用多用户多窗口方法处理多维视图动态选择

薛永生[1] 林子雨[1] 段江娇[1] 吕晓华[1,2] 张 伟[1]
[1](厦门大学计算机科学系 厦门 361005)
[2](浙江理工大学信息电子学院 杭州 310018)
(cainiu @263. net)

摘 要 多维视图动态维护是数据仓库领域当前研究的一个热点. 随着数据仓库的普及,将有越来越多的各种类型的用户使用 OLAP 工具满足各自特定的分析需求. 现有的各种视图选择方法没有考虑不同类型用户的特点,从而存在一些缺陷. 提出多用户多窗口方法,充分考虑用户的差异,利用单个用户在查询时的相对有规律性,为用户划分级别,并为每个用户设置相应级别的用户视图窗口,从而合理地利用了系统有限的资源,提高了查询响应速度,也保证了特殊用户对查询性能的特殊需求;给出了相关的定义和 MUMW 算法,并阐述了多用户多窗口方法的优点.

关键词 多用户多窗口;MUMW;实视图;OLAP;多维数据;数据仓库

中图法分类号 TP311. 13

Dynamic Selection of Materialized Views of Multi-Dimensional Data with a Multi-Users and Multi-Windows Method

XUE Yong-Sheng[1], LIN Zi-Yu[1], DUAN Jiang-Jiao[1], LÜ Xiao-Hua[1,2], and ZHANG Wei[1]
[1] (*Department of Computer Science, Xiamen University, Xiamen* 361005)
[2] (*College of Information and Electron, Zhejiang University of Sciences, Hangzhou* 310018)

Abstract Dynamic selection of materialized views of multi-dimensional data is one of the most researched aspects in the field of data warehouse; With the increasing use of data warehouse, there will be accordingly more and more different kinds of users making use of OLAP tools to complete their analytical work; The existing methods being used to select views do not take into account the characteristic of various kinds of users, and therefore have some defects; A multi-users and multi-windows method is presented here, which considers the disparity among various users and makes use of the rule of the users' queries. In the method, all users are divided into three groups with different grade and each user is accordingly endowed with user view window of certain grade. Such method leads to the reasonable use of the limited space resource and also speeds up the response of query, which stipulates satisfying some special needs of certain user; Some related conceptions and MUMW algorithm are also put forward here, and at the same time, the advantages of this method are described.

Key words multi-users and multi-windows; MUMW; materialized view; OLAP(online analytical processing); multi-dimensional data; data warehouse

1 引 言

数据仓库和联机分析处理(online analytical processing,OLAP)正日益广泛地应用于决策支持系统(decision support systems,DSS)中,为用户从大量的信息中搜寻有价值的内容,提供了强大而有力的工具. 系统每天都要接收大量的来自用户的查询,

收稿日期:2004-07-15
基金项目:福建省自然科学基金项目(A0310008);福建省高新技术研究开放计划重点项目(2003H043)

同时也在处理查询的过程中产生大量的中间结果,为了提高 OLAP 查询效率从而更好地满足用户需求,有必要在系统中保存这些查询过程中产生的中间结果. 但是,存储这些视图需要额外的空间,并且在一些公司的数据仓库里,这些视图呈指数级增长,很快就会达到几 GB 的空间,有的甚至达到几 TB. 如果不采用有效的视图选择策略,系统资源就会很快被耗尽. 由此,实视图和视图的选择,也受到越来越多的研究与重视.

在本文提出的多用户多窗口方法中,系统接收来自多个用户的查询,并在全局视图窗口(所有视图可以利用的存储空间)内建立一个用户视图窗口(所有用户可以利用的存储空间),再在用户视图窗口内为不同级别的用户建立不同级别的用户视图窗口,由于每个用户的查询集合有自己特点的规律,系统会自动收集每个用户和每个查询的相关信息,给出统计结果供视图选择算法使用. 本文结构安排是第 2 节介绍视图选择相关的研究工作;第 3 节详细论述多用户多窗口方法,其中包括相关定义和 MUMW 算法以及其他算法;第 4 节给出了性能分析;第 5 节为结束语.

2 相关工作

对视图选择的大量研究产生了丰硕了研究成果. 文献[1]给出了数据立方体格(lattice of data cube)模型,提出了以单位空间的效益为视图选择标准、时间复杂度为 $O(kn^2)$ 的贪婪算法 BPUS 来减少查询的响应时间. 在此基础上,文献[2]讨论了带有 B-树索引的实视图选择问题,文献[3]提出了以实视图的尺寸为选择标准、时间复杂度为 $O(n\log_2 n)$ 的选择算法 PBS. 这些方案都是基于这样的假设,即查询的分布情况是已知的,或者假设查询在聚集数据上是均匀分布的,或者假设用户能够给出查询的分布概率. 但实际上,均匀分布的假设常常不能成立,用户也很难给出查询的分布概率,由此,文献[4,5]提出了由系统收集查询的相关信息,其中文献[4]又提出用单位空间的频率为标准、时间复杂度为 $O(n\log_2 n)$ 的 FPUS 的选择算法;文献[5]则提出了最近实化父视图(nearest materialized parent view, NMPV)的概念,并用 B + 树创建对聚集视图的索引,还给出一个以相对效益为标准的 PVMA (progressive view materialization algorithm)算法,该算法的时间复杂度为 $O(V^2 + I + D + U)$,其中,V 表示视图的数量;I 表示各种类型的查询中属于"插入查询"的数量;D 表示"删除查询"的数量;U 表示"选择查询"的数量. 另外,文献[6～8]还提出在不考虑资源限制和性能保证情况下的筛选侯选视图的方法.

3 多用户多窗口方法

3.1 不同类型的聚集视图

由于实视图的目的就是保存数据供查询访问,所以视图的表示与查询是一致的,从而在以下的论述中,我们将查询与视图混用. 首先介绍几个概念.

定义 1. 多维数据上的查询[4]. 多维数据集合 MD 上的查询 q 是对 MD 某一数据结点的切片或切块,可以将 q 表示成由 d 个二元组组成的 d 元组:$\{(l_1, R_1), (l_2, R_2), \cdots, (l_d, R_d)\}$,其中,$d$ 为 MD 的维数;l_i 表示维 d_i 的某一级别;R_i 表示在 l_i 上的选择范围,若没有对 l_i 的范围进行限制,可以将 R_i 记为 ALL. 当 R_i 被表示为$\langle r_{i1}, r_{i2}\rangle$时,表示在$\langle r_{i1}, r_{i2}\rangle$范围上的一个切块,而当 R_i 被表示为$\{r_{i1}, r_{i2}, \cdots, r_{in}\}$时,则表示在$\langle r_{i1}, r_{in}\rangle$范围上的 n 个切片. 若$\langle r_{i1}, r_{i2}\rangle$是 $dom(l_i)$,即取值范围是整个维,则维 i 的二元组可以不出现在查询中.

定义 2. 完整聚集视图 CV. 在定义 1 中,当对于所有的 i 都有 $R_i = ALL$ 或者 $R_i = \langle r_{i1}, r_{i2}\rangle$是 $dom(l_i)$时,则此时的查询被定义为完整聚集视图. 这里用 CV 表示,可以理解为英文的 complete-aggregated view 的缩写.

定义 3. 部分聚集视图 PV. 在定义 1 中,当至少存在一个 i 使得 $l_i \neq ALL$ 时,则此时的查询被定义为部分聚集视图. 这里用 PV 表示,可以理解为英文的 part-aggregated view 的缩写.

我们以一个销售应用的数据立方体为例子对完整聚集视图和部分聚集视图的概念进行说明. 该数据立方体有 3 个维:产品维、地区维和时间维. 产品维包含 3 个产品:$P1$, $P2$ 和 $P3$;地区维包含 3 个地区:东北、西北和华北;时间维包含 3 个时间:1994, 1995 和 1996. 下面我们定义几个查询:

q_1 = {(产品, ALL), (地区, ALL), (时间, {1994, 1996})},表示"各个产品在各个地区内分别在 1994 年和 1996 年的销售量".

q_2 = {(产品, ALL), (地区, ALL), (时间, {1994, 1995, 1996})},表示"各个产品在各个地区内分别在 1994 年, 1995 年和 1996 年的销售量".

q_3 = {(产品, ALL), (地区, ALL), (时间, 〈1994, 1996〉)},表示"各个产品在各个地区内在 1994 年到 1996 年这 3 年内的总销售量".

在 q_2 中,由于{1994, 1995, 1996}是时间维的全部取值,所以可以写成 ALL,因而 q_2 等价于{(产品, ALL), (地区, ALL), (时间, ALL)}.

在 q_3 中,由于〈1994, 1996〉是 dom(时间),即时间维的整个取值范围,所以可以省略时间维二元组,即 q_3 等价于{(产品, ALL), (地区, ALL)}.

很显然,根据上面定义,q_1 属于部分聚集视图,q_2 和 q_3 属于完整聚集视图. 同时也容易看出,多维数据格文献[3]中的数据结点属于完整聚集视图.

3.2　已有的视图选择方法的缺陷

从现有的视图选择算法分析,有些是基于格的选择方法,如文献[1]中的 BPUS 算法和文献[5]中的 PVMA 算法,有些则是基于查询的选择方法,如文献[4]中 FPUS 算法. 由于基于格的选择方法过于"粗放"[4],所以文献[4]提出的基于查询的选择算法 FPUS 在性能上要优于 BPUS. 但即便如此,基于查询的选择方法还是存在缺陷.

定义 4. 全局视图窗口 GVW. 全局视图窗口是指系统提供的一个有尺寸上限限制的空间,用来存储那些根据特定算法筛选得到的所有的聚集视图. 这里用 GVW 表示,可以理解为英文的 global view window 的缩写.

定义 5. 受访视图集合. 受访视图集合用 $R(q)$ 来表示,是指一个查询中所要用到的所有视图的集合.

定义 6. 用户受访视图集合. 用户 U 受访视图集合被定义为 $R(U) = \bigcup_{i=1}^{n} R(q_i)$,其中 $q_1, q_2, \cdots, q_n$ 表示由用户 U 发起的 n 个查询,$R(q_i)$ 表示查询 q_i 的受访视图集合.

通常情况下,系统将接收来自多个用户的 OLAP 查询请求,以前的视图选择方法对所有用户不进行区分,并不考虑各个用户进行查询时的不同特点,系统一般只定义一个全局视图窗口(global view window, GVW),所有用户的查询过程中要用到的视图一起参与竞争,由特定算法决定哪些视图可以进入全局视图窗口,以及哪些应该从全局视图窗口中删除. 这种做法实际上是抹杀了每个用户查询的个性特点,在实际查询过程中,由于某些原因,有可能出现以下问题:

(1) 某些用户受访视图集合中的视图不合理地反复进出全局视图窗口. 比如说,有 n 个用户在使用系统,而且全局视图窗口已经没有剩余空间. 为了方便问题的解释,我们假设其他用户的查询始终延续原来的规律继续进行,而其中有两个用户 U_1 和 U_2 在不同时期具有不同频度的查询请求,如果在 T_1 时段内,用户 U_1 发起的查询请求比 U_2 频繁,则系统将把 $R(U_1)$ 中更多的视图放入全局视图窗口,同时从全局视图窗口中删除部分或全部那些属于 $R(U_2)$ 的视图;而在 $T2$ 时段内,用户 U_2 发起的查询请求比 U_1 频繁,则系统将把 $R(U_2)$ 中更多的视图放入全局视图窗口,同时从全局视图窗口中删除部分或全部那些属于 $R(U_1)$ 的视图;如果这种情况反复出现,则会出现视图反复进出全局视图窗口的情况,这和内存设计中出现的页面抖动有些相似. 多个用户进行查询时,这种视图的调入调出全局视图窗口的情况将更频繁,由于每个选择操作涉及的视图集合比较大,所以系统开销很大.

(2) 某些用户受访视图集合中的视图很少有机会或者根本没有机会进入全局视图窗口. 对于那些以大尺寸视图优先入选为标准的算法,则会导致某些用户出现这样的情况,即虽然他们频繁发起查询请求,但是 $R(U)$ 中的视图尺寸相对于其他用户受访视图集合中的视图尺寸来讲普遍偏小,所以不能被系统入选;而对于那些采用标准空间的频率作为入选标准的算法,也会出现某些用户虽然频繁发起查询请求,但是受访视图的尺寸相对于其他用户的受访视图的尺寸来讲普遍偏大(则标准空间的频率普遍偏小),从而不能被调入全局视图窗口. 其他种类的算法也不能避免这类问题,因为它们都把具有不同查询特点的用户放在一起进行处理.

(3) 由于没有区分不同用户的重要性,系统给予所有的用户以同等的权限使用系统资源,会造成系统资源的浪费,同时也难以保证某些特殊用户对响应速度的要求. 比如在某公司或机构内,用户可以大体分为最高层决策者、中间管理层和一般工作人员. 显然,他们的重要程度是不相同的,我们应该优先考虑最高决策者用户的查询请求,并为他们分配更多的系统资源,而一般工作人员的查询请求则可以放在最后考虑的位置,这样做更具有合理性.

3.3　多用户多窗口方法的描述

多用户多窗口方法则不同于以前的这些视图选择方法,它充分考虑了每个用户的类型和在发起一系列查询时所具有的特殊规律,并定义了公共视图窗口和用户视图窗口.

定义 7. 公共视图窗口 *PVW*. 公共视图窗口是指系统提供的一个有尺寸上限限制的空间,它分为完整聚集视图区(complete-aggregated view district, CVD)和部分聚集视图区(part-aggregated view district, PVD),分别用来存储那些被选中的实化的完整聚集视图和部分视图集合. 这里用 *PVW* 表示,可以理解为英文的 public view window 的缩写.

定义 8. 用户视图窗口 *UVW*. 用户视图窗口是指系统分配给用户的一个有尺寸上限限制的空间,用来存储那些根据特定算法筛选得到的用户的部分聚集视图. 这里用 *UVW* 表示,可以理解为英文的 user view window 的缩写.

定义 9. 等级用户视图窗口 $UVW\text{-}u$. 等级用户视图窗口 $UVW\text{-}u$ 是指在 *UVW* 中分配给不同等级用户一个有尺寸上限限制的空间,用来存储那些根据特定算法筛选得到的该用户 u 的部分聚集视图. 这里用 $UVW\text{-}u$ 表示,并且 $UVW\text{-}u_1$ 表示 1 级用户视图窗口, $UVW\text{-}u_2$ 表示 2 级用户视图窗口, $UVW\text{-}u_3$ 表示 3 级用户视图窗口.

该方法在全局视图窗口(*GVW*)中设置一个公共视图窗口(*PVW*),公共视图窗口用来存储所有被实化的完整聚集视图(*CV*)和某些满足特定条件的部分聚集视图(*PV*),完整聚集视图侯选集(*CVS*)中的视图被选中后,可以进入公共视图窗口的完整聚集视图区(*CVD*),而部分聚集视图侯选集(*PVS*)中的视图被选中后,则进入公共视图窗口的部分聚集视图区(*PVD*),本文中, *PVS* 中的某个 *PV* 被两个或两个以上用户访问是该 *PV* 进入 *PVD* 的必要条件. 另外还为每个用户都开辟了一个单独的用户视图窗口,用户视图窗口又被划分为 3 个级别:1 级、2 级和 3 级. 相应地,用户也被分为 3 级:1 级、2 级和 3 级,并且 $UVW\text{-}u_1$ 表示 1 级用户视图窗口, $UVW\text{-}u_2$ 表示 2 级用户视图窗口, $UVW\text{-}u_3$ 表示 3 级用户视图窗口. 各种视图窗口的关系如图 1 所示:

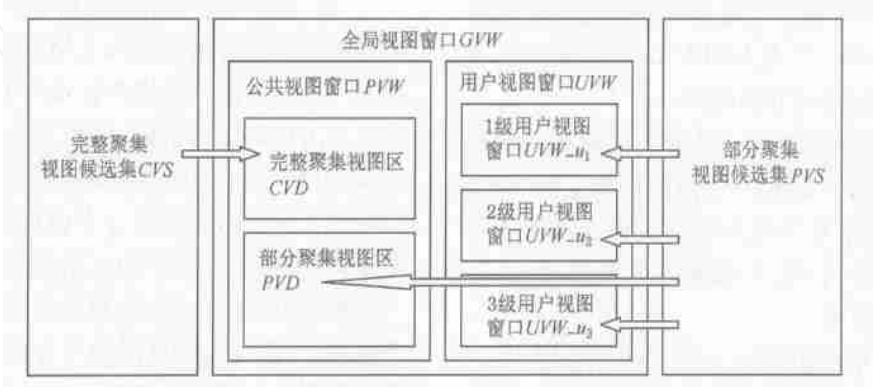

图 1 各种视图窗口之间的关系

在多用户多窗口方法中,1 级用户具有最高优先权,享受 1 级窗口待遇,该类用户通常是某公司或机构的最高层决策者,对于这类用户的查询请求,系统必须以最快的速度给予响应;2 级用户具有第 2 高优先权,享受 2 级窗口待遇,该类用户通常是某公司或机构的中间管理层,对于这类用户的查询请求,系统只要给予适当的响应速度;3 级用户的优先权最小,享受 3 级窗口待遇,该类用户通常是某公司或机构的一般工作人员,对于这类用户的查询请求,系统可以考虑或不考虑其响应速度. 在 3 个级别的用户视图窗口中,从 1 级窗口到 3 级窗口,在数量上呈递增(即 1 级窗口的数量最少,2 级窗口的数量稍多,3 级窗口的数量最多 1 级窗口的数量最少),在尺寸上呈递减. 3 个级别窗口的大小都可以改变,它们的尺寸上限值均由系统当时的可利用空间来决定,而尺寸下限值的确定则不同,其中 1 级和 2 级窗口的尺寸下限值不能小于系统设置的下限值,而 3 级窗口的下限值可以是 0,也就是系统可以不用为 3 级窗口分配空间. 另外,1 级窗口和 2 级窗口的数量应该结合公司或机构的实际情况以及系统资源来确定,而 3 级窗口的数量则是不确定的,完全取决于系统的可用资源.

3.4 多用户多窗口的空间分配

在多用户多窗口方法中,系统对用户进行分级,各级用户对应不同级别的 $UVW\text{-}u$,从而可以保证把有限的资源优先分配给具有较高级别的用户. 由此,各个窗口的空间分配就显得至关重要,分配的结果必须首先保证高级用户的需要,同时在可利用空间允许的情况下使部分或全部低级用户都能拥有自己的 $UVW\text{-}u$. 在具体阐释空间分配算法之前,需要为系统设置几个参数,包括 1 级用户和 2 级用户的数量 n_1, n_2, *PVW* 在 *GVW* 中所占的比例 r_{PG}(是

指空间大小的比例,以下皆如此),CVD 在 PVW 中所占的比例 r_{CP}($r_{PG}+r_{CP}=1$),PVD 在 PVW 中所占的比例 r_{PP},单个 1 级用户 u_1 的 $UVW\text{-}u_1$ 在 GVW 中所占的比例 r_{UP1},单个 2 级用户 u_2 的 $UVW\text{-}u_2$ 在 GVW 中所占的比例 r_{UP2},单个 3 级用户 u_3 的 $UVW\text{-}u_3$ 在 GVW 中所占的比例 r_{UP3}. 在所有这些参数中,n_1 和 n_2 是结合公司的实际情况来确定的,而其他参数则是需要结合实际应用经过多次实验后确定的最优值. 这些参数值确定后,我们就可以确定以下参数的值:

CVD 窗口的固定大小:$S_{CVD}=S_{GVW}\times r_{PG}\times r_{CP}$;

PVD 窗口的下限值:

$S_{PVD\_DOWN}=S_{GVW}\times r_{PG}\times r_{PP}$;

单个 $UVW\text{-}u_1$ 窗口的固定大小:

$S_{UVW\_u_1}=S_{GVW}\times r_{UP1}$;

单个 $UVW\text{-}u_2$ 窗口的固定大小:

$S_{UVW\_u_2}=S_{GVW}\times r_{UP2}$;

单个 $UVW\text{-}u_3$ 窗口的上限值:

$S_{UVW\_u_3}=S_{GVW}\times r_{UP3}$;

各个窗口的空间大小关系如式(1):

$$S_{GVW}=S_{CVD}+S_{PVD}+\sum S_{UVW\_u}, \quad (1)$$

式(1)中 $\sum S_{UVW\_u}$ 表示所有 $UVW\text{-}u$ 的尺寸之和;并且有式(2)成立:

$$\sum S_{UVW\_u}=\sum S_{UVW\_u_1}+\sum S_{UVW\_u_2}+\sum S_{UVW\_u_3}. \quad (2)$$

另外,系统在确定参数时,必须还应满足这样一种情况,即当所有 1 级和 2 级用户都在使用系统并且没有 3 级用户使用系统时,必须满足式(3):

$$S_{GVW}\geq S_{CVD}+S_{PVD\_DOWN}+\sum S_{UVW\_u_1}+\sum S_{UVW\_u_2}. \quad (3)$$

在各个窗口中,CVD 的空间只取决于 $S_{GVW}\times r_{PG}\times r_{CP}$,不会随着当前系统用户数量而改变;$PVD$ 的大小则是以 S_{PVD_DOWN} 为下限,空间大小可以扩展,可以利用 GVW 中未被分配给 UVW(这种情况在当前用户很少时会出现)的空间,当正在使用系统的用户增加时,系统就要把这部分被 PVD 占用的空间收回,分配给某些 $UVW\text{-}u$,这时 PVD 的空间就会收拢,但不会小于下限值. 由此,我们可以知道,在没有任何用户使用系统时,GVW 的空间被全部分配给 CVD 和 PVD,其中,$S_{CVD}=S_{GVW}\times r_{PG}\times r_{CP}$,$S_{PVD}=S_{GVW}-S_{CVD}$. 当系统中新增加一个用户 u 时,空间分配算法如下:

算法 1. 多用户多窗口的空间分配(MUMP).

输入:PVD 窗口大小的下限值 S_{PVD_DOWN},当前 PVD 窗口大小 S_{PVD},当前 UVW 中的可利用空间 S_{UVW},当前新增加用户 u,当前 3 级窗口数量 n.

输出:S_{UVW_u}.

```
case u
 u1, u2: S := S_UVW;
  /*收集到 UVW 中的可利用空间*/
  while S < S_UVW_u and n > 0 do
  /*撤消部分或全部 3 级窗口从而获得空间,
    撤消一个 3 级窗口并释放空间*/
   S := S + S_UVW_u3;
  end while
   if S < S_UVW_u then
    /*从撤消 3 级窗口获得的空间不能满
      足需求,则继续从 PVD 中索取空间
      */
     S0 := 0;/* S0 表示需要从 PVD 中获
      取的空间 */
      while S0 < S_UVW_u - S do
      删除 PVD 中尺寸最小的视图 v;
      S0 := S0 + |v|;
      S_PVD := S_PVD - |v|;
      end while
   end if
   开辟窗口 UVW-u 并分配给该窗口空间
     S_UVW_u;
 u3: if S_PVD ≥ (S_PVD_DOWN + S_UVW_u) then
     开辟窗口 UVW-u 并分配给该窗口空间
       S_UVW_u;
     end if
end case
```

3.5　各个窗口内的视图选择

在多用户多窗口方法中,在选择视图方面没有一个固定的标准,可以采用单位空间的效益为标准[1],或采用尺寸大小为标准[3],或以单位空间的频率为标准[4],也可以把相对效益作为选择视图的标准[5];正是因为这一点,我们定义了视图选择准则 M,在多用户多窗口算法的具体实现过程中,可以用具体的标准来替换准则 M.

定义 10. 视图选择准则 M. 视图选择准则 M 是用来从视图侯选集中选择视图的一种标准,它是由系统自己定义的,每个系统可以根据实际情况来选择自己的选择标准,通常的标准有视图尺寸、单位空间效益、单位空间频率、相对效益等等.

定义 11. 视图选择准则 M^*. 视图选择准则 M^* 被定义为与准则 M 的情况相反. 比如,当两个视图竞争时,若 M 以视图尺寸大者为获胜者,则 M^* 以视图尺寸小者为获胜者.

定义 12. 视图选择准则 N. 视图选择准则 N 是用来从 PVS 中选择视图到 PVD 中的一种标准,按照此标准,当两个视图竞争时,用户访问数(即有多少个用户访问过该视图)比较大的视图获胜,当两者的用户访问数相等时,总计访问数(即所有用户对该视图的所有访问的累计次数)比较大的视图获胜.

定义 13. 视图选择准则 N^*. 视图选择准则 N^* 被定义为与准则 N 的情况相反. 按照此标准,当两个视图竞争时,用户访问数(即有多少个用户访问过该视图)比较小的视图获胜,当两者的用户访问数相等时,总计访问数(即所有用户对该视图的所有访问的累计次数)比较小的视图获胜.

算法 2. 用多用户多窗口法选择实视图.

输入: CVD, PVD 和用户集合 U 以及各个用户的 UVW_u 的可利用空间 S_{CVD}, S_{PVD}, S_{UVW_u}; CVS 以及 PVS.

输出: CVD, PVD 和各个 UVW_u 内所选择的实视图集合 V_{CVD}, V_{PVD}, V_{UVW_u}.

$V_{CVD} := \varnothing$; $V_{PVD} := \varnothing$;
for each $u \in U$ do
  $V_{UVW\_u} = \varnothing$;
end for
while $S_{CVD} > 0$ do / *从 CVS 中选择视图到 CVD 中 */
  $v :=$ CVS 中符合视图选择准则 M 的视图;
  if $S_{CVD} > |v|$ then
    $V_{CVD} := V_{CVD} \cup \{v\}$; $CVS := CVS - \{v\}$;
    $S_{CVD} := S_{CVD} - |v|$;
  else $S_{CVD} := 0$;
  end if
end while
for each $u \in U$ do
/ *从 PVS 中选择视图到各个用户的 UVW 中 */
while $S_{UVW\_u} > 0$ do
  $v :=$ PVS 中符合视图选择准则 M 并且只被用户 u 访问过的视图;
  if $S_{UVW\_u} > |v|$ then
    $V_{UVW\_u} := V_{UVW\_u} \cup \{v\}$;
    $PVS := PVS - \{v\}$;
    $S_{UVW\_u} := S_{UVW\_u} - |v|$;
  else $S_{UVW\_u} := 0$;
  end if
 end while
end for
while $S_{PVD} > 0$ do
 / *从 PVS 中选择视图到 PVD 中 */
 $v :=$ PVS 中同时符合视图选择准则 M 和 N;
 if $S_{PVD} > |v|$ then
  $V_{PVD} := V_{PVD} \cup \{v\}$;
  $PVS := PVS - \{v\}$;
  $S_{PVD} := S_{PVD} - |v|$;
 else $S_{PVD} := 0$;
 end if
end while

3.6 各个窗口内视图的动态调整

对于实视图集合的即时调整而言,我们不仅需要考虑用户发起的查询 q,还要考虑在响应查询 q 的过程中系统产生的中间结果的集合 Q_0,设 $Q_1 = q \cup Q_0$,为了避免重复计算和视图不合理地进出窗口,我们要 Q_1 中那些属于集合 $P = (V_{CVD} \cup V_{PVD} \cup V_{UVW\_u})$ 的 q 从 Q_1 中删除,从而得到集合 Q.

算法 3. 单个窗口内的实视图集合的即时调整 (*WINDOW-ADAPT*).

输入:窗口可用空间 S;窗口内当前的实视图集合 V;查询集合 Q;视图选择准则 L.

输出:调整后的窗口内实视图集合 V.

$S_0 := S$;
 / *收集到的窗口中的可利用的空间 */
$V_{\text{delete}} := \varnothing$;
 / *将要从窗口中被删除的实视图集合 */
$V_{\text{add}} := \varnothing$;
 / *将要被加入到窗口中的视图集合 */
$search :=$ True; / *继续搜索标记 */
while $Q \neq \varnothing$ and $search$ do
 $q :=$ Q 中利用准则 L 竞争获胜者;

```
  while S0 <| q| and search do
    v ≔窗口中利用准则 L* 竞争获胜的视图;
    if 利用准则 M 竞争时 v 失败 then
      S0 ≔ S0 +| v| ;
      Vdelete ≔ Vdelete1 ∪{ v} ; V ≔ V - { v} ;
    else
      search ≔false;
    end if
  end while
  if S0 ≥| q| then
    Q ≔ Q - { q} ;
    Vadd ≔ Vadd + { q} ;
  end if
end while
if Vadd ≠∅then
  实化 Vadd中的视图;
  S ≔ S - | Vadd| ;
  / *| Vadd| 表示 Vadd中所有视图尺寸之和 */
end if
return V
```

算法 4. 各个窗口内实视图集合的即时调整.

输入: *CVD*, *PVD* 和各个 *UVW*-*u* 的可利用空间 S_{CVD}, S_{PVD}, S_{UVW_u}; 当前的实视图集合 V_{CVD}, V_{PVD}, V_{UVW_u}; 查询集合 Q.

输出: 调整后的实视图集合 V_{CVD}, V_{PVD}, V_{UVW_u}.

```
QCV ≔ Q 中的完整聚集视图的集合;
QPV ≔ Q 中的部分聚集视图的集合;
WINDOW-ADAPT(SCVD, VCVD, QCV, M) ;
  / *调整 CVD 中的视图 */
QUVW ≔ QPV中用户访问数为 1 的视图的集合;
QPVD ≔ QPV 中用户访问数大于 1 的视图的集合;
for each q ∈QPV1 do
  WINDOW-ADAPT(SUVW_u, VUVW_u,
  QUVW, M) ;
    / *调整相关的 UVW 中的视图,其中 u 为访问 q 的用户 */
end for
for each q ∈QPV1 do
  WINDOW-ADAPT(SPVD, VPVD, QPVD, N) ;
    / *调整 PVD 中的视图 */
end for
return VCVD, VPVD, VUVW_u.
```

4　性能分析

根据以上的论述,我们可以总结出多用户多窗口方法有如下优点:

(1) 每个用户都有自己的视图窗口,保证了查询的响应速度. 这一点对快速响应高级用户的查询速度至关重要. 一般来说,单个用户的查询在相当大的程度上是很有规律的,为用户视图开辟专用窗口存储用户视图,可以大大减少从基表或综合度较低的数据运算生成所需数据的时间.

(2) 对用户进行分级保证了有限的系统资源优先分配给高级用户,符合人们在实际运作过程中的需要. 设置用户的优先级早已被许多系统的设计所采用,因为这是保证满足特殊用户的特殊需求的有效方法.

(3) 与全局视图窗口相比较,用户视图窗口的尺寸和包含的视图数量都要少得多,从而在单个窗口(公共视图窗口或某个用户视图窗口)内选择算法效率更高. 当发生一个查询后即时调整窗口中的视图时, Q_{PVD}中的视图只要参与 V_{PVD}的调整, Q_{CVD}中的视图只要参与 V_{CVD}的调整, Q_{UVW}中的视图 q 只要参与访问 q 的用户 u 的 V_{UVW_u}的调整. 而我们已经知道, *CVD* 中只是存储了完整聚集视图, *PVD* 中只是存储了部分聚集视图,每个 *UVW*-*u* 中存放的只是本用户使用的视图,就单个窗口而言,这些窗口中的视图数量要比传统的单窗口设计方法(所有视图存储在一个全局窗口中)中存储的视图要少得多,更新速度比较快;而且,如果采用并行处理技术,这些窗口的调整是可以同时进行的,速度更快.

(4) 极大减少了上面所论述的视图不合理地反复进出视图窗口的情况. 我们已经在前面的论述中讨论过这种情形,其主要原因就是,当把多个用户的视图都放在一个窗口内来统一处理时,窗口内存放的视图的使用情况就没有规律可言,即使单个用户的查询可能很有规律. 而为每个用户开辟一个 *UVW*-*u* 后,在单个 *UVW*-*u* 内,视图的使用情况就会相对比较有规律,这样就不会造成视图不合理进出窗口的情形.

为了进一步证明多用户多窗口方法的优点,我们进行了一系列的实验,取得了预期的良好性能改善. 实验中,为了更好地与该领域中已有的研究成果进行比较,我们采用文献[4]中提出的 FPUS 算法作为视图选择准则 *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_0, D_1, D_2 和 D_3,它们分别有 4,3,4,3 个级别,各个级别的成员个数如表 1 所示:

表 1 多维数据各级别的成员个数

维	级别			
	D_0	D_1	D_2	D_3
3	1		1	
2	25	1	5	1
1	50	25	25	10
0	100	50	50	50

多维数据的基视图 DB 共有 5×10^5 行,并且假设多维数据的分布是均匀的,估算出 CUBE 的尺寸为 15×10^6 行,存储空间取为 CUBE 总尺寸的 20 %. 我们设计了 10 个用户同时按照各自的规律连续对系统发起查询请求,每个用户对系统进行的查询个数为 1000 个. 为了说明用户查询规律的重要性,我们分成两种情况:第 1 种情况为 10 个用户都按照相似的规律进行查询;第 2 种情况是 10 个用户分别按照各自截然不同的规律进行查询. 对于每种情况,把得到的各个用户的实验数据进行平均就得到我们的最终结果.

表 2 采用 MUMW 方法前后的查询相应时间的比值

前 n 个查询	采用 MUMW 方法之后的响应时间/采用 MUMW 方法之前的响应时间	
	第 1 种情况	第 2 种情况
1	1.02	1.02
5	0.99	0.93
50	0.98	0.90
100	0.97	0.81
300	0.95	0.77
500	0.94	0.74
1000	0.94	0.73

从表 2 的实验数据中,我们可以得出这样的结论:当用户的查询都遵循同一种规律时,则采用多用户多窗口方法对查询响应时间的改善并不明显,甚至在开始时性能还有些“恶化”,这是由于采用该方法的额外开销导致的. 但是,当用户的查询都有各自不同的规律时,则该方法对性能的改善是比较明显的.

另外,我们这里还需要对两个方面的问题进行简单的讨论.

(1) 用户在第 1 次使用时如何获得高的响应速度. 在实验过程中,我们发现,用户在第 1 次使用时,由于系统中并没有统计到用户的查询规律,用户惟一可以利用的只是那些处于 PVW 中的实化视图,而往往这些视图经常不是用户所需要的,这就会导致系统对用户第 1 次查询的响应时间不尽人意. 经过分析,我们采用一个小巧的办法来处理这个问题,那就是用户使用系统的优先权由两部分组成,一部分是他的身份等级,另一部分就是他是否是第 1 次使用系统,我们让第 1 次使用系统的用户具有相对较高的优先权,这样,用户在第 1 次使用系统时,既使没有享受到已有实化视图的好处,但是由于系统优先对他的请求进行处理,所以也可以获得较高的响应速度.

(2) 采用 MUMW 方法对全局查询带来的碎片太多的问题该如何处理. 这个问题也是在内存分配设计中经常遇到的问题. 当系统使用一段时间以后,肯定会产生一些“零散”的小空间,这些空间由于尺寸太小而不能被系统使用,为了解决这个问题,当使用系统时,我们需要运行一个碎片整理程序,对系统在运行过程中出现的碎片进行定期清理,并合并成较大的空间供系统重新使用. 由于这个过程实现存在一定难度,我们设计的碎片整理程序还有待进一步完善,因此没有作为成果在本文发表,这方面将是我们今后研究的重点.

5 结束语

多维视图的动态选择在以后的一段时间里仍将是数据仓库研究领域的热点,本文提出的多用户多窗口方法不同于传统的选择方法的地方就在于,该方法充分考虑了将来在公司或机构中将有更多的普通工作人员和高层管理者一样使用 OLAP 工具,也分析了不同用户本身的各自不同的查询特点,照顾了不同用户不同的查询需求,保证了对特殊用户的查询响应速度. 但该方法只是定义了一个实施的框架,方法里面有一些参数需要在实际运行的过程中根据具体情况给予确定,从而取得最佳效果. 本文只是抛砖引玉,真心希望将来有更多研究人员在这一方面做出更加深入的研究.

参 考 文 献

1 V Harinarayan, A Rajaraman, J D Ullman. Implementing data cubes efficiently. In: H V Jagadish, I S Mumick, eds. Proc of the

1996 ACM SIGMOD Int'l Conf on Management of Data (SIGMOD '96). New York: ACM Press, 1996. 205～216

2 H Gupta, V Harinarayan, A Rajaraman, *et al*. Index selection for OLAP. In: A Gray, Larson, Per-Ake, eds. Proc of the 13th Int'l Conf on Data Engineering (ICDE '97). Los Alamitos, CA: IEEE Computer Society Press, 1997. 208～219

3 A Shukla, P Deshpande, J F Naughton. Materialized view selection for multidimensional datasets. In: A Gupta, O Shmueli, J Widom, eds. Proc of the 24th Int'l Conf on Very Large Data Bases (VLDB '98). San Francisco, CA: Morgan Kaufmann, 1998. 488～499

4 **谭红星, 周龙骧. 多维数据实视图的动态选择. 软件学报**, 2002, 13(6): 1090～1096
(Tan Hongxing, Zhou Longxiang. Dynamic selection of materialized views of multi-dimensional data. Journal of Software (in Chinese), 2002, 13(6): 1090～1096)

5 U Hidetoshi, R Kanda, J T Toby. A progressive view materialization algorithm. In: Proc of the 2nd ACM Int'l Workshop on Data Warehousing and OLAP. New York: ACM Press, 1999. 36～41

6 E Baralis, S Paraboschi, E Teniente. Materialized view selection in a multidimensional database. In: M Jarke, M J Carey, K R Dittrich, *et al*, eds. Proc of the 23rd Int'l Conf on Very Large Data Bases (VLDB '97). San Francisco, CA: Morgan Kaufmann, 1997. 156～165

7 D Theodorators, T Sellis. Data warehouse configuration. In: Proc of the 23rd VLDB Conf. San Francisco, CA: Morgan Kaufmann, 1997. 126～135

8 K Ross, D Srivastava, S Sudarshan. Materialized view maintenance and integrity constraint checking: Trading space for time. In: Proc of the 1996 ACM SIGMOD Int'l Conf on Management of Data. New York: ACM Press, 1996. 447～458

薛永生　男,1946 年生,教授,主要研究方向为数据库理论与应用、分布式数据库、数据仓库、数据挖掘等.

林子雨　男,1980 年生,硕士研究生,主要研究方向为数据库理论与应用、分布式数据库、数据仓库、数据挖掘等.

段江娇　女,1972 年生,硕士,讲师,主要研究方向为据库理论与应用、分布式数据库、数据仓库、数据挖掘、网络技术等.

吕晓华　男,1970 年生,硕士,讲师,主要研究方向为分布式数据库、数据仓库与数据挖掘、网络技术.

张　伟　男, 1978 年生, 硕士研究生, 主要研究方向为数据仓库、数据挖掘、分布式数据库等.