第6卷第3期 软 件 学 报 Vol. 6, No. 3
1995年3月 JOURNAL OF SOFTWARE Mar. 1995

PROLOG－DBMS系统实现中的子句间优化技术*

李 磊 左万历 李希春

（中山大学计算机科学系，广州 510275） （吉林大学计算机科学系，长春 130023）

摘要 实用PROLOG－DBMS系统实现中的首要问题是要保证系统的正确性. 然而，从数据库中提取冗余数据不但严重地影响系统的效率，而且影响系统的正确性. 本文所提出的子句间优化技术可以解决这一问题. 因为采用这一技术可以保证仅从数据库中提取有用事实并且没有任何元组多于一次被提取.

关键词 PROLOG，数据库，演绎数据库.

目前，连接PROLOG与DBMS已不再是一个新鲜课题. 自从80年代以来，尤其是1984年以后，人们对于PROLOG与DBMS耦合的意义已经有了充分的了解和认识，并且各种实现方法、许多原型系统也相继问世. 但是由于正确性、效率、用户友好性等因素，目前尚不能说PROLOG与DBMS的连接技术已完全成熟或说已达到实用水平，因为原型系统到实用系统还相距甚远. PROLOG－DBMS系统实现中的首要问题是要保证系统的正确性. 一个PROLOG－DBMS系统是正确的，如果它与PROLOG系统(DBMS中的数据作为PROLOG的事实)对任何的一用户程序的同一目标的执行结果一致. 或者说程序的非规则部分，无论是作为PROLOG中的事实还是作为DBMS中的关系都不影响程序的语义. PROLOG－DBMS系统实现中可以采用一次一个元组的策略. 在这样的系统中，由于PROLOG标准SLD归结策略没有改变，因此正确性不成问题. 虽然这种方式被广泛采用，但是由于效率方面原因，本文不加以讨论.

保持系统的正确性是实现PROLOG－DBMS系统最基本的目标之一. 但在目前大多数PROLOG－DBMS系统的实现中却没有引起充分的重视，也很少有文献讨论这一问题，因此可以认为在这样的系统中正确性要由用户保证. 由于标准的PROLOG系统没有提供对自动回溯指针进行控制的原语，因此一般用户对于保证系统正确性是无能为力的，而且由用户保证正确性也不符合PROLOG的说明性原则.

* 本文1992－11－10收到，1993－03－26定稿

此工作部分由霍英东教育基金资助. 作者**李磊**，1951年生，教授，主要研究领域为逻辑程序设计. **左万历**，1961年生，讲师，主要研究领域为逻辑程序设计. **李希春**，1962年生，讲师，主要研究领域为逻辑程序设计与数据库.

本文通讯联系人：李磊，广州 510275，中山大学计算机科学系

1 问　题

PROLOG－DBMS 系统的实现方法可分为编译类(例如,编译、半编译方法)和解释类(例如,解释、半解释方法)[11].

由于编译类方法不可能达到"提取(从 DBMS 中)＝所需(在 PROLOG 执行中)",即可能提取重复事实又可能提取无用事实,所以本文仅讨论解释类方法,因为解释类方法可自动地克服编译类方法中提取无用事实的不足.

一个 PROLOG－DBMS 系统的实现方法是解释类的,如果数据库访问发生在 PROLOG 程序的执行期间.

一个 PROLOG－DBMS 系统的实现方法是半解释的,如果它是解释类的,并且数据库访问的基本单位(一次数据库调用)是 EDB 谓词的合取并且其长度(EDB 谓词的个数)≥1.

显而易见,当 EDB 谓词合取的长度＝1 时即为解释方法.

为了叙述方便且不失一般性,下面的讨论均以解释方法为例.

由于在解释过程中对于数据库的访问是顺序进行的,因此某些访问结果的交集可能非空.

假定 $edb(f(a,Y))$和 $edb(f(X,Y))$是 *EDB* 谓词,$edb_call(f(a,Y))$和 $edb_call(f(X,Y))$是对应的数据库提问,$\{f(a,Y)\}$和$\{f(X,Y)\}$表示执行提问得到的结果.

因为与 $f(X,Y)$相匹配元组所构成的集合包含与 $f(a,Y)$相匹配元组所构成的集合,所以有 $\{f(a,Y)\}\subset\{f(X,Y)\}$.

显然,如果 $edb_call(f(a,Y))$先于 $edb_call(f(X,Y))$执行,则执行 $edb_call(f(X,Y))$时只需提取 $\{f(X,Y)\}\subseteq\{f(a,Y)\}$. 如果顺序交换,则 $edb_call(f(a,Y))$是不需要的. 这说明,如果每次数据库访问都独立执行,则 $\{f(a,Y)\}$ 可能被多次提取,这就是使得 PROLOG－DBMS 系统正确性难以保证和效率受到影响的主要原因. 子句间优化的目标就是为了:考查各次数据库访问之间的关系;确定每次访问数据库的必要性;形成仅提取必要数据的数据库访问.

2 数据库访问之间的关系

定义 2.1. 设 A,B 是原子公式,说 A,B 是相关的,如果存在替换 θ 和 δ,使得 $A\theta=B\delta$,否则说它们是无关的.

定义 2.2. 设 A,B 是原子公式,说 A 包含(*subsume*)B,如果存在替换 θ,使得 $A\theta=B$.

例 2.1:　$edb(f(X,Y))$包含 $edb(f(a,Y))$;

$edb(f(a,Y))$包含 $edb(f(a,b))$.

显然,若 $edb(A)$包含 $edb(B)$,则$\{B\}\subseteq\{A\}$. 因为对于任何元组 $t\in\{B\}$,假定 $B\delta=t$,由于 $edb(A)$包含 $edb(B)$,因此 $A\theta=B$,于是 $A\theta\delta=t$,所以 $t\in\{A\}$,故$\{B\}\subseteq\{A\}$.

易见,如果 $edb(A)$,$edb(B)$ 无关,则$\{A\}\cap\{B\}=\varnothing$.

假设 $edb\_call(B)$,$edb_call(A)$是相继执行的,考虑一般情况有:(1)$edb(A)$与 $edb(B)$无关;(2)$edb(B)$包含 $edb(A)$;(3)$edb(A)$与 $edb(B)$相关且(2)不成立.

情况(1)时,应提取{A};情况(2)时,$edb_call(A)$是不必要的;情况(3)时,只需提取{A}−{B}.

3 meta 库管理

为了考查数据库访问之间的关系,需要对数据库访问的历史情况加以记载,这些信息可以用 POLOOG 事实来表示.例如:

$$meta\_base(A_1), meta\_base(A_2), \cdots, meta\_base(A_n).$$

这些事实的整体被称为 *meta* 库,每个事实中的 $A_i(i=1,\cdots,n)$叫作历史.

如果将所有数据库访问情况均加以保存,则 *meta* 库中事实的数量将随着程序的执行而迅速地增加,这可能引起内存空间溢出,同时影响效率.实际上这也是不必要的.

令 $edb_call(A_c)$为当前数据库访问,A_g, A_1,…,A_n 为历史并且 A_g 包含 $A_i(i=1,\cdots,n)$.考虑一般情形:

(1)A_c 与 A_g 无关,则 A_c 与 A_i 无关;

因为假定 A_c 与 A_i 相关,则有 θ 和 δ,使得 $A_c\theta=A_i\delta$,由于 A_g 包含 A_i,于是存在 π 使得 $A_g\pi=A_i$,故有 $A_c\theta=A_g\pi\delta$,这同 A_c 与 A_g 无关相矛盾.

(2)A_g 包含 A_c;

(3)A_c 与 A_g 相关,且(2)为假,则{A_c}−{A_g}={A_c}−{A_1}−…−{A_n}.

因为 A_g 包含 A_i,所以{A_i}⊆{A_g}($i=1,\cdots,n$),于是{A_g}∪{A_1}∪…∪{A_n}={A_g},故有{A_c}−{A_g}−{A_1}−…−{A_n}={A_c}−({A_g}∪{A_1}∪…∪{A_n})={A_c}−{A_g}.

情形(1)时需提取{A_c};情形(2)时 $edb_call(A_c)$是不需要的;情形(3)时只需提取{A_c}−{A_g}.总之,如果 A_i 被某一 A_g 包含,所有情形都与 A_i 无关.

事实上,对于任何历史 A_i,如果有历史 A_g,使得 A_g 包含 A_i,则 A_i 在 *meta* 库中是冗余信息.

meta 库管理算法:

设 $edb_call(A)$为当前数据库提问,$B_i(i=1,\cdots,n)$为历史,如果有 B_i 包含 A,则 A 不加入到 *meta* 库中;否则,删除所有 B_i,如果 B_i 被 A 所包含,并将 A 加入到 *meta* 库中.

此算法可用 PROLOG 描述为:

$history(A,H):-setof(A,meta\_base(A),L),!,test(A,L,H).$

$history(A,[]):-append\_clause(meta\_base(A)).$

$test(A,L,[]):-subsumed(A,L),!,fail.$

$test(A,L,L):-clear(meta\_base(A)),append\_clause(meta\_base(A)).$

其中 *subsumed* 测试是否有 $B_i\in L$,使得 B_i 包含 A; *clear* 从 *meta* 库中删除所有 B_i,如果 A 包含 B_i; *history* 参数中的 H 返回 *meta* 库中所有与 A 相关但不包含 A 的元组构成的集合,用于下一步求差运算.

4 差的提取及优化

下面讨论当前提问所对应的事实已部分地装入内存的情况.

假设 $edb_call(A)$为当前数据库提问，$A_1,\cdots,A_n$ 为历史，并且 A 与 $A_i(i=1,\cdots,n)$相关但 A_i 不包含 A. 这说明 A 所对应的事实已部分装入内存，因而仅需由 DBMS 中提取 $\{A\}-(\{A_1\}\cup\cdots\cup\{A_n\})$或者$\{A\wedge\neg A_1\wedge\cdots\wedge\neg A_n\}$它可由下述三种之一获得：

(1)利用大多数关系数据库系统所提供的差运算；

(2)形成临时关系$\{A\}$，然后由$\{A\}$中分别删除$\{A_i\}(i=1,\cdots,n)$；

(3)直接执行与$\{A\wedge\neg A_1\wedge\cdots\wedge\neg A_n\}$对应并优化了的数据库提问.

考虑效率方面因素，我们选用了方法(3).

实际上，$\{A\wedge\neg A_1\wedge\cdots\wedge\neg A_n\}$可表示为：

$$\{t_1',\cdots,t_m'\mid A(t_1,\cdots,t_m)\wedge\neg A_1(t_{11},\cdots,t_{1m})\wedge\cdots\wedge\neg A_n(t_{n1},\cdots,t_{nm})\}$$

其中 $t_k'(k=1,\cdots,m)$为新引入的变量.

由于 A 与 A_i 相关，因此它们对应于同一数据库关系，于是上式可化简为：

$$\{t_1',\cdots,t_m'\mid A(t_1',\cdots,t_m')\wedge(t_1'=t_1\wedge\cdots\wedge t_m'=t_m)\wedge\neg(t_1'=t_{11}\wedge\cdots\wedge t_m'=t_1m')\wedge\cdots\wedge\neg(t_1'=t_{n1}\wedge\cdots\wedge t_m'=t_{nm})\}$$

即为：$\{t_1',\cdots,t_m'\mid A(t_1',\cdots,t_m')\wedge(t_1'=t_1\wedge\cdots\wedge t_m'=t_m)\wedge(t_1'\neq t_{11}\vee\cdots\vee t_m'\neq t_{1m})\wedge\cdots\wedge(t_1'\neq t_{n1}\vee\cdots\vee t_m'\neq t_{nm})\}$

考虑当 t_{ij}为变量时$(i=1,\cdots,n;\ j=1,\cdots,m)$，$(t_k\neq t_{ij})\equiv$“假”$(k=1,\cdots,m)$，当 t_l 为变量时$(l=1,\cdots,m)$，$(t_k'=t_l)\equiv$“真”$(k=1,\cdots,m)$，它们可由上式中去掉，表达式得以进一步化简.

例 1：　$A=act(a,Y,Z)$，　$A_1=act(a,b,Z)$

由上述化简，$\{A,\neg A1\}$可直接表示为：

$$\{X,Y,Z\mid act(X,Y,Z)\wedge X=a\wedge(X\neq a\vee Y\neq b)\}$$

但是由于

$$X=a\wedge(X\neq a\vee Y\neq b)\equiv X=a\wedge Y\neq b$$

所以上述表达式还可以进一步化简，并容易证明：

令 A 为当前数据库提问，A_i 为历史，如果 A 的第 j 个参数为常数，则 A_i 的第 j 个参数的内容对它们的差所对应的关系表达式没有任何影响.

根据这一结论，下面两例：

例 2：　$A=act(X,b,Z)$，$A1=act(a,b,c)$

例 3：　$A=act(X,b,Z)$，$A1=act(a,Y,c)$

其差具有相同的关系运算表达式：

$$\{A,\neg A1\}\equiv\{X,Y,Z\mid act(X,Y,Z)\wedge Y=b\wedge(X\neq a\vee Z\neq c)\}$$

总结上述分析，我们可以得到一个非常简化的求差$\{A,A_1,\cdots,A_n\}$算法.

差优化算法：

步骤 1：根据 A 形成二元组集合　　$\{(x,i),(y,j),\cdots,(z,k)\}$

其中$\{x,y,\cdots,z\}=VAR(A)$并且 $i,j,\cdots,k$ 分别为代表参数 $x,y,\cdots,z$ 在 A 中位置的整数；

步骤 2：对于每个 A_i 做步骤 3；

步骤 3：对于每个$(x',i')\in\{(x,i),(y,j),\cdots,(z,k)\}$ 做步骤 4；

步骤 4：如果 A_i 的第 i' 个参量为常数 C，则构造$(x'\neq C)$并以“$\vee$”相连加到 A_i 的负子表达式中；

步骤 5：所有负子表达式以“$\wedge$”相连加到表达式中；

步骤6:根据 A 构造正子表达式并以"∧"相连加到表达式中(正子表达的构造方法是简单的).

例4: 令 $A=act(a,Y,Z)$, $A1=act(a,b,Z)$, $A2=act(X,b,c)$

则二元组集合为{(Y,2),(Z,3)}; $A1$,$A2$ 对应的负子表达式是分别为$(Y\neq b)$和$(Y\neq b \vee Z \neq c)$; A 对应的正子表达式为$(X=a)$;差优化结果为:

$$\{X,Y,Z \mid act(X,Y,Z) \wedge (X=a) \wedge (Y\neq b) \wedge (Y\neq b \vee Z\neq c)\}$$

对应的 *SQL* 提问为:

select X,Y,Z

from *act*

where $X=a$ and $Y\neq b$ and $(Y\neq b$ *or* $Z\neq c)$

5 结束语

CERI 于 1985 年提出了子句间优化的基本思想[2],但在文章中仅对 EDB 谓词之间的包含关系进行了讨论,而系统并没有真正完全实现.尽管如此,他的工作对我们有很大启发.

本文从更广的角度讨论了数据库提问之间的关系及其相关关系.我们还具体地在 SUN 工作站上实现了上述全部算法,并将其用于 PROLOG—DBMS 原型系统中,此系统是欧共体 EPSILON(ESPRIT—P530)计划研制的分布式知识库管理系统的核心部分.

显而易见,上述讨论对于半解释方法,即 EDB 表长度>1 时,也是正确的[10].

实验结果表明,采用子句间优化技术彻底避免了重复事实的提取,从而保证了系统的正确性,这对于具有聚集操作的 PROLOG 程序至关重要.在效率方面,本系统与仅采用 CERI 子句间优化技术的系统相比,通常情况下访问数据库的次数相当,但每次提取的数据量大为减少;与不采用子句间优化技术的系统相比,访问数据库的次数和每次提取的数据量都大大地减少.因此,本系统在时间、空间效率上都有显著的提高.

目前此系统在实用化方面也已取得相当程序的进展,一个具有较好应用价值基于部分求值和子句间优化技术的 PROLOG—DBMS 系统已基本完成,所使用的 PROLOG 和 DBMS 分别为 Quintus—PROLOG 和 INGRES.

致谢 法国的 J. Kouloumdjian 教授曾指导过这一研究工作,在此表示感谢.

参考文献

1 Veken R. A PROLOG meta—interpreter for partial evolution and its application to source to source transformation and query optimization ECAI, 1984.

2 Ceri S, Gottlob G, Wiederhold G. Interface relational database and PROLOG efficiently. Raporto interno, No. 85—23, 1985.

3 Ceri S, Gottlob G, Lavazza L. Translation and optimization of logic queries: the algebraic approach. Proc. of the 12th VLDB, Kyoto, 1986.

4 Bernier J *et al*. Convergences des bases de donnees et dessystemes experts. M. B. D. Volumn 5, December 1986. 17—45.

5 Coscia P, Franceschi P, Kouloumdjian J *et al*. The EPSILON DBMS: architecture and DB access optimization.

Workshop on Integration of Logic Programming and Databases, Venice, Dec. 1986.

6 Gardarin G, Simon E. Les systems de gestion de bases de donnees dedutives. TSI, 1987, 6(5).

7 Holl G H. Un language privot pour le couplage de PROLOG avec des bases de donnees. these de doctorat, Oct. 1987.

8 Li Lei, Moll G H, Kouloumdjian J. PROLOG DBMS coupling: an hybrid approach, half interpreted, half compiled. The 3rd International Conf. on Data and Knowledge Bases, Jerusalem, Israel, June 1988.

9 Li Lei. L'Evaluation partielle et son application pour coupler Prolog et les SGBD, These de doctorat, 1988.

10 李磊等. 一个实用PROLOG—DBMS系统的实现. CJCAI, 1990. 368—375.

11 李磊. PROLOG与DBMS系统连接的实现方法. 计算机学报, 1992, 15(2).

THE INTER—CLAUSE OPTIMIZATION TECHNIQUE IN THE IMPLEMENTATION OF PROLOG—DBMS SYSTEMS

Li Lei

(*Department of Computer Science, Zhongshan University, Guangzhou* 510275)

Zuo Wanli　Li Xichun

(*Department of Computer Science, Jilin University, Changchun* 130023)

Abstract　In the implementation of practical PROLOG—DBMS systems, the key issue is to maintain its correctness which is, unfortunately, often affected by extracting redundant facts from data base management systems. This paper presents a new technique called "inter—clause optimization" which can not only keep the system correct but also increase its efficiency by guaranteeing that only useful facts are loaded from DBMS and no fact is loaded more than once.

Key words　PROLOG, DBMS, deductive database.