计算机科学2000Vol. 27№. 5

基于嵌套语义的约束代数*)

Constraint Algebra Based on Nested Semantics

陈良刚

(复旦大学计算机科学系数据库研究中心 上海 200433)

Abstract In this paper, we show constraint algebra has some limitations when it is used for dealing with the set of points representing a spatial object as a whole. Rather, only point-based computations can be performed using this algebra. We extend the definition of constraint data model and constraint algebra. We show how constraint relations can be interpreted under the nested relational model. We introduce a class of constraint algebra based on the nested semantics. However, from a user point of view, it is more suitable.

Keywords Constraint databases, Relational algebra, Nested relational algebra

1 引言

约束数据库近期被 Kanellakis 等提出作为处理空间数据的一般性框架[1]。约束数据库用约束来建模和检索数据。在数据层，约束能用有限的形式来表示可能是无限的关系元组集。例如，约束 $x^2+y^2\leqslant 9$ 表示中心在点(0,0)处，半径为3的圆。在查询语言层，约束通过允许数学计算而增强了简单关系语言的表达能力，同时约束查询语言保留了关系查询语言的所有特征，如封闭性和自底向上求值。关系代数能被扩充来处理约束关系，这个新的代数叫做约束代数 CALG[2]。

然而，从处理空间数据的观点来看，我们可以发现 Kanellakis 的约束数据模型和约束代数存在一些不足。约束数据模型总是把一个约束关系看作是可能无限的点集，而不是有限的空间对象集。因此，普通的空间查询在约束查询语言中有着复杂的表示，在约束代数中也一样[3]。为了直接建模典型的空间查询，我们对约束关系提出了一个新的语义。实际上，每一个约束关系可看作是无限集的集，每一个无限集对应一个约束元组。在这种情况下，约束关系模型是一个简单的嵌套[4]，这个新的语义被叫做嵌套语义。重要的是这类无限集可通过一个元组来建模，不仅仅是合取，其他逻辑连接词也可以在约束元组中使用。例如，为了在一个元组中建模凹集，析取必须被采用。

我们提出一个基于嵌套语义的代数来处理关系语义和嵌套语义下的约束关系。在这个代数中，既可把约束关系解释为空间对象的有限集，也可解释为点的无限集。

2 约束数据模型

下列定义正式介绍了约束数据模型。

定义1(约束数据模型)[3] 设 Φ 为一类约束，D 为其论域，则：

1)变量集 $X=\{x_1,\cdots,x_k\}$ 上的约束 k 元组 t 是一个约束合取式 Ψ，$\Psi=\varphi_1\wedge\cdots\wedge\varphi_N$，其中每个 $\varphi_i(1\leqslant i\leqslant N)$ 是定义在变量集 X 上的一个约束。$\alpha(t)$ 表示集 $\{x_1,\cdots,x_k\}$。使 Ψ 为真的 D^k 的子集记为 unr(Ψ)，即 unr(Ψ)$=\{(x_1,\cdots,x_k)\mid(x_1,\cdots,x_k)\in D^k$，且 $\Psi(x_1,\cdots,x_k)$ 为真$\}$。

2)M 个定义在变量集 $X=\{x_1,\cdots,x_k\}$ 上的约束 k 元组 $\Psi_1,\cdots,\Psi_M$ 构成了一个 k 元约束关系 r。析取式 $\Psi_1\vee\cdots\vee\Psi_M$ 是 r 对应的公式 φ_r。$\alpha(t)$ 表示集 $\{x_1,\cdots,x_k\}$，称为 r 的模式。

3)约束数据库定义为有限个约束关系组成的集合。

在约束数据模型中，一个约束关系被解释为可能无限的关系元组集的有限表示，这个集表示它的语义。

定义2(关系语义) 设 Φ 为一类约束，D 为其论域。$r=\{t_1,\cdots,t_M\}$ 是一个约束关系，r 的关系语义 rel(r)为 unr(t_1)$\cup\cdots\cup$unr(t_M)。当且仅当 rel(r_1)$=$rel(r_2)时，约束关系 r_1 和 r_2 称为是 r-等价的，表示为 $r1\equiv_r r2$。

*)本文受国家自然科学基金资助，批准号69773012。

关系代数能被扩充来处理约束关系，这个新的代数叫做约束代数CALG[2]，它把约束关系处理为可能无限的关系元组集。

3 嵌套约束数据模型

约束数据模型是简单而有力的模型，可以表示关系元组的无限集。但在表示复杂的数据对象，如空间数据时，存在一些不足。

3.1 约束元组的定义和语义

约束元组的表达能力低于无量词一阶公式的表达能力，从空间数据的观点来看，一个约束元组仅能表示点的凸集。为了表示点的凹集，凹对象必须被分解为几个凸对象，如DEDALE系统[5]，每一个凸对象被一个约束元组表示。为了表示这些元组属于一个凹对象，每一个凸对象必须有一个标识符(见例1)，这种方法将导致一定的冗余。我们提出了更一般化的约束元组的概念，不仅仅是合取，其他的逻辑连接词也在约束元组中使用，例如，当析取使用时，一个约束元组既能表示点的凸集，也能表示点的凹集。

另外考虑约束关系的语义。在关系语义下，一个约束元组表示可能无限的关系元组集。实际上，约束关系也能被解释为嵌套关系[4]，包含有限个可能无限的集，每一个集对应一个约束元组。

3.2 约束代数

文[2]提出的约束代数CALG把约束关系处理为可能无限的关系元组集。这种方法迫使用户从点的思想来考虑，因此，把约束元组处理为单一对象的唯一办法是每一个约束元组有一个标识符。对于嵌套语义下的约束关系，用户不得不按照集的思想来考虑，为了能在嵌套语义下处理对象，必须引入新的代数。

从上面考虑，空间数据应从两个方面来处理：

·基于点的操作。对大多数操作，空间对象表示为点集是有用的。例如，检索对象某一部分包含于某一区域，计算两个空间对象的交集。

·基于对象的操作。对其他处理，把空间对象看作一个单一值是有用的，例如，考虑整个对象包含于某一区域或计算相交的空间对象。在这两种情况下，必须考虑所有的点属于某一区域。对象必须被看作单一值而满足这个查询，而不是它的部分。

所有基于点的计算能被约束代数CALG表示，这并不意味着基于对象的操作不能被约束代数CALG表示。然而，典型的基于对象的空间操作在约束代数CALG中，有非常复杂的表示，如例1所示：

例1 查询包含于空间区域rt内的所有空间对象。假定所有空间对象在约束关系R中表示。区域rt的约束关系表示为P。为了在约束代数中表示这个查询，约束关系R的模式必须包含一个变量ID，表示约束元组的标识符。这个查询的约束代数表示如下：

$$(\pi_{[ID]}(R)-(\pi_{[ID]}(R-\sigma_P(R))))\rhd\lhd]R$$

这个查询不是简单易懂的。虽然在空间应用中，这是一个普通的查询。这是由于这个查询把约束元组看作为一个对象，而约束代数处理的是约束元组对应的关系元组集。

3.3 嵌套约束数据模型

定义3(嵌套语义) 设$r=\{t_1,\cdots,t_M\}$是一个约束关系，则r的嵌套语义nested(r)为$\{unr(t_1),\cdots,unr(t_M)\}\setminus\{\{\}\}$。两个约束关系$r_1,r_2$是n-等价的(表示为$r_1\equiv_n r_2$)当且仅当$nested(r_1)=nested(r_2)$。

从定义3可以发现如果两个约束关系是n-等价的，则它们也是r-等价的。

引理1 $\equiv_n\subset\equiv_r$

证明：假定$r=\{t_1,\cdots,t_M\}$，$r'=\{t'_1,\cdots,t'_M\}$，则$r\equiv_n r'\Rightarrow nested(r)=nested(r')\Rightarrow\{unr(t_1),\cdots,unr(t_M)\}\setminus\{\{\}\}=\{unr(t'_1),\cdots,unr(t'_M)\}\setminus\{\{\}\}\Rightarrow\bigcup_{i=1,\cdots,M}unr(t_i)=\bigcup_{i=1,\cdots,M}unr(t'_i)\Rightarrow rel(r)=rel(r')\Rightarrow r\equiv_r r'$。

下面的例2显示了关系语义和嵌套语义的不同之处。

例2 设约束关系r_1仅有元组$1\leqslant x\leqslant 2\wedge 2\leqslant y\leqslant 4$，约束关系$r_2$有元组$1\leqslant x\leqslant 2\wedge 2\leqslant y\leqslant 3$，$1\leqslant x\leqslant 2\wedge 3\leqslant y\leqslant 4$。显然$r_1\equiv_r r_2$，然而$r_1\equiv_n r_2$，因为$r_1,r_2$所表示的集是不同的。

如果约束关系在嵌套语义下解释，在约束元组中使用的逻辑连接词集是重要的，因为它标志着能解释的无限集的类型，就象我们看到的，当在约束元组中仅使用合取时，只有点的凸集能被表示。为了克服这些缺点，我们扩充约束元组处理析取。

定义4(析取约束元组) 设Φ为一类约束，D为其论域，则变量集$X=\{x_1,\cdots,x_k\}$上的析取约束元组(简称为d-约束元组)是X上的约束元组的析取式。

当用析取约束元组取代约束元组时，约束关系和约束数据库的概念没有改变。下面我们总是考虑d-约束元组，在不引起混淆的情况下，把d-约束元组简称为约束元组。析取的使用，允许我们在一个约束元组中，表示所有类型的空间数据。

4. 嵌套约束代数NCALG

约束代数CALG是基于约束关系的关系语义的。当我们采用约束关系的嵌套语义时，一个约束元组对应一个对象，其它的操作符能被定义。下面，我们提出一个基于嵌套语义的约束数据库代数，称为嵌套约束代数NCALG。NCALG是通过对约束代数扩充新的操作而得到的，这个语言可处理关系语义和嵌套语义下

的约束元组。

·点操作符如同约束代数，唯一的不同是现在它们处理嵌套语义。因而，我们使所得的查询结果是嵌套表示的。

·集操作符如下：

①集差：设约束关系 r_1，r_2，这个操作返回 r_1 中的元组，且 r2中不存在与此等价的元组。这是嵌套关系数据库[4]中普通的减操作。

②集补：设约束关系 r，这个操作对 r 中的每一个约束元组 t，返回一个约束元组 t′，t′是公式 $\neg t$ 对应的析取式。

③集选择：这个操作从约束关系中选择满足一定条件的约束元组。条件是具有形式 $Q_1\theta Q_2$ 的无量词公式，$\alpha(Q_1)=\alpha(Q_2)$，θ 是谓词集$\{\subseteq, \triangleright\!\triangleleft\}$之一，$Q_1$和 Q_2 或者是元组 t，或者是查询表达式，其唯一参数是元组。查询 Q_i，$i=1,2$通过递归使用 NCALG 操作符$\{\sigma_p, \rho_{x/y}, \pi_{x_{i1},\cdots,x_{in}}, \neg, \cap\}$而得。

约束关系 r 上带条件 $Q_1\theta Q_2$的集选择操作，选择 r 中的约束元组 t，在 $unr(Q_1(t))$和 $unr(Q_2(t))$间存在关系 θ。下面是集操作的意思：

·$\theta\equiv\subseteq$：选择 r 中的所有元组 t，且 $unr(Q_1(t))\subseteq unr(Q_2(t))$。

·$\theta\equiv\triangleright\!\triangleleft$：选择 r 中的所有元组 t，且 $unr(Q_1(t))\cap unr(Q_2(t))\neq\varnothing$。

表1给出了嵌套语义下的点和集操作。$R_1,\cdots,R_n$是关系名，e 是代数表达式，$\alpha(R)$表示关系 R 的模式，即 R 的变量集$\{x_1,\cdots,x_k\}$。

表1 NCALG 操作

操作名	代数表达式 e	模式限制	语义 r
点操作			
元关系	R_1	$\alpha(e)=\alpha(R_1)$	$r=r_1$
选择	$\sigma_p(R_1)$	$\alpha(p)\subseteq\alpha(R_1)$，$\alpha(e)=\alpha(R_1)$	$r=\{t\wedge p \mid t\in r1, unr(t\wedge p)\neq\varnothing\}$
属性改名	$\rho_{[x/y]}(R_1)$	$x\in\alpha(R_1)$，$y\notin\alpha(R_1)$，$\alpha(e)=(\alpha(R_1)\setminus\{x\})\cup\{y\}$	$r=\{t[x/y] \mid t\in r_1\}$
投影	$\pi[x_{i1},\cdots,x_{ip}](R_1)$	$\alpha(R_1)=\{x_1,\cdots,x_M\}$，$\alpha(e)=\{x_{i1},\cdots,x_{ip}\}$，$\alpha(e)\subseteq\alpha(R_1)$	$r=\{\pi[x_{i1},\cdots,x_{ip}](t) \mid t\in r_1, unr(\pi[x_{i1},\cdots,x_{ip}](t))\neq\varnothing\}$
自然连接	$R_1\triangleright\!\triangleleft R_2$	$\alpha(e)=\alpha(R_1)\cup\alpha(R_2)$	$r=\{t_1\wedge t_2 \mid t_1\in r_1, t_2\in r_2, unr(t_1\wedge t_2)\neq\varnothing\}$
补	$\neg R$	$\alpha(e)=\alpha(R_1)$	$r=(\bar{t}_1\vee\cdots\vee\bar{t}_m \mid \bar{t}_1\vee\cdots\vee\bar{t}_m$ 是 $\neg t_1\wedge\cdots\wedge\neg t_n$ 的析取范式，$r_1=\{t_1,\cdots,t_n\}$，$unr(\bar{t}_i)\neq\varnothing$，$i=1,\cdots,m)$
集操作			
并	$R_1\cup R_2$	$\alpha(e)=\alpha(R_1)=\alpha(R_2)$	$r=\{t \mid t\in r_1$ or $t\in r_2\}$
集差	$R_1\setminus R_2$	$\alpha(e)=\alpha(R_1)=\alpha(R_2)$	$r=\{t \mid t\in r_1,$ $\nexists t'\in r_2: unr(t)=unr(t')\}$
集补	$\neg^* R_1$	$\alpha(e)=\alpha(R_1)$	$r=\{not\ t^* \mid t\in r_1, unr(not\ t)\neq\varnothing\}$
集选择	$\sigma_{Q_1\subseteq Q_2}(R_1)$ $\sigma_{Q_1\triangleright\!\triangleleft Q_2}(R_1)$	$\alpha(Q_1)=\alpha(Q_2)$，$\alpha(e)=\alpha(R_1)$ $\alpha(Q_1)=\alpha(Q_2)$，$\alpha(e)=\alpha(R_1)$	$r=\{t \mid t\in r_1, unr(Q_1(t))\subseteq unr(Q_2(t))\}$ $r=\{t \mid t\in r_1, unr(Q_1(t))\cap unr(Q_2(t))\neq\varnothing\}$

a：not t 表示公式 $\neg t$ 的析取范式

当约束关系带约束元组的标识符时，NCALG 和 CALG 是等价的。实际上，约束元组带标识符是在约束关系中集操作的唯一方法。然而，NCALG 允许用更简单和更紧凑的形式表示空间查询。更进一步，NCALG 有与 CALG 一样的复杂性。

例3 例1的查询用嵌套约束代数 NCALG 可表示

为 $\sigma_{t\subseteq F}(R)$，这个表达式比例1中约束代数表达式更为简单。

例4 下列的空间查询能用 NCALG 代数表示：

区域包含：在 R 中，选择包含于空间区域 rt 的空间对象，rt 的约束关系表示为 $p_r\sigma_{t\subseteq F}(R)$。

区域相交：在 R 中，选择和空间区域 rt 相交的空间对象，rt 的约束关系表示为 $p_r\sigma_{t[\sim]p}(R)$。

相邻查询：选择与空间对象 sp 相邻的对象，sp 的约束关系表示为 $p_r\sigma_{c_1}(\sigma_{t_2}(R))_r c_1\equiv\neg(Q_{Int}(t)\rhd\lhd Q_{Ext}(p))_r c_2\equiv(Q_{Bnd}(t)\rhd\lhd Q_{Bnd}(p))$。

空间连接(基于相交)：选择空间对象对(r，s)，$r\in R$，$s\in S$，r 与 s 相交。$\sigma_c(R\rhd\lhd\rho_{[x,x',y,y']}(S))_r c\equiv(Q_1(t)\rhd\lhd Q_2(t))$，$Q_1(t)\equiv\pi_{[x,y]}(t)$，$Q_2(t)\equiv\rho_{[x|x',y|y']}(\pi_{[x',y']}(t))$。

空间连接(基于相邻)：选择空间对象对(r，s)，$r\in R$，$s\in S$，r 与 s 相邻。$\sigma_{c1}(\sigma_{c2}(R\rhd\lhd\rho_{[x,x',y,y']}(S)))_r c_1\equiv\neg(Q_{1,1}(t)\rhd\lhd Q_{1,2}(t))$，$Q_{1,1}(t)\equiv Q_{Int}(\pi_{[x,y]}(t))$，$Q_{1,2}(t)\equiv Q_{Int}(g(t))$，$c_2\equiv(Q_{2,1}(t)\rhd\lhd Q_{2,2}(t))$，$Q_{2,1}(t)\equiv Q_{Bnd}(\pi_{[x,y]}(t))$，$Q_{2,2}(t)\equiv Q_{Bnd}(g(t))$，$g(t)\equiv\rho_{[x|x',y|y']}(\pi_{[x',y']}(t))$。

差查询：选择 R 中的空间对象 r，S 中没有一个对象 s，r 和 s 在 x 上的投影是相等的。$\pi_{[x,y]}(\sigma_c((\pi_{-x}(R)\setminus\pi_{[x]}(S))\rhd\lhd R'))$，$R'\equiv\rho_{[x|x',y|y']}(R)$，$c\equiv(\pi_{[x]}(t)=\rho_{[x|x']}(\pi_{[x']}(t)))$。

Q_{Bnd}和 Q_{Ext}是分别查询空间对象的边界和内部的简写。

结论 约束数据库用数学约束来建模可能无限的关系元组集，本文中，为了解决 Kanellakis 约束数据模型和约束代数在建模空间对象时的不足之处，在嵌套关系语义的基础上，我们提出了嵌套约束数据模型和嵌套约束代数 NCALG。通过几个例子，我们证明了这个代数比约束代数 CALG[2]更适合于时空应用。文[6]给出了嵌套语义下的约束数据库实例区间约束数据库，它以易于实现而受到约束数据库研究者的重视。

参考文献

1 Kanellakis P C, et al. Constraint Query Languages. JCSS 51, 1995

2 Goldin D Q, Kanellakis P C. Constraint Query Algebras. Constraints Journal, 1996, 1(1)

3 Kanellakis P C, Goldin D Q. Constraint Programming and Database Query Languages. In: LNCS 789: Proc. of the Int. Symp. on Theoretical Aspects of Computer Software. 1994. 96～12

4 Abiteboul S, Kanellakis P C. Query Languages for Complex Object Databases. SIGACT News, 1990, 21(3): 9～18

5 Grumbach S, et al. A Spatial Constraint Database. In: Proc. Workshop on Database Programming Language 1997

6 Chen Lianggang, et al. Interval Constraint Databases and ISQL Language. The Fifth Intl. Conf for Young Computer Scietists. 1999

(上接第103页)

一文件前先打开该文件等)。这些操作序列被检查并确定哪些表示合法操作，将合法操作删除。对剩余的序列进行分析看它们是否可以建立隐通道。

总之，尽管目前有一些形式化的方法来标识隐通道，无论采用什么方法都需要手工进行分析，所有形式化方法都无法代替经验、技巧和敏捷的头脑来完成隐通道的识别。

参考文献

1 National Computer Security Center. Department of Defense Trusted Computer System Evaluation Criteria. Dec. 1985. DoD 5200. 28-STD

2 National Computer Security Center. A Guide to Understanding Covert Channel Analysis of Trusted Systems. NCSC-TG-030, library No. S-240, 572, Nov. 1993

3 McHugh J. Covert Channle Analysis, a Chapter of the Handbook for the Computer Security Certification of Trusted Systems. NRL. Technical Memorandum 5540: 062A, 1996, Naval Research Laboratory, Washington

4 Kemmerer R A. Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channels. ACM Trans. on Computer Systems, Aug. 1983

5 Ne J, Gligor V D. Information Flow Analysis for Covert-Channel Identification in Multilevel Secure Operating Systems. In: Proc. of the 3rd IEEE Workshop on Computer Security Foundations. Franconia, New Hampshire, June 1990. 139～148

6 Tsai C R, et al. A Formal Method for the Identification of Covert Storage Channels in Secure Code. 1987 IEEE Symposium on Security and Privacy, Oakland, CA IEEE Computer Society, Computer Society Press, 1987. 74～86

7 Porras P A, Kemmerer R A. Covert Flow Trees: A Technique for Identifying and Analyzing Covert Storage Channels 1991 IEEE Computer Seciety Symposium on Research in Security and Privacy, Oakland, CA, May, 1991. 36～51

8 Sheih S P, Gligor V D. Auditing the Use of Covert Storage Channels in Secure Systems. In: Proc. of the IEEE Symposium on Research in Security and Privacy. Oakland. California, May 1990

9 Millen J K. Finite-State Noiseless Covert Channels. In: Proc. of the Computer Security Foundations Workshop Franconia, New Hampshire, June 1989. 81～85