实视图选取策略及其实现技术*

张宜红 徐宏炳 王能斌

(东南大学计算机科学与工程系 南京 210096)

摘要 研究了数据仓库中一般的实视图选取策略,定义一种利益代价图——BC图(Benefits Cost Graph),并以BC图为基础给出了选取实视图集的算法.最后,将BC图扩展为BC⁺图,解决带索引的实视图的选取问题.

关键词 实视图,数据仓库.

中图法分类号 TP311

实视图即物理上实现的视图.它通过物理上存储视图定义的数据,减少基于视图的查询时间.如今,实视图技术已在数据仓库、完整性约束、查询优化、决策支持等方面得到广泛的应用.然而,任何事物都有利有弊,实视图需要一定空间和时间上的开销,特别在基表修改时,需要对实视图进行维护.因此在决定一个视图是虚还是实时,必须权衡利弊.在具体应用环境中,特别是数据仓库的设计中,恰当选取实视图成为最重要的设计决策之一.本文讨论的实视图选取问题指的是:在一定的条件约束下,如何选取一组视图,使系统整体性能(所占空间大小、查询响应时间和维护代价的评价)最优.

对这个问题已有的工作如下.文献[1]在一组给定实视图的基础上,系统另外增加一些实视图,可以在不影响查询性能的前提下减少实视图的整体维护代价.文献[2]针对一类特殊的应用——数据立方,在有空间大小的约束条件下,选取一组实视图,使平均查询响应时间最短.Roussopoulos在文献[3]中给出用于生成最优实视图子集解的搜索空间的方法,并由此可以导出从一组基表和视图到一给定视图的所有可行的运算路径.

目前,还没有一个选取实图的普遍方法,更没有考虑带索引的实视图的选取问题.本文提出一种基于利益代价图(Benefits Cost Graph,简称BC图)的选取实视图的普遍方法,并推广到带索引的实视图的选取.

在本文中,实视图维护代价的计算基于增量维护策略.[4~6]增量维护策略的基本思想是根据基表的修改和视图的定义,求出实视图的变化,然后只修改实视图要修改的部分,不需重新生成,以确保实视图状态与源数据状态的一致性.

1 实视图的代价估算模型

实视图的代价包括系统存储实视图的空间开销、维护实视图的时间开销和视图查询的时间开销,在后面的论述中简称为:空间代价、维护代价和查询代价.本节将提出一种通用的代价估算模型——BC图.

定义1. BC图是一个有向图,它包含3类结点:基础结点、等价结点和运算结点.一个基础结点代表一张基表,一个等价结点表示一个视图,一个运算结点包含一种运算(运算是K元函数,可以是连接、交、并等);一个运算有一个参数,对应的运算结点有一条输入边连接一个等价结点,运算结果所对应的等价结点也有一条输入边连接该运算结点.

由定义可以得出:在BC图中,一条有向边连接一个等价结点(或一个基础结点)与一个运算结点,基础结点只有输出边而无输入边.一个运算结点只有一条输出边,但可以有多条输入边;一个等价结点的不同输入边代表该点的不同运算途径.图1是一个BC图.

在BC图中,只有等价结点才是实视图的候选结点.

给定一视图查询集$\{V_1,V_2,\ldots,V_K\}$,对应BC图的构造方法如下:首先,对每一个视图V_i,应用文献[3]介绍的方法生成基表和其他视图到本视图的所有运算路径,并用相应的BC图表示.然后,合并这K个BC图,即为所求.图2

---

* 本文研究得到国家自然科学基金资助.作者张宜红,女,1973年生,博士生,主要研究领域为数据仓库系统.徐宏炳,1947年生,副教授,主要研究领域为数据库应用.王能斌,1929年生,教授,博导,主要研究领域为数据库及信息系统.

本文通讯联系人:张宜红,南京 210096,东南大学计算机科学与工程系

本文1997-06-09收到原稿,1997-11-11收到修改稿

给出了一个构造BC图的例子.

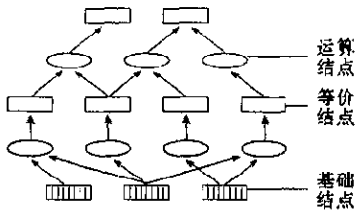

图1 一个BC图

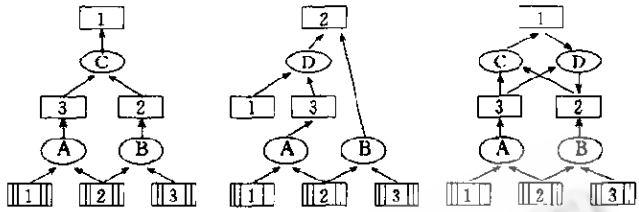

图2

为了估算实视图集的代价,我们要统计系统在一段时间内的运行数据,需统计的参数有:各等价结点对应视图的查询频率、各基础结点对应基表的数据长度、各源系统修改进程的修改频率和平均修改的记录数;然后,可以估算BC图中每个等价结点所对应视图的空间大小以及运算结点所对应操作的执行代价.[7]文献[6]介绍了实视图增址表达式的生成方法.依据这种方法,我们可以生成BC图中各等价结点所对应视图的增址表达式,并估算出增址的数据长度.同时,对于每一个运算结点,还可以估算它产生的增址运算的运算代价.[6,7]这样,在BC图中,每个等价结点有两类参数:对应视图的数据长度和每个修改进程引起的增址数据长度;每个运算结点也有两类参数:对应操作的执行代价和每个修改进程引起的增址运算代价.

BC图构造完毕,实视图的选取问题可定义为:给定一BC图G、约束条件集CS和系统综合性能评价函数$f(G,MV)$,选取图中的等价结点集合MV,设$MV=\{V_1,V_2,\ldots,V_K\}$,对$MV$中的所有视图实现实视图,使得在满足$CS$中所有条件下,$f(G,MV)$的取值最小.因为基础结点的数据可以获取,所以本文讨论的实视图选取问题不包括选取基础结点.

不同的应用会有不同的约束条件与性能评价标准,相应地,CS与$f(G,MV)$形式也会有所不同,但它们一般都与实视图的空间代价、查询代价和维护代价有关,特别是$f(G,MV)$,通常是上述三者的线性函数.因此,对于一个给定系统,如果我们能估算出实视图集的3种代价,则此实视图集的$f(G,MV)$值和CS条件集的满足情况是可计算的.所以,本文将分别讨论这3种代价的估算方法.

$$\text{实视图集的空间总代价}=\sum \text{实视图}\ i\ \text{的空间代价} \tag{1}$$

BC图已给出了所有视图的空间代价,所以,实视图集的空间代价可由对应的等价结点的数据长度求出.

对于一个视图,查询同一集合中的不同实视图,会有不同的查询代价,在BC图中表现为,从一组等价结点到任一等价结点之间都有多条路径,每条路径的代价等于该路径经过的运算结点的执行代价和.本文选取其中的最小值作为查询此实视图集的查询代价.另外,为简化计算,本文假定多个视图的查询相互独立,那么,它们的查询总代价为各个视图的查询代价和.式(2)表达了上述思想.

$$\text{实视图集的查询代价}=\sum_i \text{视图}\ i\ \text{的查询频率}\times\text{视图}\ i\ \text{的最小查询代价} \tag{2}$$

对于一些特殊的视图,如果希望查询速度快些,可以在查询频率前增加一个大于1的系数,反之,可以取一个小于1的系数.

实视图的维护代价包括计算实视图增址的代价和修改实视图的代价.基表的一个修改操作会引起多个实视图的增址修改,根据运算结点的执行代价和增址运算代价,采用多查询优化[7]可以减少实视图集的增址计算的总执行代价.对于每一个修改基表的进程,都要估算引起的实视图增址计算的执行代价,即

$$\text{实视图集的增址计算代价}=\sum(\text{修改进程}\ i\ \text{的频率}\times\text{进程}\ i\ \text{引起的增址计算代价}). \tag{3}$$

同样,实视图集的修改代价为

$$\text{实视图集的修改代价}=\sum(\text{修改进程}\ i\ \text{的频率}\times\text{进程}\ i\ \text{引起各实视图的修改代价和}). \tag{4}$$

式中"各实视图的修改代价",可以由BC图中等价结点的增址数据长度求得.

综合式(3)、(4)得

$$\text{实视图集的维护代价}=\sum \text{修改进程}\ i\ \text{的频率}\times(\text{进程}\ i\ \text{引起实视图集的增址计算执行代价}+\text{进程}\ i\ \text{引起各实视图的修改代价和}). \tag{5}$$

2 贪婪算法

基于上节给出的代价估算模型,本节提出实视图的选取算法.

文献[2]证明了数据立方的实视图选取问题属于 NP 完全问题.目前,对于更一般的情况,要想获得全局最优解,唯一的方法是遍历所有可能解.从理论上来讲,全局最优算法求得的解应是全局最优的,但在实际中,由于对查询频率、修改频率、修改代价、存储空间大小等参数的估算有一定的误差,所以,尽管全局最优算法付出的搜索代价十分庞大,但不一定能得到最优解.在许多具体应用中,只希望系统能较快地给出一个较优集合.基于这种需要,本文介绍一种贪婪算法,求解局部最优实视图集.

在介绍这种算法之前,先定义一个函数.

定义 2. 利益函数 $B=(NV,MV)=f(G,MV\cup NV)-f(G,MV)$,其中 NV 与 MV 是两个不相交的视图集.

贪婪算法采用逐步接近最优解的策略.首先,设选取的等价结点集为空,然后,每一步都选入一个新的等价结点或删除一个已选入的结点,直到无法选取又无法删除为止.选取或删除的标准基于两个门槛值 B_{in} 和 B_{out},$0\leqslant B_{out}\leqslant B_{in}$.当利益函数 $B(\{v_r\},MV)>B_{in}$ 时,相应的等价结点可以选入,若 $B(\{V_X\},MV-\{V_X\})\leqslant B_{out}$,已选入的等价结点可以剔除.通常情况下,可取 $B_{in}-B_{out}=0$.进一步的描述见算法.

条件 $0\leqslant B_{out}\leqslant B_{in}$ 可以避免贪婪算法陷入死循环.在最坏情况下,贪婪算法遍访了所有可能的实视图集,这时它等价于全局最优算法.

算法. 贪婪算法求解局部最优实视图集

```
给定 BC 图 G、约束条件集 CS 和利益函数 B(V_X,MV)
BEGIN
  M:=Set of all equivalence nodes;
  MV:=∅;
  REPEAT
    HALT:=TRUE;
    从 M-MV 中选取结点 V,使
    (1) MV∪{V}满足 CS 中的所有条件;
    (2) V 是满足(1)中 B(V,MV)取值最大的
    if (B({V},MV)>B_in)
    then MV:=MV∪{V}; HALT:=FALSE;
    endif;
    while (MV≠∅)
    do
      从 MV 中选取结点 V,使
      (1) MV-{V}满足 CS 中的所有条件;
      (2) V 是所有满足(1)中 B(V,MV-{V})取值最小的
      if (B({V},MV-{V})≤B_out)
      then MV:=MV-{V}; HALT:=FALST;
      else break;
      endif;
    endwhile;
  UNTIL HALT;
  return MV;
END
```

本文提出的贪婪算法的启发式思路与文献[2]有所不同.在文献[2]给出的算法中,每一步选取的实视图都作为将来要实现的实视图,没有考虑到选入一新的视图后,已选入的实视图的利益贡献可能变得很小.为避免上述弊端,本文在每选一个新结点时增加一个调整步骤,将贡献显著变小的实视图剔除掉,以保证已选入的实视图都是必要的.由此得出:本文中的贪婪算法的性能比文献[2]好,性能的进一步比较还有待于将来的测试.因为贪婪算法采用启发式思路,求解的是局部优最优集合,所以,从理论上说,它的性能比全局最优法差.但正如前文所述,理论上的全局最优集合,在实际中并不一定是最优的,这正是实视图选取问题的复杂性之所在.

3 用 BC⁺图选取带索引的实视图

在 BC 图中,一个视图用一个等价结点表示,并没有在不同的物理实现上加以区别.而物理实现上的不同是影响系统性能的一个重要因素,所以在选取实视图时,应考虑实视图的各种物理实现方法.本节对 BC 图进行扩充,以便考

虑到索引这一最常用的物理实现技术.为区别起见,我们称扩充后的图为 BC^+图.扩充步骤如下:

(1) 对于 BC 图中每一个等价结点 EN,构造它的兄弟结点:$EN_1,EN_2,\ldots,EN_k$.一个视图有多少种建索引的方法,与之对应的等价结点就有多少个兄弟结点,等价结点的兄弟结点也是等价结点.任一个等价结点和它的所有兄弟结点构成它的家庭;

(2) 构造每一个运算结点的兄弟结点:设正在处理的运算结点为 OP,它有一条输出边连接一个等价结点 V;另外,它还有 n 条输入边分别连接着等价结点 $V_1,V_2,\ldots,V_n$,分别属于家庭 $f_1,f_2,\ldots,f_n$,从每个家庭中任取一等价结点构成一组结点 $V_1',V_2',\ldots,V_n'$.这组等价结点参与 OP 运算,只要运算代价有变化,就要生成一个新的运算结点 $OP';V_1',V_2',\ldots,V_n'$中每个结点引出一条边指向新的运算结点,每个新的运算结点引出一条输出边指向等价结点 V;

(3) 对每个运算结点 OP,如果它的输出边连接等价结点 V,则对于 V 所属家庭中的每个等价结点,OP 都要有一条输出边与之相连;

(4) 所有新的边和新的结点都只由上述步骤产生.

图3展示了由一个简单的 BC 图变换到 BC^-图的例子.

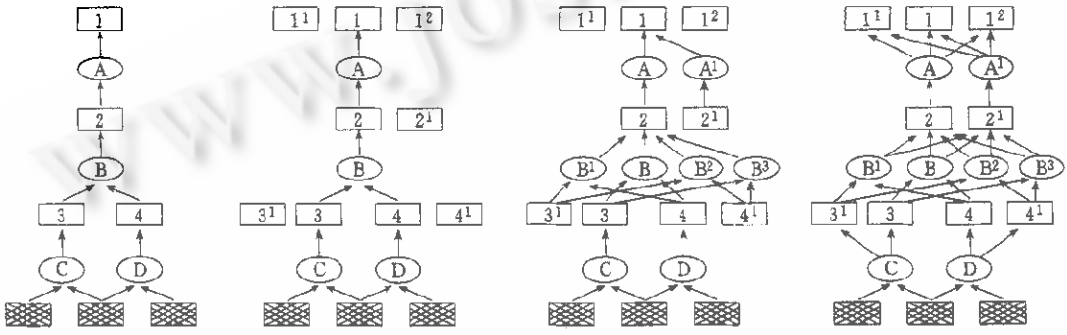

图3

由以上步骤获得的 BC^+图给出了带索引视图的所有可能形式.因此,用 BC^+图代替 BC 图,算法的搜索范围得以扩大.对于簇集、散列以及将来新的物理实现技术,都可以采用类似的方法对 BC 图进一步扩展.

4 今后的研究方向

本文研究了一般实视图的选取问题,提供了两种求解算法的框架,在具体应用中,可以根据不同的环境采用不同的算法.今后的研究方向是:

(1) 本文采用的是一元评价标准,当性能评价标准是二元或二元以上时,例如,评价标准是查询响应时间和空间大小,如何设置算法中的选取原则,还需要进一步的研究.

(2) 本文建立的实视图选取模型基于即时增量维护策略.如果 BC 图中每个等价结点附有维护频率值,就可以计算定期增量维护的代价.文中的算法对采用定期增量维护策略的实视图也适用,更进一步地,有可能提供维护方法上的建议.当然,要达到目的,还有许多的工作要做.

参考文献

1 Ross K A, Srivastava D, Sudarshan S. Materialized view maintenance and integrity constraint checking: trading space for time. In: Jagadish H V, Inderpal Singh Mumick eds. Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data. New York: Academic Press, 1996. 447~458

2 Harinatayan V, Uajaramn A, Ullman J D. Implementing data cubes efficiently. In: Jagadish H V, Inderpal Singh Mumick eds. Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data. New York: Academic Press, 1996. 205~216

3 Roussopoulos N. The logical access path schema of a database. IEEE Transactions on Software Engineering, 1982, SE-8(6): 563~573

4 Gupta A, Mumick I, Subrahmaman V. Maintaining views incrementally. In: Buneman Peter, Jajodia Sushil eds. Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data. New York: Academic Press, 1993. 157~166

5 Zhuge Y, Garcin-Molina H, Hammer J *et al*. View maintenance in a warehousing environment. In: Widom Jennifer ed. Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data. New York: Academic Press, 1995. 316～327
6 Griffin T, Libkin L. Incremental maintenance of views with duplicates. In: Widom Jennifer ed. Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data. New York: Academic Press, 1995. 328～339
7 王能斌. 数据库系统. 北京:电子工业出版社,1995. 127～132
(Wang Neng-bin. Database System. Beijing: Publishing House of Electronics Industry, 1995. 127～132)

Approach to Selecting Materialized Views and Its Implementation

ZHANG Yi-hong　XU Hong-bing　WANG Neng-bin

(*Department of Computer Science and Engineering　Southeast University　Nanjing*　210096)

Abstract　In this paper, the authors propose a general approach to selecting materialized views, and define a structure named BC graph(benefit-cost graph) as the basis for selection of materialized views. A algorithm based on BC graph is then presented for such selection. Finally, it extends BC graph to BC^+ graph to take account of indices associated with views.

Key words　Materialized views, data warehouse.