41-45 演绎数据库 规则 处理接口

计算机科学1993Vol.20№.3

演绎数据库WDeDB的规则处理接口

茹戈华 石树刚 郑振楣

(武汉大学计算机科学系 武汉430072)

摘 要

WDeDB is a deductive database based on the relational one WDDBS. After briefly presenting the rule definition language of the system, we will concentrate on the problem of the storage and retrieval of rule programs. The proposed methods will make full use of the techniques traditionally used in the database management to work efficiently.

一 引 言

关系数据库管理系统在存储、管理及访问大容量数据等方面的技术已趋成熟，但尚不适合一些基于知识的应用；演绎数据库把逻辑的方法应用于数据管理，将关系数据库管理技术与逻辑推理功能有机结合在一起，演绎地操纵大容量数据，适应了新应用的需要。

为把数据库系统和规则系统结合在一起，人们采用了不同的方法。最普遍的一种是将专家系统（如，用Prolog语言实现的系统）与数据库系统松散耦合，该方法存在以下问题：

1. 两种系统的数据模型不同，模型间的映射交给了应用程序。

2. 存在严重的执行限制。数据不得不在规则处理前装入专家系统的工作内存；数据库系统只被当成一个存贮系统，其界面仅限于"存"和"装"的命令；特别地，大量的模式匹配任务交给了专家系统推理机，而数据库操作未被充分利用。

3. 自顶向下的推理方法总是试图找出"所有的证明"而非"所有的答案"，会重复不断地产生相同的事实，而且不可能产生大规模的关系间的操作。

虽然一些系统采取了高速缓存及预编译规则程序等方法来缓解上述问题，但效果并不理想。事实上，现在许多实验系统，如LDL、NAIL! 等都将自顶向下的动态过程转换成一种静态优化算法(如魔集法)，而在推理上则采用自底向上的方式。MU-Prolog系统的改进版NU-Prolog也结合进了自底向上的推理模式。

基于以上考虑，我们在设计WDeDB时，采用了在原WDDBS（武大数据库系统）基础上扩充的策略。设计目标是尽可能利用关系数据库高效处理大容量数据的特点。设计方法是使逻辑推理在用户层、功能层及物理层与原系统紧密集成；WDeDB的界面语言LOGSQL是SQL的逻辑扩充，规则的存贮采用了关系的方法；系统的推理模式采用了自底向上的编译方法，以充分发挥关系数据库的优势。其总体结构如图1所示：

图1中，语言分析器检查语言是否符合语法要求，同时分离出与逻辑规则有关的部分，

茹戈华 硕士生，从事数据库系统与知识库系统的研究。石树刚 教授，从事计算机软件、数据库系统与知识库系统研究。郑振楣 从事计算机软件、信息分布处理与分布式数据库系统、OODB的研究。

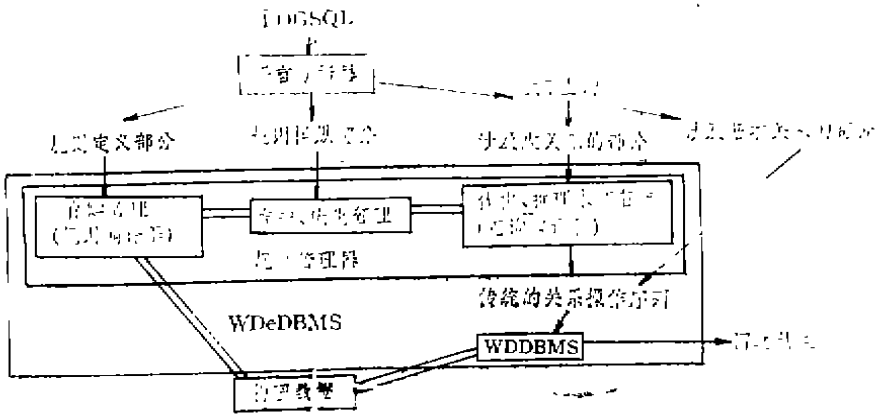

图1 WDeDB总体结构示意图

比如判断是否属规则定义、是否是对原有规则的操纵（如，修改、删除）、用户查询中是否涉及虚关系等等；规则编译器负责将规则定义语言编译成为一种规则存贮图RSG的内部表示，该图随后被以关系的形式存贮于数据库之中。当一个包含虚关系的查询提交系统时，首先访问磁盘上的规则库，检索出相关的规则（即查询求值中必然涉及的规则）；接着将这些规则调入主存，并执行优化和推理求值以得到传统的关系操作序列，这是查询编译器的任务。

二 系统WDeDB的几个基本概念

扩充的datalog数据模型是WDeDB的逻辑基础。在这个模型中，我们适当地增加了一些算术函数，并允许保证安全性的情况下，在规则体中存在否定。下面介绍贯穿系统始终的两个基本概念：

概念1 基本关系，是指一个关系的所有元组都由事实定义。

概念2 虚关系，是指一个关系中至少有一个元组必须由逻辑规则所定义。

应用级对基本关系和虚关系不作任何区分，用户可以象创建基本关系一样创建虚关系，象利用基本关系一样利用虚关系，这体现出演绎数据库的优越性。

三 规则定义语言

规则定义语言是LOGSQL逻辑扩充的核心。通过它，可以利用规则定义复杂的虚关系，支持演绎功能，使规则用于推导没有明确存储的新信息以回答用户查询。

在LOGSQL中，规则定义语言的结构保留了SQL语言规范性良好的特点，其设计基础是：SQL语言是完备的，它与带否定的、安全非递归datalog规则间存在对应关系；利用别名技术，还能借助SQL的结构定义安全的递归规则。

例1 设有关系模式ANCFS (ASC, DFS)及PARENT(PAR,YOUNG)，递归规则ancos(P, D):-parent(P, A) & ancos(A,D)。的LOGSQL表示为：

```
DEFINE ancos
AS      SELECT parent·par, anc1·des
        FROM  parent, ances anc1
        WHERE parent·young=anc1·asc;
```

LOGSQL表示逻辑程序时采用规则模块作为基本单位。规则模块是由一组语义相关的规则组成的规则集，它完全定义了某个虚关系。虚关系与规则模块间一一对应，可看作是关系数据库管理系统中视图概念的扩充。

在规则模块中，各规则的首部称为目标关系。注意，它不一定是虚关系，可能只是为定义某虚关系而用到的临时关系；规则体中用到的关系称为源关系，源关系必须是基本关系，或是已由其它规则模块定义了的虚关系，或是目标关系本身，即LOGSQL支持同一规则模块中的直接或间接递归。由于规则

模块是语义完整的，所以规则模块之间构成层次关系，这支持了规则分层的概念，规则模块的语法结构为：

```
INSERT INTO $虚关系名[(字段名[,字段名]*)]
USING 源关系名[, 源关系名]*
      DEFINE 目标关系名[(字段名|值[,字段名|,值]*)]
      AS LOGSQL子查询
      [DEFINE 目标关系名[(字段名|值[,字段名|, 值]*)]
      AS LOGSQL子查询]*;
```

其中，AS对应于一个规则的规则体，一个INSERT INTO $语句中不同的DEFINE、AS表示定义同一虚关系时用到的不同规则；[…]表示某语法结构可以省略，[…]*表示某语法结构的零次或多次重复，…|…表示或者。

例2 对例1，用LOGSQL写出定义ANCES的规则模块，

```
ances(A,D):-parent(A,D).;
ances(P,D):-parent(P,A) & ances(A,D).
INSERT INTO $ances
USING parent, ances
  DEFINE ances
  AS SELECT par, young
     FROM parent
  DEFINE ances
  AS SELECT parent.par, anc1.des
     FROM parent, ances anc1
     WHERE parent.young=anc1.asc;
```

WDeDB的规则库由规则模块组成，是LOGSQL语言规则库的最小编译单位。由于语言编译技术的支持，在同一规则模块中，规则是序无关的。

四 规则的关系存贮

演绎数据库中演绎规则的存贮有许多方法，比如，用有索引的正文文件形式存贮、用关系的方式存贮等等。我们采用的是规则程序的关系存贮方法，其主要优点在于可以一致性地操纵规则和事实，特别是当规则量大的时候，能充分利用关系数据库管理系统中高效处理磁盘上大容量数据的技术。

4.1 规则存贮图RSG

RSG是一个有向图，由谓词结点、规则结点、有向边、标记几部分组成。观察规则模块的语法结构，可以看到一条规则可划分成以下这四部分：

1. 目标关系d，由DEFINE定义。
2. 目标关系属性值与源关系属性值之间应满足的条件C_1，由AS SELECT 定义。
3. 源关系$s_1,\cdots,s_n$，由FROM定义。
4. 源关系属性值应满足的条件C_2，由WHERE定义。

与之相应地，规则存贮图RSG用谓词结点表示目标关系和源关系，用规则结点表示规则号，规则号由规则所在模块号及规则在模块中的序号组成，它唯一标记一条规则；源谓词结点到规则结点之间存在一条有向边arc1，规则结点到目标谓词结点之间存在一条有向边arc2，C_1为arc2之标记，C_2为规则结点之标记。所以，通过RSG，我们能清晰地表示出谓词与规则间的相互关系，如图2所示：

规则i：$d(C_1)\leftarrow S_1,\cdots,S_n,\ C_2$.

与规则i对应的RSG，

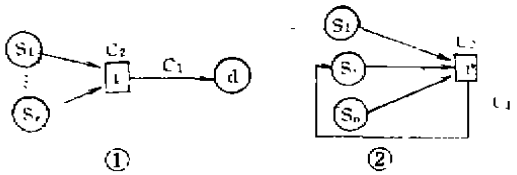

图2 RSG各组成部分示意图。图中○表示谓词结点，□表示规则结点。①$d\in\{S_1,\cdots,S_n\}$；②$d=S_j$

在RSG中，不同规则间通过谓词结点相互关联。比如图2中，d可作为其它规则j($i\neq j$)的目标谓词结点，表示i、j共同定义了d；d亦可作为j的源谓词结点，表示i的结论为j所用。同样地，S可作为其它规则j的源谓词结点，表示i与j的定义中都用到了S；S亦可作为j的目标谓词结点，表示j的结论为i所用。这样，RSG也清晰地刻画出规则之间的关

系。

当定义一个规则模块时，用户给出的形式存放于文件中，同时规则模块被编译为RSG，这是我们用关系的方法存贮规则的依据。

4.2 规则的关系存贮

尽管在规则库中，每条规则可以自由地增删、修改，象一个独立的整体，但我们认为一个完整的规则的存贮应是分开放置的，即，同一规则中所含的不同信息应存放于不同的存储区中，这样才有利于规则的存取，有利于操作时快速选取我们需要的信息。进一步地，用关系的形式存贮规则，有利于将数据库的一些重要概念，如物理和逻辑的完整性，应用于表示规则的数据；对规则库的增加、修改、删除等操作也都可以转化成对关系的操作。

对照规则模块的语法结构和规则存贮图RSG，我们的规则库包括以下五个关系模式：

1. 模块关系（模块名，#模块号），该关系用模块号唯一标识一个规则模块；

2. 相关模块关系（#模块号，#直接相关模块号），该关系表明了参与一模块定义的直接相关模块，支持了规则分层的概念。

3. 前件关系（#源关系号，#规则号），该关系指出参与某一规则定义的源关系，相当于RSG的arc1部分。

4. 后件关系（目标关系号，C_1，#规则号），该关系指明一条规则能导出什么结果，相当于RSG的arc2部分。

5. 条件关系（#规则号，C_2），该关系为一条规则定义了其源关系变量应满足的条件，相当于RSG的规则结点部分。

如第三节所述，虚关系与规则模块一一对应，所以模块名即为该规则模块定义的虚关系名，模块号即为该虚关系的标识，因此关系1、2实际上属于演绎数据库数据字典的一部分。关系3、4、5的分离是为了减少冗余，实际存储时，每个关系中同一规则模块的规则信息放在一起；当规则库更新时，我们也用规则模块为索引将它们链在一起。

五 相关规则的访问

WDeDB采用了自底向上的推理策略，有别于Prolog中将相关规则的查找与合一、回溯的证明过程融为一体的逆向推理方法，也没有Prolog中规则定序、规则子目标定序的限制。我们需要的是一次找出能产生查询所有结果的最小规则集，并对它们进行合理的分析与优化，最后产生对传统数据库调用的关系操作序列。所以，相关规则的检索是很重要的一环，特别地，当规则量大时，其访问效率显得十分突出。

先撇开具体的存贮方式，观察规则存贮图，我们发现最简明的规则检索方法是以要查询的虚关系谓词为起点，沿有向边逆向遍历，遍历中所经的规则结点即是相关的规则。算法描述如下：

算法1 基于规则存贮图的相关规则检索

输入：用户查询所涉及的虚关系Q，规则存贮图RSG

输出：与计算Q有关的所有规则之集R

步骤：1. $R\leftarrow\phi$；$P\leftarrow\{Q\}$；$T\leftarrow\phi$；

2. while $P\neq\phi$ do

i. 从P中任选一谓词i；$T\leftarrow T\cup\{i\}$；

ii. $R\leftarrow R\cup${RSG中以i为目标结点的规则r_i}；

iii. $P\leftarrow P\cup\{r_i$中的源谓词$\}-T$；

算法1中R是结论集合。集合P、T都用于存放与Q直接或间接相关联的谓词。特别地，P用于存放下一轮将要检索的谓词，T用于存放已检索过的谓词，作用是避免重复检查。

步骤1初始化P为{Q}，意指以Q为起点逆向遍历RSG；步骤2是一个搜索循环，意指只要P中还有未检索的谓词，就应将与之相关的规则存放入集合R中。该算法的终止前提是：与基本谓词相关联的规则集为空集。因为任何虚关系的数据都是从基本关系推导产生的，所以当P涉及的全部谓词皆为基本谓词时算法结束。

算法1的搜索过程与Prolog有相似之处，

但它没有繁杂的合一与回溯证明过程，也不涉及对基本数据库的任何调用。

在第16届VLDB大会上，J.P. Cheiney与C. D.Maindreville 提出了基于关系的双散列传递闭包求值算法R*，这对我们在关系的环境下具体实现算法1有很大启发。事实上，我们只要将R*中的迭代集合ΔR 初始化为某些指定的点对，R*就成为一个计算有设定起点的传递闭包算法。此时，如果我们把 R的结构定为（源谓词，规则号，目标谓词），利用算法R*就能方便地求出到虚关系谓词Q的所有谓词及规则号。

然而，我们发现上述想法存在以下问题：

1. R的结构不符合系统规则库的具体存贮形式，其冗余量大，而且易造成更新不一致；

2. 按照R*的双散列思想，我们将依据谓词域确定散列函数，并先按目标谓词、再按源谓词将R分桶。但如果这样，我们就无法保证同一规则模块中的规则物理集中在一起。结果是，当用求出的规则号查找规则库时效率会降低，因为如果Q的定义中涉及到虚关系Q_1，那么规则模块Q_1中的规则都将参与Q的定义。

根据以上分析，我们认为在访问相关规则的问题上，将规则存贮图看成由规则模块组成的抽象比较好。因为系统具体存贮规则时，是把同一规则模块中的规则物理存放在一起的，而且规则号的首字节即为规则模块号，所以用相关的规则模块号去查规则库效率很高。此时，R的结构为（#模块号，#直接相关模块号），符合第四节所述。

接下来，我们将以规则模块为单位，结合查询要求，将规则调入主存进行分析、优化，比如实施部分规则的淘汰、规则的合并、常量的下压等等。而且根据第三节所述，由于规则模块的层次性，整个相关规则图的最大强连通部分也只可能在规则模块的内部产生，从而方便了编译。

参考文献

[1] 石树刚等，WDDBS-32系统结构设计与实施，计算机研究与发展，Vol. 28, No. 11, 1991

[2] G. Kiernan, et al., Making Deductive Database a Practical Technology: a Step forward, Proc. of ACM SIGMOD Int. Conf.,Atlantic City, 1990

[3] C. Kellogg, et al., Optimizing the Rule-Data Interface in a KMS,Proc. 12th Int. Conf. on VLDB, Kyoto, 1986

[4] 茹戈华，石树刚，郑振楣，演绎数据库语言LOGSQL的设计，将刊于《计算机学报》，1993

[5] J. P. Cheiney, et al.,A Parallel Strategy for Transitive Closure using Double Hash-Based Clustering, Proc. 16th Int. Conf. on VLDB, Brisbane, Australia, 1990

[6] S. Ghosh, et al., Implementation of A Prolog-INGRES Interface, SIGMOD RECORD, Vol. 17, No.2. June 1988

[7] 周傲英，知识库系统及其研究现状，计算机科学，No.5, 1991

[8] J. D. Ullman, Principles of Database and Knowledge-Base Systems, Vol. I, I, Computer Science Press, Inc., 1988.