文章编号:1001 - 9081(2008)07 - 1689 - 03

多维数据的复杂查询聚集算法研究

刘金岭
(淮阴工学院 计算机工程系,江苏 淮安 223001)
(liujinlingg@163.com)

摘 要:对空间多维数据的复杂查询是多维数据研究的重点和难点,目前研究的结论相对较少。在传统算法的基础上,进行了几个方面的改进:按分组属性值进行数据分块;对分组数据进行有效的排序;在聚集函数的应用上进行优化。模拟数据的试验表明:改进算法较大地提高了查询效率。

关键词:复杂查询;聚集函数;粒度
中图分类号: TP311 **文献标志码**:A

Complicated query aggregation algorithms for multi-dimensional data

LIU Jin-ling
(*Department of Computer Science and Engineering, Huaiyin Institute of Technology, Huaian Jiangsu* 223001, *China*)

Abstract: Complex inquiry to the spatial multi-dimensional data is the key point and the difficulty of the multi-dimensional data research. This article has made improvement in several aspects as the following: carry out the data piecemeal according to the grouping attribute value; carry out effective sorting to the integrated data; carry out the optimization in the accumulation function's application. It indicated that the improved algorithm can enhance the inquiry's efficiency greatly via analog data' s experiment.

Key words: complicated query; aggregate functions; granularity

0 引言

近年来,对多维空间(Multi-Dimensional Space, MDS)上的数据分析成为研究的热点,多维数据模型采用数据立方体(DataCube)形式。文献[1]提出空间数据立方体高效操作的原理和空间数据库存储空间目标的详细信息,然而在许多应用中,用户仅仅需要概括性的数据。这些信息能从空间数据库中获取,但计算代价太高,致使在线处理不适用。为解决这个问题,专门构建了一个基于最佳空间粒度的空间索引分类分层算法,将分层用空间数据立方体的格模型描述,以便能处理任意的概括性数据聚集。文献[2]提出基于 Peano 树的高效空间数据 OLAP 操作算法,探讨建立一种 PD 数据立方体,在此基础上利用 Peano 树进行快速分析操作的原理和方法。文献[3]提出利用空间数据立方体进行空间目标多分辨率合并的原理和方法,由于空间目标数量很大,在线实时聚集合并空间目标耗时太多。为解决这个问题,采取预先存储多种不同分辨率的合并好的空间目标,以满足不同用户的实时需求。文献[4]提出利用四叉树集成地理空间数据到 OLAP 系统中的原理与方法等。总之,上述研究虽然对空间数据立方体的某些方面进行了初步研究和论述,但在概念、原理、结构、操作与算法等方面缺乏深入而细致的研究,没有形成较完整的空间数据立方体概念体系。由于数据立方体表示多维数据的优势在于它能在多个粒度层上表示数据,同时有利于计算聚集值,基于数据立方体的查询可以方便地进行各种 OLAP 操作[5]。由于海量数据所带来的问题,完全物化整个数据立方体是不现实的,目前很多研究工作均集中在如何有效减小立方体大小,提高聚集计算速度,从而快速响应查询等方面。这些研究一般针对的都是简单查询。简单查询大多可以直接使用分布聚集函数(如 sum(),count()等)或代数聚集函数(average()等)实现。因此聚集计算是立方体技术的核心功能和重要研究内容[6-7]。目前较多的研究只是集中于仅含简单查询任务(如单个信息)的立方体(称简单立方体)聚集技术实现,而含复杂查询任务(如多个查询)的立方体聚集的实现技术则研究得较少。文献[8]提出了一个直接的计算过程,即单独计算各个粒度和计算各个子查询任务。但对于今天的海量数据集以及高维的数据特征,显然代价是巨大的。

1 多维数据的复杂查询

现实社会中,单个信息获取已经不能满足用户对信息的要求,而构成信息流的含有多个查询任务的复杂查询顺应信息技术的发展而越来越受到用户的青睐,也是当前或未来信息查询的主流。

1.1 多维数据复杂查询聚集的例子

例如,某大型公司想要创建一个销售数据仓库 Sales,记录商品的销售,涉及维 item,branch 和 month,以记录商品的价格、销售分店和月销售额。

例 1:按{item,branch,month}的所有子集分组,对每组找出 2004 年最高价格。在最高价格的元组中,找出最小和最大的商品货架寿命。还要在所有最高价格的元组中,找出具有最小货架寿命的元组的总销售额,并且在所有最高价格的元组中找出具有最大货架寿命的元组的总销售额。

具体算法为:

1)将研究的数据分组:所有可能的分组为 8 个,记为 $R_t(t=1,2,\cdots,8)$,它们分别是:{item,branch,month},{item,

收稿日期:2008 - 01 - 02;**修回日期**:2008 - 03 - 22。
作者简介:刘金岭(1958 -),男,教授,主要研究方向:数据仓库、数据挖掘。

branch}, {item, month}, {branch, month}, {item}, {branch}, {month}, {ALL}, 其中{ALL}指不对任何维分组, 这里的每个分组都对应着SQL中的一个Group By分组。为了提高运算速度, 可以采用共享前缀排序技术, 即先按item分组, 再分别按item和branch、month分组, 最后分别按item、branch和month分组。

2)在8个不同的分组上利用聚集函数计算: 对每个分组求出在2004年的最高价格, 并将最高价格的元组依次记为: $R_{11}, R_{21}, \cdots, R_{81}$。

3)在元组集{$R_{11}, R_{21}, \cdots, R_{81}$}上利用聚集函数, 求出:

①具有最小货架寿命的元组的总销售额;

②具有最大货架寿命的元组的总销售额。

例2: 按{item, branch, month}划分, 求出2004年所有分组的最低价格, 并求出各分组中商品价格在本年度最低价格的125%, 150%, 175%以内的商品的总销售额。

本查询是首先需要求出各个分组的最低价格, 进而求出全年度的最低价格D(各分组中最低价格的最小值), 然后再分别求出各分组内的价格分别在D~1.25D, D~1.5D, D~1.75D之间的商品销售量的总和。

该查询涉及到4个子查询: 1)查询出每一组的最低价格; 2)查询出商品价格在本年度最低价格的125%以内的商品的总销售额; 3)查询出商品价格在本年度最低价格的150%以内的商品的总销售额; 4)查询处商品价格在本年度最低价格的175%以内的商品的总销售额。

根据上述复杂查询的实例, 可以归结出多维数据的复杂查询具有这样的特点: ①查询在所有可能的2^n(n为分组维属性的个数)个分组上进行计算; ②复杂查询有多个具有聚集依赖关系的简单查询构成, 这些简单查询构成了复杂查询的子查询。

1.2 Patitioned-cube算法

文献[5]将聚集函数的类型分为三种类型: 分布(Distributive)、代数(Algebraic)和整体(Holistic)。对于多维数据查询一般是采用Patitioned-cube算法, 并稍加修改后就可以应用于空间多维数据整体型的复杂查询。基本思想是: 1)水平分块为n个子方体[9]; 2)计算每个子方体上的聚集值。

算法描述为:

输入: 基本立方体R; 立方体的分组属性{B1, B2, …, Bm}; 用于聚集的测试属性A; 聚集函数G

输出: 复杂查询结果集

具体方法:

```
1)   任选择一个分块维属性: Bi, i∈[1, m]
2)   按Bi依照文献[9]的方法将R分割成n个子块R1, R2, ..., Rn
3)   for(i=0; i<n; i++)
4)     {
5)       初始化含Bi的聚集器
6)       采用PIPESORT算法[10]中的共享前缀排序技术, 按维属性值进行排序
7)       for all queries
8)         计算各个子查询
9)     }
10)  对于不含分组维属性Bi的粒度, 重复第3~8步, 直到全部完成
```

1.3 Patitioned-cube算法的改进

作为Patitioned-cube算法的改进算法笔者进行了三个方面的改进:

1)将立方体按给定的规则分化为适合内存大小的多个子方。由于数据集太大内存无法一次处理完而采取逐层分块的基本方法。利用文献[9]的方法, 按某个分组维属性的值分块, 使得同一个分组的所有数据同在一个块中。例如, 分组维属性有月份和分店, 可以先按月份分块, 最多分成12个块, 每个块为一个月份的数据, 若每个月份数据量还太大, 则对每个月份的数据再按分店分块, 这种分块方法确保了同一个分组的数据在同一个块中, 因此适合于多粒度并行计算。

2)确定粒度聚集计算的顺序。主要是确定粒度计算的次序, 采用从细粒度到粗粒度的计算策略。如分组维属性{item, branch, month}, 其粒度个数为8个, 先计算含有item的粒度{item, branch, month}, {item, branch}, {item}, 然后再计算去掉item后而含有branch的粒度, 依次进行。

3)优化聚集计算。主要是将各个子查询中的聚集函数分为两类分布或代数函数分为一类, 整体聚集函数分为一类。在子查询中, 先使用部分聚集特性来优化计算, 再计算整体聚集值。如例2中, 计算整体最小价格时, 我们可以先计算每个粒度的最小价格, 再在各个粒度的最小价格集中计算出整体的最小价格。

算法描述:

输入: 基本立方体R; 立方体的分组属性{B1, B2, …, Bm}; 用于聚集的测试属性A; 聚集函数G

输出: 复杂查询结果集

```
1)   任选择一个分块维属性: Bi, i∈[1, m]
2)   按Bi依照文献[9]的方法将R分割成n个子块R1, R2, ..., Rn
     //分块要考虑到内存的大小, 若粒度很大, 则考虑二次分解
3)   for(i=0; i<n; i++)
4)   {
5)     初始化含Bi的聚集器
6)     采用PIPESORT算法[10]中的共享前缀排序技术, 按维属性值进行排序, 并且依照先细粒度再粗粒度的原则
7)     for all queries
8)       计算各个子查询
         //在计算中要遵循先计算分布或代数函数, 再计算
         //整体聚集函数的原则
9)       end for
10)    计算去掉维属性Bi剩余子块的各粒度, 即余下的子查询
11)  }
12)  对于剩余的维属性Bi重复第2到11步
```

2 实验结果

实验的目的主要是比较算法Partitioned-Cube的修改算法(简记MPACU)与其改进算法(简记IPACU)的性能, 两算法分块的次数与子方个数是相同的, 排序过程也是相同的, 因此实验结果中的时间不含立方体分块时间和排序时间。实验是对复杂查询的例2进行的两算法的查询性能比较。在算法实现中, 对比实验均采用相同的数据结构, 没有采用数据压缩处理技术。优化策略通过简化聚集计算过程和减少查询计算量达到性能提高的目的。算法用VC++6.0实现, 在内存为2.0 GB, 主频为酷睿双核2.0 GHz, 操作系统为Windows XP的方正计算机上进行实验。为进行实验对比, 采用随机函数生成五组均匀分布的模拟数据集, 元组个数为300万条左右, 假设将立方体分块次得到的子方都适合内存, 分组维属性取4, 分

别在 5 组数据下进行实验比较。实验表明,对于不同的分组维属性个数,优化算法均能提高 200% 左右的效率。在不同的分组维属性下,局部分布聚集特性所起的优化作用基本相同。实验结果见图 1。

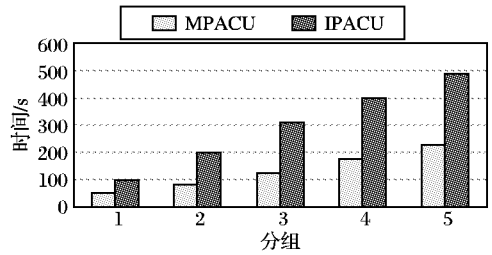

图 1　两种算法查询效率比较

3　结语

目前,人们对多维数据查询技术中涉及部分粒度的聚集计算问题也还只是集中在多维数据的简单查询上,很少对多维数据的复杂查询进行研究。本文主要对多维数据的整体型复杂查询的聚集计算方法进行了研究,针对传统方法中存在的不足,通过研究该类型的复杂查询的计算特点,进行了三个方面改进工作,即分块、排序和优化。通过对模拟数据集进行实验,结果表明改进算法能有效地提高整体型复杂查询计算的效率。

参考文献:

[1] PAPADIAS D. Efficient OLAP operations in spatial data warehouses, HKUST-CS01-01[R]. Hong Kong: [s. n.], 2001: 65 – 69.

[2] WANG B Y, PAN F. Efficient OLAP operations for spatial data using peano trees[C]//DM KD' 03. San Diego: [s. n.], 2003: 126 – 130.

[3] PRASHER S, ZHOU X F. Multiresolution amalgamation : dynamic spatial data cube generation[C]// 15th Australasian Database Conference(ADC2004), Conference in Researchand Practice in Information Technology. Dunedin New Zealand: [s. n.], 2004: 345 – 349.

[4] RAUBER A, TOMSICK P, R IEDEL H, *et al*. Integration geo spatial data into OLAP systems using a set-based quad-tree representation [C]// Proceedings Ninth SSDBM' 97. Washington, DC: IEEE Computer Society, 1997: 256 – 260.

[5] GRAY J, BOSWORTH A, LAYMAN A, *et al*. Datacube: A relational aggregation operator generalizing group-by, cross-tab, and sub-to-tals[C]// Proceedings of the IEEE ICDE. [S. l.]: IEEE Press, 1996: 152 – 159.

[6] AGARWAL S, AGRAWAL R, DESHPANDE P, *et al*. On the computation of multidimensional aggregates[C]// Proceedings of the 22th International Conference on Very Large Data Bases. Mumbai, India: Morgan Kaufmann, 1996: 506 – 521.

[7] HARINARAYAN V, RAJARAMAN A, ULLMAN J. Implementing data cubes efficiently[C]// Proceedings of ACM SIGMOD. New York: ACM Press, 1996: 205 – 206.

[8] ROSS K A, SRIVASTAVA D, CHATZIANTONIOU D. Complex aggregation at multiple granularities[C]// Proceedings of EDBT. American: Springer-Verlag, 1998: 263 – 277.

[9] ZHAO Y, DESHPANDE P M, NAUGHTON J F. An array-based algorithm for simultaneous multidimensional aggregates[C]// Proceedings of ACM SIGMOD Conference. Tucson: ACM Preww, 1997: 159 – 170.

[10] YAN YU-HONG, BEJAN A. Modeling workflow within distributed systems[C]// The 6th Intemational Conference on CSCW in Design. London: [s n], 2001.

(上接第 1685 页)

如果取 $\lambda = \frac{1}{3}, \alpha = 0.7$,依据本文所定义的模型有:

$S^{\alpha}(u_1) = \{u_1, u_{12}\}$

$S^{\alpha}(u_2) = \{u_2, u_3\}$

$S^{\alpha}(u_3) = \{u_2, u_3\}$

$S^{\alpha}(u_4) = \{u_4\}$

$S^{\alpha}(u_5) = \{u_5\}$

$S^{\alpha}(u_6) = \{u_6\}$

$S^{\alpha}(u_7) = \{u_7\}$

$S^{\alpha}(u_8) = \{u_8\}$

$S^{\alpha}(u_9) = \{u_9, u_{12}\}$

$S^{\alpha}(u_{10}) = \{u_{10}\}$

$S^{\alpha}(u_{11}) = \{u_{11}\}$

$S^{\alpha}(u_{12}) = \{u_1, u_9, u_{12}\}$

$\overline{R}^{\alpha}(Y_{\Phi}) = \{u_1, u_2, u_3, u_4, u_7, u_{10}, u_{12}\}$

$\underline{R}^{\alpha}(Y_{\Phi}) = \{u_1, u_4, u_7, u_{10}\}$

$\overline{R}^{\alpha}(Y_{\Psi}) = \{u_2, u_3, u_5, u_6, u_8, u_9, u_{11}, u_{12}\}$

$\underline{R}^{\alpha}(Y_{\Psi}) = \{u_5, u_6, u_8, u_{11}\}$

$\rho_R(Y_{\Phi})_{\alpha} = 1 - \frac{4}{7} = \frac{3}{7}$

$\rho_R(Y_{\Psi})_{\alpha} = 1 - \frac{4}{8} = \frac{1}{2}$

由此例可以看出,本文所定义的集对相似关系克服了文献[3]的缺陷,提高了相似度,更适合要求较精确的信息系统。

5　结语

本文在文献[3]的基础上定义了另一种不完备信息系统,基于集对理论的粗糙集上、下近似,将粗糙集理论在不完备信息系统方面做了进一步的推广,并通过简单的例子说明了其优点和可行性。在不完备信息系统中,当某几个对象属性值都已知或空值太多时,这种方法是实用而有效的。对于某些实际问题,$\underline{R}^{\alpha}(X)$ 和 $\overline{R}^{\alpha}(X)$ 中的 α,以及 $a + \lambda b - c$ 中的 λ 都是主观给定的,它取决于人们对同一程度的不同要求。下一步的工作,是在本文提出的集对相似关系的基础上进一步研究不完备信息系统的属性约简和规则抽取算法,为实际应用系统开发奠定理论基础。

参考文献:

[1] PAWLAK Z. Rough sets: Theoretical aspects of reasoning about data[M]. Dordrecht: Kluwer Academic Publishing, 1991.

[2] 王国胤. Rough 集理论在不完备信息系统中的扩充[J]. 计算机研究与发展, 2002, 39(10): 1238 – 1243.

[3] 黄兵, 钟斌, 周献中. 改进集对粗集模型[J]. 计算机工程与应用, 2004, 18(9): 82 – 84.

[4] 邓毅雄, 黄兆华. 不完备信息系统的基于集对分析粗糙集模型[J]. 华东交通大学学报, 2004, 22(2): 100 – 103.