半结构化查询重写的 MiniCon 算法*

陶 春+, 汪 卫, 施伯乐

(复旦大学 计算机与信息技术系,上海 200433)

MiniCon Algorithm for Semi-Structured Query Rewriting

TAO Chun+, WANG Wei, SHI Bai-Le

(Department of Computing and Information Technology, Fudan University, Shanghai 200433, China)

+ Corresponding author: Phn: +86-21-65642219, Fax: +86-21-65642219, E-mail: ctao@fudan.edu.cn, http://www.fudan.edu.cn

Received 2003-05-27; Accepted 2004-01-06

Tao C, Wang W, Shi BL. MiniCon algorithm for semi-structured query rewriting. *Journal of Software*, 2004,15(11):1641~1647.

http://www.jos.org.cn/1000-9825/15/1641.htm

Abstract: This paper addresses the semi-structured query rewriting problem for TSL (tree specification language), a language for querying semi-structured data. An algorithm that can find the maximally-contained rewriting query is presented, when a semi-structured query and a set of semi-structured views are given. The idea is borrowed from MiniCon, a scalable relational query rewriting algorithm, and some new problems for semi-structured query rewriting, e.g., object-id dependency and set value variable mapping, are solved. It is shown that the algorithm is correct.

Key words: query rewriting; OEM (object exchange model); query containment; containment mapping; MiniCon; semi-structured data

摘 要: 研究了基于半结构化数据查询语言 TSL(tree specification language)的查询重写问题.提出了一种半结构化查询重写算法,解决了在给定一个半结构化查询和一组半结构化视图的情况下,找到最大被包含重写的问题.算法借用了可伸缩的关系查询重写的 MiniCon 算法的思想,解决了半结构化数据模型之下查询重写的一些新问题(如标识符依赖、集合值变量映射等).证明了算法的正确性.

关键词: 查询重写;OEM(object exchange model);查询包含;包含映射;MiniCon;半结构化数据

中图法分类号: TP311 **文献标识码**: A

由于涉及数据管理的多个方面,如数据集成、查询优化、物理数据独立性等,使用视图回答查询的问题引起了广泛的关注.简而言之,这一问题就是在给定数据库上的一个查询 Q 和一组视图 $\boldsymbol{V}=\{V_1,V_2,\ldots,V_n\}$的情况下,

* Supported by the National Natural Science Foundation of China under Grant No.69933010 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2002AA4Z3430 (国家高技术研究发展计划(863))

作者简介: **陶春**(1974－),男,湖北麻城人,博士生,主要研究领域为数据库,知识库;**汪卫**(1970－),男,博士,教授,主要研究领域为数据库技术,数据挖掘,XML 数据管理;**施伯乐**(1935－),男,教授,博士生导师,主要研究领域为数据库系统及其应用,数据仓库与数据挖掘,数字图书馆,安全数据库.

是否可能只使用视图的结果回答查询 Q;如果可能,如何回答?针对这一问题,传统的研究工作大部分集中于关系查询重写,取得了很多成果[1],其中文献[2]研究了关系查询重写的性能问题,提出了 MiniCon 这一可伸缩的高效重写算法,引起了我们的关注.

人们提出了大量的半结构化数据模型、查询语言、视图定义语言,以满足数据集成、生物数据库、Web 数据查询和管理、数字图书馆等各个领域的半结构化数据管理[3]的需要.大量半结构化数据的出现导致了对半结构化数据的查询重写问题的研究[4,5].文献[4]在树查询语言 TSL(tree specification language)的基础上解决了基于 OEM(object exchange model)模型的半结构化数据的等价查询重写问题;而文献[5]则在提高文献[4]的算法效率方面作了一些有益的探讨.

由于半结构化数据集成系统中数据源往往是自治的,所以各个数据源数据的完备性无法得到保证,因此我们往往需要考虑最大被包含重写(maximally-contained rewriting)问题,以获得尽量完整的结果集.这个问题与文献[4]所讨论的等价重写问题有较大的不同.

本文基于 TSL 查询语言,研究了半结构化最大被包含查询重写问题,提出了半结构化查询重写的 MiniCon 算法.算法借用了 MiniCon 算法的思想,理论分析可以预见其与文献[4]处理等价查询重写的方法相比具有较高的效率.在半结构化查询重写问题中,对象标识符依赖、集合值变量的映射以及包含映射等方面,都会引入关系查询重写中没有的问题.本文研究了这些问题,并提出了相应的解决方案.

本文第 1 节介绍基本概念.第 2 节介绍算法.第 3 节是算法复杂度及正确性论证.最后是结论及展望.

1 基本概念

1.1 OEM数据模型

OEM 数据被表示为一个有根图,其中每个节点称为对象(object),都拥有唯一的对象标识符(object-id)和标签(label).对象分为原子对象和集合对象两类,前者的值(value)是原子值,后者的值是其子对象的集合.用 $\langle Object\text{-}id\ Label\ Value\rangle$ 表示 OEM 对象,这种表示形式又称为对象模式(object pattern).集合对象则表示为 $\langle Object\text{-}id\ Label\ \{SubObj_1,\ldots,SubObj_n\}\rangle(n\geq 0)$,当 $n>0$ 时 $SubObj_i(1\leq i\leq n)$是子对象.

对象标识符通常是原子值,但也可能以函数形式表示.

1.2 TSL查询

一个 TSL 查询[4]是一个规则,由规则头开始,在:-之后是规则体.规则头定义结果图中的结果对象,而规则体则包含数据源满足的条件.规则头和规则体均基于对象模式表示,其中每个叶对象模式称为子目标.

例 1:TSL 查询定义$\langle f(P)\ student\ \{\langle X\ Y\ Z\rangle\}\rangle$:- $\langle P\ person\ \{\langle R\ profession\ `student'\rangle\ \langle X\ Y\ Z\rangle\}\rangle$@db 表示:如 db 中含标识符为 P,标签为 *person* 的顶层对象;并且该对象含标识符 R,标签 *profession*,值为 *student* 的子对象,以及子对象$\langle X\ Y\ Z\rangle$,则构造结果对象,标识符为 $f(P)$,标签为 *student*,并且包含子对象$\langle X\ Y\ Z\rangle$.

TSL 查询规则头中出现的变量称为头变量;不在规则头中的变量称为体变量.当查询由头相同的多个 TSL 规则定义时,可合并起来,使用并连接各规则体,形成新 TSL 查询.不含并(∪)操作符的 TSL 查询称为连接 TSL 查询.规则体中所有集合型值域最多含一个对象模式的 TSL 查询称为规范化 TSL 查询.所有 TSL 查询均可规范化(normalize).TSL 查询规范化后,其中每个子目标称为该 TSL 查询的单路径子目标.TSL 查询如果规范化后仅含 1 个子目标,则称单路径查询.如果 TSL 查询的头是一个单路径子目标,称该查询为单路径头查询.每个 TSL 查询均可分为若干单路径头子查询.规则头中所有变量都在规则体中出现的 TSL 查询称为安全 TSL 查询(不加特别说明,本文只研究安全 TSL 查询).命名的连接 TSL 查询称为 TSL 视图.

对于 TSL 查询 Q,用 $head(Q)$记 Q 的规则头,用 $Vars(Q)$记 Q 中变量集合,用 $Subgoals(Q)$记 Q 中单路径子目标集合,用 $ObjPtns(Q)$记 Q 中对象模式集合,用 $Q[D]$记数据库 D 上 Q 的结果对象集.

对于任意半结构化数据库,生成标识符的函数不应与标识符依赖 $Object\text{-}id\rightarrow(Label,Value)$冲突.如 TSL 查询$\langle f(X)\ Z\ W\rangle$:- $\langle X\ properties\ \{\langle Y\ Z\ W\rangle\}\rangle$@db 中 f 就可能与标识符依赖 $f(X)\rightarrow(Z,W)$冲突,因此不合法.

1.3 TSL查询的包含和等价

称半结构化数据库 D_1 包含 D_2(记为 $D_1 \supseteq D_2$ 或 $D_2 \subseteq D_1$),如果:(1) D_1,D_2 根对象的标识符和标签相同;(2) $Object\text{-}id(D_1) \supseteq Object\text{-}id(D_2)$;(3) 对于 D_2 中任一原子对象,D_2 中一定存在相同的原子对象;(4) 对于 D_2 中任一集合对象 o_2,D_1 中标识符相同的对象 o_1 一定是集合对象,并且 $Label(o_1)=Label(o_2)$,$Value(o_1) \supseteq Value(o_2)$.

如果对于半结构化数据库 D_1,D_2,有 $D_1 \supseteq D_2$ 并且 $D_2 \supseteq D_1$,则称 D_1 等价于 D_2,记为 $D_1=D_2$.

对于 TSL 查询 Q_1,Q_2 及任意数据库 D,如果 $Q_1[D] \supseteq Q_2[D]$,则称 Q_1 包含 Q_2,记为 $Q_1 \supseteq Q_2$ 或 $Q_2 \subseteq Q_1$.同样,如果 $Q_1 \supseteq Q_2$ 并且 $Q_2 \supseteq Q_1$,则称 Q_1 等价于 Q_2,记为 $Q_1=Q_2$.

1.4 查询重写

给定 TSL 查询 Q 和 TSL 视图集合 $\boldsymbol{V}=\{V_1,V_2,\ldots,V_n\}$,$Q$ 使用 $\boldsymbol{V}$ 的查询重写就是 TSL 查询 Q',其规则体仅使用 $\boldsymbol{V}$ 中视图构成子目标.如对 Q 使用 $\boldsymbol{V}$ 的查询重写 Q' 满足对于任意的数据库 D,$Q'[V_1[D],V_2[D],\ldots,V_n[D]] \subseteq Q[D]$,称 Q' 是 Q 使用 $\boldsymbol{V}$ 的被包含重写.如果不存在 Q 的其他被包含重写 Q'' 满足 $Q'' \neq Q'$,并且 $Q'[V_1[D],V_2[D],\ldots,V_n[D]] \subseteq Q''[V_1[D],V_2[D],\ldots,V_n[D]]$,则称 Q' 是 Q 使用 $\boldsymbol{V}$ 的最大被包含重写.

2 TSL 查询重写的 MiniCon 算法

虽然非递归的 TSL 查询可以简化为递归的 Datalog 程序[4],但是关系查询重写的大量研究结果[1]并不能直接用于 TSL 查询重写.半结构化查询重写问题具有如下的新特点:(1) 变量之间隐含着对象标识符依赖;(2) 对象模式中值变量具有二义性,既可能映射为原子值(变量),也可能映射为对象模式的集合,因此包含映射需要更加复杂的定义;(3) 函数符号的引入也会带来额外的复杂性.这些特点在半结构化查询重写中需要解决.

查询重写算法的基本思想就是搜索所有可能的重写,然后将合格的重写加入结果集中.假设查询有 n 个子目标,视图中子目标的最大个数为 m,视图的个数为 M,查询重写搜索问题的复杂度是 $(mM)^n$,是一个指数级问题.关系查询重写的 MiniCon 算法[2]则从性能的角度研究关系查询重写问题,虽然复杂度没有变化,但实验结果表明,它比以往的算法效率要高得多,具有很好的可伸缩性.它提升效率的基本思想就是减少子目标匹配的重复性工作,尽早裁剪不必要的搜索分支.这在 TSL 查询重写中完全可以借鉴.

2.1 包含映射、对象模式融合

对于关系查询 Q_1 和 Q_2,$Q_1 \supseteq Q_2 \Leftrightarrow$存在一个从 Q_1 到 Q_2 的包含映射.TSL 包含映射的定义有点不同.将所有变量的集合记为 $Vars$,将所有对象模式的集合记为 $ObjPtns$,将所有常量集合记为 $Cons$,并用 $SubSets(S)$ 记集合 S 所有子集的集合,用 $Sets(S)$ 记集合 S 的所有单元素构成集合以及空集的集合,称一个从 $Vars$ 的有限子集 X 到 $Vars \cup Cons \cup SubSets(ObjPtns)$ 的映射 τ 为 TSL 映射,如果 τ 将 X 中的标识符变量映射为标识符变量,将 X 中的标签和原子值变量映射到 $Vars \cup C$,将 X 中的值变量映射到 $Vars \cup Cons \cup SubSets(ObjPtns)$.

定义 1(TSL 包含映射). 对于 TSL 查询 Q_1 和 Q_2,一个从 $Vars(Q_2)$ 到 $Vars(Q_1) \cup Cons \cup SubSets(ObjPtns(Q_1))$ 的 TSL 映射 τ 是一个从 Q_2 到 Q_1 的 TSL 包含映射,如果:(1) τ 将 Q_2 规则体中的任一单路径子目标映射为 Q_1 规则体中的某个子目标;(2) $\tau(head(Q_2)) \supseteq head(Q_1)$.

引理 1. 对于 TSL 查询 Q 和 Q_2,$Q_2 \supseteq Q \Leftrightarrow$从 Q_2 到 Q 的任一单路径头子查询都存在 TSL 包含映射.

证明:充分性比较明显,以下证明必要性.假设对于 Q 的某单路径子查询 Q_1,不存在 TSL 映射使得定义 1 的(1)和(2)同时满足.明显地,$Q \supseteq Q_1$.

先考虑所有映射都不满足(2)的情况,此时 $head(Q_1)$ 中的单路径子目标无法从 $head(Q_2)$ 映射得到,因此一定存在某数据库 D 使得该单路径子目标产生一个 $head(Q_2)$ 无法产生的对象,与 $Q_2 \supseteq Q_1$ 矛盾.

接着考虑任意一个满足(2)的映射 τ,假设它们都无法满足(1),如此一定存在 Q_2 的一个单路径子目标 g,使得 τ 无法将其映射到 Q_1 的某个子目标上.

如果 g 根本无法满足(1),择满足 Q_1 子目标而不满足 g 子目标的数据库 D,即得到与 $Q_2 \supseteq Q_1$ 矛盾的例子.

否则,将 τ 中映射到 Q_1 头的映射子集记为 τ',并设映射 τ'' 使得 g 映射到 Q_1 中某子目标 g_1,τ' 与 τ'' 矛盾,故 g 中

一定存在某些变量 X 使得 $\tau''(X)\neq\tau'(X)$.不失一般性,设只有这样一个子目标 g 以及一个 X,据 Q_1 中查询子目标将所有变量小写常量化,构造一个数据库 D.$Q_1[D]$有至少 1 个结果对象 o(即 $head(Q_1)$变量小写常量化),而对 Q_2,如果 g 中的 X 变量映射到 $\tau'(X)$,则 $Q_2[D]$的结果集将会是空,因 g 无法匹配;如 X 变量映射到 $\tau''(X)$,则 $Q_2[D]$的对象中 X 变量处值与 $Q_1[D]$中不同,即 $o\notin Q_2[D]$.都与 $Q_2\supseteq Q_1$ 矛盾.

假设不成立,必要性成立,证毕. □

对象模式融合[4]就是使用 Chase 技术根据标识符依赖 *Object-id*→(*Label*,*Value*)进行对象模式合并.其基本原则是,碰到相同标识符对象模式时,将集合变量扩展,将值中对象模式并到一个集合值中,将标签变量合为一个.这项操作在查询处理的过程中经常用到,有兴趣的读者可以参见文献[4],这里不再赘述.

2.2 算法第1步:形成MCD

对于 TSL 查询 Q 和视图 V,给定一个 TSL 映射 τ,称 V 的子目标 g_1 覆盖 Q 的单路径子目标 g,如果 $\tau(g)=g_1$.而一个 MCD(MiniCon 描述)是从 $Vars(Q)$的子集到 $Vars(V)\cup Cons\cup Sets(ObjPtns(V))$的 TSL 映射,且使得 Q 中的若干个单路径子目标分别映射到 V 的某个单路径子目标.直觉上,一个 MCD 代表了从 Q 到 Q 的某个连接查询重写的包含映射的一部分.构造 MCD 的方法保证以后这些部分可以无缝地组合在一起.

例 2:考虑如下 TSL 查询 $Q1$,$Q2$ 和视图 V.

$Q1$: ⟨$f(X)$ *pr* {⟨$g(Y1, Y2)$ *c Z*⟩}⟩ :- ⟨*X pr* {⟨*Y1 a Z*⟩ ⟨*Y2 b Z*⟩}⟩@*db*.

$Q2$: ⟨$f(X)$ *pr* {⟨$g(Y1, Y2)$ *c Z*⟩}⟩ :- ⟨*X pr* {⟨*Y1 a d*⟩ ⟨*Y2 b Z*⟩}⟩@*db*.

V: ⟨$f(X)$ *pr* {⟨*Y1 a Z1*⟩ ⟨*Y2 b Z2*⟩}⟩ :- ⟨*X pr* {⟨*Y1 a Z1*⟩ ⟨*Y2 b Z2*⟩}⟩@*db*.

可以看出,使用 V 可以生成 $Q1$,$Q2$ 如下的连接查询重写 $Q1'$和 $Q2'$:

$Q1'$: ⟨$f(X)$ *pr* {⟨$g(Y1, Y2)$ *c Z*⟩}⟩ :- ⟨$f(X)$ *pr* {⟨*Y1 a Z*⟩ ⟨*Y2 b Z*⟩}⟩@V.

$Q2'$: ⟨$f(X)$ *pr* {⟨$g(Y1, Y2)$ *c Z*⟩}⟩ :- ⟨$f(X)$ *pr* {⟨*Y1 a d*⟩ ⟨*Y2 b Z*⟩}⟩@V.

但无论如何,按照定义,都无法找到从 $Q1$ 或 $Q2$ 到 V 的 MCD.此例说明,有时不能找到 Q 到视图 V 的 MCD,但可以找出 Q 到 V 头变量特化后的视图的 MCD.

定义 2(头特化). 对于 TSL 视图 V,一个头特化是从 $Vars(head(V))$到 $Vars(head(V))\cup Cons\cup Sets(ObjPtns)$的一个 TSL 映射 h,它满足:如果 $h(X)\in Vars(head(V))$,则 $h(h(X))=h(X)$.

定义 3(MCD). 对 TSL 查询 Q 和视图 V,一个 V 上 Q 的 MCD C 是三元组(h_C,Φ_C,G_C),其中:

1. h_C 是视图 V 的一个头特化;
2. G_C 是 Q 规则体中某些单路径子目标的集合;
3. Φ_C 是一个从 $Vars(G_C)$到 $Vars(h_C(V))\cup Cons\cup Sets(ObjPtns(h_C(V)))$的 TSL 映射;
4. 对于 G_C 中每个单路径子目标 g,$\Phi_C(g)\in h_C(V)$.

基于对如下性质的观察,我们将提出寻找映射以及构造 MCD 的算法.

性质 1(MCD 可用性). 设 C 为 TSL 视图 V 上 Q 的 MCD.C 可用于 Q 的非冗余查询重写,如果:

C1. 对头变量 $X\in Vars(G_C)$,(1) 如果 $\Phi_C(X)\in Vars(h_C(V))$, 则 $\Phi_C(X)\in Vars(head(h_C(V)))$; 如果 $\Phi_C(X)\in Sets(ObjPtns(h_C(V)))$,则$\Phi_C(X)$包含的对象模式中出现的所有变量都属于 $Vars(head(h_C(V)))$;(2) 如果 $Vars(G_C)$中存在标识符变量 $Y\rightarrow X$,则$\Phi_C(Y)\in Vars(head(h_C(V)))$.

C2. 如果$\Phi_C(X)\notin Vars(head(h_C(V)))$,或者当$\Phi_C(X)$是对象模式值且该模式中存在 $h_C(V)$的某些体变量时,则对包含 X 的每个 Q 中单路径子目标 g,如果 G_C 的所有与 g 中决定 X 的标识符依赖同构的依赖的标识符变量不在 g 中,则:(1) $g\in G_C$;(2) $\Phi_C(g)\in h_C(V)$.

C3. 不存在某个满足如上条件的 V 上 Q 的 MCD $C'(h_C',\Phi_C',G_C')$,使得 $G_C'\subset G_C$.

条件 C1 保证视图在头特化后输出了查询中的头变量.条件 C2 的含义是,如果某个变量 X 无法在映射后的头特化视图中头部输出,则一定要把其他包含 X 的单路径子目标放到 G_C 中.条件 C3 保证覆盖子目标集的最小性.

算法 1. 构造 MCD.

输入:Q 是 TSL 查询,$\boldsymbol{V}=\{V_1,V_2,\ldots,V_n\}$是 TSL 视图集合.

输出:对于 $\boldsymbol{V}$ 中每个视图,相应 MCD 的集合 $\boldsymbol{C}$.

$\boldsymbol{C}=\varnothing$

对于 Q 中每个单路径子目标 g

对于视图 $V\in\boldsymbol{V}$ 和 V 中每个单路径子目标 v

使用映射寻找算法(限于篇幅,从略)试图找到如下的 h 和Φ:

设 h 为 V 上最小受限头特化,使得存在一个映射Φ,$\Phi(g)=h(v)$

如果 h 和Φ都存在,则可以如下构造添加到 $\boldsymbol{C}$ 中的新 MCD C

(1) Φ_C(以及 h_C)是Φ(以及 h)的扩充;

(2) G_C 是 Q 中最小子目标集使得 G_C,Φ_C 和 h_C 构成满足性质 1 的 MCD.

2.3 算法第2步:组合MCD

第 2 步是考虑组合 MCD 生成的 TSL 连接查询重写.最终的重写就是所有这些连接查询重写规则的并.

性质 2(MCD 组合条件). 给定一个 TSL 查询 Q,一组 TSL 视图集合 $\boldsymbol{V}$,以及 Q 在 $\boldsymbol{V}$ 中视图上的 MCD 集 $\boldsymbol{C}$,只有满足如下条件的 MCD $C_1,C_2,\ldots,C_n$ 组合才可能生成 Q 的非冗余查询重写:

D1. $G_{C_1}\cup G_{C_2}\cup\ldots\cup G_{C_n}=Subgoals(Q)$

D2. 对于任意的 $i\neq j$, $G_{C_i}\cap G_{C_j}=\varnothing$

算法 2. 组合 MCD.

输入:Q 是 TSL 查询,$\boldsymbol{V}=\{V_1,V_2,\ldots,V_n\}$是视图的集合,$\boldsymbol{C}$ 是所有 MCD 的集合.

输出:连接查询重写规则集.

$Answer=\varnothing$

对于 $\boldsymbol{C}$ 的每个满足以上性质 2 的 MCD $C_1,\ldots,C_n$

将各 $h_{C_i}(V_{C_i})$中的变量全部换为全新的变量,并且互不重复,并处理Φ_{C_i}

若某非标识符头变量 X 的某同构标识符依赖对应的所有 MCD C_i 都未在 $h_{C_i}(head(V_{C_i}))$输出$\Phi_{C_i}(X)$

则该 MCD 组合无效,继续下一个;

对 Q 的被不同 MCD 共享的变量逐个一致化(对于对象模式可能要进行合一)

使得这些变量映射的目标在不同共享的 MCD 一致

发生不一致时,该组合失败,继续下一个 MCD 组合

否则,设一致化以后的总映射为 Ψ

如果 $\Psi(head(Q))$:- $h_{C_1}(\text{head}(V_{C_1}))$ AND…AND $h_{C_n}(head(V_{C_n}))$是安全的

将其加入 $Answer$ 中

否则继续下一个 MCD 组合

输出 $Answer$

$Answer$ 可能无法合并为一个规则,但无须强求,$Answer$ 规则集仍可被看作是一个 TSL 查询重写,因为其头部都可以从 $head(Q)$映射过去,这决定了查询生成的所有结果对象是可以进行融合的.

3 算法复杂度、正确性证明

3.1 复杂度

设查询有 n 个单路径子目标,有 M 个视图,视图中最多有 m 个单路径子目标,则算法的复杂度为$(nMm)^n$.

虽然与类似的文献[4]中的穷举算法相比,复杂度并没有减少,但由于应用了 MiniCon 算法的思想,精心选择 MCD 使得算法在早期大大裁剪了不必要的搜索分支,从理论上可以预计实际运行效率会有大幅提升.

3.2 正确性证明

如果不加特殊说明,本节中 Q 是 TSL 查询,$\boldsymbol{V}$ 是 TSL 视图集合.*Answer* 就是算法试图找到的 Q 使用 $\boldsymbol{V}$ 的最大被包含重写.

由于 MCD 的定义及算法 2,容易看出,一致化以后的总映射 Ψ就是从 Q 到相应的查询重写规则的 TSL 包含映射的超集.

引理 2. 基于 TSL 的 MiniCon 算法结果中每个 TSL 查询重写规则 Q_1 都满足 $Q_1 \subseteq Q$.

对于 Q 使用 $\boldsymbol{V}$ 的每个 TSL 连接查询重写 Q_1,其中的每个视图子目标都是某个视图 $V \in \boldsymbol{V}$ 的头特化 h 映射后的头部.由于头特化中集合变量不一定映射到单个对象模式的集合,所以需要先将映射到多个对象模式集合头特化的情况排除.

引理 3. 对于 Q 使用 $\boldsymbol{V}$ 的某个 TSL 连接被包含查询重写 Q_1,如果其中的视图头特化映射存在集合变量映射到多个对象模式的集合的情况,则一定存在若干个 TSL 连接查询重写,使得它们的并包含 Q,并且其中每个查询重写的所有头特化映射都不存在集合变量映射到多个对象模式的集合的情况.

在找到某个 TSL 查询重写 Q_1 中各个视图的头特化以后,将使用视图规则将 Q_1 展开,并对其中各视图中的体变量的每次展开赋予一个新的变量名,并记展开后的 TSL 查询为 $exp(Q_1)$.由引理 1,一定存在一个从 Q 到 $exp(Q_1)$的 TSL 包含映射.如果在某一 TSL 包含映射下,存在 Q_1 中某个视图头特化展开后不覆盖任何 Q 中子目标;或者存在 Q 的某个单路径子目标,映射到 $exp(Q_1)$的两个不同单路径子目标,都称 Q_1 是 Q 使用 $\boldsymbol{V}$ 的冗余的 TSL 被包含查询重写.否则,称 Q_1 是 Q 使用 $\boldsymbol{V}$ 的非冗余的 TSL 被包含查询重写.

引理 4. 如果存在某个 Q 使用 $\boldsymbol{V}$ 的 TSL 连接被包含查询重写 Q_1,则一定存在一个 Q 使用 $\boldsymbol{V}$ 的非冗余的 TSL 查询重写 Q_2,使得 $Q_1 \subseteq Q_2$.

证明:如果 Q_1 本身就是非冗余的,则取 $Q_2=Q_1$,结论明显成立.因此,设 Q_1 是冗余的.

如果存在 Q_1 中某个视图头特化子目标展开后不覆盖任何 Q 中子目标,则可以去掉这个视图头特化子目标得到 Q_1',明显地,$Q_1 \subseteq Q_1'$并且 $exp(Q_1')$保持了 Q 到 $exp(Q_1)$的包含映射.不失一般性,设 Q_1'中任何视图头特化子目标展开后至少覆盖 Q 中的 1 个子目标.

对于 $exp(Q_1')$,如果存在 Q 的某个单路径子目标 g,映射到 $exp(Q_1)$的两个不同单路径子目标(设对应的视图分别是 V_1 和 V_2),要证明一定存在一个 Q 使用 $\boldsymbol{V}$ 的非冗余的 TSL 查询重写 Q_2,使得 $Q_1' \subseteq Q_2$.

按照展开重写的方法,因为视图的每次展开时候体变量都是全新的,所以该子目标 g 中所有变量映射的目标不可能包含 V_1 或者 V_2 的体变量.设 g 映射目标中包含的 Q_1'中的变量集为 *vars*,对于 V_1 头特化子目标中 *vars* 中的变量进行异化,即逐个替换为全新的变量.由此,得到查询重写 Q_1''.明显地,$Q_1' \subseteq Q_1''$.而且很容易发现,只需对从 Q 到 $exp(Q_1')$的包含映射作相应的调整,就可以得到从 Q 到 $exp(Q_1'')$的包含映射,即 Q_1''仍然是 Q 使用 $\boldsymbol{V}$ 的连接包含查询重写.不失一般性,设 Q_2 是从 Q_1'得到的对每个 Q 中单路径子目标都仅仅映射到 1 个 $exp(Q_2)$中的单路径子目标的查询重写,Q_2 就是要找的非冗余的查询重写.证毕. □

引理 5. 对于 Q 使用 $\boldsymbol{V}$ 的某个非冗余的 TSL 连接包含查询重写 Q_1,如果其中每个连接查询重写的所有头特化映射都不存在集合变量映射到多个对象模式的集合的情况,则 Q_1 的某个变量名替换一定能被本文的算法找到,即 $Q_1 \in Answer$.

要证明 Q 中单路径子目标集合可以按照性质 2 分割为 n 个子集,其中每个子集都是某个满足性质 1 的 MCD C 的 G_C;并且要证明 formMCDs 能够找到所有满足性质 1 的 MCD,而 combineMCDs 则能够找到所有满足性质 2 的分割.限于篇幅,证明从略.

定理 1. 基于 TSL 的 MiniCon 算法得到的 *Answer* 是 Q 使用 $\boldsymbol{V}$ 的最大被包含重写.

由引理 2 可知,算法产生的 *Answer* 是 Q 使用 $\boldsymbol{V}$ 的被包含重写.为了证明其最大性,需要证明任何的 TSL 被包含连接查询重写,都被 *Answer* 包含.证明时,引理 3 说明了不用考虑视图头特化映射存在集合变量映射到多个对象模式的集合的被包含重写;引理 4 则说明了可以不用考虑冗余的 TSL 被包含查询重写;引理 5 则对于非冗余的被包含连接查询重写 Q_1,且不存在视图头特化映射中集合变量映射到多个对象模式的集合的情况时,说明

了 Q_1 一定能被本文的算法找到并放到 *Answer* 中.结合引理 1,定理结论成立.

4 结论和展望

本文将 MiniCon 算法的思想融入基于 TSL 的最大被包含查询重写算法之中,对于半结构化数据的查询重写问题开展了一些突破性的工作.我们定义了基于 TSL 的最大被包含重写的概念,并证明了我们的算法能够找到这样的查询重写.由于算法融入了 MiniCon 的思想,从变量绑定的角度来看,其实 MiniCon 思想的可伸缩性仍然可以在计算半结构化查询重写时表现出来,所以我们相信,相对于其他半结构化最大被包含查询重写算法,该算法的可伸缩性是很好的.

由于半结构化数据库以及查询的复杂性,我们尚未在算法性能方面进行实际的评测工作.在以后的工作中,我们可以实现本文算法,并进行一些性能方面的评测工作.另外,由于现今的 XML 查询语言多具有正则表达式的能力,给查询重写带来新的挑战,也是一个新的研究问题.

References:

[1] Halevy AY. Answering queries using views: a survey. The VLDB Journal—The Int'l Journal on Very Large Data Bases, 2001, 10(4):270~294.

[2] Pottinger R, Halevy AY. MiniCon: A scalable algorithm for answering queries using views. The VLDB Journal—The Int'l Journal on Very Large Data Bases, 2001,10(2-3):182~198.

[3] Abiteboul S, Buneman P, Suciu D. Data on the Web: From Relations to Semistructured Data and XML. San Francisco: Morgan Kaufmann Publishers, 2000.

[4] Papakonstantinou Y, Vassalos V. Query rewriting for semistructured data. In: Proc. of the 1999 ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 1999. 455~466.

[5] Gao J, Tang SW, Yang DQ, Wang TJ. Query rewriting for semi-structured data. Journal of Computer Research and Development, 2002,39(2):165~171 (in Chinese with English abstract).

附中文参考文献:

[5] 高军,唐世渭,杨冬青,王腾蛟.半结构化数据查询重写.计算机研究与发展,2002,39(2):165~171.