数据仓库查询处理中的一种多表连接算法*

蒋旭东, 周立柱

(清华大学 计算机科学与技术系,北京 100084)

E-mail: dong@cpqmail.cs.tsinghua.edu.cn

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

摘要: 在进行数据仓库的OLAP(online analytical processing,联机分析处理)查询处理时,经常会涉及到多表连接操作,因此,提高多表连接的性能就成了数据仓库领域的关键性问题.基于数据仓库的星型模式,给出了一种新的多表连接算法(M Join).与传统关系数据库管理系统的多表连接查询处理相比,该算法充分考虑了数据仓库中的数据本身和多表连接的特点,采用对多个表进行一次性连接的方法,使得查询的性能有明显的改善.同时,还给出了算法的实验结果和分析.

关键词: 数据仓库;OLAP(online analytical processing)查询;多表连接;星型模式

中图法分类号: TP311 **文献标识码**: A

在数据仓库环境下,经常会碰到涉及大量数据的复杂查询,包括多表连接、聚集计算等.传统的关系数据库虽然对各种数据操作和查询处理进行了优化,但是还没有充分考虑数据仓库本身的特点,因此,涉及大量复杂数据操作的OLAP(online analytical processing)查询的性能还不是很好.

数据仓库中存储着大量的多维历史数据,但数据很少需要修改,主要应用于多维数据分析、OLAP、决策支持系统和数据挖掘[1].针对这些特点,目前已经提出了一些方法来提高OLAP查询的性能,主要有下面几种.

由于数据仓库中数据很少修改,维护索引的开销就很小,因此,可以根据需要添加足够多的索引,以此来提高查询的性能.文献[2]除了传统的B+树索引以外,还提出了一些新的索引技术,如位图索引(bitmap index)、多维索引、连接索引(join index)等.实验分析数据表明,这些索引确实能够提高查询的性能.

根据需要,可以事先生成一些实物化视图,这样,查询时就可以直接使用存储在实物化视图中的数据,而不需要实时计算.这是一种利用空间的代价来换取时间效率的方法.文献[3~5]给出了利用事先生成的实物化视图来实现OLAP的方法.由于数据仓库中一般只有数据的添加,因此,实物化视图维护的开销也就比数据库中的开销要小.

数据仓库中用于多维数据分析和OLAP查询的数据一般都按照星型模式进行组织,这时可以考虑采用新的多表连接、聚集操作算法来提高查询的性能.文献[6]根据星型模式的特点,设计了Star-Index索引结构和Star-Join多表连接算法,大大提高了在数据仓库环境下多表连接操作的性能.

本文基于星型模式的特点,提出一种新的多表连接算法(M-Join).与文献[6]相比,这种算法无

* 收稿日期: 1999-08-23; 修改日期: 1999-11-24

基金项目: 国家重点基础研究发展规划项目(G1998030414)

作者简介: 蒋旭东(1972—),男,江苏徐州人,博士生,主要研究领域为数据仓库,OLAP,查询处理;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,信息系统,计算机软件.

需额外的连接索引,直接用于关系数据库就能大大提高查询的性能.M-Join 充分考虑了数据仓库中数据本身的特点和多表连接的特点,并对其进行了优化.算法不需要任何额外的索引支持,所需内存很少,而且并不随数据量的增加而增加.与传统数据库的多表连接相比,算法的复杂性随连接表数目的增加变化要小得多,仅仅和多表连接中心表(即星型模式事实表)记录的数目直接相关.文献[7]分析了多种连接算法,并且指出,在某些条件下,同时进行多表连接与通过两两表连接来实现多表连接相比,具有更好的性能,但是没有给出实现的算法.本文的算法和实验数据验证了文献[7]中的观点.

我们还将在另一篇论文里给出基于 M-Join 的聚集操作算法,与文献[8]中的利用实物化视图实现 OLAP 查询相结合,给出完整的 ROLAP 实现方案.

1 多表连接算法(M-Join)

在传统的关系数据库系统中,对多表连接的处理,如 $A\times B\times C$,都是先选择其中的两个表进行连接运算,如先计算 $A\times B$,然后再将连接的结果和 C 进行连接.而能够进行的查询优化也仅仅是基于查询代价选择连接的方法(如嵌套循环、基于 Hash 的连接、基于 Join 的连接等)和选择表两两做连接的次序.

在数据仓库环境下,当数据按照星型结构进行组织时,经常会遇到涉及多个表的连接操作,要是再采用上面的连接处理方法,效率就很低了.下面用实例加以说明.

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)

该星型结构由一个事实表 Sales 和 3 个维表 DimTime,DimRegion,DimProduct 组成.其中 3 个维表的主码分别为 TimeID,RegionID,ProductID,并且作为事实表 Sales 的外码将事实表和维表相关联.事实表中的记录数远大于维表中的记录数.

对于查询 Q:

```
select year,month,day,city,ProductName,SaleNum,income
from DimTime,DimRegion,DimProduct,Sales
where DimTime.TimeID=Sales.TimeID and
    DimRegion.RegionID=Sales.RegionID and
    DimProduct.ProductID=Sales.ProductID
```

是将事实表和 3 个维表做连接.

在采用传统的多表连接处理方法时,有两种选择,或者是先将事实表和维表做连接,或者是先选择两个维表做连接.假设先将事实表和维表做连接,连接的结果记录数和事实表的记录数相等,由于事实表的记录数很大,造成需要暂存的中间结果集很大,从而带来巨大的磁盘 I/O 开销.如果先选择维表之间做连接,因为维表之间实际上没有连接字段,连接就相当于笛卡尔积运算,中间结果集也将越来越大,这同样会带来磁盘 I/O 开销的问题.

考察一下星型结构数据的组织特点.一般维表中的记录数都很有限,而事实表中的记录数却很多,与维表相比相差好几个数量级,而且表的连接都是通过一个中心表,即事实表来进行的.因此,我们可以考虑一次同时进行多个表的连接操作,首先将维表中的记录读出,存入内存中的 Hash

表、排序表,或者其他可按连接属性值进行检索的数据结构,然后依次读取事实表中的记录,对应于读出的每条记录,根据连接属性,查找内存中的各个维表形成的有序表,得到相应的元组,再利用这些元组完成连接操作.下面我们给出详细的基于星型模式的多表 Hash 连接算法 M-Hash-Join.

```
Algorithm M-Hash-Join (R,S1,S2,...,Sm)
    Input:事实表 R(ID1,...,IDm,A1,...,Ap);
          维表 S1(ID1,B11,...,B1q),S2(ID2,B21,...,B2r),...,Sm(IDm,Bm1,...,Bms).
    Output:R 和 S1,S2,...,Sm 的连接 T.
    {
    T=∅;
    FOR 每个 Si DO
      FOR 每个元组 s∈Si DO
          s 写入 HSi 的第 H(s[IDi])个 HASH 桶;  //HSi 为 Hash 函数,其输入为 IDi 属性值
      ENDFOR
    ENDFOR
    FOR 每个元组 r∈R DO
      FOR i=1 to m DO
          si=根据 r[IDi]查找 Hash 表 HSi 得到的记录;
      ENDFOR
      利用 r,s1,s2,...,sm 形成完整的连接结果元组 t;
      T=T∪{t};
    ENDFOR
    RETURN (T);
}
```

这个算法所产生的 I/O 开销为 $B_R+B_{S_1}+B_{S_2}+\ldots+B_{S_m}+U$ 块磁盘存取.其中 B_R 为关系 R 的磁盘块数,$B_{S_1},B_{S_2},\ldots,B_{S_m}$ 分别为关系 $S_1,S_2,\ldots,S_m$ 的磁盘块数,U 为结果 T 的磁盘块数.因为有 $B_R\gg B_{S_1},B_R\gg B_{S_2},\ldots,B_R\gg B_{S_m}$,所以总的磁盘存取块数近似等于 B_R+U.

对于传统的多表连接处理方法,假设两两表连接的次序为 $R\times S_1\times S_2\times\ldots\times S_m$,则因为 $B_{R\times S_1\times S_2\times\ldots\times S_i}>B_R$,最终连接运算所要求的磁盘块存取数将大于 $(2m-1)B_R+U$,其中读操作为 m 次,写操作为 $m-1$ 次,显然,当 $m>1$ 时,此磁盘存取块数比 M-Hash-Join 方法要多.类似地,也可以给出基于星型模式的多表排序连接算法 M-Sort-Join.

```
Algorithm M-Hash-Join (R,S1,S2,...,Sm)
    Input:事实表 R(ID1,...,IDm,A1,...,Ap);
          维表 S1(ID1,B11,...,B1q),S2(ID2,B21,...,B2r),...,Sm(IDm,Bm1,...,Bms).
    Output:R 和 S1,S2,...,Sm 的连接 T.
{
    T=∅;
    FOR 每个 Si DO
        将关系 Si 的所有记录读入内存;
        按属性值 Si[IDi]排序关系得到排序表 SSi;
    ENDFOR
    FOR 每个元组 r∈R DO
        FOR i=1 to m DO
            si=根据 r[IDi]查找排序表 SSi 得到的记录;
        ENDFOR
        利用 r,s1,s2,...,sm 形成完整的连接结果元组 t;
```

```
        T=T∪{t};
    ENDFOR
    RETURN (T);
}
```

同样,算法 M-Sort-Join 也要求 $B_R+B_{S1}+B_{S2}+\ldots+B_{Sm}+U$ 块磁盘存取,近似等于 B_R+U.

多表连接算法 M-Join 适用的条件为参加连接的表通过一个中心表来进行.除了中心表以外,参与连接运算的其他表的记录数较少,可以全部读入内存进行操作.当数据仓库的数据按照星型模式进行组织时,一般的多表连接查询都能够满足 M-Join 所要求的条件.

2 算法实现与实验结果分析

上文给出的多表连接算法可以采用以下方法,在关系数据库管理系统中加以实现.首先,在底层的关系数据库管理系统上构造一个查询预处理模块.该处理模块用来分析用户提交的多表连接查询语句,将其分解成多个单表查询语句(分为事实表查询和维表查询).对于每个维表查询,由查询的结果数据构造在内存中的 Hash 表或排序表.然后,对于事实表查询,对应于得到的每条记录,查询内存中的各维表的 Hash 表或排序表,完成实际的多表连接操作.

我们在数据仓库的研究中,按照这种方法实现了多表连接算法,并进行了实验.

实验用的星型模式各表结构和查询 Q 如前面所给出.系统为 Sun Sparc20 工作站(32M 内存),数据库使用 Sybase 系统 11.其中 DimTime 表的记录数为 1080,DimRegion 表的记录数为 360,DimProduct 表的记录数为 500,Sales 表的记录数满足条件 |Sales|≫|DimTime|,|Sales|≫|DimRegion|,|Sales|≫|DimProduct|.

在使用 M-Sort 算法实现上面给出的查询 Q 时,首先需要将其分解成以下 4 个子查询:

Q1:select TimeID,year,month,day from DimTime;

Q2:select RegionID,city from DimRegion;

Q3:select ProductID,ProductName from DimProduct;

Q4:select TimeID,RegionID,ProductID,SaleNum,income from Sales.

然后,查询处理程序先向 Sybase 提交对维表的查询 Q1,Q2 和 Q3,用得到的结果,分别在内存中构造 Hash 表或排序表.随后,查询处理程序提交查询 Q4,顺序对应于得到的每一条记录,根据与其他维表连接的条件,即分别使用 TimeID,RegionID 和 ProductID 查询对应维表在内存中的 Hash 表或排序表,用得到的结果构造并得到多表连接结果的一条记录.这样,查询处理程序执行完查询 Q4 以后也就处理并得到了多表连接的所有结果.

实验分析 1. 多表连接查询 M-Hash-Join 和传统数据库多表连接查询处理的比较.查询 Q 有两种处理方法,一种是将查询表示成一条多表连接的 SQL 语句,直接提交给 Sybase 服务器进行处理;另一种是使用我们自己实现的 M-Hash-Join 算法进行查询处理.实验得到的数据如图 1 所示.Sales 表记录数分别为 50 000,100 000,200 000 和 400 000.查询响应时间由 UNIX 系统命令 time 得到.多表连接查询直接由 Sybase 数据库服务器执行时所需的时间明显比使用 M-Hash-Join 时要多,使用 M-Hash-Join 算法性能提高约 5～8 倍,并且随着 Sales 表记录数的增加,算法性能提高得更多.

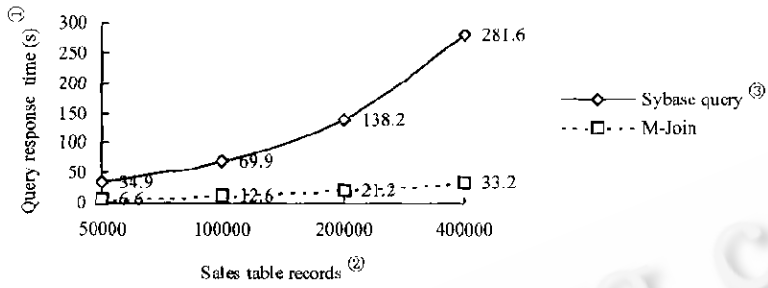

Fig. 1

图 1

实验分析 2. M-Hash-Join 和 M-Sort-Join 的比较. 查询 Q 共有 3 种实现方法:使用 M-Hash-Join 算法、使用 M-Sort-Join 算法、分别使用二分查找和插值查找. 实验得到的数据如图 2 所示. Sales 表的记录数分别为 50 000,10 0000,200 000 和 400 000,具体实验数据见表 1.

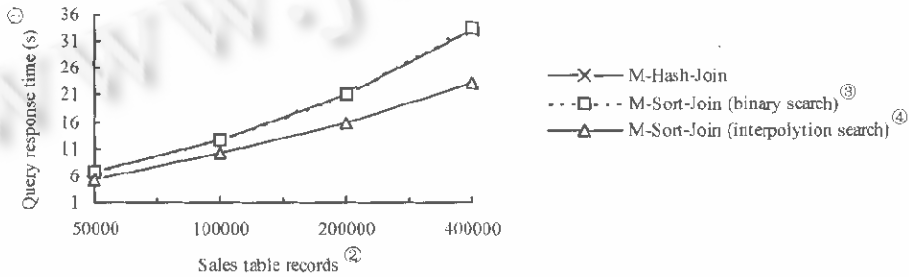

Fig. 2

图 2

Table 1 M-Hash-Join vs. M-Sort-Join

表 1 M-Hash-Join 和 M-Sort-Join 的比较

Sales table records①	50 000	100 000	200 000	400 000
M Hash Join	6.6	12.6	21.1	33.2
M-Sort-Join (binary search)②	6.7	12.7	21.2	33.6
M-Sort-Join (interpolytion search)③	5.4	10.2	15.9	23.2

①表记录数,②M-Sort-Join(二分查找),③M-Sort-Join(插值查找).

从实验结果数据可以看出,M-Hash-Join 和 M-Sort-Join(二分查找)两种算法的性能非常接近,这和前面的算法分析是相符的,因为 M-Hash-Join 和 M-Sort-Join 两种算法的磁盘存取量相同. 而 M-Sort-Join(插值查找)与 M-Hash-Join 相比,性能有明显的改善. 分析实验数据可以发现,每个维表的关键字字段(即和事实表连接的字段)的数值是均匀分布的,这样,在进行多表连接运算时,在已经排序的维表中查找对应的记录,只须比较一次即可,从而大大加快了多表连接的速度.

3 总 结

由于数据仓库中的数据按照星型模式进行数据组织本身存在的一些特点,使得我们可以采用新的多表连接算法. 实验结果表明,M-Join 算法相对于传统的多表连接处理方法具有很高的效率. 将多表连接算法和基于多表连接的聚集计算方法相结合,可以为实时的 OLAP 查询提供较高的性能.

在文献[5]中,我们也给出了利用实物化视图实现 OLAP 查询的算法,用于国家 863 高科技项

目"面向分析预测的智能数据仓库平台"[8].该算法与M-Join算法相结合,将为数据仓库系统在传统的关系数据库基础上,提供一种实现OLAP查询的解决方案.

当数据量很大时,为了提高查询性能,我们还可以考虑将数据分布到多台机器上,通过并行的方式来提高查询的性能.本文提出的多表连接算法很容易并行化.此后,我们将进一步进行研究,将单主机的OLAP查询处理算法(包括多表连接算法和聚集计算算法)并行化,从而从根本上解决在大数据量的情况下实时OLAP查询的性能瓶颈问题.

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., ed. Proceedings of the 22nd International Conference on Very Large Data Bases. San Fransisco: Morgan Kaufmann Publishers, Inc., 1996. 318~329.

[4] Gupta, A., Harinarayan, V., Quass. D. Aggregate-Query processing in data warehousing environments. In: Umeshwar, D. ed. Proceedings of the 21st International Conference on Very Large Database. San Fransisco: Morgan Kaufmann Publishers, Inc., 1995. 358~369.

[5] Jiang, Xu-dong, Zhou, Li-zhu. Answerring OLAP queries using material views. Journal of Lanzhou University (Natural Sciences),1999,35(supplement):242~247 (in Chinese).

[6] Red brick System's White Paper: Star Schemes and Star Join Technology, RedBrick Systems. Los Gatos, CA, 1995. http://www.redbrick.com/products/white/whitebtm.html

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

[8] Feng, Jian-hua, Jiang, Xu-dong, Liu, Jian-min, *et al*. The platform of market analysis and forecast-oriented data warehouse. Journal of Lanzhou University (Natural Sciences), 1999,35(supplement):236~241 (in Chinese).

附中文参考文献:

[5] 蒋旭东,周立柱.利用实物化视图实现OLAP查询.1999年全国数据库会议论文集.兰州大学学报(自然科学版),1999,35(增刊):242~247.

[8] 冯建华,蒋旭东,刘建民,等.面向分析和预测的数据仓库平台.1999年全国数据库会议论文集.兰州大学学报(自然科学版),1999,35(增刊):236~241.

A Multi-Table Join Algorithm for Data Warehouse Query Processing*

JIANG Xu-dong, ZHOU Li-zhu

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

E-mail: dong@cpqmail.cs.tsinghua.edu.cn

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

Abstract: Multi-Table join is a common operation for evaluating OLAP queries posed to a data warehouse. The performance of this multi-table join is one of the key problems in the research of data warehouses. Based on the Star Schema for a data warehouse, this paper introduces a new algorithm M-Join for the multi-table join. Compared with the traditional multi-table join processing by the Relational Database Management System, this new algorithm, taking adequate considerations on the characteristics of the data in a data warehouse environment, completes the join by scanning every table only once, thus greatly improves the performance of OLAP query processing. The paper presents and analyzes the experimental results of this comparison.

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

* Received August 23, 1999; accepted November 24, 1999

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