SQL语言 描述性语言 知识库 (15)

计算机科学1994 Vol. 21 №. 3

62-64 TP312

描述性语言与知识库系统

朱扬勇

(复旦大学计算机科学系 上海200433)

摘 要 Declarative languages are the pursuing goal of knowledge-base systems. However, after a long time research, from 1985 to now, there have been still no declarative knowledge-base programming languages published, and furthermore, this goal has somewhat blocked the practice of knowledge-base systems. This paper introduces the concept of declarative languages, explores the relationship between declarative languages and knowledge-base languages, and gives our results.

关键词 Knowledge-base, Logic programming, Declarative Languages.

1. 引言

在数据库领域,由于描述性语言SQL的成功使用,描述性更加受到重视[1,2,3,4]。描述性语言Datalog的提出,使得一些学者认为知识库语言应该是描述性的程序设计语言[4]。然而多年来的研究,不但没有开发出一个描述性的知识库程序设计语言,而且还阻碍了知识库的实用化[5]。本文研究了描述性与逻辑语言、知识库语言之间的关系,提出了三个观点:1)描述性语言是计算不完备的,因此不能作为独立的程序设计语言;2)逻辑语言是描述性语言的合适形式;3)追求语言的描述性是知识库系统实用化的障碍之一。

2. 描述性语言

描述性是与过程性相对立的概念。现行的程序设计语言都是过程性语言(如C,Pascal等),甚至连Prolog也是过程性的。用过程性语言写应用程序,不但要说明问题,而且还要告诉计算机如何去做,程序严格规定了计算机的每一步工作;而用描述性语言解题,则只要描述问题,不必告诉计算机如何去做。因此,如果有描述性语言可用,用户自然是喜欢的。

不幸的是,描述性语言是计算不完备的,即它不能独立地作为程序设计语言使用。描述性这一概念起源于"算法=逻辑+控制"的论述。用计算机解题,就要设计算法。如果算法的控制部分由计算机自动处理,那么用户只要描述出问题的"逻辑"就可以了。更直观地说,没有用户控制机制的语言就是描述性的了。然而,求解一个问题的解法严格规定了计算机的每一步工作,是问题的过程性内容(也就是控制)。而且,解法本身也可以看作一个问题,描述性语言也应该能够加以描述。用描述性语言描述过程性问题,是同描述性概念相矛盾的,所以描述性语言是计算不完备的语言,它只能描述一类问题,而不是全部问题。这也就是至今仍没有描述性程序设计语言之缘故。

尽管如此,描述性的概念仍然很有意义。传统的过程语言中有三种用户控制:顺序、分支和循环。适当减少控制仍可以给用户带来极大方便。如Prolog语言没有循环和分支结构,只有顺序结构,所以Prolog编程相对方便简单。另外,关系数据库查询语言SQL是描述性的,它给用户带来的方便是众所周知的。这样,我们将描述性概念分成两类:

1)描述性语言:是指一个计算机语言没有任何的用户控制机制(如SQL,Datalog)。

2)非过程性(或控制较少的)语言:是指一个计算机语言只有一、二种控制(不是三种)。

最后我们说的过程性语言是指包含有顺序、分支和循环三种控制结构的语言。

3. 逻辑语言——通向描述性之路

什么是逻辑语言呢?计算机所能实现的逻辑是指"数理逻辑",也就是以命题演算和一阶谓词演算为基础的公理化集合论、证明论、模型论和递归论等。所以,计算机逻辑语言是在数理逻辑基础上建立起来的,如Prolog,Datalog等都是以一阶谓词演算的Horn子句逻辑为基础。一般地,逻辑语言的程序

朱扬勇 讲师,主要研究方向:数据库、知识库和逻辑程序设计

由事实和规则组成，当提问目标时，系统根据规则（规则是问题的描述，也可以包括问题的解法）和事实，导出满足目标的事实或给出证明的结论（true 或 false）。这样，逻辑语言的描述性直观地可理解为：事实、规则及其子目标的顺序与程序的执行结果无关。

一阶谓词演算系统可以描述自然语言的很多内容以及大多数数学定理。由 Skolem 定理，一阶语言的语句集可以转化为 Skolem 标准型，后者具有某种程度的顺序无关性（标准型子句中的文字满足交换律）。作为它的一个真子集的 Horn 子句也具有这种性质。因此，适当选择求值方法就可以使得基于 Horn 子句（甚至可以是基于一阶谓词逻辑）的语言是描述性的，例如基于 Horn 子句的 Datalog（一个描述性逻辑语言）、基于一阶逻辑的描述性语言 SQL。

在人工智能领域，利用 Skolem 标准型进行定理自动证明已经提出了很多方法[6]，而逻辑程序设计也正是由定理自动证明发展而来的。因此，定理自动证明完全可以用描述性语言实现。然而，作为一种计算机语言，不仅要有能力证明定理，而且还要有能力将证明过程展示给用户（因为用户想知道定理是如何证明的），或者允许用户查看中间结果，为此，逻辑程序设计语言 Prolog 不得不向用户提供一些控制机制。所以作为独立程序设计语言的逻辑语言不可能是描述性的，至多只是非过程性的。

基于 Horn 子句的逻辑语言（如：Prolog、Datalog）的规则隐含了循环和分支两种控制结构：循环可由递归实现，而规则本身就是分支结构。这是逻辑语言的得天独厚之处，是描述性语言取逻辑语言形式的本质所在。

4. 描述性与知识库语言

描述性在数据库领域受到重视有两个方面的原因。首先，关系数据库查询语言 SQL 是描述性的，给数据库应用人员带来极大好处。用户只要给出查询什么数据、这些数据在哪些表中、而不必告诉计算机如何从诸关系表格取得所需要的数据，非专业人员可以方便地使用数据库。其次，描述性语言必须有能力并且高效地给出一个问题的所有解答结果（因为用户程序无法挑选解法和结果）。因此，数据库系统“每次一个集合”的计值模型就大大优于人工智能领域常用的“每次一个元组”的计值模型。也就是说用数据库技术实现描述性语言更为方便。

当 AI 与 DB 相结合产生知识库的时候，DB 研究人员就很自然地希望将 SQL 的优点加以保持并扩展，而且希望能够克服 SQL 与宿主语言阻抗不匹配的缺点。Ullman[4]给出了知识库系统的定义：一个知识库系统是具有下面两个特征的逻辑程序设计系统：①有一个既可以作为数据操纵语言，又可作为宿主语言的描述性语言。②支持数据库系统的主要功能，如：大批量数据的高效存取、数据共享、并发控制及错误恢复等。

这一定义过于乐观，尽管 Ullman 也强调了“带有术语‘描述性’的语言对步骤序列要求的说明比过程语言少”，但这一说明是模糊不清的。这导致了多年来一些研究者在开发知识库系统时，极力去支持一个满足条件①的逻辑程序设计语言，如：LDL、NAIL![7,8]。事实上，LDL、NAIL! 为了满足描述性，并没有起到宿主语言的作用。这些系统根本无法付诸实用。于是，LDL 中又不得不加入用户控制机制[8]；而 Glue-NAIL! 则提供了一个过程语言的窗口[9]。另外，一些系统的开发者可能注意到了这一问题，所以在实现知识库语言时没有遵循描述性要求，例如：DRL、CORAL 等[5,10]系统。

知识库语言取逻辑语言的形式是出于 AI 的要求和逻辑语言的优越性。现在的问题是描述性与操纵语言和宿主语言集成的取舍或如何并存。这有以下二个考虑：

①保持描述性，即建立一个描述性的查询语言（如 Datalog），将其与宿主语言（过程性语言）分开。这里的 Datalog 和 DML 不同，可以随意编程，定义各种递归，而 DML 只提供几种有限的操作，这就是说 Datalog 也起了一定的宿主语言的作用。

②放弃描述性条件，建立一个既作查询语言又作宿主语言的逻辑语言，某些谓词作为事实部分，另一些作为视图部分，还有一些作为单个应用程序部分。

不难看出，二者本质上差别不大。我们不会用同一种计值模型执行视图部分和应用程序部分，这也就是为什么要将这两部分分开的原因。视图部分是对事实的操纵，需要有大量优化技术（一般用户无法解决），并采用自底向上“每次一个集合”的计值模型；而应用程序部分是语言的过程性内容，常常是单步执行的，所以采用“每次一个元组”的计值模型比较合适。可见，视图部分和应用部分本质上是分离的，所以①和②基本相同，只是用户更愿意接受第②

计算机科学1994Vol. 21. №. 3

知识库查询的固有低效性

朱扬勇
（复旦大学计算机科学系　上海200433）

摘　要　This paper introduces the concepts of knowledge-base system and the scale of knowledge-base's application, analyses the method of knowledge-base query evaluation, and presents several obstacles during this evaluation. All of them cannot be well treated by current hardware techniques and software methods. Therefore inefficiency of knowledge-base queries are inherent today. Finally several research directions of the query optimization of knowledge-base systems are given.

关键词　Knowledge-base, Logic Program, Query Optimization.

1. 引言

知识库研究的重要内容之一是查询优化技术的研究。从七十年代后期开始，经过八十年代，到目前为止，知识库查询优化方面已经取得了大量研究成果，开发了许多有效的查询优化算法。例如：Semi-Naive 求值[1]、Magic_Set 重写技术和 Counting 方法[2,3]等等。这些技术的应用，大大提高了知识库的查询效率。尽管如此，知识库的查询效率仍不能令人满意，这是知识库系统不能实用的主要原因之一。

关于知识库系统，目前有两个典型的定义。一个是 J. D. Ullman 提出的[4]。他认为：一个知识库系统是具有下面两种特征的逻辑程序设计系统。

①有一个既作为查询语言又作为宿主语言的描述性语言。

②支持数据库系统的主要功能，如：大批量数据的高效存取、数据共享、并发控制及错误恢复等。

另一个定义是 D. H. Warren 给出的，他认为：一个知识库系统"能够有效地处理中等规模的知识库（由三千个谓词、三万条规则和三百万个事实组成，总存储量为30M 字节）的逻辑程序设计系统"。

总而言之，我们认为：知识库系统是一个逻辑程序设计系统，并能有效地处理（包括数据库技术的应用）一定规模的知识库。这里我们强调"一定规模"，而不是具体地指出"30MB"的存储量，是指知识库的内容在内存中放不下，必需要放到外存储器由数据库系统或采用数量库技术来管理。我们以下的讨论也是基于这一点的。事实上，如果一个应用问题的知识库可以全部装入内存，则现有的查询优化技术是不重要的，Prolog 也能很好地处理，并不需要数据库

种情形（LDL，DRL，CORAL 都用第二种形式）。

综上所述，我们认为：知识库语言是一种既作为查询语言又作为宿主语言的逻辑语言（基于 Horn 子句的语言）。我们之所以不再提描述性或非过程性，是因为逻辑语言本身就是控制较少的（或非过程）语言。

5. 结论

我们有以下结论：

①描述性程序设计语言只是一个可望而不可及的目标。适当减少一个语言的控制机制可以方便用户、缩短应用程序。但是描述性语言是计算不完备的，不能作为独立的程序设计语言。

②逻辑语言是描述性语言的合适形式。由于逻辑语言隐含了分支和循环结构，以及 Skolem 标准型的顺序无关性，其描述性语言比较容易实现（如 Datalog）。

③知识库语言要想完成宿主语言的工作，描述性条件首先要放弃。由于逻辑语言实现演绎推理方便并且控制较少，所以逻辑程序设计语言作为知识库语言是有益的。

长期以来，描述性一直阻碍了知识库系统的实用化，人们不得不在知识库语言中加入控制机制[6,7,10]。这一现象的发生源于对描述性概念的理解和 Ullman 对知识库的定义。本文在得到上述三个结论的同时，对描述性概念加以澄清，定义了描述性、非过程性和过程性等概念，并对知识库语言进行了切合实际的描述。（参考文献共10篇略）