动态更新实物化视图以提高OLAP查询效率

武 彤[1] 赵 雪[2] 赵 洵[3]

(贵州大学计算机科学与信息学院 贵阳 550025)[1] (中国传媒大学信息工程学院 北京 100024)[2]

(中国传媒大学计算机学院 北京 100024)[3]

摘 要 在数据仓库系统中,OLAP查询一般都涉及多表连接和分组聚集两部分操作,提高这些查询的性能成为提高OLAP响应速度的关键。利用实物化视图,可以准确地计算并保存表连接或聚集等耗时较多的操作的结果。研究基于查询频率的实物化视图的更新算法,可以使实物化视图得到最大效率的使用,明显地缩短查询的响应时间,从而提高OLAP的查询效率。

关键词 数据仓库,实物化视图,OLAP,多维数据查询,查询优化

中图法分类号 TP39 **文献标识码** A

Dynamically Update Materialized View in Order to Improve the Efficiency of OLAP Queries

WU Tong[1] ZHAO Xue[2] ZHAO Xun[3]

(School of Computer Science and Information, Guizhou University, Guiyang 550025, China)[1]

(School of Information Engineering, Communication University of China, Beijing 100024, China)[2]

(School of Computer Science and Technology, Communication University of China, Beijing 100024, China)[3]

Abstract In the data warehouse system, OLAP query usually involves two operations which are multi-table connection and grouping congregation. To improve the performance of queries has become the key to speed up OLAP responding. Some time-assuming operations such as table connection and congregation can be calculated and preserved by utilizing materialized view. This investigation is based on the updated arithmetic of materialized view relating query frequency. It makes materialized view is fully utilized and query time can is highly shortened so that the query efficiency is able to be highly elevated.

Keywords Data warehouse, Materialized view, OLAP, Multi-dimension data query, Query optimizing

20世纪80年代中期,"数据仓库"这个名词首次出现在号称"数据仓库之父"W. H. Inmon的"Building Data Warehouse"一书中。在该书中,W. H. Inmon把数据仓库定义为:数据仓库是面向主题的、集成的、稳定的、随时间变化的数据集合,用于支持管理决策过程[1]。

20世纪60年代,关系数据库之父E. F. Codd提出了关系模型,促进了联机事务处理(OLTP)的发展。1993年,E. F. Codd提出了OLAP概念,认为OLTP已不能满足终端用户对数据库查询分析的需要,SQL对大型数据库进行的简单查询也不能满足终端用户分析的要求。用户的决策分析需要对关系数据库进行大量计算才能得到结果,而查询的结果并不能满足决策者提出的需求。因此,提出了多维数据库和多维分析的概念,即OLAP。OLAP(联机分析处理)是针对某个特定的主题进行联机数据访问、处理和分析,通过直观的方式从多个维度、多种数据的综合程度将系统的运营情况展现给使用者[2]。

任何一种技术或应用都是在某种需求的驱动下产生和发展起来的,数据仓库也不例外。由于数据仓库技术在全局应用及复杂分析方面的强大功能,近年来基于数据仓库的决策支持系统得到了越来越广泛的应用。在这一类系统中,数据仓库主要存放OLAP分析和数据挖掘所需要的数据,通过OLAP分析可以使决策人员得到迫切需要的、准确及时的决策信息。因此,OLAP性能的优化对数据仓库系统性能的影响起着至关重要的作用。

在OLAP分析中,查询经常会涉及到多表连接,因此提高多表连接的性能就成为了提高OLAP查询处理的关键性问题。对于ROLAP来说,OLAP查询一般都涉及多表连接和分组聚集两部分操作,提高这些查询的性能成为提高OLAP响应速度的关键。本文通过研究基于查询频率的实物化视图更新算法来实现实物化视图的高效使用,以提高OLAP查询的响应速度。

1 多表连接查询分析

1.1 基于关系数据库的OLAP(ROLAP)

同专用的多维数据库相比,关系数据库尽管表达多维概念不大自然,但在现有关系数据库广泛使用的情况下也不失为一种实用可行的方案。那么,ROLAP是如何用关系数据库的二维表来表达多维概念?ROLAP将多维数据库中的多

本文受贵州省2010年工业攻关项目(黔科合GY字【2010】3061)资助。

武 彤(1964-),女,硕士,副教授,主要研究方向为数据仓库技术、OLAP、数据挖掘,E-mail:wtxx@citiz.net。

维结构划分为两类表：一类是维表，用来记录维度信息；另一类是事实表，用来存储维度交叉点处的度量信息及各个维度的码值。这样，多维数据立方体各个坐标轴上的刻度以及立方体各个交点的取值都被记录下来，因而数据立方体的全部信息就都被记录下来。通过将事实表和维表进行连接，就可以得到 OLAP 系统常用的“星型结构”。

星型结构是在 OLAP 中常用的逻辑结构，如上分析，它主要是用来在关系型数据库中模拟数据的多维模型。将关系型数据库中的数据存入星型模型的事实表和维表中，使“扁平”、没有层次的数据存储为“立体”数据。通过多表连接、分组聚集计算等操作来实现 OLAP 的查询。

1.2 多表连接查询

对星型结构进行多表连接查询时，系统对查询实施方法的不同，将导致查询性能的巨大差异。通常有 4 种基本方法。

(1)逐个将维表和事实表进行连接运算。其计算过程是先将第一个维表和事实表进行连接运算，然后将运算的结果再和第二个维表进行连接，以此类推，直到最后一个维表被连接。这种方法简单，但是每次连接一个维表，都需要对事实表进行一遍扫描，如果事实表很大，这个过程将非常消耗时间。

(2)由数据仓库系统来识别维表和事实表。先将所有需要的维表进行连接运算，然后再用所有维表连接的结果与事实表进行连接。这种方法的优点是事实表只需进行一次扫描，由于维表的长度通常远远小于事实表，因此能达到较好的优化效果。但是，如果维表的数目很多，并且其中某些维表很长，则这种方法的优势就会减弱。

(3)索引连接。这种方法中不是对表的记录直接进行连接运算，而是使用索引来进行连接运算，然后根据索引连接的结果来查找对应表的记录。在索引连接中，由于索引所占的空间通常小于表的大小，因此索引连接对存储空间的要求大大降低了。在索引连接中定义的索引通常为 B-Tree 索引。如果我们只访问某些基数很小的维表，而一次需要获取大量的记录数，则索引连接的效果会很差。

(4)位图索引。这种方法为事实表中每一个外来关键字(外键)建立一个位图索引。类似地在各个维表的关键字上也建立位图索引，然后通过将对应的位图索引进行“AND”运算得到需要的结果。这种方法对于访问基数很小的维表效果很好，但是，对于一次只从结果中选择很少的记录的查询，其效果较差[3]。

由上面分析可知，对星型结构进行查询的方法有其适用范围，很难说哪一种是最优的，需要根据实际情况选择合适的查询策略。下面提出一种实物化视图的方法来提高多表连接查询的效率。

2 实物化视图算法分析

数据仓库中的实物化视图主要用于预先计算并保存表连接或聚集等耗时较多的操作结果，这样，在执行查询时，就可以避免进行这些耗时的操作，直接对实物化视图进行操作即可，从而快速地得到查询结果。目前大多数数据仓库系统都是建立在关系数据库系统之上的，因此利用实物化视图实现 OLAP 查询优化成为一种切实可行的方案。但是 OLAP 系统中的查询是随机的，而建立实物化视图是需要知道系统中的查询集合，现有 OLAP 系统中实物化视图的选择方案都是假设所有查询在综合数据上是均匀分布的，或者可以让用户提供查询的分布概率。但在实际应用中，均匀分布的假设往往不能成立，由用户方来提供查询的分布概率也是不切实际的。此外，由于不能明确知道在系统中要进行的查询，因此实物化视图中常常不得不存储某些综合级别上的所有数据，这虽然对提高查询效率是有用的，但实物化视图中存储的数据并不是总能得到频繁的应用，甚至有些查询应用并不存在，因此实物化视图中存储的数据并不能确切地反应真实的查询需求，在实际操作中对于查询效率的提高没有达到预期的效果。由此可知使用实物化视图来提高查询效率常常要解决一个实物化视图动态更新的问题。

2.1 基于查询频率的实物化视图更新算法分析

为了能建立有效的实物化视图，在系统的使用中，需要收集查询，然后更新实物化视图。根据收集的查询信息，得到用户的查询频率，以此来动态更新实物化视图。

首先要提出一个概念。数据链表：在 OLAP 中，可以将每一个数据表表示为一个数据链。以销售数据星型模型为例，来说明数据链的构成。销售主体的事实表为 FactTable (DateID, RegionID, ProductNo, Output, Income)，另外 3 个维表为 Date, Region, Product。现在以维表 Region 的字段来说明：Region(RegionID, City, Province, Area)，含有下划线的字段为维层次的字段。于是该维表的数据表链如图 1 所示。

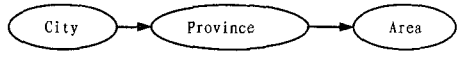

图 1 Region 维表数据表链

其中，数据节点间的偏序关系 $\bar{R}$ 为：给定结点 a 和 b，$a\ \bar{R}\ b$ 当且仅当利用 a 即可计算出 b。这意味着 a 的各维上的级别均低于或等于 b 的相应维上的级别。据此，证明一个定理：数据链为分配格。证明过程如下。

证明：设 a,b 是数据链 L 的元素，因为链中任意两个元素均含有偏序关系 $\bar{R}$，即有 $a\ \bar{R}\ b$ 或 $b\ \bar{R}\ a$。

如果 $a\ \bar{R}\ b$，则 a,b 的最大下界是 a，最小上界是 b，如果 $b\ \bar{R}\ a$，则 a,b 的最大下界是 b，最小上界是 a，故链一定是格。下面证明分配律成立即可。

对 L 中任意元素 a,b,c 分下面两种情况讨论：1)$b\ \bar{R}\ a$ 或 $c\ \bar{R}\ a$；2)$a\ \bar{R}\ b$ 且 $a\ \bar{R}\ c$。

如果是第一种情况，则 $a\cup(b\cap c)=a=(a\cup b)\cap(a\cup c)$；如果是第二种情况，则 $a\cup(b\cap c)=b\cap c=(a\cup b)\cap(a\cup c)$。

无论哪种情况分配律均成立，故 L 是分配格。

因此，将维表构造为不同的数据链，可以构造为一个格，只需要将星型模型里所有字段的集合作为这个格的上界。

在定义了数据链表的基础上，提出数据查询的具体方法：对系统的查询 Q 表示为 m 个二元组组成的 m 元组：$\{(D_1,A_1),(D_2,A_2),\cdots,(D_m,A_m)\}$，其中 m 为数据模型的维数，D_i 表示维 i 的某一个维层次，A_i 表示在 D_i 上的相同维字段选择。若没有对 D_i 的范围进行限制，可以将 (D_i,A_i) 简记为 D_i。查询 Q 所涉及的数据结点称为 D_q。若 $Q=\{(D_1,A_1),(D_2,A_2),\cdots,(D_m,A_m)\}$，则 $D_q=\{D_1,D_2,\cdots,D_m\}$。查询间的偏序关系 $\bar{R}$ 定义为：对于查询 p 和 q，若 q 的结果可以用来响应 p，则有 $p\ \bar{R}\ q$。显然，若 $p\ \bar{R}\ q$，不但要求 $D_p\ \bar{R}\ D_q$，而且用来生成 p 的数据均应包含在 q 中。

例如，查看产品 C_1,C_2 在地 B_1 销售情况的查询 $q_1=\{(ProductId,\{C_1,C_2\}),(Area,\{B_1\})\}$。若地区 B_1 中共 3 家商店 T_1,T_2 和 T_3。比较这 3 家商店中产品 C_1,C_2 的销售量

的查询 $q_2=\{(ProductId,\{C_1,C_2\}),(StoreId,\{T_1,T_2,T_3\})\}$。我们有 $q_1 \bar{R} q_2$。

由于实物化视图的目的是保存数据供查询访问，因此视图的表示与查询是一致的。我们常将查询和视图混用。

系统所收集的查询包括查询的集合 Q 和每个查询 q 的发生频率 $f(q)$。

完成了以上的基础定义，现在正式给出基于查询频率更新的算法。

在给定存储空间 Memory 的限制下，基于查询频率更新问题就是选择那些查询频率更高的查询，用以更新物化视图，来最大限度地提高总体查询的效率。

为了合理利用空间，我们采用贪心算法来求解。在系统所记录的查询 Q 中选出将被实物化的视图，选择标准是单位空间的频率，即 $f(q)/|q|$。其中 $f(q)$ 是已知的，而 $|q|=|D_q|\times\prod_1^m|q.A_i|/\prod_1^m|q.D_i|$，其中 m 为数据集的维数，$|q.D_i|$ 表示级别 D_i 中不同元素的个数，$|q.A_i|$ 表示 A_i 的基数。

算法描述(F-US)：

Step1　输入查询集合 Q；

Step2　判断内存空间是否大于 0，否则退出循环；

Step3　选择最大的 $f(q)/|q|$，将 q 里的数据集合赋给 V 集合；

Step4　如果内存空间 $M\geqslant|V|$，将 q 从 Q 集合中去除，否则 $M=0$。

返回 Step2。

该算法的时间复杂度为 $O(_n\log 2n)$，通过附加的存储器，可以大大提高查询效率，如对 f(q)/|q|建立索引等[4]。

2.2　算法分析的结果评测

为了进一步证明该算法的优点，我们进行了一系列的实验，性能改善达到了预期的效果。实验中，为了更好地与该领域中已有的研究成果进行比较，我们采用 BPUS 算法[5]作为视图选择准则 M，并采用相似的实验方法，即对于多维数据，我们也设计 4 个维 D_1，D_2，D_3 和 D_4，它们分别有 4，3，4，3 个级别，各个级别的成员个数如表 1 所列。

表 1　维层次和维成员个数

维层次	维数			
	D_1	D_2	D_3	D_4
3	5		5	
2	125	5	25	5
1	250	125	125	50
0	500	250	250	250

多维数据的基视图 DB 共有 500k 行，并且假设多维数据的分布是均匀的，估算出数据集尺寸为 15×106 行，存储空间取为数据集尺寸的 20%。我们设计了 90%访问多维数据格中 10%的结点。对于任意被访问的级别，在访问它的所有查询中，有 80%的查询访问其中 20%的成员。实验中共使用了 10000 个查询。这些查询被分为 10 组，其中每一组按所述比例在结点和维成员间重新随机分布。然后取这些查询的平均代价作为查询的代价。

我们比较了在不同的存储空间内对实物化视图查询的代价。存储空间取数据集总尺寸的 0%～20%。比较的结果如图 2 所示。

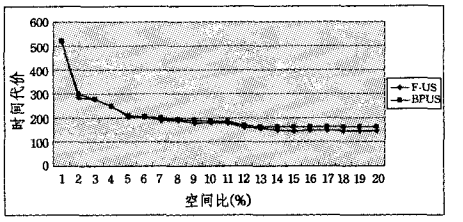

图 2　F-US 与 BPUS 算法代价比较

从图 2 中看出，随着查询数据占存储空间的增加，两种算法的代价都在下降，但是基于查询频率的算法总体优于 BPUS。基于查询频率的物化视图更新算法能够使系统获得更精确的实物化视图，节省了用户的时间，较好地满足了 OLAP 系统的需求。

结束语　通过以上分析论述，可以总结 F-US 算法的几个优点：

(1)根据用户对系统的查询，使得物化视图能得到最大效率的使用。通过最大频率的查询，可以解决用户查询均匀分布的不合理假设。

(2)可以准确地计算并保存表连接或聚集等耗时较多的操作的结果，这样，在执行查询时，就可以避免进行这些耗时的操作，从而快速地得到查询结果。

(3)目前的数据仓库容量已经增长到兆兆字节，精确的实物化视图意味着终端用户现在需要读取的行数很少，所以基于查询频率的实物化视图的更新算法，可以明显地缩短查询响应时间。

本文的研究实验基于特定的项目所涉及的数据，因此难免存在一定的局限性，对于算法的推广应用还有待进一步的研究，希望专家和同行提出宝贵的意见。

参 考 文 献

[1] Inmon W H. Building the Data Warehouse[M]. 王志海，译. 北京：机械工业出版社，2007：20-21

[2] 王丽珍，等. 数据仓库与数据挖掘原理及应用[M]. 北京：科学出版社，2005：109-110

[3] 林宇，等. 数据仓库原理与实践[M]. 北京：人民邮电出版社，2003：204-206

[4] 谭红星，周龙骧. 多维数据实视图的动态选择[J]. 软件学报，2002，13(6)：1090-1096

[5] Harinarayan V，Rajaraman A，Ullman J D. Implementing Data Cubes Efficiently[C]// Jagadish，SIGMOD' 96，Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data. Montreal：ACM Press，1996：205-216

(上接第 311 页)

[7] Fox S，Karnawat K，Mydland M，et al. Evaluating implicit measures to improve Web search[J]. ACM Transactions on Information Systems，2005，23(2)：147-168

[8] Srinivasan N，Paolucci M，Sycara K. An efficient algorithm for OWL-S based semantic search in UDDI[J]. lecture Notes in ComputerScience，2005，3387：96-100

[9] Mokhtar S B，Preuveneers D，Georgantas N，et al. EASY：Efficient SemAntic Service Discovery in Pervasive Computing Environments with QoS and Context Support[J]. Journal of Systems and Software，2007，81(5)：785-808

[10] 岳昆，王晓玲，周傲英. Web 服务核心支撑技术研究综述[J]. 软件学报，2004，15(3)：428-442