基于遗传算法的多连接表达式并行查询优化*

曹 阳, 方 强, 王国仁, 于 戈

(东北大学 信息科学与工程学院,辽宁 沈阳 110004)
E-mail: {wanggr,yuge}@mail.neu.edu.cn
http://www.neu.edu.cn

摘要: 多连接表达式的并行查询优化是提高数据库性能的关键问题之一.提出了使用遗传算法来解决多连接表达式的并行查询优化问题.为了提高查询处理器的执行效率,采用启发式规则来搜索最优的多连接表达式并行调度执行计划.文中给出了详细的测试结果和性能分析.实验结果表明,结合启发式知识的遗传算法是解决多连并行查询优化的有效途径,对提高数据库的性能起到重要作用.

关 键 词: 遗传算法;多连接表达式;查询优化;并行调度
中图法分类号: TP311 文献标识码: A

多连接表达式的并行查询优化涉及到多个连接的连接顺序问题,还和处理机的有效分配密切相关,因此可能存在相当多的并行执行计划供选择.到目前为止仍未有令人满意的通用方法能快速找到计算多连接表达式最小代价的并行执行计划.本文使用遗传算法求解多连接表达式的连接顺序,为兼顾查询优化器的处理效率,结合启发式方法,进行并行查询优化.

在并行处理机平台上寻找多连接表达式的并行查询最优执行计划可分为两个步骤:(1) 如何构造一棵代价最小的多连接查询树,(2) 如何为该多连接查询树分配处理机.代价最小的多连接查询树的查找可以通过枚举法、启发式算法、搜索算法和遗传算法构造多连接树.枚举法通过解空间中的每个可能解来找到最优解.当解空间增大时求解效率相当低.启发式算法则是寻求一种能产生可行解的启发式规则来找到一个最优解或近似最优解.但该方法在不同的限定条件,不同的并行环境和体系结构要有不同的启发式规则,不具有通用性,同时不能保证所得的结果最优[1].搜索算法在可行解集合的一个子集内进行搜索,搜索算法的随机性和盲目性不能保证得到最优解,该算法结合一些启发知识才能在一定程度上达到要求.遗传算法是模拟生物在自然环境中的遗传和进化过程而形成的一种自适应全局优化概率搜索算法[2].它将原问题的解空间映射到位串空间中,然后再实施遗传操作,强调个体基因结构的变化对其适应度的影响.现有遗传算法主要存在下面问题:仅考虑使用简单的遗传算法;对演化计算的其他方法没有加以讨论和改进;只涉及到多连接表达式的执行顺序,没有进一步研究多处理机的并行调度问题.

为多连接树分配处理机是复杂的组合优化问题,涉及的限制条件包括每个连接的负载大小、连接关系的物理存储情况、连接操作间的依赖、并行资源以及调度策略本身的性能等.文献[3]给出了 4 种连接树的处理机分配策略:顺序并行执行策略(SP)、同步执行策略(SE)、分段右深树的执行策略(RD)和完全并行的执行策略(FP).

顺序并行执行策略(SP).SP 方法只有连接内的并行操作,没有连接间的并行操作,把所有的处理机全部分给一个连接操作,操作完成之后再分配给下一个连接操作.

* 收稿日期: 2001-04-20; 修改日期: 2001-09-24
基金项目: 国家教育部高等学校骨干教师资助项目;教育部高等学校优秀青年教师教学和科研奖励基金资助项目;教育部跨世纪人才基金资助项目

作者简介: 曹阳(1976－),女,辽宁大连人,硕士,主要研究领域为并行数据库技术;方强(1975－),女,湖北武汉人,硕士,主要研究领域为并行数据库技术;王国仁(1966－),男,湖北崇阳人,博士,教授,主要研究领域为并行数据库,对象数据库,Web 数据库技术;于戈(1962－),男,辽宁大连人,博士,教授,博士生导师,主要研究领域为数据库理论和技术,分布式系统.

同步执行策略(SE).SE 方法也称为自上而下的分配策略,把处理机同时分配给彼此独立的不同的连接操作,利用了操作内的并行性和操作间的并行性.并行度和处理机的利用率较高,适用于大型的并行系统[4].

分段右深树的执行策略(RD).RD 方法利用管道的并行性,对于连接树上的右深子树用流水线的方式并行处理,其他部分用 SP 方法处理.

完全并行执行策略(FP).FP 方法是 SE 和 RD 方法的结合,把处理机按一定比例分配给每个连接结点,兼顾操作间的并行性和流水线的并行性.

根据相关工作和上面提出的问题,可以使用遗传算法选出一些相对较优的多连接表达式的连接操作符树,然后根据限定条件和不同调度策略的启发式算法,找到代价最小的并行执行计划.本文有关并行调度的讨论和代价估算是基于无共享资源的体系结构.

本文第 1 节介绍遗传算法.第 2 节介绍基于遗传算法的多连接表达式的查询优化.第 3 节介绍了结合启发式规则的多连接并行查询优化技术.第 4 节是仿真实验的设计和性能评价.第 5 节小结.

1 遗传算法

1.1 遗传算法简介

20 世纪 60 年代中期,美国 Michigan 大学的 John Holland 从对生物例子的研究中提出了遗传算法.它的基本思想是基于 Darwin 的进化论和 Mendel 的遗传学说.由于该算法具有很好的适应性,尤其适合解决组合优化及规则获取等非线性问题,因而被应用于许多不同的领域.

用遗传算法求解最优化问题时,首先对可行域中的点进行编码,然后在可行域中随机挑选一些编码组成作为进化起点的第一代编码组,并计算每个编码组的适应度.接着利用选择机制从编码组中随机挑选编码作为繁殖过程前的编码样本.选择机制应保证适应度高的解能够保留较多的样本;而适应度低的解则保留少的样本,甚至被淘汰.在繁殖过程中,使用交叉和变异两种算子对挑选后的样本进行交换,产生下一代编码组.重复上述选择和繁殖过程,直到结束条件得到满足为止.进化过程最后一代中的最优解就是用遗传算法解最优化问题所得到的最终结果.

遗传算法具有下述特点:(1)针对问题参数的编码组进行操作,而不是直接对参数本身.(2)从问题解的编码组开始搜索,而不是从单个解开始.(3)使用目标函数值(适应度)进行搜索,而不需其他专业信息.(4)使用的选择、交叉、变异这 3 个算子都是随机规则,而不是确定规则进行搜索.

1.2 遗传算法的形式化定义

遗传算法可形式化定义为一个 8 元组:$SGA=(C, E, P_0, M, \Phi, \Gamma, \Psi, T)$,式中的 C––个体的编码方法,E––个体的适应度评价函数;P_0––初始种群;M––群体大小;Φ––选择算子;Γ––交叉算子;Ψ––变异算子;T––遗传运算终止条件.

1.3 遗传算法的构成要素

(1) 染色体编码算法

把一个问题的可行解从其解空间转换到遗传算法所能处理的搜索空间的转换方法就称为编码.编码方法决定了个体的染色体排列形式,也影响到交叉算子、变异算子的运算方法.它在很大程度上决定了如何进行群体的进化以及遗传运算的效率.常用编码方法有三大类:二进制编码方法、浮点数编码方法和符号编码方法.

(2) 个体适应度评价

遗传算法中使用适应度来度量群体中各个体在优化计算中有可能达到或接近于最优解的优良程度,从而得到下一步的搜索信息.度量个体适应度的函数称为适应度函数.

(3) 选择算子

遗传算法使用选择算子对群体中的个体进行优胜劣汰操作.选择操作建立在对个体的适应度基础之上,主要目的是为了避免基因缺失、提高全局收敛性和计算效率.常用的选择算子的操作方法如下:比例选择、最优保存策略、无回放随机选择.

(4) 交叉算子

交叉运算是指按一定的交叉概率 p_c 随机的从种群中选择两个个体(染色体),部分交换其中的某些基因位,生成新的个体.交叉运算是产生新个体的主要方法,决定了遗传算法的全局搜索能力.目前,常用的有单点交叉、多点交叉和算术交叉.

(5) 变异算子

变异运算是按一定的变异概率 p_m 将个体的某些基因位用该基因座的其他等位基因来替换,从而形成一个新个体.变异运算是产生新个体的辅助方法,它决定了遗传算法的局部搜索能力.交叉算子与变异算子相互配合,共同完成对搜索空间的全局和局部搜索.常用的变异方式有基本位变异和均匀变异.

2 结合遗传算法的多连接表达式的查询优化

下面从遗传算法的编码、建立优化模型、运算算子设计和确定遗传算法的运行参数 4 个方面介绍用遗传算法解决多连接表达式的查询优化技术.

2.1 多连接表达式的编码

对于多连接表达式用连接树的形式表达不同的执行计划.由于连接树不仅涉及到不同的连接关系还有树的形状,不能简单的使用二进制编码方式.为保留连接树的特征,直接以树的形式编码.对任意连接树采用先根序遍历方法,非叶子结点用 0 表示,叶子结点用非 0 整数表示.

图 1 是以 5 个连接关系为例的表示法.这种编码方式便于对树进行各种变形操作,不需要将染色体转换成其他形式就可直接使用前面提到的 4 种调度策略对连接树分配处理机.该编码方式不同于 Steinbrunn 等人提出的编码方式,他们的编码方式不含有表示中间结点的 0,从染色体个体本身不能得到树的形状,不能直接对其进行处理机分配.

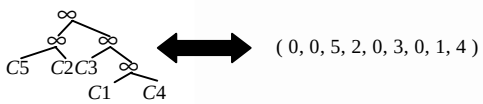

Fig.1 Join tree corresponds to non-zero integer

图 1 连接树与非负整数串一一对应

对于 N 个关系构成的连接表达式,随机产生长度为 $(2N-1)$ 的初始种群不一定能够构成二叉树,如 (0,0,1,2,3,4,5,0,0)就不能构成一棵二叉树.所以必须给出限定条件,产生适合遗传算法的搜索空间.

2.2 优化模型建立

问题的最终目标就是要找到代价最小的连接树,这实际上是求目标函数的最小值优化的问题.计算每个连接表达式的代价需要确定:连接关系的个数 N,连接关系的大小 C_i,可用处理机的数目 S,可用内存大小 $AvailMem$,并行调度策略 SS.因此代价函数可用 $Fcost(N, \Sigma C_i, S, AvailMem, SS)$ 来表示,也就是每棵连接树的适应度函数.

2.3 遗传算法的运算算子设计

遗传算子是对生物遗传和自然进化过程的模仿,实现对群体中个体的优胜劣汰.遗传算子主要有选择、交叉和变异 3 种算子[5].下面是针对多连接查询树设计的 3 种算子的运算规则.

(1) 选择算子

研究表明保留最佳个体策略的遗传算法收敛于最优解的概率为 1[5].采用最优保存选择策略,当前种群中代价最小的连接树不参加交叉和变异运算,代价大的个体被淘汰.这样使当前最优个体模式不会被交叉、变异运算破坏,保证了遗传算法的收敛.

(2) 交叉算子

因为每个个体表示不同的连接表达式的连接树型,简单的交叉会导致交叉后的个体不能构成一棵树,达不

到交叉产生新个体的目的.考虑树操作的特殊性,采用等大小子树交叉策略.随机产生等大小的子树进行交换,可较大的改变连接树型.对于交叉之后产生的重复基因要进行“修补”,将树上重复的部分以随机的方式改成其他未出现的基因(连接关系).具体描述如下:

```
Function CrossOver( ) {
随机为种群配对,
do{
        取出的一对个体(p_i,p_{i+1}),
    Psize --,    /*Psize 记录了交叉对的数目*/
        do{
        随机产生一个奇数 NUM (NUM<2N-1),
        HasSubTreeFlag1=findsubtree(p_i,NUM,subtree1),
        HasSubTreeFlag2=findsubtree(p_{i+1},NUM,subtree2),
        /*HasSubTreeFlag1 和 HashSubTreeFlag2 是能否含有规定数目的子树的标志*/
    }while(!HasSubTreeFlag1||!HasSubTreeFlag2),
    /*随机的寻找长度为 NUM 的子树 subtree1,subtree2*/
    exchange(p_i,subtree1,p_{i+1},subtree2),
}while(Psize>0)
}
```

(3) 变异算子

考虑到变异后的非负整数串需要满足构成一棵树的要求,将基本位变异的方法改进为交换位的操作,且交换的两位上的值必须大于零,从而保证树型不变,其描述如下:

```
Function mutation( ) {
do {
        随机取出一变异个体 p_i,
    Msize --;    /*Msize 是需要变异的个体数目*/
        do{
        num1=rand( );
        num2=rand( );
    }while(isZero(p_i,num1)|| isZero(p_i,num2)|| num1==num2 );
  /*找到连接树上的两个非零的位进行交换*/
  exchange(p_i,num1,num2);
}while( Msize>0 );
}
```

2.4 确定遗传算法的运行参数

遗传算法作为一种随机优化算法,影响性能的参数有很多,如:个体编码串长度 L,群体大小 pop_size,交叉概率 p_c,变异概率 p_m,终止条件 T 等.这些参数对性能的影响较大,需要认真选取.

(1) 编码长度 L.由连接关系的数目确定.对于 N 个关系的连接表达式,$L=2N-1$.

(2) 群体大小 pop_size.表示群体中所含个体的数量.当 pop_size 较小时,提高了遗传算法的运算速度却降低了群体的多样性,有时会引起遗传算法的早熟现象,当 pop_size 过大时,降低算法的运行效率.一般取值 20-100.

(3) 交叉概率 p_c.因为交叉操作是产生新个体的主要方法,决定了初始种群有多少个体参与交叉运算.所以 p_c 一般取值较大.仿真实验中 p_c=0.4.

(4) 变异概率 p_m.变异概率取值较大时,可以产生较多的新个体,但也可能破坏掉很多较好的模式,使得遗传

算法的性能近似于随机搜索算法的性能,若 p_m 取值较小,则变异操作产生新个体的能力和抑制早熟的现象的能力就会较差.一般建议取值范围为 0.0001~0.1,仿真实验中,大多取值为 0.1.

(5) 终止条件 T.在仿真实验中设计了两种结束条件:(a) 给定时间找到最佳解;(b) 给定进化代数,寻找最佳的并行调度策略.

3 采用启发式算法的多连接表达式的并行查询优化

计算连接树的代价需要知道每个结点左右子树的大小、中间结果的代价、结点之间的关系等.而遗传算法中的编码形式不能满足这样的要求,我们设计了代价树这一中间结构,把非负整数串转换成代价树的形式,把所需信息填到代价树中.

初始种群经过选择、交叉、变异运算之后每个个体转化为代价树,对于每棵代价树分别按照前面介绍的 4 种并行调度策略(SP,SE,RD,FP)计算代价,进行排序,保留代价小的个体,参与下一次的遗传操作.这样经过若干代之后可以得到最优的并行连接调度树即并行执行计划.整个优化过程如图 2 所示.

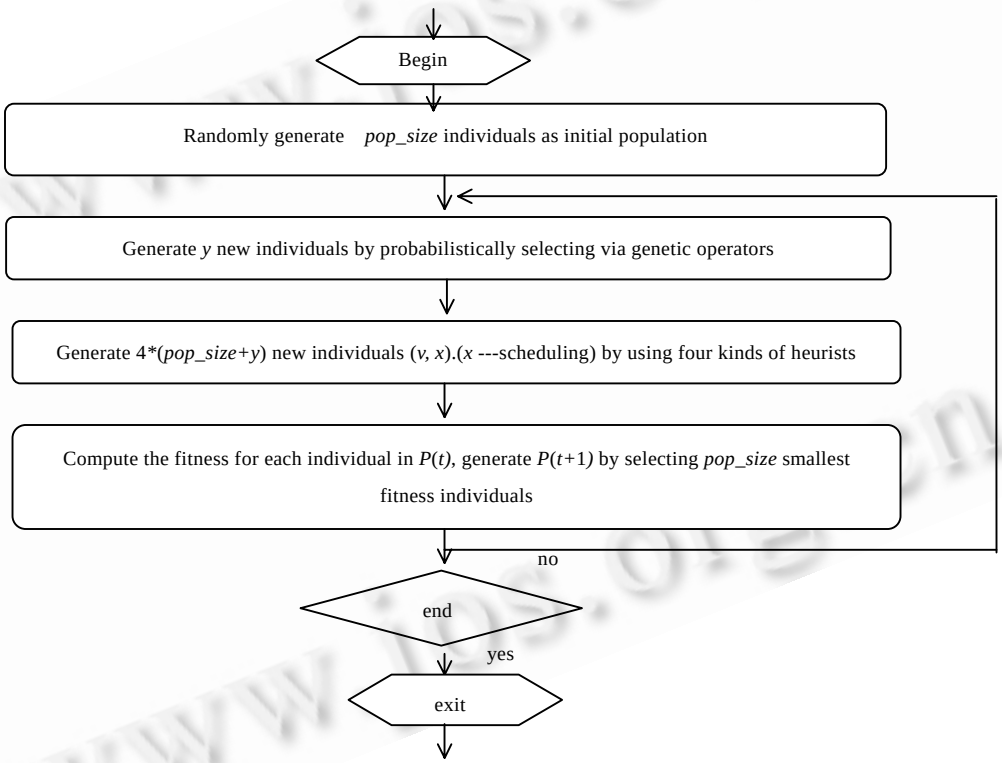

Fig.2 Flow chart of parallel schedule based on genetic algorithm

图 2 基于遗传算法的并行调度流程图

4 仿真实验的设计和性能分析

为了测试上述算法的正确性和有效性,结合实际的系统和并行调度策略,设计专门的仿真实验.测试从两个方面入手:(1) 设置不同的参数观察遗传算法的运行时间和运行结果的并行调度执行计划代价的关系,(2) 不同的连接树型用 4 种调度策略进行调度时处理机数和运行结果的并行调度执行计划代价的关系.

4.1 遗传算法的参数控制

遗传算法中的几个参数如交叉率 P_c、变异率 P_m、种群大小 *pop_size* 等的取值不同直接影响到遗传算法的效率.合理的参数取值有助于算法的快速收敛,我们分别测试了交叉率和种群大小对遗传算法性能的影响.图 3 和图 4 是在指定运行时间(1、2、3、4、5、10、15、20 秒)和遗传算法找到的最优解的时间代价之间的关系图.

图 3 显示了交叉率为 0.4、0.6 和 0.8 时,遗传算法的性能情况.其中当 P_c 为 0.4 时,遗传算法的收敛最慢,而且找到的最优解也比其他两种情况稍差,P_c 为 0.8 时,收敛较快,最优解的代价比前一种情况好,而 P_c 等于 0.6 时,收敛速度好于其他两种情况,最优解的随机波动也小于 0.4 和 0.8 的情况.由于交叉运算影响到遗传算法扩展搜索空间,当 P_c 较大时,遗传算法能在更大的空间内寻找最优解,当然,P_c 太大,在一定程度上使算法过多的扩展解空间,忽视了局部和区域搜索,同样不能较快地找到最优解.从搜索连接表达式的最佳执行调度计划上,P_c 取 0.6 是比较好的选择.

图 4 是不同的 *pop_size* 下的遗传算法的性能.*pop_size* 和 p_c 的功能相似,图中显示了 *pop_size* 在取值为 60 和 80 时,算法性能好于 40,而取值为 60 和 80 的性能相差不大.一般说来,*pop_size* 较小,可提高遗传算法的运算速度,却降低了群体的多样性,可能引起遗传算法的早熟现象,而 *pop_size* 取值较大时,降低效率,但有利于在更大的空间中求解,跳出局部范围.

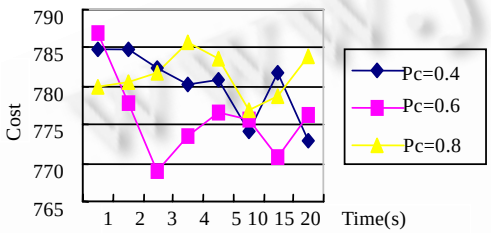

Fig.3 Performance for different P_c
图 3 不同交叉率下时间和最优解关系

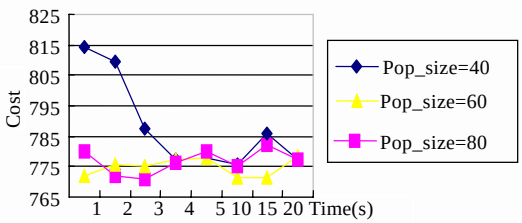

Fig. 4 Performance for different *pop_size*
图 4 不同种群大小下时间和最优解关系

4.2 并行调度策略的对比

为比较在不同的连接树形下 4 种调度策略的优劣,设计了有 8 个连接关系的连接表达式构成的浓密树、类左深树和类右深树如图 5 所示.

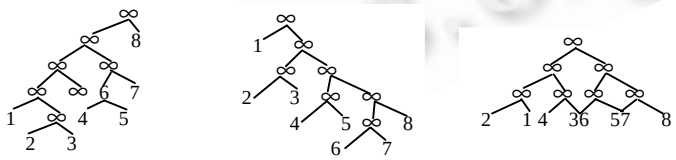

Fig.5 Left-Deep tree,right-deep tree and bushy tree
图 5 测试调度策略的类左深树、类右深树和浓密树

图 6~图 8 描述了浓密树、类左深树和类右深树情况下处理机调度和执行代价的关系,*x* 轴是可用处理机数目,*y* 轴是执行时间代价.图 6 显示在浓密树的情况下 4 种调度策略的执行代价都随处理机的增多而减少(这里不考虑并行处理机间相互影响带来的时间开销),当处理机的数目较小时,顺序执行和同步执行调度的时间代价较少,而处理机较多时,分段右深树和完全并行调度显示出较好的性能.这是因为顺序执行和同步执行主要以连接内的并行为主,当较多的处理机处理同一个连接时,处理机处理效率较低,处理机不能充分得到利用,而分段右深树和完全并行方法的流水线并行随处理机增多,性能增加,连接间的并行度得到提高.

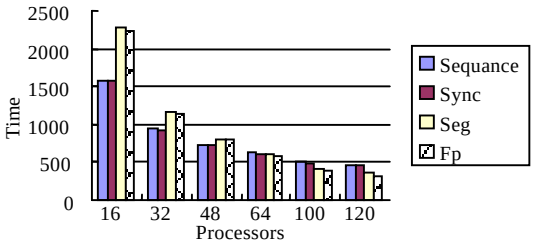

Fig.6 Performance for bushy tree
图 6 浓密树处理机数和响应时间的关系

图 7 是类左深树情况下处理机数目和执行时间代价的关系.4 种调度策略执行代价都随处理机的增多而减少,其中完全并行的调度策略的时间代价远远小于其他 3 种,其他 3 种的代价相近,代价差别相对而言较小.这是因为其他 3 种策略在类左深树的情况下,并行度很小,顺序执行和分段右深树的策略没有连接间的并行,而同步执行的策略由于中间结点的左右两个孩子的负载相差太大,因而连接间的并行度很小.完全并行策略在左深树的情况下,每两个相邻中间结点存在类似流水线的并行,只是长度仅限于 2.

对类右深树的调度情况如图 8 所示.完全并行和分段右深树的调度策略的时间代价远远小于顺序执行和同步执行的方法.类右深树的流水线长,完全并行和分段右深树的优势得到充分的发挥,由于完全并行策略还存在长度为 2 的左深树的并行,使得完全并行策略的性能能够稍好于分段右深树策略.同步执行和顺序执行方法时间代价几乎相同,这是因为类右深树的左右孩子的负载很不平衡,使同步执行策略的连接间的并行执行很难发挥好处,因而性能和顺序执行方法相似.

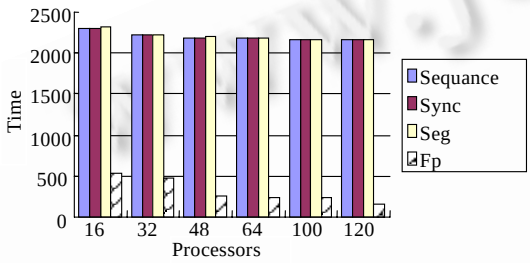

Fig.7 Performance of left-deep tree
图 7 类左深树处理机数和响应时间的关系

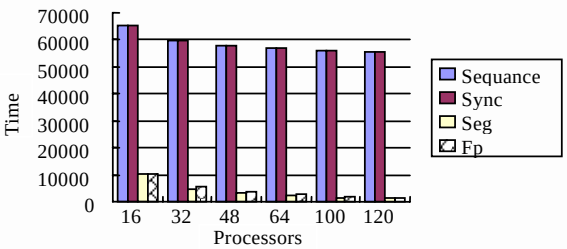

Fig.8 Performance of right-deep tree
图 8 类右深树处理机数和响应时间的关系

5 结束语

多连接表达式的并行调度是并行数据库中研究的重点之一.本文提出了一种结合遗传算法和启发式规则的搜索多连接表达式的最优并行调度执行计划的方法.仿真实验证明结合启发式规则依据顺序调度、同步并行调度、分段右深树调度和完全并行调度四种方法,在优化连接树上分配多处理机使整个连接表达式的并行调度执行代价最小.

References:

[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] Steinbrunn, M., Moerkotte, G., Kemper, A. Heuristic and randomized optimization for the join ordering problem. VLDB Journal, 1997,6(3):191～208.

[3] Wilschut, A.N., Flokstra, J., Apers, P.M.G. Parallel evaluation of multi-join queries. In: Michael, J.C., Donovan, A.S., eds. Proceedings of the ACM-SIGMOD'95. San Jose, CA: Academic Press, 1995. 115～126.

[4] Chen, M.S., Yu, P.S., Wu, K.L. Scheduling and processor allocation for parallel execution of multi-join queries. In: Proceedings of the 8th International Conference on Data Engineering. Arizona: I.C.S. Press, 1992. 58～67.

[5] Zhou Ming, Sun Shu-dong. The Principle and Application of Genetic Algorithm. Beijing: National Defense Industry Press, 1999 (in Chinese).

附中文参考文献:
[5] 周明,孙树栋.遗传算法原理及应用.北京:国防工业出版社, 1999.

Parallel Query Optimization Techniques for Multi-Join Expressions Based on Genetic Algorithm*

CAO Yang, FANG Qiang, WANG Guo-ren, YU Ge

(*School of Information Science and Engineering, Northeastern University, Shenyang* 110004, *China*)

E-mail: {wanggr,yuge}@mail.neu.edu.cn

http://www.neu.edu.cn

Abstract: The parallel query optimization for multi-join expressions is one of the key factors to improve the performance of database systems. In this paper, an approach to solve the problems of the parallel query optimization for multi-join expressions by adopting GA algorithms is proposed. To improve the execution efficiency of the query processors, the authors exploit heuristics to seek the optimum parallel scheduling execution plan for multi-join expressions. The detailed testing results and performance analysis are presented. The experiment results show that the GA algorithm with heuristic knowledge is effective for parallel query processing of multi-joins, and plays an important role in improving the performance of database systems.

Key words: genetic algorithm; multi-join expression; query optimization; parallel scheduling

* Received April 20, 2001; accepted September 24, 2001

Supported by the Foundation for University Key Teacher; the Teaching and Research Award Program for Outstanding Young Teachers in Higher Education Institution; the Cross Century Excellent Young Teacher Foundation of the Ministry of Education of China