

93-94/61 一种外联接优化的设计与实现

A Kind of Implementation of Outer Join Optimization

冯玉才 黄琳倩

(华中理工大学计算机科学与工程学院 武汉 430074)

TP311.13

Abstract In this paper, we present the strategy of outer join query optimization of DM3 database management system (DBMS), also present the implementation method of query optimization of DM3.

Keywords Outer join, DBMS, Query optimization

一、引言

关系代数中联接运算将两个关系中相匹配的元组结合起来,生成新的元组对。如果,一个关系中的元组在另一个关系中没有相匹配的元组,那么这个元组中的数据就不会出现在联接结果中。Codd 在 1979 年提出了关系数据库的扩展模型,引进了外联接 (outerjoin) 的概念,作为普通联接的补充。外联接保存了进行联接运算的两个关系中的一个或两个中的所需要的信息。

外联接已被列入 ISO-ANSI 的 SQL92 标准。目前流行的关系数据库管理系统如 Sybase, SQL Server, ORACLE 等均支持这种联接方式。在我们自行研制开发的数据库管理系统 DM3 中,也实现了这种联接方式,并研究了其优化方法。

对于 select/project/join 查询来说改变运算顺序是一种常用的有效的优化技术,在一般情况下可以大幅度地减少运算时间。在普通的联接运算时,可以利用其结合性和交换性变换运算顺序。但是,当查询中既有普通联接又有外联接时,问题就比较复杂,具体在于:(1)在进行运算顺序变换时,要决定每一步使用何种联接方式。(2)与普通联接不同,外联接并不满足结合性,这就造成了确定运算顺序的困难。为了解决这一问题,我们利用了 Galindo-Learia 提出的优化算法^[1],将其融入了 DM3 的联接查询的优化器中。

二、DM3 系统的联接优化策略

2.1 简化算符树

普通联接的处理相对比较简单,而且满足结合性,简化的目的是尽量减少外联接,并且将双向外联接转化为单向外联接。要达到这个目的,就需要重新改写查询中的联接组成。重写后使联接类型发生了变化,一些外联接将转化为普通联接,各种联接的个数顺序是:普

通联接最多,单向外联接其次,双向外联接最少,从而可以充分利用已有的技术,在 DM3 系统中我们采用的简化方法是根据 Galindo-Learia 提出的基于 NULL—拒绝的思想,重写联接。根据 Galindo-Learia 给出的两个基本等式:

$$\sigma_{p_1}(R_1 \xrightarrow{p_2} R_2) = \sigma_{p_1}(R_1 \xrightarrow{p_2} R_2) \quad (1)$$

$$\sigma_{p_1}(R_1 \xrightarrow{p_2} R_2) = \sigma_{p_1}(R_1 \xrightarrow{p_2} R_2) \quad (2)$$

if p_2 在 $\text{sch}(R_2)$ 中拒绝 NULL

可以看出,若进行联接运算的谓词满足非 NULL 条件,则联接运算可以得到一定的简化。

2.2 重新构造算符树

重排列查询次序,减小中间结构的规模,对数据量很大的表的联接查询,可以有效地提高查询效率。在 DM3 原有的优化器中,首先按算符树生成查询图,每个关系作为图中的一个结点,关系之间的联接运算以无向边表示。这样的查询图表明了关系之间是否存在联接,并没有指出哪个联接先做,哪个联接后做,但在此基础上可以方便地自底向上生成新的查询树,实现联接顺序的调整。算符树的生成算法如下:

算法 A: 生成算符树

输入: 具有 n 个关系的查询图 G

输出: 算符树

说明: 1. For (G 中的每个结点)

1. 创建一棵只有叶结点的树;

2. For ($k = n+1, \dots, 2n-1$)

{ 选择满足以下条件的子树 T_1, T_2 ,

(1) $\text{leaves}(T_1) \cap \text{leaves}(T_2) = \emptyset$

(2) $G \setminus \text{leaves}(T_1) \setminus \text{leaves}(T_2)$ 是连通的

(3) T_1, T_2 是 $k-1$ 个子树中根结点的规模最小的两个子树

2. 1 创建树 $T_k = (T_1 \odot T_2)$

}

3. 输出有 n 个叶结点的算符树

引入外联接后,由于外联接不满足普通联接的结合性,并非任意的查询次序都可以得到正确的结果。这

就需要确定哪种次序是可行的,并且满足优化的目的。因此,我们对算法 A 进行了一定的改进。

首先我们扩展了查询图,引进有向边,使其能表达外联接。在扩展了的查询图中,以无向边表示普通联接,单向箭头表示单向外联接,双向箭头表示双向外联接,这样的查询图不仅指出了关系之间是否存在联接,而且进一步指出了联接方式。

在算法 A 中,对于只有联接的查询,算符 $\odot$ 是 join,但对于包含联接和外联接的查询来说,算符 $\odot$ 则需要加以确定。Galindo-Learia 中提出了一种方法,在满足一定的约束条件时,能解决算符 $\odot$ 的确定问题。下面就引入 conf 集和 away 集的概念,对算法 B 的 2.1 步进行求精。

说明: FULL JOIN 的约束最强, LEFT (RIGHT) JOIN 的约束较弱, JOIN 的约束最弱。

定义 1 设查询图 G, 有路径 $P = e_M \cdots e_L, e_M$ 的约束比 e_L 强, 则称 e_M 与 e_L 冲突, 称 P 是冲突路径。其中 e_L 是 $\leftrightarrow$ 或 $\leftarrow$ ($\rightarrow$), P 中其他边是 $\rightarrow$ 或 $\leftarrow$ ($\rightarrow$)。

定义 2 设查询图 G, 对给定的边 e_M 的冲突集 $\text{conf}(e_M) = \{e_L | e_M \cdots e_L \text{ 是冲突路径}\}$ 。

定义 3 $G - e_L$ 得到 G 的两个子图 G_1, G_2 , 设 G_1 不包含边 e_M , $\text{away}_M(e_L) = \{G_1 \text{ 中的关系的属性}\}$ 。

算法 A 的 2.1 步求精:

1. 令 T_1, T_2 是由算法 A 的第二步所得的两个子树; E 是 G 中 $\text{leaves}(T_1)$ 和 $\text{leaves}(T_2)$ 之间的边; $e_0 \in E$; p 是 E 中联接谓词; $S = \text{leaves}(T_1) \cup \text{leaves}(T_2)$; $A_S = \{S \text{ 中的关系的属性}\}$; $\{e_1, \dots, e_m\} = \{e_L | e_L \in \text{conf}(e_0) \text{ 且 } e_L \text{ 是 } S \text{ 中两个结点的边}\}$;
2. switch(e_0)
 - case 双向边: 创建树 $T_s = (T_1 \leftrightarrow T_2)$;
 - break;
 - case 单向边: 假设 $e_0 = R_i \rightarrow R_j, R_i \in \text{leaves}(T_1)$
 - if ($m=0$) 创建树 $T_s = (T_1 \leftrightarrow T_2)$
 - else 令 $A_i = \{\text{leaves}(T_1) \text{ 中的关系的属性}\}$, 创建树 $T_s = T_1 \text{GOJ}[p, A_i, \text{away}_{e_0}(e_L) \cap A_S, \dots, \text{away}_{e_0}(e_m) \cap A_S]T_2$;
 - break;
 - case 无向边: if ($m=0$) 创建树 $T_s = (T_1 \odot T_2)$;
 - else 创建树 $T_s = T_1 \text{GOJ}[p, \text{away}_{e_0}(e_L) \cap A_S, \dots, \text{away}_{e_0}(e_m) \cap A_S]T_2$;
 - break;

三、DM3 系统中外联接优化的实现

根据传统的试探性优化原则, DM3 优化系统遵循以下规则:

- (1) 选择运算尽可能靠近算符树的叶结点;
- (2) 不删重复的投影尽早进行, 但不先于选择;
- (3) 删重复的投影尽量向算符树的根部靠近;
- (4) 多个联接运算的计算顺序参考参加联接的关系的代价决定。

在 DM3 系统中, 对于外联接的优化尽量利用已有

的查询优化系统, 嵌入对外联接的优化, 并对原系统作相应修改。

引入了外联接后, 投影操作不会受到影响, 而选择操作则不然。由等式 (1) 和 (2) 可知, 选择运算可用于降低联接的约束。对于外联接, 只有在选择谓词拒绝 NULL 时, 才能被下移向叶结点。

进行联接优化时, 我们首先对联接表达式进行简化, 降低外联接的约束, 得到新的算符树来表示相对简单的联接表达式。对这个表达式的计算就可以充分利用已有的优化方法。

然后根据新的算符树, 生成扩展的有向查询图。根据改进后的算法 A, 由查询图自底向上重新构造算符树。在有的优化器中, 列举所有的联接排列, 选择最佳联接顺序, 这样做增加了查询优化的开销, 甚至有时比查询的执行时间长。产生这样的结果的原因在于优化搜索空间太大。因此, 有的优化器为了削减搜索空间, 采用 "left-leaning" 树, 即只允许联接的左边是前一联接的结果。这样的顺序选择方法有可能造成较大的中间结果。在 DM3 系统中, 为了兼顾搜索空间和联接代价两个方面, 我们采用折衷方法, 每次选择根结点规模最小的两棵子树进行联接, 产生新的一棵子树, 这样的方法, 保证每次进行的联接代价较小, 即局部代价优化, 算符树生成的同时确定联接顺序, 而不考虑所有的联接方案。

但是, 在进行外联接优化时, 无论是算符树的简化还是上述算法 A 的 2.1 步求精, 其依据的转化等式并不总是成立的, 其必须满足以下条件:

(1) 外联接的谓词对其所涉及的关系中的属性拒绝 NULL;

(2) 所有的外联接的谓词仅涉及两个关系中的属性, 即外联接是二元的。

可见, 这种优化方法并不是在任何情况下均成立。为此, 优化系统中进行了条件判断, 只对满足上述条件的外联接或没有外联接的联接表达式才进行优化, 否则按照联接表达式中的顺序进行联接运算。判断方法较为简单, 针对条件 (1), 判断谓词中是否有谓词 IS-NULL (属性), 对于条件 (2), 对谓词所涉及的不同关系进行计数, 若大于 2 则不满足条件。目前对外联接优化的研究中, 有的方法可以减少约束条件, 能进行较复杂的谓词优化, 但是算法复杂, 开销较大, 不易实现。在对比了优化的普遍适用性和优化效率、实现的困难程度后, 我们采用了这种较为简单的方法, 尽管其对外联接的优化是不完全的。

结论 外联接是 SQL92 标准的重要运算, 但目前的许多数据库管理系统对外联接的优化支持较弱。在

(下转第 61 页)

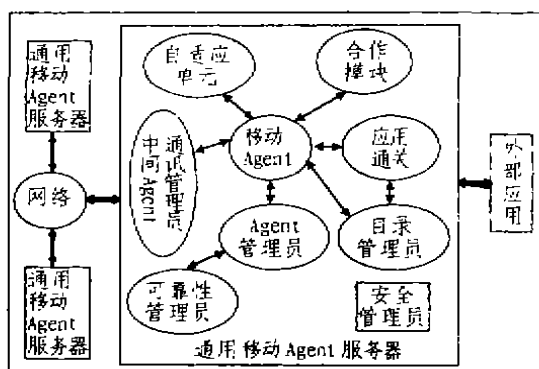


图2 通用移动 Agent 系统结构

的变化,满足用户对服务质量的需求。

·合作模块实现 Agent 与其它移动 Agent 之间的通讯合作,当移动 Agent 接受的任务超出它的求解能力时,移动 Agent 通过合作模块完成多个移动 Agent 间的任务求解。

·中间 Agent 通讯管理员在 Java 和其它系统中促进分散在网络各处的移动 Agent 的通讯,最简单的应用是实现多 Agent 计算和多相关 Agent 集成通讯,移动 Agent 系统典型工作是提供信息和分布事件。

·应用网关服务作为 Agent 与应用服务器(如遗留数据库)进行交互的安全入口。Agent 可利用基于 JNDI 的目录管理员验证应用服务器的位置并移动到服务器所在的主机,通过网关访问常驻服务器,安全管理利用网关认证 Agent 和应用服务器。这个通用结构对于多数应用领域是足够的,相信扩展和提高分布计算技术将使移动 Agent 更有效。当前 Agent 框架支持具有移动性、自主性、传递性和适应性的分布系统的实现。

结束语 目前,国际上一些高等院校和研究机构对移动 Agent 技术进行大量的研究,并开发了应用领域的移动 Agent 系统,如 Dartmouth 学院开发的

Agent TCL 移动 Agent 系统^[6],IBM 公司用 Java 语言开发的 Aglet 移动 Agent 模型^[11],Stuttgart 大学基于 Java 开发的 Mole^[10],Ottawa 大学多媒体和移动 Agent 研究室的 SHIP-MAIL^[12]等。随着这些实用系统的研究与开发,移动 Agent 技术将日趋成熟,Internet 与移动 Agent 技术的结合将给远程学习、电子商务和多媒体通讯领域带来无限发展机遇。

参考文献

- 1 Pham V A, Karmouch A. Mobile Software Agents An Overview. IEEE Commun. Magazine, 1998(7): 26~37
- 2 Wong D, et al. Java-based Mobile Agents. CACM, 1999, 42(3): 92~102
- 3 Jamah N, et al. An Actor-Based Architecture for Customizing and Controlling Agent Ensembles. IEEE Intelligent Systems, April, 1999
- 4 Sandholm T. Agents in electronic commerce. Component technologies for automated negotiation and coalition formation. In: Proc of the Second Workshop on Cooperative Information Agents (Paris, France, July 4-7). Springer-Verlag, Berlin, 1998. 113~134
- 5 Shebory O, et al. Agent Cloning an Approach to Agent Mobility and Resource Allocation. IEEE Commun. Magazine, 1998(7): 58~66
- 6 Kotz D, et al. AGENT TCL: Targeting the Needs of Mobile Computers. IEEE Internet Comp., 1997, 4(1): 58~67
- 7 Rothermel K, et al., eds. Mobile Agents, Lecture Notes in Comp. Sci., Series, vol. 1219, Springer, 1997
- 8 Morreale P. Agents on the Move. IEEE Spectrum, April, 1998
- 9 Arisha K A, et al. Impact: A Platform for Collaborating Agents. IEEE Intelligent Systems, April, 1999
- 10 Available at: <http://www.informatik.uni-stuttgart.de/ipvr/vs/projecte/Ara/index.e.html>
- 11 Available at: <http://www.trl.ibm.com/aglets>
- 12 Available at: <http://www.deneb.genie.uottawa.ca>

(上接第94页)

DM3系统中我们实现了外联接,并利用文[1]中提出的算法思想,实现了外联接的优化,在联接顺序的选择上,采用了折衷的方法,进行局部优化,减小搜索空间。在这个优化系统中,待改进的地方在于因所用的算法要求联接的谓词拒绝 NULL,并不适用于所有的查询,因此需要更新算法来减少约束。

参考文献

- 1 Galindo-Learia, Arnon Rosenthal. Outerjoin Simplification and Reordering for Query Optimization[J]. ACM Transactions on Database Systems, 1997, 22(1)
- 2 Selinger P G. Access path selection in a relation database system[C]. In: Proc. of ACM SIGMOD 1979 Intel. Conf. on Management of Data
- 3 周龙骧. 数据库管理系统实现技术[M]. 中国地质大学出版社