

新型物化视图选择算法

李 明, 刘青宝, 陆昌辉

(国防科学技术大学 信息系统与管理学院, 长沙 410073)

(rush_lee@163.com)

摘 要:针对现有物化视图选择算法无法很好解决 OLAP 随机查询的问题,提出了一种新型的两阶段物化视图选择算法(2-PMVS),将传统的静态选择算法与动态选择算法相结合,使其可以动态矫正用户随机查询与预估查询之间的偏差。经实验证明,该算法切实有效。

关键词:数据仓库;物化视图;两阶段物化视图选择算法

中图分类号:TP311 **文献标志码:**A

New materialized view selection algorithm

LI Ming, LIU Qing-bao, LU Chang-hui

(College of Information System and Management, National University of Defense Technology, Changsha Hunan 410073, China)

Abstract: To address the problem that the current materialized view selection methods couldn't solve random OLAP query efficiently, this paper proposed a 2-Phases Materialized View Selection (2-PMVS) algorithm which combined static materialized view selection algorithm with dynamic materialized view selection algorithm. It could dynamically bridge the gap between random queries and default queries. The experiments validate the effectiveness of the algorithm.

Key words: Data Warehouse (DW); materialized view; 2-Phases Materialized View Selection (2-PMVS)

0 引言

数据仓库是一个面向主题的、集成的、非易失的且随时间变化的数据集,用来支持管理人员的决策^[1]。数据仓库作为一个巨大的数据容器,包含了大量的历史性事务数据,因而在辅助决策和大规模数据库等领域有着广泛的应用。然而随着数据仓库中的数据量不断膨胀,如何保证快速响应复杂的多维查询成为研究热点。在现有方法中,物化视图技术被广泛地采纳并取得了明显的效果。简单来说,物化视图就是将查询中可能遇到的聚合值进行预计算,并将结果存储在磁盘中,以起到加速查询的目的。在现实应用中,往往因为存储空间和维护代价而不可能将数据仓库中的所有视图进行物化,因而需要从中选取获益最大的一组聚合视图来进行物化,同时满足存储和维护条件的约束^[2],这样就将问题转化为一个典型的 NP 完全问题,此问题的求解复杂度为 $O(2^n)$ ^[3]。目前关于这个问题也没有精确的求解方法,只能通过改变代价模型来优化问题,求出近似最优解。

关于物化视图选取问题的研究,目前已有许多成果。在文献[4]中提出了一种基于贪心算法的 BPUS 算法,并首次提出了立方体格模型(Lattice Model),它可以表述视图之间的导出(偏序)关系;在文献[5]中对 BPUS 进行了改进,提出了更为简单的 PBS 算法,将运算速度提高了几个数量级;文献[6-7]将遗传算法应用于求解物化视图选择问题,在降低运算复杂度方面取得进展。为了进一步优化视图选择算法,文献[8]深入地研究了物化视图的预选择算法,缩减了物化视图选择算法的搜索域。

经过分析可以发现,这些算法都是基于预先已知用户查询分布的算法,可以称之为静态选择算法,而对于动态物化视图的研究相对较少。在文献[9]中给出了一种在线的物化视图选择算法,但其每做一次查询,就要执行收益值排序算法,即进行一次静态排序算法,显然运算太过密集,很难满足视图实时选择的需要。由于现有方法存在的不足,本文提出了一种将静态算法与动态算法相结合的算法 2-PMVS,通过此算法可以很好满足 OLAP 随机查询响应的需求。算法首先利用用户的预估查询需求对原始数据表进行预计算,即执行静态物化视图选择算法;随后对用户的突发访问进行动态物化,这里主要应用了 Cache 机制,通过以上两个步骤,可以使得物化过程具备很好的自适应性和动态调整性,有效弥补了空间一性能饱和的问题^[10],使算法具备了一定的智能性。

1 算法介绍

1.1 问题描述

本文主要是利用两阶段物化视图选择算法,对用户查询需求经常变更的问题,给出了合理的解决算法。在本文的算法中仅考虑空间的约束,并未考虑更新代价的约束。

定义 1 物化视图选择算法的优化模型可以表述为:

$$\begin{cases} \min \text{AVG}(\text{Response}(MV, q_i)), & q_i \in Q \\ \sum_{mv_i \in MV} \text{Space}(mv_i) < \text{Space}_{\text{total}} \end{cases} \quad (1)$$

其中: $\text{Response}(MV, q_i)$ 表示物化视图集 MV 对查询 q_i 的响应时间,可以看出这是一个典型的 NP 问题,在视图所占空间受到约束的条件下,对查询响应时间的均值求最小值。在本文的算

收稿日期:2008-12-10;修回日期:2009-02-24。

基金项目:国家 863 计划项目(2007AA1123);国家自然科学基金资助项目(70771110)。

作者简介:李明(1985-),男,河南洛阳人,硕士研究生,主要研究方向:数据仓库、多维分析、辅助决策;刘青宝(1968-),男,江西景德镇人,副教授,博士研究生,主要研究方向:辅助决策、数据挖掘;陆昌辉(1976-),男,湖南常德人,讲师,博士研究生,主要研究方向:多维分析、数据挖掘。

法中,也是基于此优化模型进行求解。为了便于求解,可将目标函数进行变换,将其转化为求收益值的最大值。

定义 2 对于物化视图 mv_i 而言,若其可以间接或直接回答查询集 Q 中的 q_i ,我们则称物化视图 mv_i 获益。

根据定义 2 对于收益的描述,可以定义物化视图的收益值函数。

定义 3 物化视图的收益值函数可以描述为:

$$G(mv_i) = f(mv_i) \times (\text{row}(\text{fact}) - \text{row}(mv_i)) \quad (2)$$

其中: $f(mv_i)$ 代表 mv_i 的使用频率,取决查询集 Q ; $\text{row}(\text{fact})$ 表示原数据表的行数; $\text{row}(mv_i)$ 表示视图 mv_i 中行数,通过物化 mv_i 可以直接响应查询集 Q 而不必访问原始数据表,因而可以从中获益,又由于查询代价与数据表行数近似满足正比关系^[5],因而收益值函数可以定义为式(2)。

由定义 2.3 可以将式(1) 中的目标值函数转化为:

$\max \sum_{mv_i \in MV} G(mv_i)$, 它与式(1) 中的优化模型是等价目标优化问题。本文提出 2-PMVS 算法正是基于这个优化模型,通过静态选择算法和动态选择算法来求解最优物化视图集。

1.2 第一阶段物化视图选择算法

第一阶段主要进行静态物化视图选择,完成预计算的功能。关于静态物化的算法前面已经提到,这方面的研究也十分成熟,本文采用了改进贪心算法^[11]进行静态物化视图选择。显然,静态物化视图的计算成本很低,并且存储空间也十分廉价,通过第一阶段的初步物化,可以满足用户的预估查询需求,为加速查询奠定了基础。

为了引出这个静态物化视图选择算法,首先给出如下定义。

定义 4 若物化视图 mv 可以直接或间接回答查询 q_i ,则称 mv 为 q_i 对应的物化视图的祖先,即 $mv \in \text{Ancestor}(q_i)$ 。

根据定义可以看出 $\text{Ancestor}(q_i)$ 也包括 q_i 本身。注:由于 q_i 与物化视图 mv 一一对应,因而可以用 q_i 表示与其相对应的物化视图。

算法 1 统计每个物化视图的使用频率。

输入:查询集 Q 以及每一个查询的频率 $f(Q)$

输出:每一个物化视图的使用频率 $f(MV)$

第 1 步

/* 初始化每个物化视图的使用频率 */

Foreach mv_i in $\{MV\}$

$f(mv_i) = 0$

第 2 步

/* 统计每个视图的访问频率 */

Foreach q_i in $\{Q\}$

Foreach mv_i in $\{\text{Ancestor}(q_i)\}$

$f(mv_i) = f(mv_i) + f(q_i)$

return $f(MV)$

算法 1 十分简单,就是通过循环累加计算每一个物化视图的概率,主要为物化视图的静态选择做准备。

算法 2 静态物化视图选择。

输入:物化视图集 MV 每个物化视图的使用频率 $f(MV)$

输出:静态物化视图集 SMV

第 1 步

/* 初始化静态物化视图集 SMV */

$SMV = \emptyset$

Insert mv_{fact} in SMV

$\text{Space}(SMV) = \text{Space}(SMV) + \text{Space}(mv_{\text{fact}})$

第 2 步

/* 将收益值大于 0 的插入静态物化视图集 */

While ($\text{Space}(SMV) \leq \text{Space}_{\text{total}}$)

{

Foreach mv_i in $\{MV\}$

{

if ($G(mv_i) > 0$) // $G(mv_i)$ 参见式(2)

{

Insert mv_i in SMV

$\text{Space}(SMV) = \text{Space}(SMV) + \text{Space}(mv_i)$

// 计算静态物化视图占用的空间

}

}

return SMV

算法 2 主要通过判断视图的收益值来确定静态物化视图集,在空间约束的条件下,将候选视图集中收益值大于零的予以物化。经过静态物化阶段,对于预判的查询集 $\{Q\}$ 可以具备满意的响应速度,但对于一些突发的访问则无法取得良好的效果,这就要依赖第二阶段的动态物化视图算法。

1.3 第二阶段物化视图选择算法

依照本文算法的要求,在第一阶段的静态物化视图选择之后,还要进行第二阶段的动态物化视图选择。在这个阶段主要是为了适应用户需求的偏移,即补充静态物化视图中遗漏的高收益值物化视图。这个算法依靠 Cache 缓存机制,在用户的即席查询下动态调整,即动态物化的视图是不断变化的,因而替换算法是关键。

相比静态物化视图的存储,动态物化视图存储代价更高,为了反映物化视图的收益,根据定义 3 将收益值函数改进为单位存储下收益值,即:

$$UG(mv_i) = \frac{f(mv_i) \times (\text{row}(\text{fact}) - \text{row}(mv_i))}{\text{Space}(mv_i)} \quad (3)$$

定义 5 若存在一组物化视图集 MV ,则:

$$UG(MV) = \frac{\sum_{mv_i \in MV} f(mv_i) \times (\text{row}(\text{fact}) - \text{row}(mv_i))}{\sum_{mv_i \in MV} \text{Space}(mv_i)} \quad (4)$$

根据式(4) 关于收益值的描述,动态物化视图选择算法如下:

算法 3 动态物化视图选择。

输入:用户查询集 Q

输出:动态物化视图集 DMV , DMV 初始值为空

第 1 步

/* 计算查询 q_i 出现的频率,便于计算 UG 值 */

$f(q_i) = \text{Statistic}(q_i) / \text{Count}(Q)$

第 2 步

/* 动态视图的选择和替换算法 */

if (q_i in $SMV \parallel q_i$ in DMV)

return DMV ;

// 跳出算法函数

else if ($\text{Space}(DMV) + \text{Space}(q_i) \leq \text{Space}_{\text{Cache}}$)

Insert q_i in DMV

else

Sort_{UG}(Array(DMV))

// 将 DMV 中视图按 UG 值升序排列

Pre_del = \emptyset

// 预删除队列初始值为空

$\text{Space}_{\text{free}} = \text{Space}_{\text{Cache}} - \text{Space}(DMV)$

for ($i = 0$; $i \leq \text{Array}(DMV). \text{Count}; i++$)

```

{
    Insert Array(DMV)[i] in Pre_del
    Space_free = Space_free + Space(Array(DMV)[i])
    if (Space_free >= Space(q_i))
        break
}
// 跳出循环

if (UG(q_i) > UG(Pre_del))
    // 若单位存储的收益值大于预删除队列,则插入 q_i
{
    Delete Pre_del from DMV
    Insert q_i in DMV
}

return DMV
// 输出动态视图集

```

算法3描述了用户执行查询 q_i 后对动态物化视图集 DMV 的调整过程,首先若Cache中剩余空间足够并且视图 q_i 不在物化视图(静态和动态)当中,则直接将 q_i 插入 DMV 中;否则将 DMV 中的视图按单位存储收益值 UG 升序排列,把 UG 值较低的逐个放入预删除队列直到剩余空间满足视图 q_i 的大小;最后计算预删除队列的 UG 值,方法如定义5,将其与 q_i 的 UG 值作比较,若 q_i 的 UG 值较大,就将预删除队列从 DMV 中删除,并将 q_i 插入 DMV ,反之保留预删除队列而放弃 q_i 。最后一步是为了保证优化模型中的目标函数始终保持最大,通过循环迭代,最终将 $G(MV) = G(SMV) + UG(DMV) \times Space(DMV)$ 的值不断逼近最大值,即全局最优解。

因为用户的查询需求和预期有偏移,算法3的核心思想就是将静态物化中没有物化但却有较高收益值的视图进行动态物化,以用来弥补这种偏移。用户每做一次查询,都会触发算法3的执行,保证动态物化视图集的最优性,即收益值最大。在算法3中还对每个查询的出现概率进行的动态统计,通过这个动态统计数值,也可以反作用于静态物化视图的选择,即作用于算法2,由于静态物化视图的选择主要在磁盘中完成,因而十分耗时,所以可以放在用户的查询间隙定期执行静态物化视图的更新,从而使得用户的查询得到最佳的响应时间。

2 实验分析

2.1 实验设计

由于2-PMVS算法涉及静态物化与动态物化两个阶段,因而采用开源代码Mondrian来作为测试平台。Mondrian是一个基于ROLAP的数据仓库分析引擎,在Mondrian中包含了Cache管理器,可以通过它来插入动态物化视图选择算法,它本身提供了一个基于内存管理的LRU动态物化视图选择算法,可将本文的算法与其进行对比分析。在静态物化视图选择上主要依赖关系数据库,即通过静态物化视图选择算法在关系数据库中生成静态物化视图,以此可以对仅进行静态选择算法的效率和2-PMVS算法效率进行对比。

实验硬件环境:

CPU为Inter core2 1.8 GHz,内存512 MB。

实验软件环境:

操作系统为Windows XP SP2,关系数据库为MySQL,分析引擎为Mondrian,测试数据集为Mondrian中自带的Foodmart测试数据集,算法采用Java代码开发。

2.2 实验结果分析

实验一 对比Mondrian中LRU动态物化算法和2-PMVS算法,静态物化存储空间定为100 MB,查询频率0.5次/秒,静态物化选择算法均为本文的第一阶段算法,对其不同的

Cache大小下的响应时间进行对比,如图1所示。

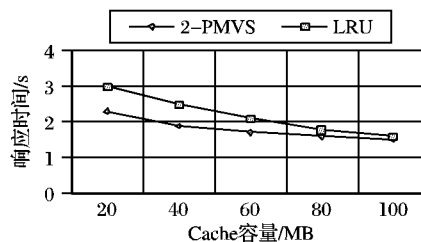


图1 LRU与2-PMVS对比

由图1可以看出在Cache存储空间较小时2-PMVS具有明显的优势,原因是2-PMVS在动态物化选择时收益值中考虑了物化视图集的大小对查询响应时间的影响,而LRU中仅考虑了查询频率,因而2-PMVS在存储空间有限时比LRU存储更为有效,随着存储空间不断加大,而者的效率基本持平。

实验二 对比2-PMVS和仅进行静态选择算法的效率,静态选择算法采用较为流行的PBS算法^[5],两者静态物化存储空间定为100 MB,查询频率0.5次/s,2-PMVS的Cache容量为40 MB,随机生成6个5000次的查询集,对比两个算法的优劣,如图2所示。

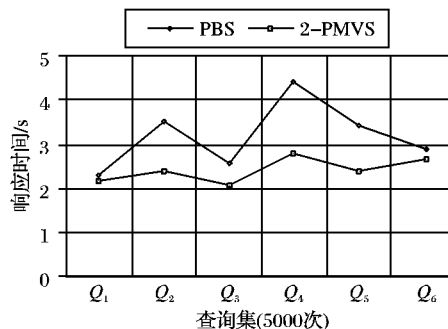


图2 PBS与2-PMVS对比

通过随机产生的6个5000次查询集可以看出,若查询集查询与预先估计分布相似,则PBS与2-PMVS查询效率相似,反之则2-PMVS要大大优于PBS,可以看出PBS的曲线有较大抖动,而2-PMVS通过附加40 MB Cache缓存很好地缓解了抖动,对偏离预估的查询也有不错的响应效率。

实验三 分析2-PMVS的抗压性,即分析在不同查询频率下查询响应时间的变化趋势,如图3所示。

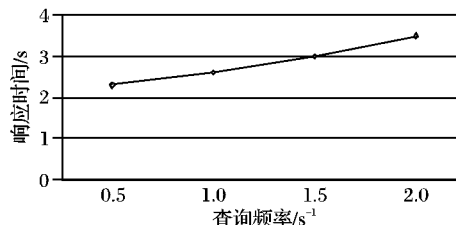


图3 查询频率对算法的影响

从图3可以看出,随着查询频率的不断提高,查询响应的时间也不断增长,原因是每次查询都要执行动态物化选择算法,当查询频率逐渐增高时,动态物化选择算法就会占用大量的CPU时间,从而使得查询效率下降。

3 结语

本文提出的2-PMVS两阶段物化视图选择算法充分考虑了数据仓库投入使用后查询需求的偏移,将静态物化选择算法与动态物化选择算法相结合,使得算法具备了自适应性和

(下转第1611页)

$$U/IND(a_4, D) = \{\{1, 8\}, \{2, 6, 14\}, \{3, 4, 5, 9, 10, 13\}, \{7, 11, 12\}\}$$

$$QG(a_1, D) = |a_1| / |a_1 \cup D| = 66/42 = 1.5714$$

$$QG(a_2, D) = |a_2| / |a_2 \cup D| = 68/38 = 1.7895$$

$$QG(a_3, D) = |a_3| / |a_3 \cup D| = 98/62 = 1.5806$$

$$QG(a_4, D) = |a_4| / |a_4 \cup D| = 100/58 = 1.7241$$

$QG(a_1, D)$ 最小,故从粗粒度 a_1 即天气开始约简。因为 a_1 的三个等价类中,第2个等价类 $\{3, 7, 12, 13\}$ 中各个元组均属于 P 类,第1个和第3个等价类中即有 N 元也有 P 元,因此,可去除对象 $\{3, 7, 12, 13\}$ 元组参与运算,减少 $4/14 = 28.6\%$ 元组数。

表2 气象信息决策系统表

U	天气(a_1)	气温(a_2)	湿度(a_3)	风(a_4)	运动(D)
1	晴	热	高	无风	不适合
2	晴	热	高	有风	不适合
3	多云	热	高	无风	适合
4	雨	适中	高	无风	适合
5	雨	冷	正常	无风	适合
6	雨	冷	正常	有风	不适合
7	多云	冷	正常	有风	适合
8	晴	适中	高	无风	不适合
9	晴	冷	正常	无风	适合
10	雨	适中	正常	无风	适合
11	晴	适中	正常	有风	适合
12	多云	适中	高	有风	适合
13	多云	热	正常	无风	适合
14	雨	适中	高	有风	不适合

2) 用同样方法求细粒度商:

$$QG(a_1, a_2, D) = |a_1 \cup a_2| / |a_1 \cup a_2 \cup D| = 22/14 = 1.5714$$

$$QG(a_1, a_3, D) = |a_1 \cup a_3| / |a_1 \cup a_3 \cup D| = 26/21 = 1.2380$$

$$QG(a_1, a_4, D) = |a_1 \cup a_4| / |a_1 \cup a_4 \cup D| = 26/20 = 1.3$$

$QG(a_1, a_3, D)$ 最小,故从细粒度 $\{a_1, a_3\}$ 即 $\{\text{天气, 湿度}\}$ 开始约简,同理可去除对象 $\{1, 2, 8, 9, 11\}$, 减少 $5/10 = 50\%$

元组数。

3) 用同样方法求最细粒度商:

$$QG(a_1, a_3, a_2, D) = |a_1 \cup a_2 \cup a_3| / |a_1 \cup a_2 \cup a_3 \cup D| = 9/5 = 1.8$$

$$QG(a_1, a_3, a_4, D) = |a_1 \cup a_4 \cup a_3| / |a_1 \cup a_4 \cup a_3 \cup D| = 7/7 = 1$$

故得最约简 $\{a_1, a_3, a_4\}$ 即 $\{\text{天气, 湿度, 风}\}$, 其中平均减少 39.3% 个元组。

8 结语

面对复杂的、难于准确把握的问题,人们通常是通过由粗到细、不断求精、逐步尝试的办法达到比较合理的目标,也就是取得所谓足够满意的解,避免了计算复杂度高的困难。粒度计算理论是人工智能领域中很有价值的解决问题的思维方法,也是当前研究的热点之一。本文把粒度知识应用到粗糙集中,从动态的角度分析了粗糙集的边界域,提出了一种新的属性约简算法,并用实验证明是有效。但没有研究粗细不同粒度世界之间的联系,这方面的工作有待于进一步研究。

参考文献:

- [1] 饶斐,张广明,费宏举,等. 基于粗糙集人工神经网络的锅炉故障诊断模型[J]. 计算机工程与设计, 2008, 29(9): 2333-2335.
- [2] 杜伟林,苗夺谦,李道国,等. 概念格与粒度划分的相关性分析[J]. 计算机科学, 2005, 32(12): 181-183.
- [3] 李道国,苗夺谦,张红云. 粒度计算的理论、模型与方法[J], 复旦学报: 自然科学版, 2004, 43(5): 837-841.
- [4] 袁晓峰,许化龙,陈淑红. 一类最小代价模糊决策系统及其算法[J]. 计算机工程, 2008, 34(7): 200-202.
- [5] 钱宇华,梁吉业. 基于粗糙集的粒度计算理论与方法研究[D]. 太原: 山西大学, 2005.
- [6] 苗夺谦,胡桂荣. 知识约简的一种启发式算法[J]. 计算机研究与发展, 1999, 36(6): 681-684.
- [7] UCI machine learning repository [EB/OL]. [2008-10-10]. <http://www.ics.uci.edu/~mllearn/>.
- [8] Rosetta. A rough set toolkit for analyzing data [EB/OL]. [2008-10-10]. <http://www.idi.ntnu.no/aleks/rosetta>.
- [9] 朱玉全,杨鹤标,孙雷. 数据挖掘技术[M]. 南京: 东南大学出版社, 2006: 108-109.

(上接第1607页)

智能性。通过实验检验,该算法在查询性能上优于基于LRU的动态物化视图选择算法以及纯静态物化视图选择算法PBS,但是随着查询密度的不断增加,该算法的效率开始下降,这还有待于在下一步的研究中予以解决。

参考文献:

- [1] INMON W H. 数据仓库[M]. 王志海,译. 北京: 机械工业出版社, 2002.
- [2] GUPTA H. Selection of views to materialize in a data warehouse [C]// Proceedings of the 6th International Conference of Data Theory. Heidelberg: Springer-Verlag, 1997: 98-112.
- [3] GUPTA H, MUMICK I S. Selection of views to materialize under a maintenance cost constraint [C]// Proceedings of 8th International Conference of Data Theory. Heidelberg: Springer-Verlag, 1999: 120-132.
- [4] HARINARAYAN V, RAJARAMAN A, JEFFREY F, et al. Implementing data cubes efficiently [C]// Proceedings of the 1996 ACM International Conference on Management of Data. New York: ACM Press, 1996: 205-227.

- [5] SHUKLA A, DESHPANDE P, JEFFREY F, et al. Materialized view selection for multidimensional datasets [C]// Proceedings of the 24th Very Large Data Base Conference. New York: ACM Press, 1998: 488-499.
- [6] ZHANG C, YAO X, YANG J. An evolutionary approach to materialized views selection in a data warehouse environment [C]// IEEE Transactions on Systems, Man and Cybernetics. Singapore: IEEE Society Press, 2001: 282-294.
- [7] HORNG J T, CHANG Y J, LIU B J. Applying evolutionary algorithms to materialized view selection in a data warehouse [J]. Soft Computing, 2003, 7(8): 574-581.
- [8] 张柏礼,孙志挥,孙翔. 物化视图选择的预处理算法[J]. 计算机研究与发展, 2004, 41(10): 1645-1651.
- [9] 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.
- [10] 张柏礼,孙志挥,周小云,等. 静态物化视图的动态 Cache 优化算法[J]. 软件学报, 2006, 17(5): 1213-1221.
- [11] 杨少军,范金存,李庆忠. 数据仓库中物化视图的选择[J]. 计算机应用, 2003, 23(9): 58-60.