重庆大学学报
第12卷 第2期 JOURNAL OF CHONGQING UNIVERSITY vol.12 №2.1989

偶查询优化的图论方法

OPTIMIZATION OF BIPARTITE JOINS WITH GRAPH THGORY

曾永宁　　童　富
Zeng Yongning　Tong　Fu
（计算机系）

摘　要　本文针对一类特殊的多关系查询——偶查询，提出了一种建立在图论、偶图和匹配理论基础上的查询优化方法，这种方法具有多项式复杂性。

关键词　数据库；查询处理；查询优化；算法；图论；偶图；配对

ABSTRAC　An optimization approach for multi-relational joins based on the matching concept in graph theory is introduced. The basic idea lies in constructing a join graph from an expression of N-relation query, and seeking for a maximun matching with minimun total weight. The algorithm for bipartition(X, Y) is proposed in this paper.

KEY WORDS　database; algorithm; graph theory; bipartite graph; matching; query processing; query optimization

一、引　言

关系数据库的查询处理效率极大地影响数据库的系统性能。因此，查询优化问题受到了广泛的重视，不少人对它进行了深入广泛的研究〔2、3、4、5、6、7、8、9、10、11、13〕。其研究结果表明查询优化问题是NP一完全的〔1〕。为了寻找有效的优化方法，通常把查询分为树查询和环查询两类。所谓树查询是指这类查询所对应的查询图是树；而环查询所对应的查询图中包含着回路。对树查询已经找到到了具有多项式复杂性的优化方法，但对环查询还没有找到有效的优化方法。

本文从图论的角度出发，把查询分为偶查询和非偶查询两类。所谓偶查询是指其查询所对应的查询图是偶图（Bipartite Groph）。不难证明树查询属于偶查询类，而且一些环查询也属偶查询类。我们应用图论的匹配（Mathinp）理论，导出了一个有效的偶查询优化算法。本文的第二节介绍图论有关的概念，给出了若干新的定义；第三节讨论估算多关系查询

本文于1988年1月23日收到。

处理开销的方法；第四节给出了偶查询优化算法。

二、预 备 知 识

定义2.1：设$D=(R_1, R_2, \cdots, R_n)$为数据库模式，$Q=(q=q_1 \wedge q_2 \cdots \wedge q_n)$为查询表达式，$q$为查询约束条件。查询图$G_q$是一个有序三元组$(V_q, E_q, \varphi_q)$。其中$V_q=\{R_1, R_2, \cdots, R_n\}$为顶点集。$E_q$代表边集，$\varphi_q$是映射函数，它确定了边集$E_q$和顶点集$V_q$间的映射关系：$\varphi_q(e)=R_iR_j$当且仅当$R_i$和$R_j$在$Q$中是可联接的。例如，设$Q=(R_1 \cdot A=R_2 \cdot B) \wedge (R_1 \cdot C=R_3 \cdot D) \wedge (R_2 \cdot E=R_4 \cdot F) \wedge (R_2 \cdot G=R_5 \cdot H)$为一查询表达式，它的查询图如图2.1所示。

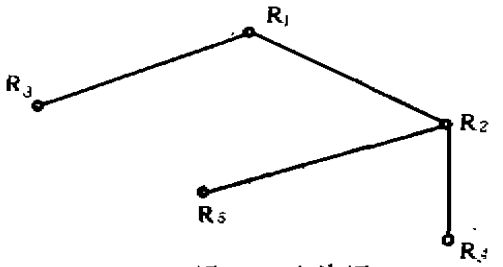

图 1 查询图

定义2.2：联接图G_j是一个带权查询图。即$G_j=(V_q, E_q, \varphi_q, W_q)$其中$W_q$是边的权集：

$$W_q=\{w_{ij} \mid \text{对所有的边} R_iR_j \in E_q\}$$

其中w_{ij}是边R_iR_j所带的权。

定义2.3[6]：设M为边集E_j的一个子集，如果M满足没有任何两条M中的边是相邻的。则称M为E_q的一个匹配（Matching），并称M中任一边所关联的两个顶点在M下是配对的。

若匹配M的某条边与顶点R_i关联，则称M饱和顶点R_i，并称R_i是M饱和的，否则R_i是M非饱和的。若G_j的每个顶点为M饱和的，则称M为G_j的完全匹配。若G_j没有另外的匹配M'，使$|M'|>|M|$成立，则称M为G_j的最大匹配。

设M是G_j的匹配，G_j的M交错路经是指其边在$E_q \backslash M$和M中交错出现的路径。M可扩路径是指其起点和终点都是M非饱和的M交错路径。

定理2.1：G_j的匹配M是最大匹配，当且仅当G_j不含M可扩路径。

定义2.4[6]：若图G_j的顶点可划分为两个子集X和Y，使得G_j的任一条边的两个顶点总是分属X和Y，则称图G_j为偶图（Bipartite Graph），而(X, Y)称为G_j的一个二分类。

完全偶图是一个具有二分类(X, Y)的简单偶图，并且X的任一个顶点总是和Y的每一个顶点相关联。

定义2.5：如果一个查询Q所对应的查询图G_j是一个具有二分类(X, Y)的偶图，则称Q为偶查询。

树查询属于偶查询类，因为根据图论，树总可转换为偶图。有一些环查询，也可以转换为偶查询。例如，对查询图为如图2.2(a)所示的一个环查询，不难把它转换成其查询图为如图2.2(b)所示的偶查询。

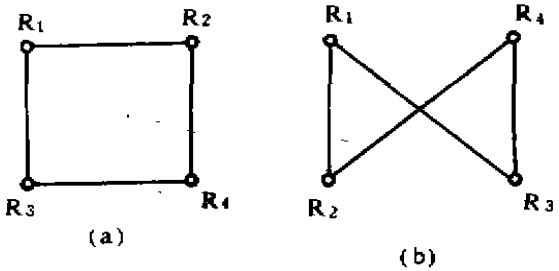

图 2 环查询向偶查询的转换

$$X=\{R_1, R_4\}$$
$$Y=\{R_2, R_3\}$$

定义2.6：设S为联接图G_j的顶点集V_q的一个子集。S的邻集$V_n(s)$是由所有S相邻的顶点构

成的顶点集。

若G_i是一个具有二分类（X，Y）的偶图，我们希望找到一个能够饱和X的每一个顶点的匹配，存在这样的匹配的充要条件由下面的定理给出。

定理2.2：设G_i是具有二分类（X，Y）的偶图，G_i包含饱和X的每一个顶点的匹配的充要条件是

$$|V_n(S)_i| \geqslant |S| \qquad \text{对所有的} S \subset X$$

三、查询处理开销评估

用$\phi(R_i \cdot A)$表示关系R_i的属性A的值域。$R_i[A]$表示关系R_i在属性A上的投影。$R_i[A]$是$\phi(R_i \cdot A)$的一个随机子集。所有值域的集合构成了一个值域层次结构。每一个层次由一个最大值域ϕ_{max}和若干值域ϕ组成，满足$\phi \subset \phi_{max}$。如果值域ϕ_i和ϕ_j同属一个层次，则称ϕ_i和ϕ_j是可联接的。

下面给出三个假设作为讨论的前提。

（1）　如果$\phi_i \subset \phi_j$，则ϕ_i是ϕ_j的随机子集；

（2）　关系R_i的元组在$R_i[A]$上是均匀分布的；

（3）　用$\mathrm{attr}(R_i)$表示R_i的属性集，则对关系R_i和属性$\{A, B\} \subset \mathrm{attr}(R_i)$，$R_i[A]$和$R_i[B]$是相互独立的。

由以上假设，可导出任一元素$x \in \phi_{max}$满足$x \in R_i[A]$的概率$p(R_i \cdot A)$为：

$$p(R_i \cdot A) = \eta(R_i[A]) / \eta(\phi_{max}) \qquad (1)$$

其中，$\eta(R_i[A])$表示$R_i[A]$的不同值的数目。$\eta(\phi_{max})$表示ϕ_{max}的不同值的数目。

一般地，多关系查询的约束条件可表示为$q = q_1 \wedge \cdots \wedge q_m$。$q_k (k = 1, 2, \cdots, m)$的形式为：$R_i \cdot A\theta v$，或$R_i \cdot A\theta R_j \cdot B$，$\theta \in \{=, <, \leqslant, >, \geqslant, \neq\}$，$v$是一个算术表达式。$R_i \cdot A\theta v$是单变量约束，不少文献已对它作了深入的讨论[12]。本文只讨论联接约束$R_i \cdot A\theta R_j \cdot B$。

多关系查询处理的开销主要取决于在查询处理过程中访向存储页面的总数。一旦数据库建立起来后，各个关系所占用的存储页面总是可计算的。设N_i，t_i分别为关系的元组数和元组长度。P_i为关系R_i占用的存储页面。则：

$$P_i = N_i t_i / b \qquad (2)$$

b表示页面大小。

另一方面，在多关系查询处理过程中，要形成若干中间临时关系，这些临时关系所占用的总页数随着选择执行二元联接的次序的不同而变化。每一个约束子条件为$R_i \cdot A\theta R_j \cdot B$的子查询$Q'$都要形成一个临时关系$R_{ij}$。处理$Q'$的联接开销$C_{ij}$取决于关系$R_i$、$R_j$的大小和联接算法。然而，当我们从一个多关系联接表达式确定二元联接的次序时，不仅要考虑二元联接的开销C_{ij}，而且也要考虑形成的中间临时关系R_{ij}的大小。因为后者对多关系查询处理的总开销有很大的影响。因此，我们把联接图G_j各边的权W_{ij}定义为：

$$W_{ij} = C_{ij} + P_{ij} \qquad (3)$$

而

$$P_{ij} = N_{ij} t_{ij} / b \qquad (4)$$

其中N_{ij}是R_{ij}的元组数，它是未知的。下面我们来讨论N_{ij}的计算方法。

首先定义R_{ij}的选择因子。

定义3.1：临时关系R_{ij}的选择因子p_{ij}是由关系R_i和R_j的元组构成的元组对满足约束子条件$q_{ij}=R_i\cdot A\theta R_j\cdot B$的概率。

因此只要p_{ij}已知，N_{ij}容易由下述公式求得：

$$N_{ij}=p_{ij}N_iN_j。 \tag{5}$$

在相等联接（equijoin）的情况下，$p_{ij}=p(R_i\cdot A)\cdot p(R_j\cdot B)$。这时（5）式变为：

$$N_{ij}=p(R_i\cdot A)p(R_j\cdot B)N_iN_j \tag{6}$$

为了在多关系查询处理过程中，重新计算压缩后的联接图的各边的数，需要估算联接属性$R_{ij}\cdot AB$的不同值的数目$\eta(R_{ij}[AB])$，以及R_{ij}的其他属性$R_{ij}\cdot A'$的不同值的数目$\eta(R_{ij}[A'])$。

$$\eta(R_{ij}[AB])=p(R_i\cdot A)p(R_j\cdot B)\eta(\phi_{max}) \tag{7}$$

然而，估算$\eta(R_{ij}[A'])$较困难。这里我们利用文献[5]提出的一个近似公式：

$$\eta(R_{ij}[A'])=\begin{cases}N_{ij} & \text{当 } N_{ij}<\frac{1}{2}\eta'(R_{ij}[A'])\\ 1/3(N_{ij}\cdot\eta'(R_{ij}[A'])) & \text{当}\frac{1}{2}\eta'(R_{ij}[A])<N_{ij}<2\eta'(R_{ij}[A'])\\ \eta'(R_{ij}[A']) & \text{当}2\eta'(R_{ij}[A'])<N_{ij}\end{cases} \tag{8}$$

其中

$$\eta'(R_{ij}[A'])=\begin{cases}\eta(R_i\cdot A') & \text{当}R_{ij}\cdot A'=R_i\cdot A'\\ \eta(R_j\cdot A') & \text{当}R_{ij}\cdot A'=R_j\cdot A'\end{cases} \tag{9}$$

四、偶查询优化算法

Kuhn-Munkres曾提出了一个有效的算法[6]，在一个具有二分类(X,Y)，$X=\{x_1、x_2、\cdots、x_n\}$，$Y=\{y_1、y_2、\cdots、y_n\}$的带权完全偶图上寻找具有最大权的完美匹配。我们汲取了Kuhn-Munkres算法的基本思想，提出了在具有二分类(X^*,Y^*)，$X^*=\{Rx_1,Rx_2,\cdots,Rx_k\}$，$Y^*=\{Ry_1,Ry_2,\cdots,Ry_r\}$的带权偶图$G_i$上寻找具有最小权的最大匹配的算法，从而导出了偶查询优化算法。该算法从以下三点对Kuhn-Munkres算法作了修改：

（1） 取消了带权偶图必须是完全图的限制；

（2） 我们的算法是求具有最小权而不是最大权的最大匹配；

（3） 放宽了对二分类的限制。X的元素数目可以不等于Y的元素数目。即$k\leqslant r$。

1. 偶查询优化算法的基本思想

首先从查询表达式构造联接图G_i。并根据式（1）—（9）计算各边的权。然后在图上寻找一个具有最小权的最大匹配M。M中的边代表了一组相互独立的二元联接操作。然后，把M中的边压缩为一个顶点，G_i变为一个新的联接图G_i'，G_i'的边数和顶点数都要比G_i少。重新计算各边的权。重复上述过程，直到G_i变成一个平凡图（只有一个顶点的图）。至此我们得到了一个二元联接的最优次序，算法终止。

不难看出，算法的关键在于寻求具有最小权的最大匹配（以下简称最佳匹配）。下面介绍寻找最佳匹配的算法的思想。

设G_2为具有二分类(X^*, Y^*)的带权偶图。$X^* = \{Rx_1, Rx_2, \cdots, Rx_k\}$，$Y^* = Ry_1, Ry_2, \cdots, Ry_r\}$，$k \leqslant r$。$G_1$的每条边的权$W_{ij}$为：

$$W_{ij} = C_{ij} + p_{ij} \tag{10}$$

然后人为地在G_1上加若干虚边，使之成为一个完全偶图。这些虚边的权定义为：

$$W'_{ij} = W_{max} + 1 \tag{11}$$

$W_{max} = max\{W_{ij}$ 对所有的 $e(Rx_i, Ry_j) \in E_1\}$。有了以上假想，在以后的讨论中，都把G_1看成一个完全偶图。

定义4.1：若在顶点集$X^* \cup Y^*$上的实值函数L满足下述条件：对所有的$R_x \in X^*$，$R_y \in Y^*$，均有

$$L(R_x) + L(R_y) \leqslant W(R_x R_y) \tag{12}$$

则把这个函数定义为图G_1的一个可行顶点标号。实数$L(v)$称为顶点v的标号。可行顶点标号是这样的标号，它使每条边的两个顶点的标号之和不会超过这条边的权。不管边的权是什么，总存在一个可行顶点标号：

$$\begin{aligned} L(R_x) &= min\ w(R_x R_y) \quad 若 R_x \in X^* \\ L(R_y) &= 0 \qquad\qquad\quad 若 R_y \in Y^* \end{aligned} \tag{13}$$

若L是可行顶点标号，则用El表示使（12）式等式成立示的那些边的集。即：

$$El = \{R_x R_y \in E_2 \mid L(R_x) + L(R_y) = w(R_x R_y)\} \tag{14}$$

具有边集El的生成子图不妨称为对应于可行顶点标号的相等子图，用Gl表示。相等子图Gl与最佳匹配M间的联系由下述定理给出。

定理4.1：设L是G_1的可行顶点标号，若Gl包含最大匹配M^*，则M^*是G_1的最佳匹配。

证：假设G_1包含最佳匹配。由于G_1是G_1的生成子图，所以M^*也就是G_1最佳匹配。于是

$$w(M^*) = \sum_{e \in M^*} w(e) = \sum_{v \in V_2} l(v) \tag{15}$$

这是因为每个边$e \in M^*$都属于这个相等子图，并且M^*的边的端点覆盖V_2的每个顶点恰好一次。另一方面，若M是G_1的任一最佳匹配，则有：

$$w(M) = \sum_{e \in M} w(e) \leqslant \sum_{v \in V_2} V_2 l(v) \tag{16}$$

从（15）式和（16）式推出$w(M^*) \geqslant w(M)$。于是M^*是最佳匹配，证毕。

下面叙述最佳匹配算法的思想。

首先给出任一可行顶点标号l，然后决定G_1。在G_1中选择任一匹配M，并且利用匈牙利方法[6]，若在G_1中已经找到一个最大匹配，则由定理4.1，该匹配就是最佳的。否则匈牙利方法将终止于一个非完全的匹配M'和一棵既不包含M'可扩路径，又不能在G_1中进一步生长M'的交错树H。随后，把L修改为具有下述性质的另一个可行顶点标号L'：M'和H都包

含在G_i'中，并且H能够在Gl'中伸展。每当必要时，连续不断地进行这种可行顶点标号的修改，直到一个最大匹配在某个相等子图中找到为止。

2. 寻找最佳匹配的算法OPTIM

从任一可行顶点标号开始，然后决定G_1，并且在G_1中选取任一匹配M。

1）. 若X^*是M饱和的，则M是最大匹配，并由定义4.1可知M是最佳匹；转到第4步。否则，令$u\in X^*$是一个M非饱和顶点，置$s=\{u\}$，$T=\phi$。

2）. 若$V_n(s)\supset T$，则转到第3步。否则，$V_n(s)=T$，计算：

$$dl=\min_{\substack{R_x\in S\\R_y\in T}}\{\omega(R_xR_y)-(L(R_x)+L(R_y))\} \tag{17}$$

且由

$$L'(v)=\begin{cases}L(v)+dl & \text{若}v\in S\\ L(v)-dl & \text{若}v\in T\\ L(v) & \text{其它}\end{cases} \tag{18}$$

给出可行顶点标号L'（注意$dl>0$，且$V_n(S)\supset T$）。以L'代替L，G_i'代替G_i。

3）. 在$V_n(S)\backslash T$中选择一个顶点v'，考察v'是否M饱和。若v'是M饱和的，则一定存在某条边$e\in M$，其关联的两个顶点为v'和$Z\in X^*$，用$S\cup\{Z\}$代替S，用$T\cup\{v\}$代替T，再转到第2步。否则，设P是G_i中的M可扩$(u,\ v')$路，用$M'=M\Delta E(P)$代替M，并转到第1步。

4）. 从M中删除那些具有权为$W_{ij}=W_{max}+1$的边。算法终止。

下面来分析算法OPTIM的复杂性。

定理4.2：算法OPTIM的时间复杂性为$O(r^4)$。

证：在算法的第2步，计算Gl所需的计算次数显然是$|V_0|^2=(r+k)^2$级的。由于在找出M可扩路之前，算法的循环最多可能有$|X|$次。所以算法总的计算次数为

$$|V_0|^2*|X|^2=(r+k)^2k^2=k^4+2rk^3+r^2k^2 \tag{19}$$

所以，算法的复杂性为$O(r^4)$。

3. 偶查询优化算法GEN

在第1节已经介绍了偶查询优化算法的基本思想，下面给出算法描述。

```
ALGORITHM GEN
INPUT: G_i
OUTPUT: An Optimal Join Order.
    1. WHILE G_i NOT a isolated vertex
       BEGIN
    2. OPTIM(G_i);
    3. WHILE M NOT NIL
       BEGIN
    4. FOR all e(u, v)∈M, REDUCE(e(u, v))
    5. M = M - {e(u, v)};
       END
```

6. COMPUT-WC(G_i);
   END
7. END of ALGORITHM。

过程REDUCE把M中的每条边压缩为一个顶点，并去掉新形成的联接图G_i'中的重复边；过程COMPUT-W根据式(1)～(9)重新计算边的权。计算时，只有那些受到影响的边的权才需重新计算。

五、结　语

本文针对一类特殊的多关系查询——偶查询，提出了一种有效的优化方法。这种方法是建立在图论的偶图和匹配理论基础上的。把确定最优次序问题转化为在一个带权的偶图上寻求一个具有最小权的最大匹配问题。算法具有多项式复杂性。

参考文献

[1] Aho A V, Sasiv Y, Uliman J D. Efficient Optimzation of A Classof Relational Expression. ACM trans, Database Syst, 1979; 4(4): 435～454

[2] Apers P M G, Hevner A R, Yao S B. Optimization Algorithms for Distributed Queries. IEEE trans. on Soft. Eng., 1983; 9(1): 57～68

[3] Bernstein P A, Chiu D W. Using Semi-joins to Solve Relational Queries. J. ACM, 1981; 28(1): 25～40

[4] Bernstein P A etc. Query Processing in a System for Distributed Databases (RDD-1). ACM trans. Databases Syst., 1981; 6(4): 602～625

[5] Bondy J. A, Murty U R. Graph Theory with Applications. The MACMILLAN Press ltd, 1976;

[6] Chandra A K, Merlin P M. Optimal Implementation of Conjunctive Queries in Relatioal Data Bases. Proc. Ninth Ann. ACM Symp. on Theory of Uompt., 1976; 77～90

[7] Epstein R, Stonebraker M. Distributed Query Processing in A Relational Data Base System. Proc. of ACM-SIGMOD, 1978; 169～180

[8] Goodman N, Shmueli O. Tree Queries: A Simple Class of Relationl Queries. ACM trans. Database Syst., 1982; 7(4): 653～677

[9] Ibaraki T, Kameda T. On the Optimal Nesting Order for Computing N-Relational Joins. ACM trans. Database Syst., 1984; 9(3): 428～502

[10] Selinger P G, Adiba M. Access Path Selection in Distributed Data Management System. Proc. Conf. Databases, 1980; 204～215

[11] Stonebraker M, Wang E, Kreps P. The Design and Implementation of INGRES ACM trans. Database Syst., 197(; 1(3): 189～222

[12] Wong E, Youssefi K. Decomposition-A Strategy for Query Processing. ACM trans.Databases Syst., 1976; 1(3): 223～241