编译型PROLOG数据库的结构模型及其分析

车敦仁 （100083北京航空航天大学计算机科学系）

摘 要

In this paper, the compiler-based PROLOG database structures are separated into four categories on the different levels at which two sub-databases, CDB and SDB, are coupled. And further, they are abstracted into four models, i.e. MTCM(Most Tightly Coupled Model), TCM(Tightly Coupled Model), LCM(Loosely Coupled Model) and MLCM (Most loosely Coupled Model). TCM, the best, si firstly introduced, then the other three models are presented, and at last the four models are analysed and compared in detail on their space needed, management cost, searching and compiling time, particularly on their convenience of implementation, hoping to give some assistance to those developers of PROLOG systems.

作为一种非常重要的人工智能系统构造工具，PROLOG是近十年来被研究得最多的一门程序设计语言。当今，围绕PROLOG的研究已经经历由解释实现到编译实现，由顺序

不再是语句而是一个响应数据集，因此转换过程实现的是数据到相应知识的转换。

五、结束语

AI与DB技术的结合既是早期计算机领域的一个研究课题，又是智能系统实用化的一个新的研究方向，AI正在向通用系统的方向转移，而DB技术则在向特殊应用领域的方向转移，以扩展其功能。两者的逐步渗透必将导致新型的智能信息处理系统的出现。本文给出的基于框架的耦合知识系统模型FBC是作者在这方面的尝试，还有不少细节问题有待进一步发展，恳望同仁不吝赐正。

参考文献

[1] Turning Data Into Knowledge, IntelliCorp, 1989

[2] 马立兴，顾学春，数据库与知识库的结合策略，计算机科学，1988，6

[3] 韩建超，史忠植，知识库系统研究，计算机科学，1988，6

[4] 吴信东，智能数据库研究，中国人工智能90，吉林大学出版社，1990

[5] 萨师煊等，面向新的应用领域的数据库技术，计算机科学，1989，4

[6] M.L.Brodie，未来的智能信息系统：AI与DB技术的结合，计算机科学，1989，3

[7] 姚卿达，纪岳，数据模型智能化扩充研究，计算机科学，1989，3

[8] L.Kerschberg, Expert Database Systems, Proc.of the 1st Intenal. Workshop on Expert Database Systems, U.S.A, 1986

[9] H.Craig Howard, D.R.Rehak, KADBASE: Interfacing Expert Systems with Databases, IEEE Expert, FALL 1989

[10] Yannis Vassiliou, Knowledge-Based and Database Systems, Enhancements, Coupling or Integration? On Knowledge Base Management Systems

[11] J.D.Ullman, Principles of Database Systems, Computer Science Press, 1982

PROLOG机到并行PROLOG机的过渡，追求PROLOG系统的运行效率一直是这些研究的驱动力。PROLOG系统的编译实现是研究PROLOG推理机(包括顺序推理机和并行推理机)的基础，但由于PROLOG子句具有作为程序(被运行)和作为数据（被操作，亦称库操作）的双重特性，使得PROLOG系统的编译实现完全不同于一般算法语言的编译实现。它要求同时持有子句的两种形式，即编译后的代码形式(用于运行)和编译前的源形式(亦称源子句或子句项，用于数据库操作)。如何高效地保持子句的两种形式，直接关系到PROLOG程序的运行效率和数据库操作的快速实现，这是PROLOG编译实现的一个核心课题。

PROLOG数据库实质上由两个子库构成，即代码库CDB(Code DataBase)和源子句库SDB(Sourceclause DataBase)。本文按照CDB与SDB相互耦合的不同程度，将PROLOG数据库的结构分成四类并抽象成四个模型，分别命名为最紧耦合模型MTCM(Most Tightly Coupled Model)，次紧耦合模型TCM (Tightly Coupled Model)，次松耦合模型LCM(Loosely Coupled Model)和最松耦合模型MLCM (Most Loosely Coupled Model)。文章首先介绍了TCM模型，接着介绍了另外三种模型，最后，着重从支持数据库操作谓词的快速实现上将四种模型的主要特性进行了详细的分析比较。

一、次紧耦合模型(TCM)

TCM是基于WAM[1]的顺序PROLOG推理机实验模拟系统YH-SIM[4]中所采用的数据库模型。为了有效地支持数据库操作，在YH-SIM中对WAM的指令集进行了扩充[5]，对有些原有指令进行了修改并新增加了一些指令，其中有两条关键的说明性指令，使数据库操作谓词(简称DBOP)的实现具有高效性和简洁性。这两条指令是：

Pdb指令：procedure PDB_i

Cdb指令：clause CDB_j

其中procedure和clause分别为两条指令的操作码，PDB_i(Procedure Declare Block)和CDB_j(Clause Declare Block)分别为两条指令的操作数。

PDB_i作为过程定义表PDT中过程i的表项，含有相应过程的有关说明信息，如过程名Name，过程的变元数Arity，过程的子句数Nclause，指向过程代码的指针Pcode等。而CDB_j起着联接子句j的代码块(Code Block)与源子句块(Term Block)的界面，含有相应子句的说明信息，如子句类型标志Clause-tag，子句删除标志Doomed-tag，子句代码长度Code-length，指向子句代码的指针Code-Ptr，子句项的大小Term-length，指向子句项的指针Term-Ptr等。由于CDB_j在CCB(Clause Code Block)与CTB(Clause Term Block)间的中介作用(在数据库中CDB_j是第j条子句的前缀)，使每个过程的CCB与CTB可共享同一索引框架，这给从CCB到CTB的查询和因数据库操作而需要的索引重构都带来了方便，从而使DBOP的实现具有高效性和简洁性。简洁性体现在DBOP可借助PROLOG系统的正常执行机制，用PROLOG编程实现。TCM是在子句一级将CDB与SDB联系起来，又名子句级耦合模型。

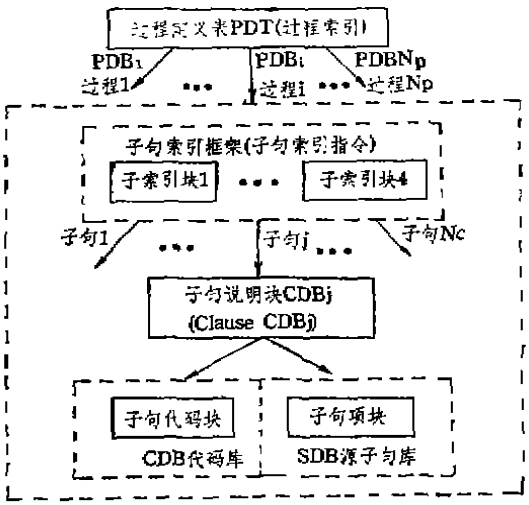

图1 次紧耦合模型（TCM）

二、其它模型

1.最紧耦合模型(MTCM)

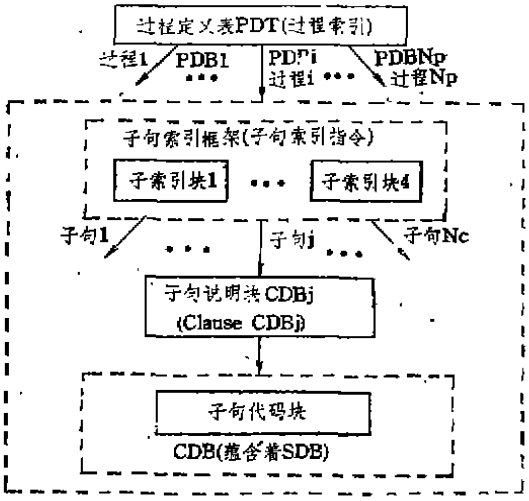

图2 最紧耦合模型(MTCM)

在基于编译的PROLOG系统中，获取源子句的方法至少有三种[2]。TCM中采用的是将子句项与子句代码并存的方法，MTCM则依赖于获取子句项的反编译方法。在该模型中，CDB与SDB紧耦合成一个实体，或者说源子句库SDB只是一个虚拟意义上的数据库，当进行数据库操作需要子句的源形式时，在另一种执行状态下(如反编译状态)，按照另一种操作语义[3](在反编译状态下，指令的操作语义与正常执行时不同)执行子句的代码块，从而获得子句的源形式。MTCM是在代码一级将CDB与SDB耦合在一起，亦称为代码级的耦合模型。

2.次松耦合模型(LCM)

LCM是在过程一级将子句的代码形式与源形式联系起来，所以又称为过程级的耦合模型。在LCM中，表示源子句的方法除了并存子句代码与子句项(如图3所示)外，还可采用并存子句代码与项代码方法，这是Source谓词方法[2]的一种变体形式，整个系统不是只设置一个主谓词Source

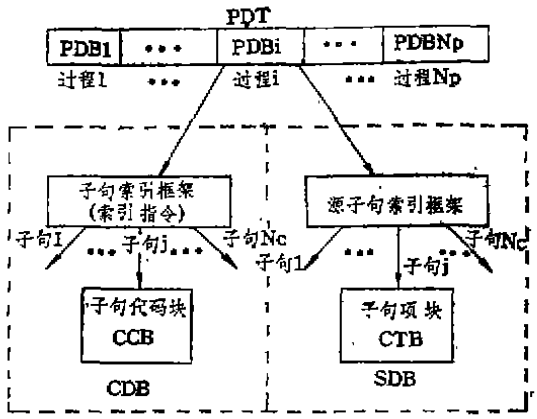

图3 次松耦合模型(LCM)

/5，而是要对每个过程，比如"abc/2"，都指定一个相关的主谓词，如"Source-abc-2/5"，这样可以减少一级源Source方法中的过程索引。所以说，LCM也是最松耦合模型的改进型。

3.最松耦合模型(MLCM)

MLCM是一种基于表示子句的并存子句代码与项代码方法[2]的数据库模型。这种模型直接取自于Clocksin的Source谓词方法，该方法为了得到子句的源形式，专门设置一个主谓词Source/5，每当编译生成一条子句的代码之后，紧接着再以此子句本身作为Source谓词的变元之一，将Source子句再进行一次

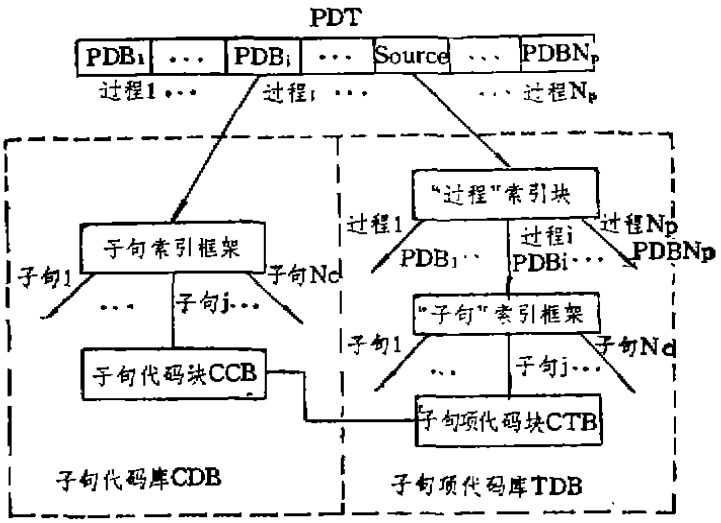

图4 最松耦合模型(MLCM)

编译，生成的代码称为项代码。当进行库操作需要子句的源形式时，执行相应的项代码动态生成所需的源子句。图4的项代码库TDB中"过程"索引框架和"子句"索引框架实际上都是Source谓词的子句索引框架，"过程"索引框架是对除了过程Source以外的其他过程(PDB_1，…，PDB_{N-1})的索引，"子句"索引框架是对任一过程PDB_i的子句的索引。子句项代码库TDB是变象的源子句库SDB。最松耦合模型可以认为是在过程级之上的(通过Source谓词)耦合模型。

三、四种模型的比较

耦合类型	最紧耦合	次紧耦合	次松耦合	最松耦合
空间开销	1	2	2^+	2^{++}
查找开销	2	2	3	5
管理开销	2	3	4	4^+
编译时间开销	1	1	1或2	2
实现的方便性	最差	好	差(或较好)	较好

图5 四种模型的比较

1.空间开销

空间开销是指整个数据库所占用的内存空间。造成空间开销的主要因素是子句代码块和子句项块，子句索引框架(索引指令)则是次要因素。MTCM中由于只有子句代码块和一套子句的索引框架，其空间开销最小，不妨设其为1；TCM中由于子句项与子句代码相并存，空间开销加倍，记为2；LCM比起TCM又多了一套子句项的索引框架，我们记其空间开销为2^+；MLCM比TCM又多了一级项代码库中的"过程"索引，空间开销有相应的增加，我们记为2^{++}。

2.查找开销

查找开销是指在进行数据库操作时查找到子句的代码块和对应的子句项(或项代码)块所导致的时间开销。决定查找开销大小的主要因素是索引的层数。查找一个给定过程的子句的代码块和子句项块(或项代码块)，在MTCM中，需要经过一级过程索引和一级子句索引，记其开销为2；TCM与MTCM的索引规模完全相同，并存的子句代码块与子句项块共享同一索引框架，其查找开销与MTCM的相同，亦记为2；在LCM中，子句代码与子句项各自拥有一套独立的子句索引框架，二者共享过程索引框架(PDT)，查找子句代码与子句项时，需经过一次过程索引，一次子句索引和一次子句项索引，三次索引的总开销记为3；在MLCM中，查找子句代码块时需经两次索引，一次过程索引和一次子句索引；查找子句项代码时需经三次索引，第一次过程索引(找到Source谓词)，第二次"过程"索引(找到过程PDB_i的项代码)和一次子句项代码索引；整个查找共需5次索引，总的查找开销记为5。

3.管理开销

管理开销是指在进行数据库操作时，因修改数据库而导致的开销。管理开销可分为两个方面，一是重构索引框架的开销，二是插入或删除子句时，申请或释放数据库空间的开销。在本文讨论中，假定这两方面的管理开销均等。对于MTCM插入或删除一条子句需重构子句索引框架一次，申请或释放子句代码块空间一次其总开销记为2；对于TCM，插入或删除一条子句需重构子句索引框架一次，申请或释放子句代码块空间和子句项空间各一次，其总管理开销记为3；对于LCM，插入或删除一条子句需重构子句代码索引框架和子句项索引框架各一次，申请或释放子句代码块空间和子句项空间各一次，其总管理开销记为4；对于MLCM总管理开销除包括LCM的全部管理开销外，还可能涉及到一次"过程"索引框架的重构(当一个过程的唯一一条子句被删除或插入一个新过程时)，所以其管理开销应有所增加，记为4^+。

4.编译时间开销

编译时间开销是指把源程序(PROLOG子句)编译成目标代码所需的时间。对于

MTCM和TCM，由于数据库中只需生成一种形式的目标代码(CDB)，编译只需进行一次，其时间开销记为1；对于MLCM由于需要生成两种形式的代码(CDB与TDB)，根据Source谓词方案[2]，需要进行两次编译量几乎相等的编译，时间开销也相应加倍，记为2；对于LCM，若采用并存子句代码与子句项方法，只需一次编译，时间开销为1；若采用并存子句代码与项代码方法，则需两次编译，时间开销为2。

5.实现的方便性

实现的方便性是指实现DBOP时是否还需要提供另外的特别支持以及提供这些支持的容易程度。为了给利用各种耦合模型实现DBOP的方便性作一个相对恰当的评价，应从利用每一种模型实现DBOP的途径入手加以分析：

(1) MTCM对DBOP的实现提供的支持较少。DBOP的实现在很大程度上依赖于反编译技术的完善。以库操作谓词retract(Clause)为例，其实现算法利用GKD-PROLOG[6]描述如下：

retract(Clause)：—decompile(Clause, Ref)erase(Ref).(算法1)

其中decompile和erase是系统专用的两个内部过程。decompile的实现较为复杂，需要为大部分指令指定第二种操作语义，且难以保证目标代码关于编译的完全可逆性，同时对编译进行的优化要有一定程度的限制，这势必会影响系统的运行效率。decompile(Clause, Ref)通过反编译查到一个与Clause匹配的子句，并将指针Ref指向子句代码块，其执行具有不确定性。erase(Ref)将Ref所指代码块删除。

(2) TCM实现retract的算法如下：

```
retract(Clause)：—get-ref(Clause,
                    Ref-c, Ref-t),
                    erase(Ref-c),
                    discard(Ref-t).(算法2)
```

其中内部过程get-ref(Clause, Ref-c, Ref-t)充分利用了代码库中的选择点指令，查找一个可以与Clause匹配的子句，并将指针Ref-c指向子句代码块，将指针Ref-t指向子句项，其执行具有不确定性。内部过程discard(Ref-t)删除Ref-t所指的子句项。

(3) LCM实现retract的算法描述形式与算法2相同，但内部过程get-ref的实现完全不同，需要单独研究源子句库SDB的结构，为子句项设计专门的索引机制，而且一般不支持不确定性执行。

(4) MLCM实现谓词retract的算法描述形式亦同算法2。但内部过程get-ref的实现略有不同，它利用项代码库TDB中选择点指令执行项代码，获得一个能与Clause匹配的子句项，同时获得指向项代码和指向子句代码的指针Ref-t和Ref-c。get-ref的执行亦带有不确定性。

在MTCM中实现decompile (Clause, Ref)和在LCM中实现get-ref(Clause, Ref-c, Ref-t)都是较为困难的事，因此在这两种模型中DBOP实现的方便性较差，但如在LCM中采用改进的Source谓词方案，实现DBOP谓词的方便性会有所改进，TCM实现DBOP的方便性最好，MLCM次之。

以上分析的结果(图5)表明，数据库的结构对数据库的性能影响很大。CDB与SDB耦合越松散，数据库的主要性能随空间开销，查找开销，管理开销和编译的时间开销越大而越差。MTCM应该是最佳模型，其次是TCM。但在实现DBOP的方便性上，MTCM远不如TCM与MLCM。综合考虑各种因素，TCM相对来说是一种较好的数据库模型，并且已在YH-SIM中得到实现[5]。此外，由于TCM支持DBOP的不确定性，YH-SIM实现的DBOP都能在不确定状态下执行，具有较强的功能。在用户使用上，DBOP与其它谓词完全相同。

主要参考文献

[1] Warren, D.H.D, An Abstract PROLOG Instruction Set, Technical Note 309,

支持多介质数据库的NF²方法*)

田沧海　林钧海　（210016南京航空学院计算机系）

摘　要

由于NF² (Non-First Normal-Form) 理论的实质在于允许关系的属性是另外一个关系，本文提出了用规范关系存贮各种介质的基本数据并用NF²的嵌套结构构造复杂的多介质数据的方法，从而有效地支持了多介质数据库的管理。

一、引言

近年来，非第一范式NF² (Non-First Normal-Form)理论的巨大进展[1,2]和应用，使数据库系统不仅能够支持商业领域中使用的简单表格，而且能够支持大而复杂的结构对象，包括文字、图形、图像及声音数据等等，从而能有效地利用数据库在工程领域和其它领域中实现管理。一般认为，NF²数据模型是关系和层次模型的高度统一。由于该模型可以达到外模式、概念模式和内模式的高度一致，保证数据存取的有效性和完整性，所以它优于扩充的关系数据模型[3]；也由于NF²模型易于实现，有较为成熟的理论基础，并且还能够描述抽象的语义[5]，所以亦优于面向对象的数据模型[4]。

NF² 理论的本质在于允许关系的属性是另外一个关系，从而可以描述复杂的结构。为了支持各种介质数据(如数字/字符、文字、图形、图像、声音等)，我们用规范关系描述和存贮各种介质的基本数据[6]，如一段文字、一个图形等，通过NF² 构造生成各种应用，形成复杂的多介质数据。我们的实践表明，NF² 方法支持多介质数据库是可行的。本文最后介绍了我们待进一步开展的工作。

二、基本数据存贮

各种介质的基本数据可以通过关系数据库加以存贮，例如，一段文字可以用(行号，内容)这两个属性值来描述，一个圆可以用(圆的名字，半径）来描述等等。为了能够统一地对所有介质的基本数据进行描述，我们给出关系数据字典和属性数据字典的描述形式[7]：

可以看出，这种描述提供了一种完整的、一致的、无冗余的表示，可以统一存贮各种介质的基本数据：

1. 数字/字符存贮：最基本的数据存贮

SRI International, Oct,1983.

[2] Clocksin, W.F. Implementation Techniques for PROLOG Database, SOFTWARE-PRACTICE and EXPERIENCE, Vol 15(7), PP.669—675(July 1985)

[3] Kevin A.Buettner, Fast Decompilation of Compiled Prolog Clauses,Logic Programming Research Group, School of Computer & Information Science, Syracuse University.

[4] 李良良，一种顺序PROLOG机系统结构的研究，国防科技大学计算机系，博士论文，1987(12)

[5] 车敦仁，编译型PROLOG数据库的实现技术的研究，国防科技大学计算机系，硕士论文，1988

[6] GKD-PROLOG/VAX780用户手册，国防科技大学，人工智能研究室，1985,12

*) 航空航天部科学基金资助课题