• 软件与算法 •

基于贪婪策略的分布式数据库查询优化研究

李志伟

(空军第一航空学院 计算机教研室，河南 信阳 464000)

摘 要：针对分布式数据库系统复杂的多连接查询问题，分析了查询系统的目标要求，研究了查询优化的代价模型。结合具体实例，通过问题简化，构造出代价模型的查询图，提出了利用贪婪算法实现数据库查询的迭代方案。采用多步决策，按照一定的算法依次优化查询图，使得每一步优化都能得到最小的查询中间代价，从而确保了全局查询的最优。分析比较结果表明，该算法能以最小的代价实现对数据库的查询优化，缩短查询时间，提高查询效率。

关键词：贪婪算法；查询优化；查询图；代价；优化

中图法分类号：TP311 **文献标识码**：A **文章编号**：1000-7024 (2010) 17-3838-03

Study on distributed database query optimization based on greedy strategy

LI Zhi-wei

(Office of Computer Teaching and Researching, First Aeronautical Institute of Air Force, Xinyang 464000, China)

Abstract: Aimed at the issues of complicated multi-join query in the distributed database system, the target demand is analyzed, and the cost model of query optimization is studied. Based on a provided case, the query graph of cost model is constructed by simplifying the issue, and an iterative scheme based on greedy algorithm to realize database query is proposed. In order to obtain the lowest query middle-cost in each step of optimization, and insure the global-query optimum, the multi-decision is employed, and according to the definite arithmetic take turns optimize the query graph. Analysis and comparison show that the proposed algorithm could realize query optimization to database in lowest cost, shorten the query time, enhance the query efficiency.

Key words: greedy algorithm; query optimization; query graph; cost; optimization

0 引 言

在数据库管理系统(DBMS)中，查询是数据库应用的一个重要环节，查询优化方法的选择则是影响系统性能的关键因素。在分布式数据库系统中，连接操作以及连接顺序的选择是关系其查询优化的核心任务之一，如何寻找多个关系连接的最佳执行顺序，使查询代价最小，是一个 NP-Hard 问题[1]。随着分布式数据库应用的日益广泛，近年来，人们研究并提出了大量的优化算法，如用于商业数据库系统中的动态编程法，基于智能预取技术的蚁群算法等[2]，这些算法都从某些侧面研究了查询优化的局部问题，但其时间复杂度和空间复杂度相对较高，在资源有限的情况下，对数据库查询的优化往往受到一定的限制。

为了更好地解决查询优化问题，本文从查询系统的逻辑结构入手，提出了一种基于贪婪策略的数据库查询优化算法，并采用多步决策的方法将系统的查询结点进行代价处理。在每一步决策时，都要根据问题满足的全部约束条件构成一个最优解。对于数据库查询问题，这里用查询图对问题空间进行描述，通过算法设计构造问题的解空间，通过代价估算，检验算法的优越性。通过对整个问题空间的搜素迭代，达到问题的全局最优。

1 查询优化

1.1 查询优化的目标

对于分布式 DBMS，既提供了系统高速执行的潜力，如可以利用多结点进行并行处理，可以分散特别忙碌的结点的任务，以均衡负荷等，同时也带来了一些不利的方面，如增大的通信开销，特别是广域网中大量数据的传输，多复本的更新等。因此，全局优化的主要任务是发挥有利的一面，减少不利的一面，以提高分布式数据库系统的性能。

根据分布式数据库系统性能指标的不同，优化目标也不相同，通常主要有两个目标[3]：

(1)最小响应时间：即事务的执行时间最短，也就是要求系统尽可能地“快”。

(2)最大吞吐量：即在一定时间内，系统完成的数据库处理任务最多。

这两个指标既有联系又不等价，具体以哪种指标作为优化指标，取决于 DBMS 的用途。本文主要以最小响应时间作为优化目标进行讨论。

1.2 优化方案的选择

当对一个 DBMS 进行查询优化时，可供选择的查询方案

收稿日期：2009-10-09；修订日期：2009-12-09。

作者简介：李志伟（1964－），男，河南宜阳人，副教授，CCF 会员，研究方向为计算机网络与数据库技术。E-mail：xylzw@sohu.com

很多，而影响查询方案的因素主要有以下几个方面[3]。

(1)多种复本选择：为了提高访问的局部性和/或系统的可用性，有时关系或裂片常设有多个复本，分布在不同的结点上。在查询处理时，就存在一个复本选择问题。复本的选择可按照如下原则进行：

一是尽可能提高访问的局部性，减少远距离访问；

二是尽可能减少通信开销，尤其要减少大量数据的传送；

三是应当适当考虑结点负荷的平衡。

(2)多种执行次序：当复本较多时，复本选择的可能方案也很多，因此，复本的选择是一个复杂的问题。如图 1 所示，假设有 4 个关系分别分布在不同的 4 个结点 R1—R4 上，则根据不同的选择方案，可以有多种不同的执行顺序。

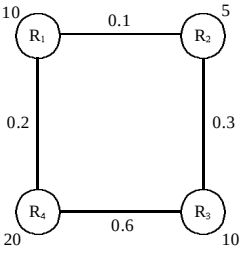

图 1 查询图示例

(3)多种执行地点：以图 1 为例，图中的任意一个结点都可以作为系统的执行地点，适当选择执行地点，可以减少系统的查询代价，提高系统的执行效率，即对查询进行优化。

查询优化的方案主要有两种，一是静态优化，二是动态优化。静态优化是指在事务执行之前进行的优化，它主要是根据数据目录中的统计数据对中间结果的大小进行估计，通常情况下，这种估计不一定准确，因此优化质量可能不会太高。而动态优化的基本方法是根据实际中间结果进行优化，即边执行边优化。动态优化的质量虽然比静态优化高，但优化须在执行时进行，执行一次，优化一次，因此可能会增加系统的通信开销。在实际应用中，往往采取折衷的方案，先按静态优化的结果执行，如果发现静态优化所依据的统计值与实际偏差太大，再以动态优化的结果取代静态优化的执行方案。对于执行方案的选择，可用穷举法对各种方案进行代价比较，但工作量太大。通常采用的方法是先用启发式规则进行方案初选，然后再用代价比较法从中进行选择。

2 查询执行代价

2.1 查询执行代价的组成

查询执行代价是指执行查询时的时间开销，主要由以下 3 部分组成：①访问辅助存储器的代价(简称 I/O 代价)；②计算代价(简称 CPU 代价)；③通信代价。

无论是集中式 DBMS，还是分布式 DBMS，I/O 代价和 CPU 代价都是执行任何查询所必须付出的代价。

在集中式数据库系统中，I/O 代价又是主要的，并且是 DBMS 性能的瓶颈。一般情况下，伴随着一定的 I/O 操作，总是需要一定的 CPU 操作。I/O 代价也能部分地反映 CPU 代价，而且如果综合考虑 I/O 和 CPU 代价，不能将两者简单地相加，还必须对两种代价适当地加权。如何加权，这也是难以决定的问题，需要根据具体的问题构造相应的代价模型。

在分布式数据库系统中，通信代价往往是主要的，是查询执行代价应该考虑的主要方面。

2.2 查询执行代价模型[3]

(1)I/O 代价模型

目前，使用较多的辅助存储器主要是磁盘，其一次访问所需的代价可表示为

$$C_{I/O} = D_0 + D_1 \cdot X$$

式中：X——存取数据的大小；D_0——与 X 无关的 I/O 代价，包括寻道时间和等待时间；D_1——单位数据的传输时间。一般地，$D_0 >> D_1 \cdot X$，故 $C_{I/O} \approx D_0$；I/O 代价≈I/O 次数 · D_0。

在具体优化时，代价只供比较执行方案之用，没有必要进行精确计算，只需估算出其相对值即可。由于同一磁盘的 D0 是常数，一般只用 I/O 次数作为 I/O 代价。为了估算代价，系统必须提供必要的数据，根据这些数据再进一步进行计算。需要注意的一点是，系统提供的数据通常是一些统计数据，这些数据会随数据库状态的改变而改变。

(2)通信代价模型

分布式数据库系统中的通信代价与网络的类型有关，通常可以粗略地表示为[4]

$$C_c(X) = C_0 + C_1 \cdot X$$

式中：X——传输数据的大小；C_0——传输一次数据所必需的初始代价；C_1——单位数据的传输代价(代价系数)。C_0、C_1——般随网络的类型而变化，对于某一具体的网络，它们为常数。

对于分布式 DBMS，在进行代价估算时，除了考虑 I/O 代价和 CPU 代价外，还要考虑通信代价。

不同的网络对查询处理的准则不尽相同，通常情况下，网上数据通信要比各网络结点内部的通信及处理慢得多，因此在进行代价估计时，需要重点考虑通信代价。为了简化问题，本文暂不考虑查询的局部 I/O 代价和 CPU 代价，而把通信代价作为系统查询代价的主要部分进行讨论。

3 基于查询图的贪婪优化方案

3.1 查询图模型

如图 1 是一个查询图的实例，其中 R_1、R_2、R_3、R_4 表示参与查询的所有基本关系，在该图中称为结点，结点旁边的数字表示该查询的数据大小，每个查询表示为结点之间的边，边上的数字表示该边所连接的两个结点之间进行数据传输时单位数据的传输代价，不妨定义为代价系数。

对于一个数据库系统，要想通过查询图求解出一种最优的查询关系，需要根据某种规则对该查询图进行变换，构造出一个表示特定查询顺序的查询树[5]，并用叶子结点表示基本的查询关系，分支结点表示查询连接，其对应的数值表示查询的中间结果，最后，对于所有连接操作的中间结果相加，求得的总和即表示整个系统的查询代价。

3.2 贪婪算法

对于一个给定的问题，要想对其进行完整地求解，通常情况下需要提供 n 个输入和一些约束条件，任何满足这些约束条件的子集均称之为一个可能解。

贪婪算法的基本思想是对问题求解采取多步决策,每一步的选择都必须能构成问题的一个可能解(即满足问题的全部约束条件),同时又使目标函数的值增加最快(当目标函数最大时)或增加最慢(当目标函数最小时)。这种选择过程是以某些最优化量度为依据的,而最优化量度有时可以是目标函数本身,有时也可能是别的量度。最优化量度的选择是贪婪法的关键[6]。

基于查询图的贪婪查询实际上是一种动态优化方案,在具体查询过程中,可以用中间查询结果的大小近似地表示当前通信代价的大小,因此,对于不同结点之间进行查询连接时,应当选取查询运算最小的中间结果,从而降低当前查询代价,达到局部最优。

3.3 算法设计

根据上述贪婪优化策略,对于如图 1 所示的查询图,其优化步骤如下:

(1)对于相邻的结点进行连接查询时,首先需要找出中间结果最小的连接运算。在图 1 中,当相邻两个结点进行连接时,其相应的查询代价分别为

$$C_{R1\infty R2} = 10*5*0.1=5$$

$$C_{R2\infty R3} = 5*10*0.3=15$$

$$C_{R3\infty R4} = 10*20*0.6=120$$

$$C_{R1\infty R4} = 10*20*0.2=40$$

可见,当 R_1 和 R_2 连接时,当前的查询代价最小,因此可将该二结点进行合并,结果如图 2 所示[7]。

这里,当对原结点进行合并时,新的查询图中合并后结点旁边的数字表示当前查询结果的中间值。

(2)在图 2 中,采用与(1)同样的方法可以判定出,当 R_{12} 与 R_3 两个结点合并时,其中间结果最小,相应的查询代价为:$C_{R12\infty R3}$=5*10*0.3=15,合并后结果如图 3 所示。

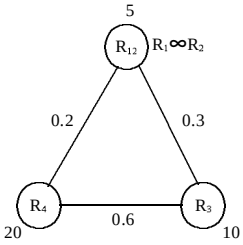

图 2 结点 R_1 与 R_2 合并后的查询

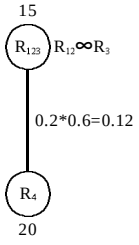

图 3 结点 R_{12} 与 R_3 合并后的查询

这里,R_{12} 与 R_3 两个结点合并时将引起线段 $[R_{12}, R_4]$ 和 $[R_3,R_4]$的合并,合并后的代价系数取原来两个代价系数的乘积作为近似值,即 0.2*0.6=0.12。

(3)最后,执行 R_{123} 与 R_4 的合并查询,查询代价为 $C_{R123\infty R4}$=15*20*0.12=36。

至此,整个查询的连接策略已经确定,系统的查询总代价近似为 $C_{R1\infty R2}+C_{R12\infty R3}+C_{R123\infty R4}$=5+15+36= 56。

根据上述算法得出的查询结果,相应的连接表达式可表示为 $((R_1\infty R_2)\infty R_3)\infty R_4$。

根据贪婪算法的规则,对于图 1 所示的分布式 DBMS 的查询,其解空间即所有可能的查询计划,每个查询计划都可看作解空间中的一个状态,它表示在求解问题的过程中某一步骤或某一时刻的状况。因此,对于该算法得出的查询连接可以更直观地用图 4 所示的二叉树进行表示[8]。

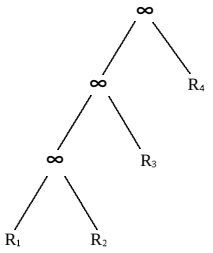

图 4 基于贪婪策略的查询树

该二叉树又称为查询树,在完整的解空间中,查询树的形状是任意的,根据具体的算法设计而定。

由此可以看出,该算法的设计思想实际上是按照贪婪策略的算法规则,将图 1 所示的查询图分步转换,实现对数据库查询的优化处理,最后得出图 4 所示的查询树结构。

3.4 算法分析

对于图 1 所示的查询图,根据采用策略的不同,得到的连接方案也不相同,所要付出的查询代价也将不同,表 1 列出了各种不同连接方案对应的查询代价。

表 1 查询代价分析

方案	查询连接	查询中间代价	总查询代价
1	$((R_1\infty R_2)\infty R_3)\infty R_4$	5,15,36	56
2	$(R_1\infty R_2)\infty(R_3\infty R_4)$	5,120,36	161
3	$(R_1\infty R_4)\infty(R_2\infty R_3)$	40,15,36	91
4	$((R_1\infty R_2)\infty R_4)\infty R_3$	5,20,36	61
5	$(R_1\infty(R_2\infty R_3))\infty R_4$	15,15,36	66
6	$R_1\infty((R_2\infty R_3)\infty R_4)$	15,180,36	231
7	$R_1\infty((R_3\infty R_4)\infty R_2)$	120,180,36	336
8	$((R_1\infty R_4)\infty R_2)\infty R_3$	40,20,36	96

从该表可以看出,有以下两个方面的特点:

(1)对于任何一种查询连接,最后一步的查询中间代价完全相同(均为 36),这是由于对于整个查询图来说,不管采用何种方式进行连接,最终结果是要对整个系统中各个结点沿着各种不同的查询路线,全部查询一遍,即完成一次遍历。

(2)对于不同的查询连接,由于采用的策略不同,除最后一步的中间代价之外,其它查询的中间代价也不完全相同,从而导致查询的总代价也不相同。从表 1 可以看出,总查询代价最小的查询连接是方案 1,这正是本文介绍的贪婪算法得出的查询结果,从而验证了贪婪算法在解决查询优化问题时的有效性。

从图 4 所示的查询树结构可以看出,贪婪算法实际上是一种自底向上的启发式查询优化算法,在选择连接顺序时,总是使用一种简单而严格的选择方法,每次都是选取当前代价最小的一个连接,这样便可使整个系统最终查询的总代价达到最小。同时,该算法也充分利用了查询图本身包含的语义信息,使得整个系统的执行效率较高。

4 结束语

本文提出的贪婪查询优化算法在进行每步决策时,总是

(下转第 3875 页)

4 结果分析

为验证算法有效性，本文以计算机学院一个学期的课程数据为依据对算法进行评估，测试种群规模对算法性能的影响。设种群规模为 50、70、90、110、130，1500，2000，分别对计算机学院的 52 门课程进行测试。根据测试得到的近似最优解的适应度值来衡量粒子群数量对性能的影响，并比较加入免疫算子的粒子群算法与标准粒子群算法在处理问题上的不同。测试结果如图 3 所示。

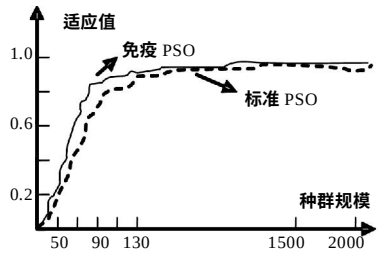

图 3 种群规模和算法采用对适应值的影响

由图 3 可以看出，随着种群规模的增加，得到的近似最优解的适应度值逐渐变大，当种群规模增加到一定程度如 1500、2000 时，适应度值基本保持不变。适应度值越高，表明粒子所处位置越优，从而课表的质量越高。同时，免疫粒子群算法在寻优能力上优于标准粒子群算法，它能快速地达到较高适应度值，对于排课优化来说，起到积极作用。

另外，假设有 A、B、C、D 这 4 个学院，现对每个学院一学期的课程进行排课，分别测试其适应度值。设定种群规模为 80，学院基本信息及测试结果如表 2 所示。

由表 2 可知，教学资源越充足(如表中的教师数、教室数)，得到的适应度值越优，越有利于提高课表质量。

表 2 教学资源对适应值的影响

教学资源相关信息					
学院	教室数	教师数	班级数	课程数	适应值
A	16	30	8	32	0.9433
B	15	27	9	36	0.9387
C	20	32	11	41	0.9546
D	24	37	12	46	0.9618

5 结束语

本文研究免疫粒子群算法在解决教务排课问题中的应用，将免疫系统中的记忆和选择机制加入到粒子群算法中。由于排课问题是一个非线性多目标优化问题，应用该算法操作简单，设置参数少，特有的记忆功能确保达到全局最优解，调节功能可提高局部搜索能力。本文只给出了求解问题域的基本方法，在实际应用中往往需要依赖人们对排课经验的分析和总结，这样才能取得较好的效果。

参考文献:

[1] 陈章辉,黄小晖,任文艺,等.基于双倍体遗传算法求解大学排课问题[J].计算机应用,2008,28(12):3074-3076.

[2] 张林.基于蚁群算法的排课系统研究与设计[D].合肥:安徽大学,2005.

[3] 刘永涛.基于粒子群算法的排课系统的设计与实现[D].上海:华东师范大学,2008.

[4] SHI Y,EBERHART R.A modified particle swarm optimizer[C].Proc of IEEE International Conference on Evolutionary Computation,1998:69-73.

[5] 韩琳.免疫粒子群算法研究及其应用[D].西安:西安工程大学,2008.

[6] 贾岩峰.基于免疫算法的粒子群优化算法的研究[D].吉林:东北电力大学,2006.

[7] 韦玉,冯速.免疫遗传算法在排课问题中的应用[J].北京师范大学学报(自然科学版),2008,44(2):168-173.

[8] 刘晶晶.粒子群优化算法的改进与应用[D].武汉:武汉理工大学,2008.

(上接第 3840 页)

作出在当前看来是最好的选择，也就是说，该算法并不是从整体最优上加以考虑的，它所作出的选择只是在某种意义上的局部最优选择。当然，我们希望该算法得到的最终结果也是整体最优的。实际上，本文示例得到的结果就是一个整体最优解。但是贪婪算法并不是对所有问题都能得到整体最优解。尽管如此，本文提出的查询优化算法，由于其简单、方便，且优化效率较高，对于大多数问题都能产生最优解，即使有些情况不能得到整体最优解，但其最终结果却是最优解的很好的近似解，因此具有普遍的实用价值。

参考文献:

[1] 宋静静,贾智平.一种嵌入式实时数据库系统查询优化算法[J].计算机工程,2007,33(11):90-92.

[2] 朱鸿宇, 刘瑰, 唐福华, 等. 数据库查询优化中的智能预取技术[J].计算机应用研究,2007,24(5):35-37.

[3] 王能斌.数据库系统教程[M].北京:电子工业出版社,2008.

[4] 徐济惠.嵌入式数据库多连接查询优化算法的研究[J].宁波大学学报(理工版),2008,21(2):206-210.

[5] 李瑞轩,霍晓丽,文珠穆,等.多数据库系统中的全局查询转换方法研究[J].计算机工程,2005,31(8):4-6.

[6] 李志伟.利用贪心法实现对磁盘文件的最佳存储[J].计算机工程与应用,2003,39(11):103-105.

[7] 卢凌云,莫洪灵,魏芳.分布式DCI数字电影图像压缩系统研究[J].计算机工程与设计,2009,30(13):3124-3127.

[8] 李志伟.基于二进制数据库的信息搜索算法[J].计算机工程与设计,2005,26(10):2759-2761.