计算机科学2003Vol.30№.5

并行数据仓库ParaWare系统的查询优化

肖 震 陈 红 王 珊
(中国人民大学数据与知识工程研究所 北京100872)

The Query Optimization of Parallel Data Warehouse System ParaWare

XIAO Zhen CHEN Hong WANG Shan
(Institute of Database and Knowledge Engineering, Renmin University of China, Beijing 100872)

Abstract A Data Warehouse System has a more large scale of data, and rather simple process logic than a DBMS. So those optimize means of DBMS Query system are not so valid. This paper discusses the optimize policy used in to parallel DW system ParaWare, such as query division, public sub query mergence, and fill in the query result. It also gives out the right algorithm.

Keywords Parallel data warehouse, Query optimization

1. 引言

数据仓库作为支持OLAP应用的系统,它所面对的数据往往都具有很大的规模。如何快速地得到查询结果,一直是人们研究的一个重要方向。在已有的研究中,已经相继提出了使用实体化视图和各种有效的索引技术来提高查询响应性能。使用实体化视图是通过数据的冗余存储,对于常用的聚集数据进行预计算,使用预计算结果响应用户查询。而索引技术也主要是应用在存储层上的,如在文[4]中提到在存储层中使用bitmap索引提取相关数据的方法。而在查询处理逻辑层次上的优化技术目前研究得还比较少。在文[3]中提到了使用缓存技术,减少磁盘I/O及通信代价,提高查询效率的方法,这个方法虽然是在查询处理的层次上进行的优化,但是主要还是从存储方面进行考虑的。本文主要针对在并行处理环境下,从查询处理层的角度提出相关优化技术,以充分利用机器的并行能力,在存储层已做优化的基础上进一步提高查询效率。

数据库系统中成熟的优化技术可以作为数据仓库查询优化的一个很好的借鉴,但是由于数据仓库具有独特的特征,所以对于查询优化提出了自己的要求。本文主要针对并行数据仓库ParaWare系统中所使用的一些查询优化技术进行分析,并对这些技术提出相关策略。在以下的部分里首先对ParaWare系统查询处理层的体系架构以及系统接口、对象模型进行相关介绍。之后将重点讨论ParaWare系统中使用的查询优化策略以及相关算法。在并行数据仓库系统ParaWare中已经实现了这些算法。

2. ParaWare查询查询接口及对象模型

2.1 查询接口

ParaWare的查询接口参照的是微软公司的OLE DB for Online Analytical Processing(OLAP)。它是微软公司为了扩展OLE DB访问多维数据的能力而提出的一套对象和接口的标准。使用这个标准可以使得OLAP数据的提供者能够通过其定义的接口为所有的OLAP客户端所访问。

针对多维数据的查询,OLE DB for OLAP定义了一组语义丰富的语句,称为多维表达式(Multidimensional Expressions),简称MDX。

例如对于如下MDX语句:

```
SELECT {CROSSJOIN({销售代表1},{北京.CHILDREN,
        上海}),{销售代表2,北京},{销售代表3,北京
        .海淀},{销售代表3,上海},{销售代表4,北京.
        朝阳}}ON COLUMNS,
      {CROSSJOIN({1月,2月},{电器,电器.CHIL-
        DREN}),{3月,电视},{一季度,电器}} ON
        ROWS
FROM SalesCube
WHERE (Sales,[1991]);
```

查询所表示的结果集如图1所示。

			销1	销1	销1	销2	销3	销3	销4
			北京		上海	北京	北京	上海	北京
			海淀	朝阳			海淀		朝阳
1月	电器								
	电器	洗衣机							
		电视							
		电冰箱							
2月	电器								
	电器	洗衣机							
		电视							
		电冰箱							
3月	电器	电视							
一季度	电器								

图1

2.2 对象模型

MDX语言具有很强的灵活性,对于一个用户MDX查询,可能包含有多个轴,每个轴上又会有多个维,每个维上用户又可以指定若干个成员。因此,需要有一套灵活的对象模型来表示它。ParaWare参照OLE DB标准,定义了用于系统内部处理的各种对象类型。

Query对象:描述一个用户的查询信息,即查询计划;

Axie对象:描述查询中的一个轴的信息;

AxieSeq对象:描述轴上的查询序列信息,即查询在轴上的组合序列;

MemberQuery对象:查询所涉及的某一维上的所有成员

的集合，这些成员按照维值序顺序排列。

例如对于上面的例子，它在 Row 轴上的 AxieSeq 和 MemberQuery 如下所示：

AxieSeq：

销1	销1	销1	销2	销3	销3	销4
海淀	朝阳	上海	北京	海淀	上海	朝阳

MemberQuery：

维	销售人员维		地区维	
	销售代表1	0	北京	0
	销售代表2	1	上海	1
	销售代表3	2	海淀	2
	销售代表4	3	朝阳	3

这些类型间存在如下关系：一个用户查询对应于一个 Query 对象，一个 Query 对象可以包含若干个 Axie 对象，一个 Axie 对象对应一个 AxieSeq，包含若干个 MemberQuery 对象。如图2所示：

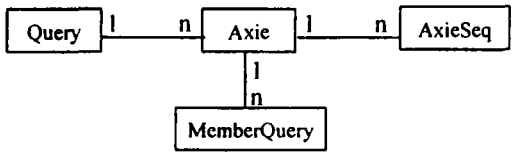

图2

3. ParaWare 的查询优化策略

作为并行环境下的数据仓库系统，要提高查询的效率，就必须充分利用系统的并行处理能力。PareWare 针对并行处理环境选择了相关的查询优化策略，包括查询的划分，公共子查询的合并，以及查询子计划索引表达式的优化。

3.1 查询的划分

用户的查询可能关心的是维上的不同级别的数据，而在存储层上，因为经过数据分片和数据划分，这些数据可能被存放在不同的实体化视图里。将一个大的查询划分为若干个相对小的子查询，再对这些子查询进行并行处理，将很大程度上减少查询的反应时间。这时查询的总延迟时间将是最后一个完成的子查询的响应时间，而如果各个子查询能够独立地进行数据填充，则将进一步减少用户的等待时间。

ParaWare 对子查询的划分是按照查询成员在维上所属级别来进行的，即一个子查询在每一个维上所涉及的成员必须在同一个级别上。这样的划分要求使得一个子查询所涉及的数据落在同一个实体化视图上。

由于 ParaWare 使用这样的查询划分策略，所以必然地子查询的查询结果可能对应着原查询中彼此互不相邻的各个单元。为了使得子查询结果能够被填充到正确的位置，必须保存子查询单元格与展现结果单元格的映射信息，事实上只要保存轴序列的映射信息，就可以通过计算实现查询的映射。

ParaWare 系统使用一个数组来存放轴序列的对应信息。这个数组对应于轴上的 MemberQuery 里的成员的笛卡尔积。顺序扫描轴序列，对于每一个组合，在数组相应的位置填写组合的序号。对于不存在的组合，数组相应位置置为－1。而通过查询各个轴上的这种轴序列数组，就得到用户查询与查询内部处理结构的对应关系。

例子中的 Row 轴上的 MemberQuery 构成一个4×4的空间，根据上面的方法所建立起来的轴对应关系如表1所示。在 column 上的对应关系如表2所示。

表1

	北京	上海	海淀	朝阳
销售代表1	－1	2	0	1
销售代表2	3	－1	－1	－1
销售代表3	－1	5	4	－1
销售代表4	－1	－1	－1	6

表2

	电器	洗衣机	电视机	电冰箱
一季度	9	－1	－1	－1
1月	0	1	2	3
2月	4	5	6	7
3月	－1	－1	8	－1

建立起这样的轴映射数组之后，就可以开始进行查询的划分。将轴上 MemberQuery 里的成员按照层次进行划分，并用起止点的绝对位置来标示这些划分，不同层次的组合构成不同的空间。比如{北京，上海}在地区的 MemberQuery 里位置是{0，1}，{洗衣机，电视机，电冰箱}在产品 MemberQuery 上的位置是{1，3}，等等。通过这样的方法，上例的 row 轴被划分为2个空间，column 轴被划分为4个空间。各个轴上的这种空间的组合最后构成子查询。

例如({北京，上海}，{销售代表1，销售代表2，销售代表3，销售代表4})×({电器}，{一季度})这个子查询可以表示为({0，1}，{0，3})×({0，0}，{0，0})。

这种组合的数目为 row 轴上空间数与 column 轴上空间数的乘积，亦即划分出的子查询数目。

通过子查询各维上起止点位置，就可以从轴的映射空间里取出相对应于这个子查询的部分，即子查询映射数组，如表3，4所示。其中，括号中表示相应 bitmap 位的取值。

表3

	北京	上海
销售代表1	－1(0)	3 (1)
销售代表2	4(1)	－1(0)
销售代表3	－1(0)	6(1)
销售代表4	－1(0)	－1(0)

表4

	电器
一季度	9(1)

如上，子查询的数组中存在－1，即说明存在用户不关心的组合，对于这类数据，并不需要从存储层获取。ParaWare 系统在查询处理层与存储层间通过一个数据的 bitmap 序列来标识这种取舍情况。

通过子查询映射数组，可以很方便地得到这个 bitmap：值为－1的单元格相应的 bitmap 位设置为0，否则相应位置设置为1。上例子查询的 row 轴上的 bitmap 则为01100100，column 轴上为1。将这两个 bitmap 发给存储层，存储层即可计算出所需要返回值的各点，查出相应的值，返回给查询处理层。

在计算 bitmap 时，对于某些轴上的空间可能会存在全零

的情况，如上例中表2的

	洗衣机	电视机	电冰箱
一季度	-1(0)	-1(0)	-1(0)

这说明对于包含该轴空间的子查询不存在用户所关心的数据，则该子查询为空子查询。对于空子查询，则无需作进一步处理。

经过划分的子查询独立地执行，从存储层获得一个结果数组，该结果数组种存放的是查询处理层所要求的点的数据。由于使用了上面的轴映射数组，子查询结果的填充变得非常简单。下面给出它的算法：

```
I=0;
Datap=结果集数组头指针;
while(I<各轴映射数组单元数的乘积)
      计算 I 值所对应的在各轴上的坐标;
      if(不存在一个轴在坐标点位置的轴映射单元数组==-1)
        根据坐标点上的单元值计算填充位置;
        将 Datap 所指对象，填充到该位置;
        Datap 指针后移一个单元;
```

3.2 公共子查询的提取

在多用户并发查询的时候，对于同一个主题不同用户可能会有类似的查询兴趣，因此查询计划中会有公共的部分。通过前面子查询的划分，可以得到最细粒度的子查询，这样公共子查询的判断也就变得非常容易。只要不同的子查询对应的子 Cube 定义相同就可以作为公共子查询提取出来。

公共子查询的合并只在具有相同的查询空间，也就是具有相同维的子查询上才有意义。因为不同的查询空间对于存储层来说是不能同时取出的。合并子查询的算法如下：

```
输入：查询空间相同的子查询集合
输出：合并所得到的公共子查询
算法流程：
生成一个空查询对象 PublicQuery;
for(集合中的每一个子查询 Query)
  for(查询中的每一个 MemberQuery)
    for(MemberQuery 里的每一个成员 Member)
        在 PublicQuery 的相应 MemberQuery 里查找维值序小于该
            成员的最大 LTMember
        将该成员插入到 LEMember 后
根据 PublicQuery 各 MemberQuery 里的成员数，生成 bitmap;
for(集合中每一个子查询 Query)
  for(查询的轴空间映射数组每一个单元)
      if(单元内值==-1)
        下一次循环
      else
          获得该单元相应的各 MemberQuery 的成员;
          查找这些成员在 PublicQuery 的相应 MemberQuery 中的
              序号;
          计算相应的 bitmap 位
          设置 bitmap 位
返回 PublicQuery
```

公共子查询的合并由优化器来实现，优化器在子查询个数超过一个阈值，或达到某一时间间隔时执行公共子查询的合并，并执行这个查询，待查询结束，返回结果后，将结果头指针及公共查询的 bitmap 头指针传给相应的各个子查询，并将各子查询状态设置为已完成。

由于经过公共子查询的合并，子查询所获取的结果集大于自己所需要的结果集，因此要对结果集进行筛选填充。下面给出包含筛选操作的子查询的处理算法：

```
将子查询加入到优化器里;
等待子查询状态变为已完成;
for(bitmap 中的每一位)
  if(该位为0)
    进行下一次 bitmap 扫描循环
  else
      计算该位在公共查询的各 MemberQuery 上对应的成员;
      在子查询相应的 MemberQuery 上查找对应的成员;
      if(某一个 MemberQuery 上未找到相应成员)
        进行下一次 bitmap 扫描循环
      else
          计算子查询的各轴上 MemberQuery 成员组合相应的查询
              轴映射数组单元
          if(该单元上的值为-1)
            进行下一次 bitmap 扫描循环
          else
              根据取出地轴映射单元的值，计算原查询结果单元格
                  位置
              将结果集指针指向的值填入该单元格;
              结果集指针后移一位;
```

由于合并子查询时对子查询的空间进行了扩充，因此不能使用原来的那种简单的填充算法。在上面的算法里对轴上的 MemberQuery 进行了查找，在效率上比原先的线性填充要低一些，但是由于公共子查询可以同时为多用户共享查询数据，将较大地提高查询性能，而且通过选择较好的查找算法，会很大程度上降低查找所带来的消耗。进一步地可以使用一个估计的合并代价值来进行权衡，从而在代价值与效率间取得较优的结果。

结束语 查询响应时间是数据仓库处理能力的一个很重要的因素。本文简单介绍了并行数据仓库系统 ParaWare 查询处理的相关概念及体系结构，并深入讨论了 ParaWare 中所采取的子查询划分和合并的优化方法。通过子查询的划分，可以充分利用多处理机的并行处理能力，降低查询的总的延迟时间。查询的合并可以提高在多用户并发查询时的处理能力，配合 ParaWare 的缓存机制，减少数据存取次数，从而提高查询效率。

在 ParaWare 中已经对上面所提出的算法进行了实现，在下一步研究工作中，将进行合并代价的估算，以及对查询子计划索引表达式的优化技术，将用户指定的查询展现成员按照优化的结果提交给存储层，以利于存储层更快地获取数据的策略。

参 考 文 献

1 OLE DB for Online Analytical Processing. microsoft technique document, 1999

2 王珊，等. 数据仓库技术和联机分析处理. 科学出版社，1998

3 郑霄. 基于语义数据块的两层缓存策略及其在 ParaWare 中的实现：[中国人民大学硕士毕业论文]. 2001

4 蒋岳龙，王珊，陈红. ParaWare 存储结构设计. 计算机科学，2001，28(8. 增刊)

5 O'Neil P, Quass D. Improved Query Performance with Variant Indexe. In: Proc. of the ACM SIGMOD Intl. Conf. on Management of Data, 1997. 38～49

6 Johnson T. Performance Measurements of Compressed Bitmap Indices, In: Proc. of the 26th Int. VLDB Conf., 2000. 278～289