法;若满足则用 Memory - Cube 算法。Partitioned - Cube 算法使用的是分散存储、“各个击破”的思路。它将具有  $k$  个维度、 $T$  个元组的数据立方体“化整为零”到  $n+1$  个子立方体中。(前  $n$  个子立方体有  $T/n$  个元组,  $k$  个维度;最后一个子立方体不超过  $T$  个元组,  $k-1$  个维度。)

5) BUC 算法 (Bottom - up Computation)<sup>[9]</sup>。算法非常适合解决一些“冰山”查询 (遍历的数据量很大,但是输出数据量很小,例如:查询语句中有 HAVING COUNT(\*) >= X)。它的特点是计算节点树的过程是从下往上的。

在 BIAS 系统的实现中,对立方体计算的算法采用的是基于矩阵的算法,由 Gray 提出。其优点是:简单、查询方便。缺点:受内存大小限制。该算法将所有的数据元组以  $k$  维数组的形式存入内存。但是一般来说即使元组  $R$  满足内存大小,  $k$  维数组也可能不满足 (因为数据立方体是稀疏的,数组空间可能远大于  $R$  元组的大小)。因此该算法一般适用大内存。

文献[5]中给出了一个优化算法。这里先给出一个简化的算法版本 (Naive)。首先举例说明构建方案:若以一个三维的立方体为例, ABC 是一个  $16 * 16 * 16$  的多维矩阵,可以将其分为  $4 * 4 * 4$  的 64 个矩阵块。如图 2 所示<sup>[4]</sup>。

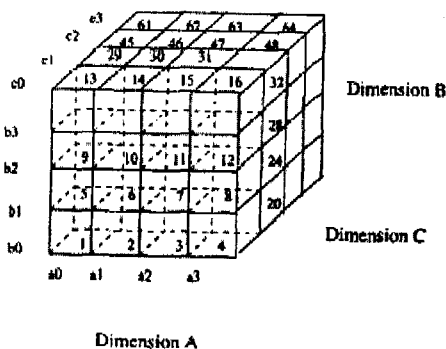


图 2 三维立方体

可以将存在磁盘上的小数据立方体,分批读入内存进行计算。

生成一个最小构建树 (原理和 PIPESORT 类似)。将立方体分块后,内存大小的限制就没有了,但是增加了磁盘读取的次数。显然,在一次汇总过程中会重复调用一些小数据块,因此,数据节点的维度顺序决定调用的次数。

关于节点继续顺序的选择优化,文中给出 MMST 算法 (Minimum Memory Spanning Tree) 的一种 Multi - pass 算法:

- (1) Create the MMST  $T$  for a dimension order 0
- (2) Add  $T$  to the Tobecomputed list
- (3) For each tree  $T'$  in Tobecomputed list

- (3.1) Create the working subtree  $W$  and incomplete subtree  $Is$
- (3.2) Allocate memory to the subtrees
- (3.3) Scan the array chunk of the root of  $T'$  in the order 0
  - (3.3.1) aggregate each chunk to the groupbys in  $W$
  - (3.3.2) generate intermediate results for  $Is$
  - (3.3.3) write complete chunks of  $W$  to disk
  - (3.3.4) write intermediate results to the partitions of  $Is$
- (3.4) For each  $I$ 
  - (3.4.1) generate the chunks from the partitions of  $I$
  - (3.4.2) write the completed chunks of  $I$  to disk
  - (3.4.3) Add  $I$  to Tobecomputed

该算法中,使用多路径来完成数据立方体的计算。将构建树  $T$  分成一个工作子树和一个未完成子树集。将工作子树的每个节点在内存中分配存储空间,对每个未完成子树,分配其相当于一个子立方体块的空间,将计算的中间结果存入磁盘。

## 4 结 论

文中在分析数据仓库和 OLAP 概念、特征和体系结构的基础上,设计了一个 OLAP 系统——BIAS,分析了其内部的结构,探讨了数据立方体的聚合运算的一些算法。该系统已在多个行业运用,像银行总帐分析、个人征信查询分析系统、企业财务费用分析等系统中。数据仓库实现的重点和难点是要解决数据立方体的存储和计算两个问题。对于提高查询、分析能力,还需要对算法做优化,以适合更广泛的需求。

## 参考文献:

- [1] Han J W, Kamber M. 数据挖掘概念与技术[M]. 北京:机械工业出版社,2001.
- [2] Inmon W H. Building the Data Warehouse (Second Edition) [M]. 北京:机械工业出版社,2000.
- [3] OLAP Council. OLAP AND OLAP Sever Definitions [EB/OL]. <http://www.olapcouncil.org/research/glossary.htm>, 1997.
- [4] Corey M. Oracle8i Data Warehousing [M]. [s.l.]: The McGraw - Hill Inc, 2001.
- [5] Zhao Yihong, Deshpande P M, Naughton J F. An Array - Based Algorithm for Simultaneous Multidimensional Aggregates [EB/OL]. <http://citeseer.ist.psu.edu/zhao97arraybased.html>, 1997.
- [6] Sarawagi S, Agrawal R, Gupta A. On computing the data cube [M]. [s.l.]: IBM Almaden Research Center, 1997.

(下转第 106 页)

截取,最终形成标准码流。

### 2.3 整体 EBCOT 并行处理

在 JPEG200 系统编码过程中,考虑到离散小波变化和 EBCOT 之间的处理匹配问题,提出了一种有效的解决方法,就是针对小波变化后的三个子带系数分别采用三套 EBCOT 系统来对应处理,这样就更进一步提高了处理速度。其系统结构设计如图 4 所示。

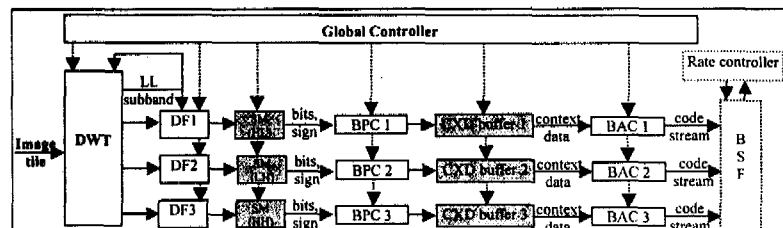


图 4 整体 EBCOT 结构设计图

整个结构由三套 EBCOT 组成,参见文献[2],而每个 EBCOT 又是由 5 部分组成:

- 1)数据转换(DF),它是把 DWT 变换后的子带分成小的编码块,其系数转换成符号-量值型数据,且进行量化处理的单元;
- 2)子带存储器(SM),用来存储编码块的符号-量值数据;
- 3)位平面编码(BPC),是进行三通道编码生成上下文-量值对的部件;
- 4)上下文量值缓存器(CXD Buffer);
- 5)算数编码器 MQ(BAC)。

在压缩过程中,首先把原始图像划分成拼接块(tiles),再进行预处理,然后进行小波变换,每一级的小波变换都生成三个高频子带(HL, LH, HH)和一个低频子带(LL),LL 子带继续进行下一级的变换,其他三个高频子带数据被输出进行熵编码,每个子带被送入一套熵编码设备中实现并行编码,当 DWT 变换到最后一级时,LL 子带被最后送到第一套设备中进行编码。每个编码块编码完成后生成的码流送入位流转换器中(BSF),为了实现所需图像要达到的分辨率和质量,通过率控制器(rate controller)控制 BSF 来实现最终的码流。

### 3 三种并行方法的讨论

在这三种并行结构中,因为位平面并行处理和位平面数相关,且成正比,所以此方法的硬件开销最大,处理速度

也最快,通道并行处理方法的硬件最少,处理速度相对而言也比较慢。在整体 JPEG2000 编码过程中,不仅要考虑编码处理速度,而且还得考虑方案软硬件结合的可行性与实用性,对于位平面并行处理,可以采用软件进行前期和后期处理来弥补软件比例,从而达到最佳处理方案。从不同应用场合来说,通道并行处理和整体 EBCOT 并行处理适合于嵌入式便携设备中,而位平面并行处理则比较适合

于固定压缩设备。在实际的图像处理中,可以根据现实条件的要求而选择不同的并行处理方法,也可以同时应用几种并行处理方式的组合,以达到更高的处理速度。

### 4 结论

随着 JPEG2000 标准的逐渐推广和应用,提高 EBCOT 处理速度的方法也在不断的涌现,文中就其三种并行结构做了深入的描述,因为 EBCOT 并行处理大大提高了 JPEG2000 图像处理速度,降低了 EBCOT 瓶颈效应,所以灵活地运用几种并行方法及其组合会得到更广泛的应用前景。

#### 参考文献:

- [1] JPEG2000 Final Committee Draft (FCD). JPEG2000 Committee Drafts[EB/OL]. <http://www.jpeg.org/CDs15444.htm>, 2005-07.
- [2] Taubman D. High performance scalable image compression with EBCOT[J]. IEEE Trans Image Processing, 2000(9): 1158-1170.
- [3] Lian Chung-Jr, Chen Kuan-Fu, Chen Hong-Hui, et al. Analysis and Architecture Design of Block-Coding Engine for EBCOT in JPEG 2000[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2003, 13(3): 219-230.
- [4] Chiang Jen-Shiun, Lin Yu-Sen, Hsieh Chang-Yo. Efficient Pass-Parallel Architecture for EBCOT in JPEG2000[A]. IEEE International Symposium on Circuits and Systems[C]. [s.l.]: [s.n.], 2002. 773-776.
- [5] 刘凯, 吴成柯, 李云松, 等. 比特平面并行的 EBCOT 编码及其 VLSI 结构[J]. 计算机学报, 2004, 27(7): 928-935.
- [6] Andra K, Chakrabarti C, Acharya T. A High-Performance JPEG2000 Architecture[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2003, 13(3): 209-218.

(上接第 103 页)

- [7] Agarwal S, Agrawal R, Deshpande P M. On the Computation of multidimensional aggregates[A]. Proceedings of the 22nd VLDB Conference[C]. Mumbai, Bombay, India: [s.n.], 1996.
- [8] Ross K, Srivastava D. Fast computation of sparse data cubes

[EB/OL]. <http://citeseer.ist.psu.edu/ross97fast.html>, 1997.

- [9] Beyer K. Bottom-up computation of sparse and Iceberg CUBEs[EB/OL]. <http://citeseer.ist.psu.edu/beyer99bottomup.html>, 1999.