一种改进的多连接查询优化方法*

钟 武 胡守仁

（长沙工学院计算机系 长沙 410073）

摘要 M.S.Chen 提出了用于产生具有较低计算代价的join丛树的启发式方法G_{MC}和G_{MR}.本文在分析相关join操作的次序与计算代价的关系后，给出了时间复杂度为$O(n^2)$的对G_{MC}和G_{MR}的改进算法.由于在该算法生成的join丛树中，任意两个相邻的内部结点（join操作结点）的操作次序是最优的，因此，它比G_{MC}和G_{MR}能进一步降低join丛树的计算代价.

关键词 关系数据库，多元连接查询，查询优化，并行执行，执行依赖.

中图法分类号 TP311.13

由于一般的SRJ查询在经过投影和选择下推优化之后，往往可以表示为多元连接查询MJ(multi-join query)，所以，目前无论是传统的数据库还是并行数据库，查询优化都是围绕着多元连接查询进行.

文献[1]探讨了一种MJ查询并行执行的优化方法.该文在一定的模型基础上，通过模拟对比和启发方式的比较，阐述了MJ查询并行执行的优化步骤：(1)像在单处理器系统的数据库系统所做的那样，利用启发方式G_{MC}或G_{MR}生成join操作丛树，使该树的总计算代价尽可能地小；(2)在同步执行时间思想的指导下，用自顶向下的方式为join丛树中的内部结点分配处理器，以降低查询执行时间，最终产生并行执行计划.

对上述优化步骤的第1步，文献[1]探讨了4种启发式优化方法，并根据文中给定的模型进行模拟统计后的结果，阐述了G_{MC}和G_{MR}是比较理想的优化方法.因为，①它们的复杂度较低，为$O(n^2)$；②用它们优化的结果接近于用全局优化方法所获得的结果.

关于在单处理机系统上如何以低计算代价执行多join的问题已得到广泛的讨论.文献[2]讨论了采用线性执行策略，避免笛卡尔乘积的优化策略；文献[3,4]阐述了减小优化的复杂度而确立搜索子空间，以获最优解的优化思想.然而，文献[1～4]均未能在具体的估算代价模型的基础上，就join操作次序与计算代价的关系作深入的研究.因此，它们也就不能从代价估算模型的特征上发掘出低复杂度的降低多join执行代价的更好的方法.我们通过对文献[1]所使用的模型进行研究后，发现该模型具有一些特征.它们能进一步改善G_{MC}和G_{MR}局部优化方法.

1 问题描述及模型

同文献[1]，本文讨论的join操作是等连接操作.对于待连接的关系R_1和R_2，它们之间存在同名的等连接属性.一个join查询图用$G=(V,E)$表征.其中，对于每个结点$R\in V$，R表征待连接的关系；对于边$(R_1,R_2)\in E$，它表示需将R_1和R_2进行join操作，边上所附带的字母表征等连接的属性名.图1(a)是一个join查询图.当执行图1(a)中的R_1 join R_3后，产生查询图1(b).其中$R_1\,\theta\,R_3$表示R_1 join R_3生成的关系.

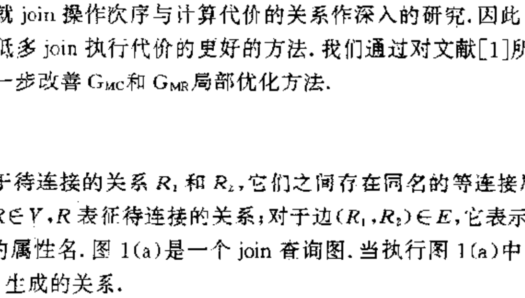

图1

定义1. R_1 join R_2的计算代价cost(R_1,R_2)为：$|R_1|+|R_2|+|R_1\,\theta\,R_2|$.其中$|R_1\,\theta\,R_2|$表示关系$R_1\,\theta\,R_2$的元组数；$|R_1|$和$|R_2|$的含义同上（参见文献[1,5]）.

根据文献[1]，若$G_B=(V_B,E_B)$是查询图G的一个连通子图，令$R_1,R_2,\ldots,R_n$是对应于V_B中的结点的关系，$A_1,A_2,\ldots,A_q$是依附于E_B中的边上的不同属性，m_i是属性$A_i(i=1,\ldots,q)$对应的边所邻接的关系的数目，则将$R_1,R_2,\ldots,R_n$进行等连接后，所产生的关系R的大小为：

* 本文研究得到国防预研基金资助.作者**钟武**，1965年生，讲师，主要研究领域为数据库技术，并行与分布处理技术.**胡守仁**，1926年生，教授，博士导师，主要研究领域为计算机组织与系统结构，并行与分布处理技术，数据库技术.

本文通讯联系人：胡守仁，长沙410073，长沙工学院计算机系

本文1996-12-09收到原稿，1997-03-03收到修改稿

$$|R|=\prod_{k=1}^{n}|R_k|/\prod_{i=1}^{q}|A_i|^{m_i-1} \tag{1}$$

关于公式(1),可参阅文献[6].

2 多个相关 join 的操作次序与计算代价

为方便讨论,本节的讨论都针对两个关系之间存在一个等连接属性的情况.读者将会看到,对两个关系间存在多个等连接属性的情况并不影响本节讨论的结果.

定理 1. 围绕关系 R,对于要执行的两个相关的 join 操作 R_1 join R 和 R_2 join R,其等连接属性分别为 A_1 和 A_2(如图 2 所示).在操作执行的次序上有:R_1 join R 先执行的总计算代价低于(等于)R_2 join R 先执行的总计算代价的充要条件为 $|R_1|/|A_1|<(=)|R_2|/|A_2|$.

证明:R_1 join R 先执行的操作序列为:R_1 join R,$R_1\,\theta\,R$ join R_2,其总计算代价为

$$|R_1|+|R|+|R_1\,\theta\,R|+|R_1\,\theta\,R|+|R_2|+|R_1\,\theta\,R\,\theta\,R_2| \tag{2}$$

同理,R_2 join R 先执行的总计算代价为

$$|R_2|+|R|+|R_2\,\theta\,R|+|R_2\,\theta\,R|+|R_1|+|R_2\,\theta\,R\,\theta\,R_1| \tag{3}$$

由(2)式−(3)式<(=)0 $\Leftrightarrow|R_1|/|A_1|<(=)|R_2|/|A_2|$. □

定义 2. 对围绕关系 R 的要执行的两个相关的 join 操作 R_1 join R 和 R_2 join R,当 $|R_1|/|A_1|<|R_2|/|A_2|$ 时,R_1 join R 应先于 R_2 join R 执行,该关系记为:R_1 join R before R_2 join R;当 $|R_1|/|A_1|=|R_2|/|A_2|$ 时,不论 R_1 join R 先执行还是 R_2 join R 先执行,总计算代价都相等,该关系记为:R_1 join R equal R_2 join R.下面我们将看到:当围绕 R 存在多个 join 操作时,操作间的 before 关系和 equal 关系存在传递性.

定理 2. 围绕关系 R,对于要执行的 3 个相关的 join 操作 R_1 join R,R_2 join R 和 R_3 join R,其等连接属性分别为 A_1,A_2 和 A_3(如图 3 所示).若存在 R_1 join R before R_2 join R 和 R_2 join R before R_3 join R,则 R_1 join R before R_3 join R.

证明:由

$$R_1 \text{ join } R \text{ before } R_2 \text{ join } R \Rightarrow |R_1|/|A_1|<|R_2|/|A_2| \tag{4}$$

由

$$R_2 \text{ join } R \text{ before } R_3 \text{ join } R \Rightarrow |R_2|/|A_2|<|R_3|/|A_3| \tag{5}$$

由(4)式、(5)式 $\Rightarrow|R_1|/|A_1|<|R_3|/|A_3|$. □

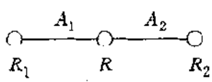

图2

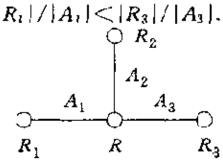

图3

定理 3. 围绕关系 R,对要执行的 3 个相关的 join 操作 R_1 join R,R_2 join R 和 R_3 join R,其等连接属性分别为 A_1,A_2 和 A_3(如图 3).若存在 R_1 join R before R_2 join R 和 R_2 join R equal R_3 join R,则 R_1 join R before R_3 join R(证明略).

定理 4. 围绕关系 R,对于要执行的 3 个相关的 join 操作 R_1 join R,R_2 join R 和 R_3 join R,其等连接属性分别为 A_1,A_2 和 A_3(如图 3 所示).若存在 R_1 join R equal R_2 join R 和 R_2 join R before R_3 join R,则 R_1 join R before R_3 join R(证明略).

3 优化算法

不论是 G_{MC} 还是 G_{MR},其思想均为局部选优.这里,我们就 G_{MC} 的改进做进一步的讨论.对 G_{MR} 的改进的讨论可类似地进行.

在描述改进的算法以前,我们就图 $G=(V,E)$ 的定义做如下的扩充:(1) 对 $R\in V$,R 关系中的元组个数用 $R.value$ 表示;(2) 对于边 $(R,R')\in E$,增加属性 *weight* 和 *lock*.这里,$(R,R').weight=\prod_{A\in Att(R,R')}|A|$.其中,$Att(R,R')$ 表示 R 和 R' 之间的等连接属性集合.$(R,R').lock$ 的类型为结点集合类型,其值由算法 *set_lock* 确定.

定义 3. 对于边 (R,R'),若 $(R,R').lock=\{\}$,则称 (R,R') 为自由边;否则,称上锁边.若 $(R,R').lock=\{R\}$,则称边 (R,R') 被上了结点 R 锁.

对于结点 R 及其关联的边 e 和 e',若 e before e',则 e' 对应的 join 操作必须滞后于 e 对应的 join 操作,即便是 $cost(e)<cost(e')$ 时也是如此.这就克服了 G_{MC} 的局部选优所带来的不足.为此,通过对 e' 边上 R 锁,使得 e' 成为非自由边,达到禁止 G_{MC} 优先考虑对 e' 的选择的目的.由于一条边只与两个结点关联,因此,一条边最多被上两把锁.对与 R

关联的边设锁的算法如下：

```
algorithm set_lock(R)
begin
  e:=ε; /* 变量 e 的类型为边类型. ε 为特殊边,对任意边 e' 有 e' before ε */
  L:={}; /* 变量 L 的类型为边集合类型. 它用于存放与 e 有 equal 关系的边 */
  for all 与 R 相邻的结点 R' do
  if (R,R') before e then
    begin
      for all 边(R,Rx)∈L do /* 对存放在 L 中的边上锁. 因为(R,R') before e,则有(R,R') before (R,Rx)(见定理 3) */
      (R,Rx).lock:=(R,Rx).lock+{R};
      L:={(R,R')}; e:=(R,R'); (R,R').lock:=(R,R').lock-{R};
    end
       else
    if (R,R') equal e then
      begin
        L:=L+{(R,R')};(R,R').lock:=(R,R').lock-{R};
      end
        else (R,R').lock:=(R,R').lock+{R}; /* 当 e before (R,R'),对 L 中的任意边 e',必有 e' before (R,R')(见定理 4)
        */
  return
end set_lock
```

以下是改进的 G_{MC} 算法.

```
algorithm improved_GMC(G) /* G=(V,E) */
(1) 用 set_lock 算法对图 G 中所有的结点所关联的边设锁:
    for all R∈V do
    set_lock(R);
(2) 在图 G 中的自由边中,根据 GMC 选择 join 操作:
    m:=最大值;
    e:=ε; /* 同 set_lock 算法 */
    for all (R,R')∈E do
    if  (R,R').lock={} and cost(R,R')<m then
        begin
        m:=cost(R,R'); e:=(R,R')
        end
(3) 若 e=ε,则转(7);
(4) 根据第(2)步最终选择的边 e(设为(R,R')),确定 R join R'为所选操作. 修正图 G:
    产生新结点 Rnew;
    Rnew.value:=R.value×R'.value/(R,R').weight; /* 计算新关系的大小 */
    V:=V+{Rnew}; /* 向图 G 加入新结点 */
    for all 与 R 或 R'相邻的结点 Re do /* 向图 G 中加入与新结点 Rnew 关联的边 */
    begin
      E:=E+{(Rnew,Re)};(Rnew,Re).lock:={};
      if Re 与 R 和 R'均相邻 then (Rnew,Re).weight:=(R,Re).weight×(R',Re).weight
              else
        if Re 与 R'相邻 then (Rnew,Re).weight:=(R',Re).weight
                      else (Rnew,Re).weight:=(R,Re).weight;
                      end;
                      从图 G 中消去结点 R,R'及其与它们关联的边;
(5) 对与 Rnew 关联的边设 Rnew 锁;由于 Rnew 的产生,与其相邻的结点所关联的边之间的 before 关系和 equal 关系有可能变化,因
    此,需重新设锁:
    set_lock(Rnew);
    for all 与 Rnew 相邻的结点 Re do
     set_lock(Re);
(6) 转(2);
(7) 结束.
```

定理 5. improved_$G_{MC}(G)$算法对图 G 的每一个结点 R 执行 set_lock(R)不会导致图 G 中的所有边被锁住.

证明：在图 G 中任意选取一条边(R_1,R_2). 若$(R_1,R_2).lock=\{R_1,R_2\}$，则可在 R_2 关联的边中找到一条边(R_2,R_3)，有$(R_2,R_3).lock=\{R_3\}$或$\{\}$. 若为$\{\}$，则存在没有锁住的边；若为$\{R_3\}$，我们可按如下方法继续搜索：

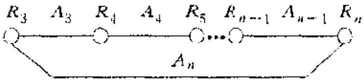

图4

(1) $i \leftarrow 3$;

(2) 搜索关联于结点 R_i 的边(R_{i+1},R_i),R_i 对边(R_{i+1},R_i)没有上锁或(R_{i+1},R_i)是自由边;

(3) 若(R_{i+1},R_i)是自由边,搜索结束;若$(R_{i+1},R_i).lock=\{R_{i-1}\}$,则 $i \leftarrow i+1$,转第(2)步.

上述搜索过程不会陷入无限循环之中,它会找到自由边而终止搜索.因为:由于图 G 中的结点个数是有限的,因此,搜索要进入无限循环的条件是:结点 $R_3,R_4,\ldots,R_n$ 必构成回路(如图 4 所示),且$(R_3,R_4).lock=\{R_4\}$,$(R_4,R_5).lock=\{R_5\}$,…,$(R_{n-1},R_n).lock=\{R_n\}$,$(R_n,R_3).lock=\{R_3\}$.由该条件,可获得如下不等式组:

$$\begin{cases} |R_3|/|A_3| > |R_4|/|A_4| \\ |R_4|/|A_4| > |R_5|/|A_5| \\ \quad\vdots \\ |R_{n-1}|/|A_{n-1}| > |R_3|/|A_n| \\ |R_n|/|A_n| > |R_3|/|A_3| \end{cases}$$

将不等式组中的不等式的左部和右部分别相乘后,得出不等式:$\prod_{k=3}^{n}|R_k|/\prod_{k=3}^{n}|A_k| > \prod_{k=3}^{n}|R_k|/\prod_{k=3}^{n}|A_k|$.由于该不等式不可能存在,因此,图 4 中的回路必不存在. □

简要的算法复杂度分析略.读者不难看出算法的复杂度为 $O(n^2)$.

4 结束语

通过分析文献[1]给出的计算代价模型和估算 join 操作所产生关系的大小的模型,给出了模型所具有的一些特征.借助于该特征,提出了能进一步降低 MJ 查询计算代价的优化方法.该优化方法与文献[1]提出的 G_{MC}和 G_{MR}优化方法相比,它们的不同之处在于:后者只考虑了局部选优这种传统性的方法,而前者是在局部选优的过程中,还考虑了优选出来的 join 操作应与下一次优选出来的 join 操作在执行次序上应是最优的.也就是说,后者不光从局部考虑选优,还从全局上考虑了任意两个有依赖关系的 join 操作所构成的操作序列是最优的.另外,由于两者的复杂度均为 $O(n^2)$,因此,前者的优化方法较后者更优.

参考文献

1 Chen M S, Yu P S, Wu K L. Optimization of parallel execution for multi-join queries. IEEE Transactions on Knowledge and Data Engineering, 1996,8(3):416～428

2 Tay Y C. On optimality of strategies for multiple joins. In: Proceedings of the 9th ACMSIGACT-SIGMOD-SIGART Symposium on Principle of Database Systems. 1990. 124～131

3 Swami A, Gupta A. Optimization of large join queries. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. 1988. 8～17

4 Swami A. Optimization of large join queries: combining heuristics and combinatorial techniques. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. 1989. 367～376

5 Wolf J L, Dais D M, Yu P S. A parallel sort merge join algorithm for managing data skew. IEEE Transactions on Parallel and Distributed Systems, 1993,4(1):70～86

6 Chen M S, Yu P S. Combining join and semijoin operations for distributed query processing. IEEE Transactions on Knowledge and Data Engineering, 1993,5(3):534～542

An Improved Optimizing Method for Multi-join Queries

ZHONG Wu HU Shou-ren

(Department of Computer Science Changsha Institute of Technology Changsha 410073)

Abstract M. S. Chen has put forward heuristics G_{MC} and G_{MR}, which are used to produce a join bushy tree with less total cost. On the basis of his work, the paper gives an improved algorithm with complexity of $O(n^2)$, by means of analysing relationship between the order of join operations and computing costs. The algorithm can reduce more total cost of a join bushy tree than G_{MC} and G_{MR}, which benefits from the following: the operation order of two arbitrary adjacent internal nodes (join operations) is optimum.

Key words Relational database, multi-join queries, query optimization, parallel execution, executinon dependency.

Class number TP311.13