联机分析查询处理中的一种聚集算法*

蒋旭东, 冯建华, 周立柱

(清华大学 计算机科学与技术系 软件研究所,北京 100084)
E-mail: fengjh@tsinghua.edu.cn
http://www.cs.tsinghua.edu.cn

摘要: 联机分析处理(online analytical processing,简称OLAP)查询是涉及大量数据的即席复杂查询,从SQL(structured query language)角度来看,这些查询通常都包含多表连接和分组聚集操作.从 OLAP 查询处理角度出发,提出一种新的基于排序的聚集查询算法 MuSA(sort-based aggregation with multi-table join).该方法充分考虑到数据仓库星型模式的特点,将聚集操作和新的多表连接算法 MJoin 相结合,排序时采用关键字映射技术对排序关键字进行压缩,从而可以显著地提高排序速度.此外,通过预先估计聚集分组的数目,优化选择不同的排序方法,使得算法对不同的分组聚集查询都进行优化.算法实验数据表明,这种聚集查询算法与传统的聚集查询处理方法相比,其性能有显著的提高.

关 键 词: 数据仓库;OLAP(online analytical processing)查询;多表连接;聚集查询;星型模式
中图法分类号: TP311 文献标识码: A

OLAP(online analytical processing)是数据仓库系统最主要的一种应用,OLAP 查询一般都是涉及大量数据的即席复杂查询[1].用户通过提交 OLAP 查询对数据进行分析,辅助决策,通常需要较快的查询响应速度.提高 OLAP 查询处理的性能是数据仓库领域的关键性研究问题.

目前主要有 MOLAP(multi-dimensional OLAP)和 ROLAP(relational OLAP)两种方式可用于 OLAP 查询的实现.ROLAP 方式是将数据仓库中的数据直接存储在关系表中,目前已经有一些方法能提高 ROLAP 查询的响应速度,如新的索引技术[2]、实物化视图技术[3]等.OLAP(ROLAP)查询中通常都会包含多表连接和分组聚集操作,提高这些操作的性能是提高 OLAP 响应速度的关键.目前,有关聚集查询处理的代表性文献都集中于聚集查询的并行处理[4]和结合聚集操作的查询优化方法[5].本文从优化 OLAP 聚集查询处理本身出发,提出了一种新的基于排序的聚集算法——MuSA(sort-based aggregation with multi-table join).

MuSA 算法充分考虑到数据仓库星型模式的特点,将聚集查询和新的多表连接算法 MJoin[6]相结合,排序时采用关键字映射技术对排序关键字进行压缩,从而显著地提高了排序速度.此外,通过预先估计聚集分组的数目,优化选择不同的排序方法,使得算法对不同的分组聚集查询都进行优化,进一步了提高算法的总体性能.算法实验结果中的数据表明,我们提出的聚集查询算法 MuSA 与传统的聚集查询处理方法相比,在性能上有显著的提高.

* 收稿日期: 2000-04-04; 修改日期: 2000-07-26
基金项目: 国家重点基础研究发展规划 973 资助项目(G1998030414)

作者简介: 蒋旭东(1972－),男,江苏徐州人,博士,工程师,主要研究领域为数据仓库,OLAP,查询处理;冯建华(1967－),男,山西运城人,博士生,副教授,主要研究领域为数据库,数据仓库,WWW 环境下的信息处理;周立柱(1947－),男,江苏连云港人,教授,博士生导师,主要研究领域为数据库,数字图书馆,海量信息处理.

1 聚集算法 MuSA

1.1 星型模式

数据仓库中的数据一般都按照星型模式进行组织,如本文使用的星型结构实例:

事实表: Sales(TimeID,RegionID,ProductID,SaleNum,income)

维表: DimTime(TimeID,year,month ,day)

DimRegion(RegionID,area,province,city)

DimProduct(ProductID,type,ProductName,price)

该星型结构对应于数据仓库的销售主题,由事实表 Sales 和 3 个维表 DimTime,DimRegion,DimProduct 共同组成,维表的主码分别为 TimeID,RegionID,ProductID,并作为事实表 Sales 的外码,将事实表和维表进行关联.事实表中的记录数远远大于维表中的记录数.

当按照星型结构组织数据时,各种数据仓库应用提交的 OLAP 查询一般都同时涉及分组聚集计算和多表连接操作.如下面的查询实例 Q1:

```
SELECT year,area,type,sum(SaleNum),sum(income)
FROM   DimTime,DimRegion,DimProduct,Sales
WHERE DimTime.TimeID=Sales.TimeID AND
      DimRegion.RegionID=Sales.RegionID AND
      DimProduct.ProductID=Sales.ProductID
GROUP BY year,area,type
```

查询 Q1 要求按年、地区和产品类型得到总销售数量和总的销售收入.该查询同时涉及多表连接和分组聚集计算,是一个比较典型的 OLAP 聚集查询.

1.2 聚集算法MuSA

当前分组聚集计算主要有基于排序和基于 Hash 的两种方式[7].这里提出的 MuSA 算法采用基于排序的方法.其核心思想是,首先对各个维表进行处理,得到维表每条记录对应的分组序号,然后使用我们在文献[6]中提出的 MJoin 算法进行多表连接操作,并采用各个维表的分组序号组合成的关键字进行排序,最终进行聚集计算.由于分组序号组成的关键字很短,所以就相当于对原分组字段组成的排序关键字进行了数据压缩.这样不但能提高排序的速度,并且由于关键字长度变短,还可以采用基数排序方法来代替原来适用的快速排序方法,可以进一步提高排序的性能.

下面我们以维表 DimTime 为例来说明如何对维表进行预处理,从而得到维表每条记录的分组序号.首先获得维表 DimTime 分组字段 year 的所有取值,并按其值进行排序,给出每个取值的分组序号.结果见表 1.

Table 1 Group number of DimTime

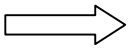
表 1 DimTime 的分组序号

Year	Group No.
1997	0
1998	1
1999	2

在得到每个分组字段值对应的分组序号以后,就可以相应给出维表 DimTime 每条记录对应的分组序号 Group No.,如图 1 所示.此外,对于维表 DimTime 我们还可以得到按字段 Year 进行分组的分组数目 $Groups_{year}$,表示分组数 $Groups_{year}$ 的最小二进制位数 $GroupsBit_{year}$ 以及分解组合关键字时所需的位运算掩码 $GroupsMask_{year}$.对应于给定的 DimTime 表实例,这 3 个值分别为 $Groups_{year}=3$,$GroupsBit_{year}=2$,$GroupsMask_{year}=3$.

对于维表 DimRegion,DimProduct 也将与 DimTime 同样处理.

TimeID	Year	Month	Day
1	1997	1	1
2	1997	1	2
…	…	…	…
366	1998	1	1
…	…	…	…



TimeID	Group No.
1	0
2	0
…	…
366	1
…	…

Fig.1 Group number of every record for DimTime

图 1 DimTime 表的每条记录的分组序号

下面我们给出 MuSA 算法的处理步骤:

算法 **1**. MuSA

输入: 事实表 FT,维表 $DT_1,\ldots,DT_m$,分组字段为 $GA_1,\ldots,GA_m$,聚集字段为 Sum(A);

事实表和维表进行连接的字段分别为 $ID_1,\ldots,ID_m$,排序缓冲区为 B.

(1) 将 Q 分解为单表查询 $Q_1,\ldots,Q_m$,其中 Q_i 为对维表 DT_i 的简单查询,仅包含原查询 Q 中与维表 DT_i 有关的查询条件和相关字段(包括连接字段和分组字段);

(2) For i=1 to m

提交查询 Q_i,得到维表记录,然后计算出每条记录对应的分组序号,并同时得到 $Groups_i$,$GroupsBit_i$ 和 $GroupMask_i$ 的值;

(3) While 顺序扫描事实表 FT,对应于得到的每条记录 R:

(3.1) Key=0;

(3.2) For i=1 to m

(a) 通过连接字段 ID_i 值,查找相应维表 DT_i,得到分组序号 $GroupNO_i$;

(b) Key=Key<<$GroupsBit_i$; /* Key 左移 $GroupsBit_i$ 位 */

(c) Key=Key|$GroupNO_i$; /* |为按位或操作 */

(3.3) 将 Key 和聚集字段 A 组合成的记录 R(Key,A)放入排序缓冲区 B

(3.4) If 排序缓冲区 B 满

对排序缓冲区中的记录按照 Key 值进行排序,并将结果写入磁盘

(4) 将步骤(3)生成的若干有序记录序列归并成一个完全有序的记录序列;

(5) 进行聚集计算:

(5.1) 从有序记录序列中读一条记录 R(Key,A),生成分组 G(GKey,SumA),并设定 GKey=Key,在地 SumA=A;

(5.2) While 从有序记录序列中依次读入剩余的一条记录 R(Key,A)

If Key=GKey

SumA=SumA+A;

Else

For i=m to 1

(a) $GroupNO_i$=GKey & $GroupsMask_i$; /* &是按位与操作 */

(b) GKey=GKey>>$GroupsBit_i$; /* GKey 右移 $GroupsBit_i$ 位 */

(c) 根据 $GroupNO_i$ 查找位于内存中的相应维表,得到分组字段值 GA_i;

将 $GA_1,\ldots,GA_m$ 和 SumA 作为分组聚集计算的一条结果记录写入磁盘;

生成新的结果分组 G(GKey,SumA),并设定 GKey=Key,SumA=A;

End If

在上面给出的算法步骤中,为清晰起见,我们将步骤(5)从步骤(4)分离出来.实际上,为了减少因暂存中间结果而导致的不必要的磁盘读写开销,应该在归并的同时就进行聚集计算.

MuSA 算法使用各个维表的分组序号组合成的分组排序关键字来替代分组字段直接形成的关键字,这虽然增加了某些额外操作,包括分组排序关键字的生成以及聚集计算完成后,由分组排序关键字反算分组序号并

得到分组字段值,但是,由于仅涉及简单的整数算术运算,增加的额外开销并不是很大.从另一方面来说,分组排序关键字长度变短,不但能直接提高排序性能,而且也减少了写入磁盘的中间结果的数据量,此外,必要时还可以选用基数排序代替快速排序方法进一步提高排序的性能,因此,MuSA 算法与传统的基于排序的聚集算法相比,总体性能得到显著的提高.

1.3 基于分组数估计的查询优化

当事实表记录数目过多时,基于排序的基本 MuSA 算法的开销主要是在外存归并排序操作上.但事实上,对于结果分组数目较少的聚集查询,可以通过使用插入排序方法来避免外存归并排序,从而提高聚集查询处理的性能.使用这种方法对聚集算法进行优化的关键在于结果分组数目的正确估计,直接估计查询结果的分组数比较困难,但是我们可以很容易地得到每个维表按分组字段进行分组的分组数目(如上一节对于维表 DimTime 的分组数 $Groups_{year}$),然后通过简单地将每个维表分组数相乘而得到聚集查询结果的估计分组数.例如,对于查询 Q1 的估计分组数应为

$$Groups=Groups_{year}*Groups_{region}*Groups_{type}.$$

由前面的分组数估计方法可知,估计出的分组数必然大于或者等于实际分组数.这样,我们可以估计出保存所有结果分组所需的最大内存空间,如果比查询可用的物理内存小,就可以采用插入排序方法来进行聚集计算.这样,通过避免外存归并排序可以极大地提高聚集计算的性能.当所需内存比可用物理内存大,但分组数比事实表记录数小得多时(我们在实际算法实现时,是判断事实表记录数是否为估计分组数的 100 倍以上),可以使用排序结合预先分组的方法,即对缓冲区中的记录进行排序之后就进行分组聚集计算.这样,通过减少生成的外存归并段的大小来减少磁盘读写次数,从而提高聚集算法的性能.

2 算法实验结果分析

我们在数据仓库的研究中,实现了聚集查询算法 MuSA,并进行了算法实验.实验所用的星型模式各表结构和查询 Q1 前面已经给出,另有聚集查询 Q2,如下所示:

```
SELECT year,city,ProductName,sum(SaleNum),sum(income)
FROM   DimTime,DimRegion,DimProduct,Sales
WHERE  DimTime.TimeID=Sales.TimeID AND
       DimRegion.RegionID=Sales.RegionID AND
       DimProduct.ProductID=Sales.ProductID
GROUP BY year ,city,ProductName
```

该查询要求按年、城市和产品得到总的销售数量和总销售收入.一般来说,查询 Q1 的结果分组数目较少,而 Q2 的结果分组数目较多.

首先我们将算法 MuSA 和 Sybase 的聚集查询进行了比较,实验所用环境为 SunSparc20(32M 内存),数据库使用的是 Sybase 系统 11.

表 2 为查询 Q1 的实验比较结果.Q1 的估计分组数为 900,实际分组数也为 900,MuSA 实际采用以插入排序方法进行的聚集计算.MuSA 算法与 Sybase 查询处理相比,其加速比为几十倍,并且随着事实表记录数的增加而逐渐增大.表 3 为查询 Q2 的实验比较结果.Q2 的估计分组数为 540 000,实际分组数目与事实表 Sales 的记录数有关,MuSA 算法与 Sybase 查询处理相比,其加速比为几百倍,并且随着事实表记录数的增加而逐渐增大.

Table 2 The result of experimentation for query Q1

表 2 查询 Q1 的实验结果

Records number of sales table①	Response time of MuSA② (s)	Response time of Sybase③ (s)
100 000	3.0	99.7
200 000	5.1	192.0
400 000	7.9	391.3

①Sales 表记录数,②MuSA 算法的响应时间,③Sybase 算法的响应时间.

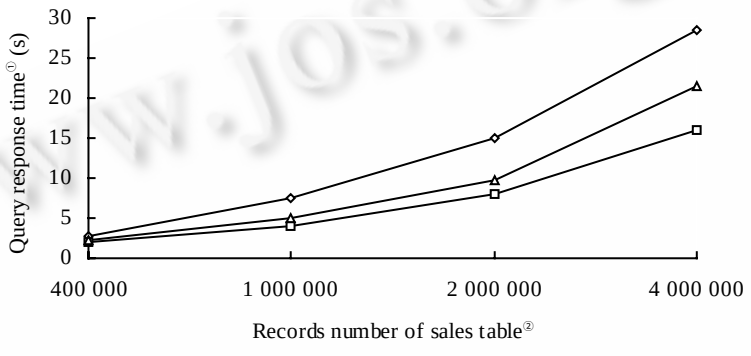
Table 3 The result of experimentation for query Q2
表 3 查询 Q2 的实验结果

Records number of sales table①	Response time of MuSA② (s)	Response time of Sybase③ (s)
100 000	4.5	924.2
200 000	8.0	2507.1
400 000	16.3	>6000

①Sales 表记录数,②MuSA 算法的响应时间,③Sybase 算法的响应时间.

从以上比较结果可以看出,MuSA 算法与传统的聚集查询处理算法相比,在各种分组结果情况下,其查询性能都有显著的提高.

下面,我们再给出在分组数较小时,各种聚集计算方法的实验结果比较.实验环境为 SunUltra30(内存为 128M),实验结果如图 2 所示.对于聚集查询 Q1,我们分别给出了 3 种查询处理方法(基本的 MuSA 算法、直接插入排序方法和排序结合预先分组方法)在查询处理时的响应时间.由于 Q1 的估计分组数为 900,因此使用直接插入排序方法的聚集查询性能最优,相对于基本的 MuSA 算法,其加速比约等于 2.由此可知,基于聚集分组数估计的查询优化能够进一步提高 MuSA 算法的性能.



①查询响应时间,②Sales 表记录数,③基本的 MuSA 算法,④直接插入排序方法,⑤排序结合预先分组方法.

Fig.2 Comparison of query Q1 based on three aggregation algorithm
图 2 3 种不同聚集算法对查询 Q1 的比较

3 总 结

本文提出的 MuSA 算法充分考虑了数据仓库星型模式数据组织的特点,将多表连接和聚集计算结合起来,生成较短的分组排序关键字来加快基于排序的聚集算法的排序速度,并采用基于结果分组数估计的方法优化选择不同的排序策略,使得算法对各种分组聚集查询都进行优化.在算法实现时,我们还考虑了面向缓存、磁盘存取进行优化等问题.算法实验结果表明,相对于传统的聚集查询处理算法,MuSA 算法的性能有显著的提高.

目前,MuSA 算法主要适用于数据按照星型模式进行组织的情况.今后,我们还将进一步研究,扩展其适用范围,使得对于一般的聚集查询(如 TPC-D 查询)也能够适用.

References:

[1] Chaudhuri, S., Dayal, U. An overview of data warehousing and OLAP technology. ACM SIGMOD Record, 1997,26(1):65~74.

[2] O'Neil, P, Quass, D. Improved query performance with variant indexes. ACM SIGMOD Record, 1997,26(2):38~49.

[3] Srivastava, D., Dar, S., Jagadish, H.V., *et al*. Answering queries with aggregation using views. In: Vijayaraman, T.M., Buchmann, A.P., Mohan, C., *et al*, eds. Proceedings of the 22nd International Conference on Very Large Data Bases. San Francisco: Morgan Kaufmann Publishers, 1996. 318~329.

[4] Sameet, A., Rakesh, A., Prasad, M.D., *et al*. On the computation of multidimensional aggregates. In: Vijayaraman, T.M., Buchmann, A.P., Mohan, C., *et al*, eds. Proceedings of the 22nd International Conference on Very Large Data Bases. San Francisco: Morgan Kaufmann Publishers, 1996. 506~521.

[5] Weipeng, P.Y., Per-Ake, L. Eager aggregation and lazy aggregation. In: Umeshwar, D., *et al*, eds. Proceedings of the 21st International Conference on Very Large Data Bases. Zürich: Morgan Kaufmann Publishers, 1995. 345~357.

[6] Jiang, Xu-dong, Zhou, Li-zhu. A multi-table join algorithm for data warehouse query processing. Journal of Software, 2001,12(2): 190~195 (in Chinese).

[7] Graefe, G. Query evaluation techniques for large databases. ACM Computing Surveys, 1993,25(2):73~130.

附中文参考文献:

[6] 蒋旭东,周立柱.数据仓库查询处理中的一种多表连接算法.软件学报,2001,12(2):190~195.

A Novel Aggregation Algorithm for Online Analytical Processing Query Evaluation*

JIANG Xu-dong, FENG Jian-hua, ZHOU Li-zhu

(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084, *China*)

E-mail: fengjh@tsinghua.edu.cn

http://www.cs.tsinghua.edu.cn

Abstract: The OLAP (online analytical processing query) queries are ad-hoc, complex queries, as expressed in SQL, these queries include multi-table join and aggregate operation. In this paper, a novel sorting based aggregation algorithm, MuSA (sort-based aggregation with multi-table join), is given for OLAP query evaluation. In this algorithm, by taking the characteristics of star schema into consideration, the aggregation operation is combined with a novel multi-table join algorithm, MJoin, and the key words mapping technique is used to compress the sorting key which can obviously speed up sorting. Further by estimating the group number of query result, the proper sorting methods which can optimize the algorithm for different aggregation queries can be chosen. As being illustrated by the experimental result, compared with original methods for aggregation query evaluation, the performance of the new algorithm can be improved dramatically.

Key words: data warehouse; OLAP (online analytical processing) query; multi-table join; aggregation query; star schema

* Received April 4, 2000; accepted July 26, 2000

Supported by the National Grand Fundamental Research 973 Program of China under Grant No.G1998030414