第34卷 第10期 计 算 机 工 程 2008年5月
Vol.34 No.10 Computer Engineering May 2008
·软件技术与数据库· 文章编号：1000—3428(2008)10—0067—03 文献标识码：A 中图分类号：TP391

基于 Caching 重用的复杂数据立方体聚集方法

唐培和，王日凤，刘 浩
(广西工学院计算机工程系，柳州 545006)

摘 要：基于数字立方体的复杂查询是立方体技术的发展方向。该文针对复杂立方体查询中可能存在的3种聚集依赖，分别给出3种基于Caching重用技术的解决方法。在模拟数据集和真实数据集上的实验结果验证了该方法的有效性和正确性。
关键词：立方体查询；复杂查询；粒度计算

Aggregation Approaches for Complex Data Cube Based on Caching Reusing

TANG Pei-he, WANG Ri-feng, LIU Hao
(Dept. of Computer Engineering, Guangxi University of Technology, Liuzhou 545006)

【Abstract】 Complex query based on data cube is the development direction of cube technique. This paper detects three types of the dependent-relationships in complex data cube query, and proposes three methods based on cache reusing accordingly. The experiments on synthetical and real datasets illustrate the approaches proposed are efficient and promising.
【Key words】 data cube query; complex query; granular computing

数据立方体是多维空间数据的一个有效模型。用立方体表示多维数据直观且有利于计算聚集值。随着信息处理技术的发展，从简单立方体查询的实现(含 1 个子查询)到复杂立方体查询的快速响应是决策支持系统的必然趋势和发展目标。基于数字立方体的复杂查询是立方体技术的发展方向，聚集依赖性是复杂立方体查询的主要特性[1]。基于立方体的查询是OLAP(On-Line Analysis Processing)技术的核心功 能[2]。本文研究了复杂立方体查询的3种聚集依赖关系(完全依赖，部分依赖和互斥依赖)并给出相应算法(完全Caching重用、部分Caching重用和反Caching重用机制)。

1 相关工作

立方体查询技术始于1996年，由于立方体中多维数据聚集计算复杂且难度较大，因此多数研究集中于简单立方体查询。复杂立方体查询研究目前处于起步阶段，相关文献较少。1998 年文献[2]首次提出复杂立方体查询，并用扩展的 SQL 语言对其进行描述；文献[1]将复杂立方体查询分为分布型、代数型和整体型 3 类，并提出了分布型和代数型复杂立方体查询的计算方法；文献[3]针对计算难度较大的整体型复杂立方体查询提出解决方法，利用部分分布聚集特性优化计算整体型复杂查询；文献[4]在文献[3]的基础上增加了冰山查询重用技术，提高了整体型复杂立方体查询的效率；文献[5]根据整体型复杂立方体查询的特点，提出基于 Cache 重用的方法，初步分析了复杂立方体查询中可能存在的 3 种聚集依赖关系，但没有给出具体解决方法或相应算法。

2 复杂立方体查询及其聚集依赖

2.1 复杂立方体查询及其特征

简单立方体查询是在多个粒度上计算且仅含一个子查询的查询；复杂立方体查询是在多个粒度上计算且含有 2 个或 2 个以上子查询的查询。计算复杂查询的立方体也称为多特征立方体方，简称多特征方。

复杂立方体查询除了具有立方体查询的典型特征，即多个粒度聚集计算外，还有一个独有的主要特征，即聚集依赖。这种聚集依赖特征和复杂立方体查询的提出有关，本文研究的复杂查询是由用户连续提出的、在时间上具有同时性或连续性、其中含多个子任务且它们之间具有一定内在逻辑性的查询。这与不同用户间断提出的查询流不同，后者只是一种按时间顺序排列的查询序列，内容上可以毫不相关，可以不存在内在逻辑性特征，因此，能随意交换。而立方体查询的多个子查询任务的执行顺序一般不能任意交换。

由上述分析可知，复杂立方体查询中的聚集依赖性是指构成查询的多个子任务间的逻辑依赖关系，例如后一子查询的聚集依赖于前一子查询的聚集结果。

2.2 复杂查询中的聚集依赖

设数据库为销售数据库Selling，取其 4 个维{time, customer, price, sale}。设R_a, R_b分别表示在同一粒度层的不同子查询的查询数据集/结果集，则3种子查询间的聚集依赖关系分别如下：

(1)完全重叠

若$R_a \subset R_b$，则聚集结果为完全重叠依赖。完全重叠可分为 2 种情况：1)前一个子查询的数据集/结果集包含后一个子查询的数据集/结果集；2)后一个子查询的数据集/结果集包含前一个子查询的数据集/结果集。例如：

查询 $Q1$：按{month, customer, item}的所有分组，求出2006年的最低价格，并求出各分组中最低价格商品的总销售量。

基金项目：广西自然科学基金资助项目(0481016)
作者简介：唐培和(1964－)，男，副教授，主研方向：软件工程，人工智能；王日凤，博士研究生；刘 浩，讲师、硕士
收稿日期：2008-02-27 **E-mail：**tangpeihe@163.com

查询 $Q2$：按{month, customer, item}的所有分组，求出 2006 年所有分组的最低价格，并求出各分组中商品价格在最低价格的 125%, 150%, 175%以内的商品的总销售量。

$Q1$ 和 $Q2$ 属于完全重叠情形。在查询 $Q1$ 中，第 1 个子查询先求出 2006 年的最小价格，第 2 个子查询在第 1 个子查询的结果集(即价格为最低价格的元组)中求出总销售量。查询 $Q2$ 的第 2～第 4 个子查询，都存在后一个子查询的数据和结果包含了前一个子查询的数据和结果的情况。

(2)部分重叠

若$R_c=R_a \cap R_b$且$R_c \neq \phi, R_c \neq R_a, R_c \neq R_b$，则聚集结果为部分重叠依赖，即前后结果中有部分结果相同，而其余的不相同。例如：

查询 $Q3$：按{month, customer, item}的所有分组，求出第 1 个月～第 2 个月所有商品的销售变化量及第 2 个月～第 3 个月所有商品的销售变化量。

$Q3$ 的第 1 个子查询需要查找第 1 个月～第 2 个月的商品销售变化量，第 2 个子查询需要查找第 2 个月～第 3 个月的销售商品变化量，因此，2 个子查询中均要用到第 2 个月的销售数据。

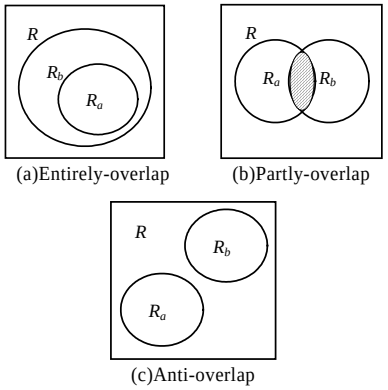
(3)互斥重叠

若$R_a \cap R_b = \phi$，则聚集结果为互斥重叠依赖。这种情况表明一个子查询的数据集/结果集不包含另一个子查询的数据集/结果集，并且一个子查询的数据集/结果集需要在剔除另一个子查询的数据集/结果集基础上进行。例如：

查询 $Q4$：按{month, customer, item}的所有分组，求出 2004 年的总销售量，并求出各分组中占总销售量第 1 个 10%的最高销售量商品和占第 2 个 10%的次高销售量商品。

$Q4$ 的第 2 个子查询查找占销售量 10%的最高销售量商品，而第 3 个子查询则查找占销售量第 2 个 10%的次高销售量商品。具体做法是将商品按销售量排序后，先选取最高的占总销售量 10%的商品，然后再在余下商品中找满足第 2 个子查询的商品。

3 种查询任务间的聚集依赖关系如图 1 所示。



(a)Entirely-overlap (b)Partly-overlap

(c)Anti-overlap

图 1　3 类聚集依赖

3　基于 CACHE 重用的聚集技术

针对上述 3 种立方体聚集依赖关系，本文提出基于 Cache 重用的聚集技术。

3.1　Cache 重用技术

Cache 本义是高速缓冲存储器，是解决 CPU 计算速度与外存读取速度不匹配而使用的一种特殊存储器子系统。Cache 重用技术指存储当前 data 以供后续计算重复使用，从而节约重新处理和载入的时间，提高效率。Cache 重用技术广泛用于查询处理，包括查询流和传统的数据库查询。

3.2　3 种基于 Cache 重用的依赖聚集技术

复杂立方体查询中可能存在的 3 种依赖聚集有一个共同特点，即存在于前后子查询任务的聚集结果集中。因此，本文提出了基于 Cache 重用的依赖聚集方法。

(1)使用完全 Cache 重用机制解决完全重叠依赖。对于复杂立方体查询中的完全重叠，由于前后子查询的结果集是完全包含的关系，因此在计算前一子查询任务后，不清除当前 Cache 中的 data，为后续子查询完全重用。

例如，在查询 $Q1$ 中，实现了子查询任务 1 后，已经保存了所有粒度所有分组的最小值(可能达几万个甚至更多)，因此，在实现子查询任务 2 时，可以直接使用这部分聚集结果，无须重新计算，对查询 $Q2$ 可做相同处理。图 2 描述了此实现过程：将子查询任务 1 的聚集结果 Result1 及子方 $C1$ 都完全重用到子查询任务 2 中去，后续做相同处理。

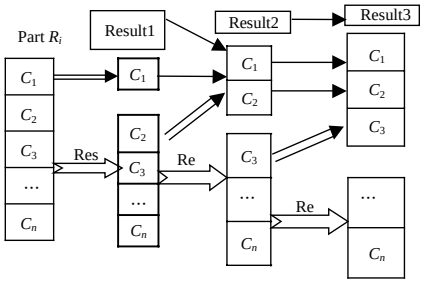


图 2　完全重叠型 Caching 重用示意图

(2)使用部分 Cache 重用机制解决部分重叠依赖。因为子查询结果集是部分重叠，所以只须部分重用结果集。而且，由于已经聚集的子查询任务的结果可以被后续子查询任务重用，因此不必重新使用已经计算过的方体数据，只须对其余数据继续后面的聚集过程。称这样的部分结果和方体重用为部分方体和结果集 Cache 重用。例如，在查询 $Q3$ 中，实现了查询任务 1，输出聚集结果后，只须保留第 2 个月各个分组的 Sum(Sales)，用于子查询 3 的聚集过程。

(3)使用反 Cache 重用机制来解决互斥重叠依赖。在互斥依赖中，由于不同子查询中的结果集之间是不重叠的，且后续子查询需要在剔除前一个子查询任务的聚集结果集的基础上进行，因此在实现了当前子查询任务后，必须清除这部分结果和数据，以便后续子查询任务能顺利实现。这种清除机制相应地称为反 Cache 重用。例如，查询 $Q4$，在实现了子查询 1 和子查询 2 后，必须把子查询 2 中满足第 1 个 10%Sum(Sales)的商品从当前 Cache 中剔除，这样子查询 3 的第 2 组商品才能被顺利找出。

4　实验和分析

为验证方法的可行性及执行效率，本文在模拟数据和 Weather真实数据集上进行了效率对比实验。对于基本算法，分布型和代数型复杂立方体查询采用来源于计算分布型和代数型简单立方体查询的BUC算法的Partitioned-Cube实现[3]；整体型复杂立方体查询采用以Partitioned-Cube为基础的算法[1]。对于改进的比较算法，前 2 类采用在Partition-Cube基础上增加Caching技术的改进算法；而整体型则采用在PDIC算法[2]基础上修改的算法。为提高效率，本文同时对 3 种类型的复杂立方体查询均采用了冰山查询技术(iceberg query)[3-4]，该技术的目的是解决大数据集的选择物化问题，采用一定的冰山条件，只物化部分立方体。基本数据集是冰山，筛选出的数据集为冰山顶。由于是有条件的选择数据，因此能将搜

索的范围尽可能缩小到用户感兴趣的部分，从而提高查询效率。

4.1 数据集

为验证本文方法的有效性，笔者在模拟数据集和真实数据集上进行了试验。模拟数据采用数据生成器产生，数据量为 5×10^{6}条记录，分稠密型和稀疏型 2 类；真实数据集采用 Weather数据集，数据量为 0.98×10^{6}条记录，如表 1 所示。

表 1　真实的 weather 数据及属性选取

	项	基
分组维属性	Date	30
	Low-cloud type	12
	Total-cloud cover	9
维属性	Station number	100 000
测试属性	Change Code	10
	longitude	[0, 36 000]

4.2 基本算法

对于分布型和代数型复杂立方体查询，采用 Partitioned-Cube 算法实现。Partitioned-cube 算法思想主要有以下 2 点：

(1)将大的基本数据集分化为多个适合内存大小的子方(Partitioned-Cube 到 Memory-cube)；

(2)在每个子方上独立执行各个复杂操作(independent operation)。

4.3 复杂立方体查询优化算法

优化算法也相应分 2 类：(1)分布型和代数型优化算法；(2)整体型优化算法。第(1)类优化算法是在 Partitioned-Cube 算法基础上增加 Caching 重用技术，即在计算粒度的聚集函数时，增加如下判断：

```
if   TypeAggreQuery(q_i,q_j) = Entire_Caching
     then   call AggreQuery(q_i,q_j, 0);
     else   if   TypeAggreQuery(q_i,q_j) = Part_Caching
            then   call AggreQuery(q_i,q_j, 1);
            else   if   TypeAggreQuery(q_i,q_j) = Anti_Caching
                   then call AggreQuery(q_i,q_j,2);
```

其中，AggreQuery(Sub-query1, Sub-query2, Caching_type)中的 0, 1, 2 分别代表 Entire_Caching, Part_Caching, Anti_Caching。第(2)类优化算法可以通过将文献[7]PDIC 算法中的相应语句用该 if 语句替换来实现。

4.4 实验结果

实验目的主要是测试采用了Caching技术后算法的效率情况。在稀疏、稠密和真实数据集下的实验结果如图 3 所示(共 5×10^{6}条记录)。在实验结果图中，BasicAlg代表基本算法；ImproAlg代表改进算法。

由实验结果可以看出，在 3 种数据集下改进算法的效率均高于基本算法，且真实数据集中效率的提高程度低于模拟数据集，原因是模拟数据集的数据量远大于真实数据集，数据量越大，Caching 重用技术的优势越明显。

通过对分布型和代数型复杂立方体查询的实验比较发现，其效率提高程度明显低于整体型复杂立方体查询，原因是整体型复杂立方体查询除了采用 Caching 重用技术外，还采用了部分分布聚集性质及冰山查询技术。

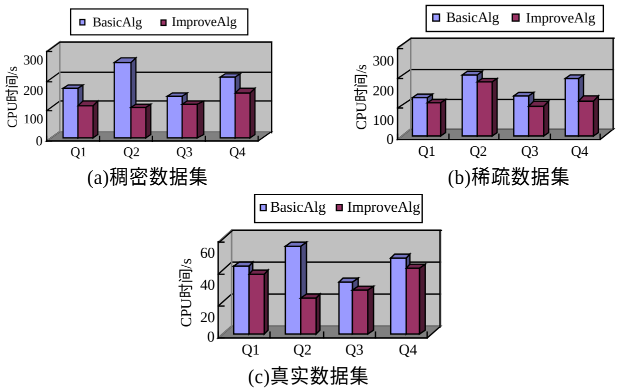


图 3　实验结果

5 结束语

本文分析了复杂立方体查询不同于简单立方体查询的多个优势，并针对已有研究的不足，对复杂立方体查询进行了深入研究，提出基于 Cache 重用的聚类依赖解决方法。

参考文献

[1] Ross K A, Srivastava D, Chatziantoniou D. Complex Aggregation at Multiple Granularities[C]//Processings of the 6th Int'l Conference on Extending Database Technology. Valencia, Spain: Springer Verlag, 1998: 263-277.

[2] Chatziantoniou D, Ross K A. Querying Multiple Features of Groups in Relational Databases[C]//Proceedings of the 22nd International Conference on Very Large Data Bases. [S. l.]: Morgan Kaufmann Publishers Inc., 1996.

[3] 曾德胜, 覃　泽, 王日凤. 一种基于立方体的复杂查询的高效算法[J]. 计算机应用研究, 2007, 24(3): 30-33.

[4] 覃　泽, 王日凤, 张师超, 等. 多特征方查询优化策略[J]. 计算机应用, 2006, 26(7): 1655-1658.

[5] Zhang Shichao, Wang Rifeng, Guo Yanping. Efficient Computation of Multi-feature Data Cubes[C]//Proceedings of the 1st International Conference on Knowledge Science, Engineering and Mangement. Guilin, China: [s. n.], 2006.

~~~~~~

(上接第 66 页)

参考文献

[1] McGregor C, Schiefer J. A Web Service Based Framework for Analyzing and Measuring Business Performance[J]. Information Systems and E-business Management, 2004, 2(1): 89-110.

[2] Fu S, Chieu T, Yih J, et al. An Intelligent Event Adaptation Mechanism for Business Performance Monitoring[C]//Proc. of ICEBE'05. Beijing, China: [s. n.], 2005.

[3] Felber P, Chan C, Garofalakis M, et al. Scalable Filtering of XML Data for Web Services[J]. Internet Computing, 2003, 7(1): 49-57.

[4] Bebawy R, Sabry H, Kassas S, et al. Nedgty: Web Services Firewall[C]//Proc. of ICWS'05. Orlando, Florida, USA: [s. n.], 2005.

[5] Diao Yanlei, Fischer P, Franklin M, et al. YFilter: Efficient and Scalable Filtering of XML Documents[C]//Proc. of ICDE'02. San Jose, CA, USA: [s. n.], 2002.
~~~~~~