PostgreSQL查询优化器分析研究

孙振兴[1],向 阳[1],刘增宝[2]

(1. 同济大学 电子与信息工程学院,上海 201804;
2. 兖州矿业集团东滩煤矿,山东 邹城 273512)

摘 要:作为开源数据库的代表,PostgreSQL的应用范围越来越广泛。文中的目的是研究PostgreSQL查询优化器的工作原理,介绍了PostgreSQL查询优化器的工作流程,分析了PostgreSQL查询优化器的工作原理,深入剖析了PostgreSQL查询优化器实现的具体细节和采用的两种优化算法。结合图论中查找最小生成树的算法提出了改进策略,并简要论证了可行性。研究发现,PostgreSQL查询优化器可以处理任意复杂的请求,并能尽快地给出比较合理的执行路径。

关键词:PostgreSQL;查询优化;System-R;基因优化

中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2011)08-0141-04

Analysis and Research on Optimizer of PostgreSQL

SUN Zhen-xing[1], XIANG Yang[1], LIU Zeng-bao[2]

(1. College of Electronic and Information Engineering, Tongji University, Shanghai 201804, China;
2. Dong Tan Coal Mine, Yanzhou Mining Group, Zoucheng 273512, China)

Abstract: As a representative of open-source database, PostgreSQL has been used more and more widely. The purpose is to study the work principle of PostgreSQL optimizer. Introduces the architecture of PostgreSQL and the workflow of the optimizer, then analyses the optimization principle. After that, dissect the two algorithms as well as their implementation details applied in PostgreSQL. Finally, gives an improved strategy based on MST. The conclusion is that PostgreSQL's optimizer can handler arbitrarily complex SQL request and can give very reasonable execute path as soon as possible.

Key words: PostgreSQL; optimizer; System-R; GEQO

0 引 言

PostgreSQL是一种对象—关系型DBMS,在一些涉及复杂数据的系统具有广泛的应用,例如GIS[1]等。它设计时采用了很多至今看来仍然很先进的理念,其中优化器(planner/optimizer)可以说是PostgreSQL中最复杂的部件。

PostgreSQL的优化器采用两种算法来处理复杂度不同的查询,对于比较简单的查询,优化器采用动态规划算法,这是一种近似穷举的搜索算法,但是由于搜索范围较小,因此效率并不差。对于复杂的查询,则采用基因算法。基因算法是一种自适应的算法,可以在较短的时间内给出一个较优的解。

收稿日期:2011-02-16;**修回日期**:2011-05-28

基金项目:国家自然科学基金资助项目(70771077);国家高技术研究发展计划(863计划)项目(2008AA04Z106);上海市科委项目(08DZ1122300);上海市信息化发展专项资金项目(200901015)

作者简介:孙振兴(1986-),男,硕士生,研究方向为数据库、数据挖掘;向 阳,博士,教授,博士生导师,研究方向为决策支持系统、人工智能、数据挖掘。

1 PostgreSQL工作流程

PostgreSQL采用的是比较经典的C/S(client/server)架构,即服务器端有一个守护进程侦听来自客户端的请求,当有请求到达时,服务器端便获取这个请求,并产生一个独立的进程处理该请求以及后续的操作。在PostgreSQL中,守护进程被命名为postmaster,每当postmaster获取到新的请求便把该请求交给PostgreSQL服务器进行处理[2],整个流程如图1所示。

由图1可以清晰地看出PostgreSQL各模块的主要作用:客户端负责发起连接并发送查询请求;连接器接收客户端的请求并将请求转交给解析器;解析器对请求进行语法分析并生成查询树;优化器接收查询树并据此找到最优的执行路径;执行器负责对执行路径的具体实施,并且获取存储器的结果,然后将结果返回客户端;存储器则是负责对具体数据的管理和操作。

2 查询优化器分析

严格地讲,PostgreSQL的查询优化器应该包括规划器和优化器两部分。一个特定的SQL语句可以以

多种不同的方式执行，并且每种方式都生成完全相同的结果集，所不同的是获取结果集需要的代价存在很大的差异。查询优化器中的规划器就是负责根据特定的 SQL 语句生成所有可能的执行路径（也就是执行方式），而优化器则负责检查这些执行路径并计算每种执行路径获取结果集所花费的代价，最终选择一个代价最小的执行路径，然后根据这个执行路径制作一个完整的规划树传递给执行器。

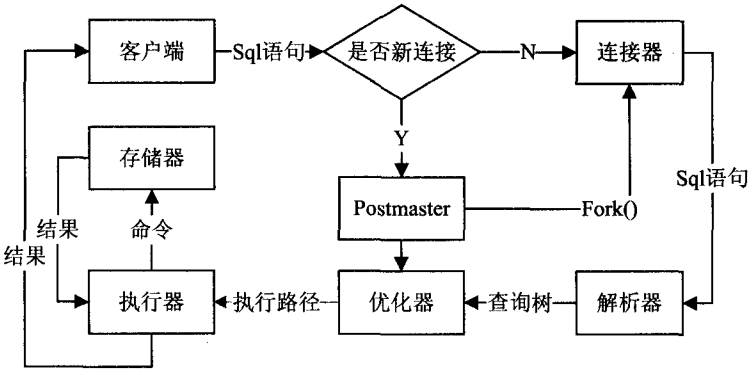

图 1 PostgreSQL 工作流程图

2.1 主要概念说明

查询优化器在对 SQL 语句进行优化时会涉及到几个重要的概念，下面将介绍这些重要的概念，然后再对查询优化器的算法和工作流程进行研究与分析。

1）路径[3]：在 PostgreSQL 中，查询优化器用路径（path）来表示处理一条 SQL 语句的一种方案。一条语句可以有很多条不同的执行路径，路径的不同主要是由以下几方面的原因造成的：

第一，基本表的扫描方式不同。扫描方式是指优化器从数据库表中读取（更确切地说是检索）数据的方法。基本上有两种，第一种称为顺序访问方式，是一种连续读取表中的数据的方法；另一种就是使用索引，称之为索引访问方式，这种方法会有更好的效率，当查询的数据量较多时优先考虑这种查询方法。

第二，关系间的链接方式不同。关系间链接方式包括嵌套循环链接、哈希链接和归并链接，当用户提交的 SQL 语句涉及有像 Where 子句这样的约束时，优化器会考虑使用上述链接方式处理子句中涉及的关系。

第三，关系间的链接顺序不同。当子句中涉及的关系基本表较多时，优化器就会考虑它们之间的链接关系，因此尽早地约减数据可以大大地提高查询效率。

PostgreSQL 规划器根据上述三个方面针对特定的 SQL 语句生成路径，并以链表的方式将它们组织起来。

2）代价[4]：优化器在生成执行路径的时候采用执行代价对候选路径进行评估，并从中选择执行代价最小的路径。下边将对代价数据结构进行简要的说明。

一条路径的执行代价总是由以下几个基本代价组成的：（1）获取一个连续的页所需要的代价；（2）获取一个非连续的页所需要的代价；（3）处理一组数据所需要的标准 cpu 时间；（4）执行一次操作或者运行一个函数所需要的标准 cpu 时间。

在具体的实现中，每条路径都会两个代价，一个是获取所有数据所需要的代价，另外一个是在获取第一条数据之前所做的初始化工作需要的付出代价。因此一条执行路径的代价可以说由这两种代价组成，但是并不是简单的将两种代价相加，因为一些操作并不需要获取所有的数据，例如 exist 子句。

3）规划树：SQL 语句在经过查询优化之后，将会以规划树（plan tree）的形式输出，并被传送给执行器。规划树本质上是一个需求拉动的元组处理操作流水线，每个节点被调用时将会在它的输出队列产生下一个元组。

2.2 查询优化的流程

其实 PostgreSQL 对查询语句的优化贯穿了整个服务器端的处理过程，例如重写模块对 SQL 查询的处理等，而查询优化器则是整个优化系统的核心和精华，查询优化器对 SQL 语句的优化采取的措施主要有：

1）表达式的预处理。表达式的预处理有两个作用，第一就是找出查询中所有的子句并依次进行预处理，第二就是对子句进行分析进而进行析取或者合取。

2）查找索引。通常来说，在查询数据量较大时，使用索引的方式会比顺序查找的效率高很多，因此优化器会查找关系上的索引来提高查询的效率。

3）代价评估策略。PostgreSQL 利用代价评估策略分析查询语句的每一条执行路径，并给出路径的执行代价，供优化器进行选择。

4）基因查询优化。PostgreSQL 利用基因算法处理涉及很多关系的查询语句，并在可接受的时间内给出一条较优的执行路径。

为了更好地理解查询优化器对 SQL 语句的优化过程，下边将给出 PostgreSQL 查询优化器的工作流程图，如图 2 所示。

2.3 查询优化的算法

总体来讲，PostgreSQL 查询优化器的算法可以概括为以下三步：

（1）接收查询树，生成所有可能的执行路径；

（2）遍历所有的执行路径，给出执行每条路径的代价；

（3）选择最优的执行路径，并据此生成规划树。

在 SQL 中，多个连接操作（join）可以按任意的顺序进行，同时所有的子查询都可以转变成连接操作[5]，

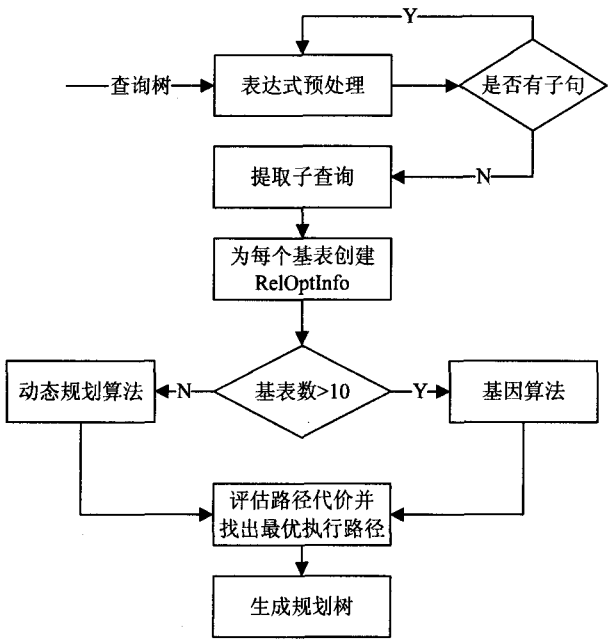

图 2　查询优化的流程

PostgreSQL 就是充分地利用这一点,将用户提交的 SQL 语句最终转化成几个 join 操作,然后计算每个 join 操作需要的代价,最后利用 System-R 算法或者基因算法找出最优的组合,即最优的执行路径。

2.3.1　System-R 算法

System-R 算法是一种典型的穷尽搜索算法[6],在 PostgreSQL 中该算法的目的是找到一棵最优的 join 树(即代价最小的树),它以左线性树作为搜索空间,采用动态规划(dynamic programming)的方法,逐步构造最优 join 树。

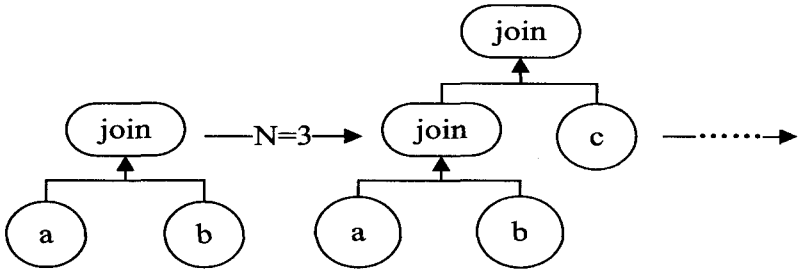

图 3　System-R 算法构造最优 join 树

System-R 算法的基本思想是:要想构造包含 n 个关系的最优 join 树,必须先构造包含 $n-1$ 个关系的最优树。因此,PostgreSQL 的查询优化器首先构造包含两个关系的最优 join 树,然后在此基础上构造包含三个关系的最优 join 树,依次类推,直到找到包含所有关系的最优 join 树,整个构造过程如图 3 所示[7]。

由以上的分析可以知道 System-R 算法的时间复杂度是 $O(2^n)$,在关系数较少时具有很好的效率,但是随着关系数目的增加,该算法的时间开销迅速增加,同时,对存储空间和 I/O 的要求也越来越大,因此 System-R 算法只适合处理关系数小于阈值(视具体情况而定)的情形,对于关系数目大于阈值的情形,PostgreSQL 采用基因算法进行处理[8]。

2.3.2　基因算法(GEQO)

基因算法(GA)是一类基于统计理论基础上的组合优化和搜索技术。一个简单的基因算法从形式上看并不复杂,但却要包含如下基本运算:变量的编码、群体成员的繁殖、个体成员之间基因交换以及个体成员的变异。PostgreSQL 查询优化器采用基因算法主要用来解决多连接查询优化的问题[7]。

基因算法在 PostgreSQL 中实现有以下特点:

(1)基因查询优化对执行计划进行编码,字串中的每一位代表一个关系。

(2)忽略基因突变,这大大简化了算法复杂度。

(3)使用稳定状态(steady state)策略替换全体中最小健康度的个体,提高算法的效率。

(4)边缘重组交叉(edge recombination crossover)的使用避免了算法处理过程中较优路径的丢失。

基因算法成功地解决了多连接查询优化的问题,让 PostgreSQL 可以使用非穷举的方式处理比较大的连接查询,但是仍然有很多问题亟待解决,例如当连接数目变大时,基因算法对存储空间的要求越来越大,同时处理这类查询所需的时间也显著增加,因此,有必要对基因算法进行改进,这将是后续工作。

3　PostgreSQL 优化算法与 MySql 优化算法的对比

作为目前最成功的两个开源数据库,PostgreSQL 和 MySql 都有其他数据库所不具备的优点,例如 PostgreSQL 对用户自定义类型、几何类型的支持和 MySql 轻巧快速等,下面将通过对 MySql 的优化算法的分析给出两个数据库优化算法的比较[9]。

在 MySql 优化算法中,每个查询都是一系列的连接(仅对一个表进行的查询被视为一个退化的连接),具体的是算法流程可以分成两部分:确定最佳连接顺序;执行一个嵌套循环来实现连接。

对于连接的实现(也就是算法的第二步)是通过一系列嵌套循环,而算法的第一步也就是查找最佳连接顺序的阶段和 PostgreSQL 查找最优执行路基的问题相似。MySql 采用一种改进的动态规划的算法查找最佳执行顺序,该算法定义了一个阈值,以控制查找最佳连接顺序的次数,若在小于阈值的查找次数中找到了最佳连接顺序则算法返回,否则从当前的最佳连接顺序候选集中选择一个最优的连接顺序[10]。

通过以上的分析可以看出，PostgreSQL 优化算法的效率高于 MySql 优化算法的效率，也就是说最终找到的执行树的执行速度是 PostgreSQL 快于 MySql，但是考虑到执行时间会抵消部分发现时间，因此 PostgreSQL 的返回结果的时间不一定短于 MySql。

4 结束语

一个 SQL 语句在 DBMS 中可以有很多条执行路径，虽然执行这些路径返回的结果完全相同，但是执行它们所花费的代价却又有很大的差异，而查询优化器的作用就是找出那条花费代价最小的路径。

PostgreSQL 查询优化器在生成所有可能的执行路径时会综合考虑基本表的访问方式、关系间的链接方法和关键间的链接顺序等因素，而在搜索最优的执行路径时，PostgreSQL 会视关系的多少选择不同的算法，对于关系较少的查询，用 System-R 算法进行处理，而对于关系较多的查询，则利用基因优化算法进行处理，最终找出最优的执行路径并生成规划树。

由于查找优化算法的工作流程和查找图的最小生成树算法相似，而经典的最小生成树查找算法的效率不会随着图的节点的增加而成几何增长，因此下一步的研究重点将会是参考查找最小生成树算法设计一个适合作为 DBMS 查询优化的算法，该算法将会对所有的查询进行统一的处理，另外，模拟退火算法在全局查询优化中有广泛的应用[11]，遗传基因算法也有相应的优化策略[12]，因此 PostgreSQL 查询优化器算法仍有很大的改进余地。

参考文献：

[1] 林慧君，彭 宏. 基于 WebGIS 的城市基础设施管理系统的设计[J]. 计算机技术与发展，2010，20(1)：221-224.

[2] 何伟平. PostgreSQL 的昨天今天和明天[EB/OL]. 2004-04-18[2010-05-25]. http://blog.csdn.net/beick/archive/2005/04/18/353063.aspx.

[3] 刘长浩，孙玉芳. PostgreSQL 请求优化机制研究[J]. 计算机科学，2005，32(4)：163-167.

[4] The PostgreSQL Global Development Group. postgresql-8.4.3 source code[EB/OL]. 2010-03-12[2010-05-25]. http://www.postgresql.org.

[5] 李桂杰，梅 红. 多关系 SQL 查询中连接顺序的优化[J]. 杭州电子科技大学学报，2006，26(2)：31-34.

[6] Jose S. A history and evaluation of System-R[J]. Computing Practice，1981，24(10)：632-646.

[7] The PostgreSQL Global Development Group. PostgreSQL 8.4.3 Documentation[EB/OL]. 2010-03-12[2010-05-25]. http://www.postgresql.org.

[8] 姜国伟. 数据库电子商务性能测评系统 ACID 子系统实现和 MySQL 与 PostgreSQL 查询处理分析[D]. 吉林：吉林大学，2005.

[9] 朱 斌，黄水清. PostgreSQL 与 MySQL 比较研究初探[J]. 计算机时代，2003，12：5-7.

[10] Pacbev S. Understanding MySql Innternals[M]. 北京：中国电力出版社，2007.

[11] 林慧君，彭 宏. 模拟退火算法在全局查询优化中的使用[J]. 计算机技术与发展，2006，16(4)：154-157.

[12] 路 景，周春艳. 基于遗传算法的混合优化策略研究[J]. 计算机技术与发展，2007，17(3)：144-149.

(上接第 140 页)

needs: a study and analysis of user queries on the web [J]. Information Processing and Management，2000，36：207-227.

[3] Broder A. A taxonomy of web search [C]//In SIGIR Forum. [s.l.]：[s.n.]，2002.

[4] Lee U，Liu Z，Cho J. In automatic identification of user goals in Web search[C]//Proc of World Wide Web Conference. Japan：Chiba Press，2005：10-14.

[5] Jansen B J，Booth D L，Spink A. Determining the informational，navigational and transaction intent of Web queries [J]. Information Processing & Management，2008，44(3)：1251-1266.

[6] 余慧佳，刘奕群，张 敏. 基于大规模日志分析的搜索引擎用户行为分析 [J]. 中文信息学报，2007，21(1)：109-114.

[7] 陈红涛，杨放春，陈 磊. 基于大规模中文搜索引擎的搜索日志挖掘 [J]. 计算机应用研究，2008，25(6)：1663-1665.

[8] Liu Y Q，Cen R W，Zhang M，et al. Automatic Search Engine Evaluation Based On User Behavior Analysis [J]. Journal of Software，2008，19(11)：3023-3032.

[9] Zhang Zhitao，Yang Muyun，Li Sheng. Sogou Query Log Analysis：A Case Study for Collaborative Recommendation or Personalized IR [C]//International Asian Language Processing. [s.l.]：[s.n.]，2010：304-307.

[10] 张 磊，李亚楠，王 斌，等. 网页搜索引擎查询日志的 session 划分研究[J]. 中文信息学报，2009，23(2)：54-61.

[11] 王 攀，张顺颐，陈雪娇. 基于动态行为轮廓库的 Web 用户行为分析关键技术[J]. 计算机技术与发展，2009，19(2)：20-23.

[12] 王继民，彭 波. 搜索引擎用户点击行为分析 [J]. 情报学报，2006，25(2)：154-162.

[13] Liu Yuting，Gao Bin，Liu Tieyan . BrowseRank：Letting Web Users Vote for Page Importance [C]//the 31st Annual International ACM SIGIR Conference on Research & Development on Information Retrieval. Singapore：[s.n.]，2008.