基于有向图的对象范式生成算法*

刘国华[1,2+], 汪 卫[1], 张 亮[1], 施伯乐[1]

[1](复旦大学 计算机与信息技术系,上海 200433)

[2](燕山大学 计算机科学与工程系,河北 秦皇岛 066004)

An Algorithm for Creating an Object Normal Form Based on Directed Graph

LIU Guo-Hua[1,2+], WANG Wei[1], ZHANG Liang[1], SHI Bai-Le[1]

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

[2](Department of Computer Science and Engineering, Yanshan University, Qinhuangdao 066004, China)

+ Corresponding author: Phn: +86-21-65642219, E-mail: liugh@hotmail.com, http://www.cit.fudan.edu.cn

Received 2003-04-09; Accepted 2003-05-27

Liu GH, Wang W, Zhang L, Shi BL. An algorithm for creating an object normal form based on directed graph. *Journal of Software*, 2004,15(5):730~740.

http://www.jos.org.cn/1000-9825/15/730.htm

Abstract: In this paper, the major ideas of the normalization theory for object-oriented data models proposed by Tari et al. is introduced, their methods for creating an object normal form are analyzed, and the problems in these methods are pointed out. In order to develop a new method for creating such an object normal form, the meaning of a vertex of the directed graph is extended, such that it is not only a simple vertex, but also a directed graph. Based on this extended directed graph, an algorithm for creating it is proposed, the time complexity analysis of the algorithm is given, and its correctness is proved.

Key words: object-oriented data model; normalization; normal form; directed graph

摘 要: 对 Tari 等人提出的面向对象数据模型规范化理论的基本思想进行了介绍,分析了他们给出的对象范式生成方法,指出了这些方法所存在的问题.为了研究新的对象范式生成方法,对有向图顶点的含义进行了扩充,使其不仅可以是一个简单的顶点,还可以是一个有向图.基于这种扩充有向图,提出了一种对象范式生成算法,并给出了算法的时间复杂度分析和正确性证明.

关键词: 面向对象数据模型;规范化;范式;有向图

中图法分类号: TP311 **文献标识码**: A

1997 年,Tari 等人在文献[1]中首次提出了面向对象数据模型的规范化理论.与关系模型规范化理论相类似,

* 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.2002AAA444130 (国家高技术研究发展计划(863))

作者简介: 刘国华(1966－),男,黑龙江齐齐哈尔人,博士,教授,博士生导师,主要研究领域为数据库理论,数字图书馆,Web 数据管理;汪卫(1970－),男,博士,教授,主要研究领域为数据库,数据仓库;张亮(1963－),男,博士,教授,主要研究领域为信息集成,支持多媒体应用的数据库技术;施伯乐(1935－),男,教授,博士生导师,主要研究领域为数据库理论及应用.

它也是以数据依赖为基础.但两者的数据依赖在种类和内涵上有所不同.关系模型的数据依赖包括函数依赖、多值依赖和连接依赖,它们被用来表示关系中属性间的联系.函数依赖是关系模型规范化理论中最重要的数据依赖,2NF 至 BCNF 都是基于函数依赖来定义的.面向对象数据模型的数据依赖有 3 种,即路径依赖(path dependency)、局部依赖(local dependency)和全局依赖(global dependency).它们是关系模型中函数依赖的扩展,用它们不仅可以表示对象内部属性间的联系,还可以表示对象间及复杂对象与其成员间的联系[1].全局依赖(global dependency)是面向对象数据模型规范化理论中最重要的数据依赖,对象范式是根据全局依赖(global dependency)来定义的.关系数据模型规范化理论主要应用于逻辑模型设计阶段,解决影响数据库系统性能的冗余和更新异常问题[2],即它是面向系统的.面向对象数据模型规范化理论主要解决如何设计出正确的数据库模式问题.所谓正确的数据库模式是指它能够反映出用户(即领域专家)关于论域(UoD)的解释(interpretation),因此,面向对象数据模型规范化理论是应用于概念模型设计阶段的[1,3],而不是面向系统的.在概念模型设计阶段,衡量面向对象数据库模式设计正确与否的标准是对象范式.Tari 等人把用户(即领域专家)关于论域(UoD)中客观事物的解释形式化为一个全局依赖(global dependency)集,并称其为用户解释,把对象反映的解释(interpretation)也形式化为一个全局依赖(global dependency)集,并称其为对象解释.一个对象可能有多个对象解释,把对象的所有对象解释并在一起后构成的全局依赖(global dependency)集称为对象模型(object model).如果一个对象的对象模型能够推导出它所对应的用户解释,那么,该对象就是对象范式.Tari 等人的最大贡献是把判断数据库模式是否能够反映出用户(即领域专家)关于论域(UoD)的解释(interpretation)问题,转化为判断一个全局依赖(global dependency)集(对象模型)是否蕴含另一个全局依赖(global dependency)集(用户解释)的问题.

在文献[1]中,Tari 等人提出了两种不同的对象范式生成方法.第 1 种把现存的面向对象模式当作一个初始设计,然后,不断地对它进行调整直到正确为止,即把它变成了一个对象范式.这是一种自然迭代的方法,它与面向对象方法学相一致[4].第 2 种方法不把现存的面向对象模式作为主要信息来源,仅当作一个“起点”.其原理是把现存的对象模式中的所有属性都放在同一层上,然后,从用户解释开始,利用簇的关键字来确定属性间的层次关系.这是一个根据用户解释来构造对象结构的过程,所得到对象的对象模型(object model)一定能够推导出用户解释,即它能够生成一个对象范式.

在现实世界中,用户解释有时会含有冲突,这给对象规范化带来了一定的困难.文献[1]中的第 1 种对象范式生成方法没有检测用户解释含有冲突的机制.当用户解释含有冲突时,它将进入死循环.第 2 种对象范式生成方法检测用户解释含有冲突的策略是看算法的执行时间是否超过了时间复杂度 $O(s\times r)$.显然,这是不可行的.可见,文献[1]中的方法对于用户解释含有冲突的情况处理得不是很得当.因此,这两种方法是有局限性的.用户解释中冲突的识别问题是文献[1]所遗留的问题之一.为了解决这个遗留问题,我们进行了较深入的研究,取得了一些成果[5,6].这些成果为本文方法的提出奠定了基础.

本文提出的对象范式生成方法与 Tari 等人的第 2 种规范化方法相似,也是根据用户解释来构造对象结构.但两者的根本区别在于,用来确定属性间层次关系的方法有所不同.Tari 等人利用簇的关键字来确定属性间的层次关系.本文是利用对象结构图(后面将给出其定义和构造方法)来确定属性间的层次关系.

1 面向对象数据模型的规范化理论

下面介绍文献[1]提出的面向对象数据模型规范化理论的基本思想.

1.1 面向对象数据模型和面向对象数据库

1.1.1 面向对象数据模型

文献[1]中采用的面向对象数据模型仅涉及复杂对象(complex object)、对象标识(object identity)、继承(inheritance)和类/类型(class/type)等与结构有关的面向对象概念,而不考虑与行为有关的概念,如封装(encapsulation)、替换(overriding)和延迟绑定(late binding).实体对象和属性对象都统称为对象,并用 $O(\Delta_O)$来表示,Δ_O为对象的类型(type).

定义 1.1(对象实例). 根据对象类型Δ_O,对对象 O 的每个成员都赋予一个值,就能得到对象 O 的一个实例.

每个对象实例都有一个对象标识符.实体对象实例的标识符是系统给的,属性对象实例的标识符(即对象关键字,具体定义后面将给出)是通过依赖约束推导出来的.

1.1.2 面向对象数据库

定义 1.2(面向对象模式). 面向对象模式(object-oriented schema)可以定义为一个带有依赖约束(后面将对其进行说明)的对象集合,用$\langle \odot,\Sigma\rangle$表示,其中,$\odot$是对象的集合,$\Sigma$是$\odot$上的依赖约束集.

定义 1.3(面向对象数据库). 面向对象数据库(object-oriented database)可形式化定义为$\langle\langle \odot,\Sigma\rangle,\Psi\rangle$,其中$\Psi$是$\odot$中元素的对象实例.图 1 表示的是一个面向对象模式.

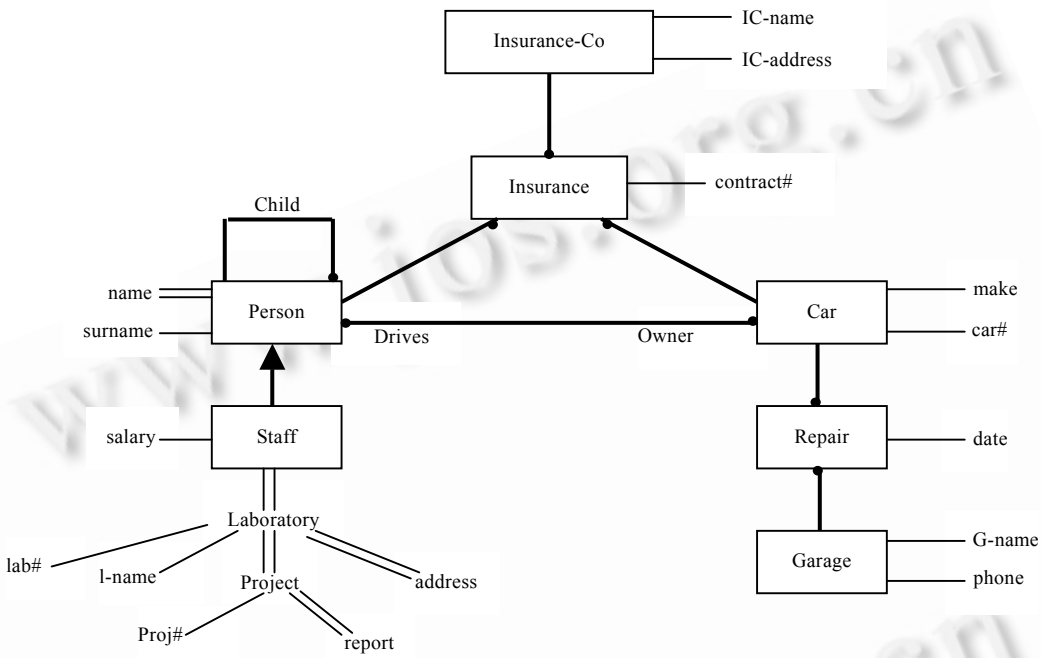

Fig.1 Diagram of an example of object-oriented schema

图 1 面向对象模式图例

在图 1 中,方框表示一个实体对象,每一个实体对象都带有属性对象.单线表示单值属性,双线表示多值属性.实体对象间的联系用粗实线来表示.一端带圆点的粗实线表示聚集(aggregation)关系,两端带圆点的粗实线表示联合(association)关系,粗箭头表示继承(inheritance)关系.

1.2 路径和依赖约束

1.2.1 路 径

在面向对象模式中,对象间不仅可以通过聚集(aggregation)关系、联合(association)关系、继承(inheritance)关系和属性关系相关联,还可以通过路径(path)相关联.面向对象数据模型的数据依赖正是基于路径这种关联关系提出来的.

定义 1.4(路径). 路径就是模式中对象的序列.在该序列中,每一个对象与其前驱(若有)和后继(若有)之间或存在聚集(aggregation)、联合(association)、继承(inheritance)关系,或存在属性关系.对象 O_1 与 O_n 间的路径表示为 O_1-O_2-...-O_n(n≥2).根据对象与其前驱和后继之间的关系,可以把路径分为以下 3 种形式:

- 垂直路径(vertical path):路径中的对象与其前驱和后继之间只存在属性关系.图 1 中的 Staff-Laboratory-Project 就是一个垂直路径.
- 水平路径(horizontal path):路径中的对象与其前驱和后继之间只存在聚集(aggregation)、联合(association)、继承(inheritance)关系.图 1 中的 Person-Car-Insurance 就是一个水平路径.
- 混合路径(composite path):路径中的对象与其前驱和后继之间或存在聚集(aggregation)、联合

(association)、继承(inheritance)关系,或存在属性关系.图 1 中的 Person-Car-Insurance-contract#就是一个混合路径.

定义 1.5(路径实例). 路径 O_1-O_2-...-O_n($n\geq 2$)的路径实例是形如 τ_1-τ_2-...-τ_{n-1} 的实例序列,其中,τ_i 是 O_i 的一个实例,$i\in\{1,...,n-1\}$.在图 1 中,$P1$-$C2$-$I2$ 就是路径 Person-Car-Insurance-contract#的一个实例,其目的是要找出 $P1$ 为其所驾驶的车 $C2$ 上的全部保险.

定义 1.6(投影). 对象沿着路径或路径实例的投影是一个多重投影的嵌套过程.令 τ 为对象 O 的一个对象实例,ρ 为一条路径或路径实例,记 $\tau[O_i]$ 为 τ 在对象 O_i 上的投影,$\tau_\rho[O_i]$ 为 τ 沿着 ρ 在对象 O_i 上的投影.若 ρ 为 O-O_1-O_2-...-O_q,τ 为对象 O 的一个实例,则 $\tau_\rho[O_q]=(\tau_\rho[O_{q-1}])[O_q]=((...((\tau_\rho[O_1])[O_2])...)[O_{q-1}])[O_q]$.

1.2.2 依赖约束

在面向对象模式$\langle \odot,\Sigma\rangle$中,$\Sigma$ 为 $\odot$ 上的依赖约束集.根据对象间的路径,把依赖约束分为路径依赖(path dependencies,简记为 PD-约束)、局部依赖(local dependencies,简记为 LD-约束)和全局依赖(global dependencies,简记为 GD-约束).路径依赖通过水平路径来表示实体对象间的依赖关系.局部依赖和全局依赖通过垂直路径来表示对象内部的依赖关系.下面给出这些依赖约束的定义.

定义 1.7(PD-约束). 对象 X 和 Y 沿路径 ρ 的路径依赖成立,当且仅当对于 X 的每一个实例,通过 ρ 的所有以 X 为起点的路径实例得到的 Y 的实例都相同.记为 $X|^{\rho}\rightarrow Y$.

形式化定义为 $X|^{\rho}\rightarrow Y\Leftrightarrow\forall x_0,\forall\rho_1,\rho_2$,有 $x_{0\rho 1}[Y]=x_{0\rho 2}[Y]$,其中,$\rho$ 为 X 和 Y 间的一条路径,x_0 为 X 的一个实例,ρ_1 和 ρ_2 为 ρ 的任意两条以 x_0 为起点的路径实例.

如果规定每个人开的车都必须在同一个保险公司上保险,那么,如图 1 所示,沿着路径 Person-Car-Insurance-Insurance-Co 有 $\text{Person}|^{\text{Person-Car-Insurance-Insurance-Co}}\rightarrow$Insurance-Co 这样一个 PD-约束.

定义 1.8(LD-约束). 对象 Y 局部依赖于对象 X,当且仅当对于 X 的每一个实例,通过 X 和 Y 间的任何路径得到的 Y 的实例都相同.记为 $X+\rightarrow Y$.

形式化定义为 $X+\rightarrow Y\Leftrightarrow\forall x_0,\forall\rho_1,\forall\rho_2,\forall\tau_1,\forall\tau_2$ 有 $x_{0\tau 1}[Y]=x_{0\tau 2}[Y]$,其中,$\rho_1,\rho_2$ 为 X 和 Y 间的任意两条路径,x_0 为 X 的一个实例,τ_1 和 τ_2 分别为 ρ_1,ρ_2 的任意两条以 x_0 为起点的路径实例.

如果规定实验室里一个项目只能有一组报告,那么,(Laboratory,Project)+→{report}就是图 1 中的一个 LD-约束.

定义 1.9(GD-约束). 对象 Y 全局依赖于对象 X(简称为 Y 依赖于 X),当且仅当,Y 局部依赖于对象 X,且对于 X 的任何两个相等的(指广义上的相等,包括普通属性相等和标识符属性(OID)相等,用"=="表示)实例,其对应的 Y 的实例相同.记为 $X\rightarrow Y$.

形式化定义为 $X\rightarrow Y\Leftrightarrow X+\rightarrow Y$ 且 $\forall x_0,x_1,\forall\rho_1,\forall\rho_2,\forall\tau_1,\forall\tau_2$ 有 $x_{0\tau 1}[Y]=x_{0\tau 2}[Y]$,其中,$\rho_1,\rho_2$ 为 X 和 Y 间的任意两条路径,$x_0\in\Psi[X],x_1\in\Psi[X]$ 且 $x_0==x_1$,τ_1 和 τ_2 分别为 ρ_1,ρ_2 的任意两条以 x_0,x_1 为起点的路径实例.

Ψ 由数据库的全部实例构成,当 X 为一个实体对象时,$\Psi[X]=\{\tau|\tau\in\Psi,\tau$ 为 X 的实例$\}$,可见 $\Psi[X]\subseteq\Psi$;当 X 为一个属性对象时,$\Psi[X]=\{y|E$ 为 X 所属的实体对象,对于 $\forall\tau,\tau$ 为 E 的实例,$y\in\tau path_X(E)[X]\}$,$path_X(E)$ 为从 E 到 X 的最短路径.

如果规定实验室里一个项目只能有一组报告,不同的项目有不同的报告,实验室编号 lab#为对象(Laboratory,Project)的标识符属性(OID),那么,在图 1 中,LD-约束(Laboratory,Project)+→{report}成立,但 GD-约束(Laboratory,Project)→{report}不成立.在图 1 中,GD-约束 Staff→{address}成立,这是因为 Staff 对象实例的标识符都不相同.

定义 1.10(关键字约束). K 为对象 O 的一个候选关键字,当且仅当:

(1) K 是对象 O 的成员集合;

(2) $\exists S_1,\exists S_2$ 使得

(a) S_1,S_2 和 K 构成 O 的一个划分;

(b) $\forall O_i,O_i\in S_1,K\rightarrow O_i$;

(c) $\forall O_j,O_j\in S_2,\exists K_{A_1},K_{A_2},...,\exists K_{A_r}$,其中,$K_{A_i}$ 是 A_i 的一个关键字,A_i 是 O 的一个祖先($1\leq i<r$),A_r 是 O 的双亲,

$K \cup K_{A_1}...K_{A_r} \rightarrow O_j$.

(3) K 的任何真子集 K_1 都不满足条件(1)和条件(2);

关键字约束为对象的复杂属性提供了一种标识符.因此,在面向对象数据模型的规范化理论中,无论是实体对象还是属性对象,都有标识符的概念.

在图 1 中,Laboratory 的成员集是{lab#,l-name,Project,address}.因为有全局依赖 lab#→l-name,(Staff,lab#)→{Project},(Staff,lab#)→{address},所以,lab#为 Laboratory 的一个关键字.

定义 1.11(一致依赖). 依赖 $A \rightarrow B$ 是一致的(coherent),当且仅当,A-B 是一条路径;否则,称其为不一致的(incoherent)依赖.

在图 1 中,(Staff,lab#)→{Project},(Staff,lab#)→{address}是一致的,(Staff,Project)→{report}是不一致的.

1.3 对象范式

定义 1.12(对象解释). 对象解释(object interpretation)是对象的一致 GD-约束集.称一个对象解释是最大的,当且仅当它不能被其他任何对象解释推导出来.

定义 1.13(对象模型). 令 $I_1,I_2,...,I_n$ 为对象 O 的所有最大对象解释,称 $I_1 \vee I_2 \vee ... \vee I_n$ 为对象 O 的模型(model).把用户关于对象 O 的解释记为 Σ^O_{user},O 的模型记为 Σ_O.

定义 1.14(对象范式). 对象 E 是一个对象范式(object normal form)当且仅当:

- E 的每一个复杂成员都有一个标识符;
- $\Sigma_E \vdash \Sigma^E_{\text{user}}$.

下面是文献[1]给出的对象范式判定定理,它是对象规范化的基础.

定理 1.1. 对象 E 是一个对象范式,当且仅当:

(1) 如果 $A \rightarrow B$ 是一个用户依赖(即属于 Σ^E_{user}),那么,以下条件之一成立:

(a) A-B 是 E 的一条路径;

(b) $O_1,...,O_n$ 是 E 的成员,O_{i+1} 是 $O_i(1 \le i \le n-1)$的直接成员,B 是 O_n 的一个直接单值成员,A 是 $O_1,...,O_n$ 的关键字的集合;

(c) 对象 O 的基数是 1:n,A 是 O 的直接成员,B 是 O 的直接单值成员.

(2) 如果 $A \rightarrow \{B\}$是一个用户依赖(即属于 Σ^E_{user}),那么,以下条件之一成立:

(a) A-B 是 E 的一条路径;

(b) $O_1,...,O_n$ 是 E 的成员,O_{i+1} 是 $O_i(1 \le i \le n-1)$的直接成员,B 是 O_n 的一个直接多值成员,A 是 $O_1,...,O_n$ 的关键字的集合;

(c) 对象 O 的基数是 1:n,A 是 O 的直接成员,B 是 O 的直接多值成员.

由定理 1.1 可知,对于一个用户解释,它的每一个 GD-约束在对象范式中都一定是一致的,即对象范式完全体现出了用户所需的存取路径.反之,若一个对象不是对象范式,则用户解释中必有 GD-约束在对象中是不一致的.这说明该对象并没有完全体现出用户所需的存取路径,因此,需要对其进行重构,直到完全体现出用户所需的存取路径为止,这就是规范化的目的.

2 Tari 等人的对象范式生成算法

在文献[1]中,Tari 等人提出了两种不同的对象范式生成算法.第 1 种称为调整算法,它把现存的面向对象模式当作一个初始设计,然后,不断地对它进行调整直到正确为止,即把它变成了一个对象范式.第 2 种称为重建算法,它不把现存的面向对象模式作为主要信息来源,仅当作一个“起点”.其原理是把现存的对象模式中的所有属性都放在同一层上,然后,从用户解释开始,利用簇的关键字来确定属性间的层次关系.这是一个根据用户解释来构造对象结构的过程,所得到对象的对象模型(object model)一定能够推导出用户解释,即它能够生成一个对象范式.在具体介绍这两个算法之前,先给出基于定理 1.1 的两个规范化规则.它们是 Tari 等人对象范式生成算法的基础.

2.1 规范化规则

规则 1. 给出对象 E,如果存在一个不一致的依赖 $A\rightarrow B$(或 $A\rightarrow\{B\}$)以及包含 A 和 B 的路径,则路径 $path_B(A)$ 被拆分成两条子路径:ρ_1 和 ρ_2,ρ_1 包含仅对象 A 和 B,ρ_2 包含对象 A 和($path_B(A)$–B)的对象.

规则 2. 给定对象 E,如果 E 的所有利用复杂对象的依赖是一致的,此外,如果存在一个 GD-约束 $A\rightarrow B$,它不能被 Σ_E 推导出来,则 E 是通过创建一个新对象 O_{new} 来转化的.新对象有如下特征:

• O_{new} 是 E 的直接成员,基数为($\min(k_{O_1},...,k_{O_n})$,$\min(K_{O_1},...,K_{O_n})$),其中,$A=O_1...O_n$ 且(k_{O_j},K_{O_j})是属性 O_j 的最小基数和最大基数.属性 O_j 变成 O_{new}($1\leq j\leq n$)的一个直接单值成员;

• B 是 O_{new} 的直接成员;

• A 是 O_{new} 的一个关键字.

规则 1 说明了如何把存在于同一条路径上的不一致依赖转变为一致的依赖,规则 2 说明了如何把存在于不同路径上的不一致依赖转变为一致的依赖.

2.2 Tari等人的对象范式生成算法

2.2.1 调整算法

2.2.1.1 算法描述

算法 2.1. 调整算法.

输入:对象 E 和关于对象 E 的用户解释 Σ^E_{user};

输出:对象范式.

(1) for each $A\rightarrow B$(或 $A\rightarrow\{B\}$)$\in\Sigma^E_{user}$ do

if A-B 不是一条路径,但 A 和 B 属于同一路径时

then 在对象 E 上应用规则 1 来表示 A-B

else 在对象 E 上应用规则 2 来进行调整;

(2) return (E).

2.2.1.2 算法分析

该算法的时间复杂度为 $O((m-1)!p)$,其中,m 为模式中路径的最大长度,p 为对象可能包含的垂直路径的最大数目.假设对象 E 包含 n 个成员,这 n 个成员构成一条垂直路径,且该路径也是模式中惟一一条路径.它的路径长度应为(n+1).在这种情况下,该算法的时间复杂度变为 $O(n!)$.我们知道,时间复杂度为阶乘级的算法不是一个实用的算法.

除了不实用以外,该算法还存在不完善之处.它没有检查用户解释 Σ^E_{user} 是否含有冲突的机制,当用户解释含有冲突时,算法 2.1 将进入死循环.例如,假设 $\Sigma^E_{user}=\{E\rightarrow A,A\rightarrow B,B\rightarrow C,C\rightarrow A\}$.从语义上讲,该用户解释是有冲突的,因为 A 不能既做 B 的祖先,又做 B 的直接成员.无论初始模式是什么样的,算法 2.1 都无法把它调整为一个对象范式且会无休止地调整下去.

2.2.2 重建算法

2.2.2.1 算法主要思想的描述

重建算法包括 3 个子算法,与这些子算法对应的有一些预备知识.由于篇幅所限,不能对它们一一加以介绍,这里仅对其主要思想进行描述.

第 1 个子算法先通过移去任何嵌套来使对象 E 的所有成员都在同一层上,然后,从与每个成员相关的用户 GD-约束中标识成员可能的分簇.这一工作是由簇关键字来实现的.该算法的输出是若干个不可分解的簇.

第 2 个子算法通过利用标识簇的关键字及所有和簇对象相关的 GD-约束,为每个不可分解的簇建立了树结构.该算法的输出是若干棵树.

第 3 个子算法根据这些树的共同结点和边,对它们进行合并,最后的合并树就是对象范式.

2.2.2.2 算法分析

假设 s 是对象 E 所包含的全部成员的个数,r 是 Σ^E_{user} 中所有 GD-约束的个数.第 1 个子算法的时间复杂度

为 $O(s)$,因为对象 E 最多有 s 个簇(每个簇仅含一个成员).第 2 个子算法的时间复杂度为 $O(s\times r)$,因为一棵树最多有 s 个结点和 r 条边.第 3 个子算法的时间复杂度为 $O(s)$,因为最多要合并 s 棵树.因此,整个重建算法的时间复杂度为 $O(s\times r)$.

虽然比调整算法在时间上有了很大的改进,但重建算法还是存在一些不足.第 1,簇关键字是该算法的主要工具,它是在 GD-约束集的最小覆盖基础上求出来的,但 Tari 等人仅给出了最小覆盖的定义,并没有给出求解方法,因此,影响了其实用性;第 2,Tari 等人所提出的检测用户解释含有冲突的策略是看算法的执行时间是否超过了时间复杂度 $O(s\times r)$,显然,这是不可行的;第 3,对于有些含有冲突的用户解释,算法是可以在 $O(s\times r)$时间内结束并输出一个结果,但这个结果是不正确的,因此,该算法是不完善的.例如,对于前面含有冲突的用户解释 $\Sigma^E_{\text{user}}=\{E\rightarrow A,A\rightarrow B,B\rightarrow C,C\rightarrow A\}$.第 1 个子算法可以正常结束,其结果是$\{E,A\},\{A,B\},\{B,C\},\{C,A\}$这样 4 个簇.第 2 个子算法可以正常结束,其结果是 *E-A*,*A-B*,*B-C*,*C-A* 这样 4 棵仅含有 1 个根和 1 个孩子结点的树.第 3 个子算法也可以正常结束,其结果是 *E-A-B-C-A*,这是一个带有回路 *A-B-C-A* 的图,并不是一棵树,因此,该结果不正确.

3 基于有向图的对象范式生成算法

我们将借鉴超图[7]的思想对有向图进行扩充,并称其为扩充有向图.与超图相反,扩充有向图中顶点的含义被扩充了,它允许是一个有向图,弧的含义不变.下面给出扩充有向图的形式化定义.

定义 3.1(扩充有向图). 设 $V=\{v_1,v_2,...,v_n\}$为一个有限集合.$G_e=(V_e,E)$是集合 V 上的一个扩充有向图,其中,V_e 是扩充顶点集,V_e 中的元素或者是集合 V 中的元素,或者是集合 V 上的一个扩充有向图.E 是笛卡尔积 $V_e\times V_e$ 的子集,E 中的元素称为弧.若 E 中存在弧$(v_{e1},v_{e2}),(v_{e2},v_{e3}),...,(v_{em-2},v_{em-1}),(v_{em-1},v_{em})$,则称 v_{e1} 到 v_{em} 间存在一条路径,表示为$(v_{e1},v_{e2},v_{e3},...,v_{em-2},v_{em-1},v_{em})$.若路径中的每一个顶点都是 V 中元素,则称该路径为普通路径.

例 3.1:设一个顶点集为 $N=\{A,B,C,D,E,F,H\}$,图 2 给出了 N 上的一个扩充有向图.其中,普通路径有(A,B)及(D,E,F).

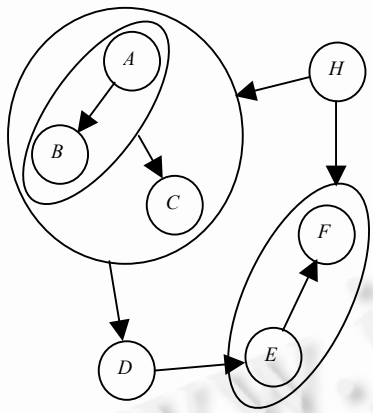

Fig.2 An extended directed graph on N
图 2 N 上的一个扩充有向图

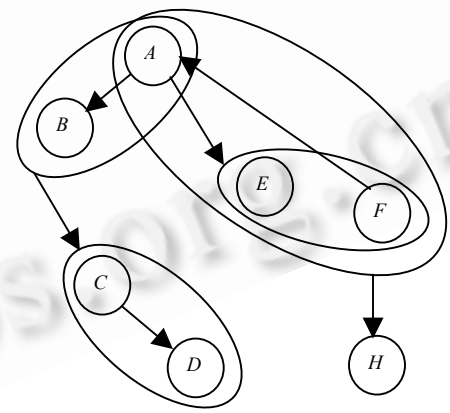

Fig.3 The GD-constraint graph corresponding to G
图 3 G 对应的 GD-约束图

定义 3.2(GD-约束图). 设 E 为一个实体对象,U 为 E 的成员集,G 为对象 E 上的 GD-约束集.若把 G 中每一个 GD-约束的左部和右部分别作为扩充有向图中的顶点,并称为左部顶点和右部顶点,然后,用由左部顶点指向右部顶点的弧将它们连接起来,就构成了一个扩充有向图,称这个扩充有向图为 G 对应的 GD-约束图,称 U 中元素对应的顶点为对象顶点,称 E 对应的顶点为实体对象顶点.

例 3.2:设 GD-约束集为 $G=\{AB\rightarrow CD,AEF\rightarrow H,C\rightarrow D,A\rightarrow B,A\rightarrow EF,F\rightarrow A\}$.图 3 给出了其对应的 GD-约束图.

定义 3.3(对象结构图). 设 O 为一个对象,U 为 O 的成员构成的集合,$G(V,E)$为一个有向图.若 $V=\{v_i:v_i\in U\cup\{O\}\}$,$E=\{(v_i,v_j):v_i,v_j\in V\wedge v_j$ 为 v_i 的直接成员$\}$,则称 $G(V,E)$为对象 O 的结构图.若 O 为一个对象范式,则称 $G(V,E)$为 O 的规范化对象结构图.

算法 3.1. 规范化对象结构图的生成算法.

输入:对象 E 和关于对象 E 的用户解释 Σ^E_{user};

输出:若 Σ^E_{user} 中不含有冲突,则输出一个规范化对象结构图 G';否则,输出∅.

GENERATE NORMALIZED OBJECT STRUCTURE GRAPH (Σ^E_{user})

(1) /* 为 Σ^E_{user} 建立 GD-约束图 G */

令 G 初始为空;

for 每一个 GD-约束 $X\rightarrow Y\in\Sigma^E_{user}$ do

[

在 G 中,为 X 建立一个左部顶点;

为 Y 建立一个右部顶点;

建立一条由 X 指向 Y 的弧

]

(2) /* 对 G 进行变换得到 G' */

令 G'初始为空;

for G 中每一条弧(X,Y) do

[在 G'中建立顶点 X,Y;

if (X 为一条普通路径) and (Y 为一条普通路径)

then

[

在 G'中,建立一条弧,该弧的尾是 X 内普通路径的终点,头是 Y 内普通路径的起点

]

else return (∅) /* Σ^E_{user} 中含有冲突 */

(3) /* 顶点检查 */

for G'中每一个对象顶点 O do

if O 为实体对象顶点且入度不为 0 then return (∅) /* Σ^E_{user} 中含有冲突 */

else

if O 为实体对象顶点且入度为 0 then 把该顶点记为 v_e

else

if O 为成员对象顶点且入度不为 1 then return(∅);

/* Σ^E_{user} 中含有冲突 */

(4) return (G').

算法 3.2. 对象范式生成算法.

输入:对象 E 和关于对象 E 的用户解释 Σ^E_{user};

输出:若 Σ^E_{user} 中不含有冲突,则输出一个对象范式 E';否则,输出∅.

GENERATE OBJECT NORMAL FORM (Σ^E_{user})

(1) 令 E'初始为空;

(2) if GENERATE NORMALIZED OBJECT STRUCTURE GRAPH (Σ^E_{user})≠∅ then

[$E'=v_e$;从 v_e 开始,对于 G'的每一个顶点 v,若存在一条弧(v,v'),则在 E'中把 v'作为 v 的直接成员]

(3) return (E').

定理 3.1. 设 E 为实体对象,Σ^E_{user} 为关于 E 的用户解释.算法 3.1 得到的有向图 G'必为一个规范化对象结构图.

证明:只要证明存在一个对象范式 E',G'为对象 E'的结构图,就说明 G'为一个规范化对象结构图.

这里先由 G'构造一个对象 E'.把 G'中顶点 v_e 作为对象 E',E'的成员由 G'中其他顶点构成.若(v_i,v_j)为 G'中的

一条弧,则在E'中,v_j就作为v_i的直接成员.由算法 3.1 可知,G'的顶点集是由E及其成员构成的,因此,E与E'具有相同的成员,且$E=E'$.这样,可以把Σ^E_{user}看作关于E'的用户解释.根据算法 3.1 可知,Σ^E_{user}中的每一个 GD-约束$A_1A_2...A_n \rightarrow B_1B_2...B_m$,在$G'$中都有普通路径$(A_1,A_2,A_3,...,A_{n-1},A_n,B_1,B_2,B_3,...,B_{m-1},B_m)$.对应于$E'$,必有路径$A_1$-$A_2$-...-$A_n$-$B_1$-$B_2$-...-$B_m$.说明在$E'$中,$A_1A_2...A_n$与$B_1B_2...B_m$构成了一条路径.根据定理 1.1 可知,$E'$为一个对象范式.由定义 3.3 可知,$G'$为一个规范化结构图. □

定理 3.2. 设E为实体对象,Σ^E_{user}为关于E的用户解释.算法 3.2 所生成的对象E'必为E对应的对象范式.

证明:因为$E=E'$且E与E'的成员完全相同,所以,E'是E对应的对象范式. □

定理 3.3. 算法 3.1 是可终止的,它的时间复杂度是$O(n\times m\times \log_2 m)$,其中,$m$为$E$中成员对象的个数,$n$为$\Sigma^E_{user}$中 GD-约束的个数.

证明:(可终止性)在算法 3.1 中,第(1)步含有一个 for 循环语句,其循环次数为$|\Sigma^E_{user}|=n$.循环体内再没有其他循环.因为Σ^E_{user}中 GD-约束的个数是有限的,所以,该 for 循环不是一个死循环.因此,算法 3.1 的第(1)步可以在有限的时间内结束.第(2)步含有一个 for 循环语句,其循环次数是图G中弧的个数.由算法 3.1 可知,图G中弧的个数应为$|\Sigma^E_{user}|=n$.循环体内再没有其他循环.该语句结束有两个条件,一个是循环次数超过n,另一个是Σ^E_{user}中含有冲突.因此,它也是可以在有限的时间内结束的.第(3)步含有一个 for 循环语句,其循环次数是图G中弧的个数.由算法 3.1 可知,图G中弧的个数应为$|\Sigma^E_{user}|=n$.循环体内再没有其他循环.因此,它也是可以在有限的时间内结束的.由此可见,算法 3.1 是可终止的.

在算法 3.1 中,第(1)步的语句频度为$|\Sigma^E_{user}|=n$.对于第(2)步,考虑最坏情况,假设每一对X,Y所含的对象个数都接近E中成员对象的个数m,则其语句频度接近于$(n\times m\times \log_2 m)$.第(3)步的语句频度为$m$.因此,算法 3.1 的时间复杂度为$O(n\times m\times \log_2 m)$. □

定理 3.4. 算法 3.2 是可终止的,它的时间复杂度是$O(n\times m\times \log_2 m)$,其中,$m$为$E$中成员对象的个数,$n$为$\Sigma^E_{user}$中 GD-约束的个数.

证明:(可终止性)在算法 3.2 中,只有第(2)步生成对象范式时含有循环,其循环次数为m.因此,算法 3.2 是可终止的.

在算法 3.2 中,第(2)步的语句频度最高,它与$(n\times m\times \log_2 m+m)$为同一数量级,故算法 3.2 的时间复杂度为$O(n\times m\times \log_2 m)$. □

下面通过一个在文献[1]中出现过的例子来说明本文算法的执行过程.

例 3.3:设E为实体对象,其成员集为$U=\{A,B,C,D,F,G,H,J,K,L,P,Q,R,S,T,U,V,W,Z\}$,关于$E$的用户解释为$\Sigma^E_{user}=\{E\rightarrow B, EB\rightarrow K, B\rightarrow R, E\rightarrow C, EC\rightarrow D, C\rightarrow F, C\rightarrow A, ACE\rightarrow H, AC\rightarrow G, A\rightarrow P, ACG\rightarrow U, ACEG\rightarrow V, GU\rightarrow W, G\rightarrow L, U\rightarrow Z, E\rightarrow JEJ\rightarrow T, J\rightarrow Q, E\rightarrow S\}$.

算法 3.2 在执行完第(1)步以后,调用算法 3.1.因为本例中的Σ^E_{user}不含有冲突,所以,算法 3.1 先后生成了 GD-约束图和规范化对象结构图,然后,算法 3.2 根据算法 3.1 生成的规范化对象结构图构造对象范式.图 4 为算法 3.1 第(1)步生成的 GD-约束图,图 5 为算法 3.1 第(2)步生成的规范化对象结构图,图 6 为算法 3.2 生成的对象范式.它与文献[1]中第 2 种方法的结果完全相同.

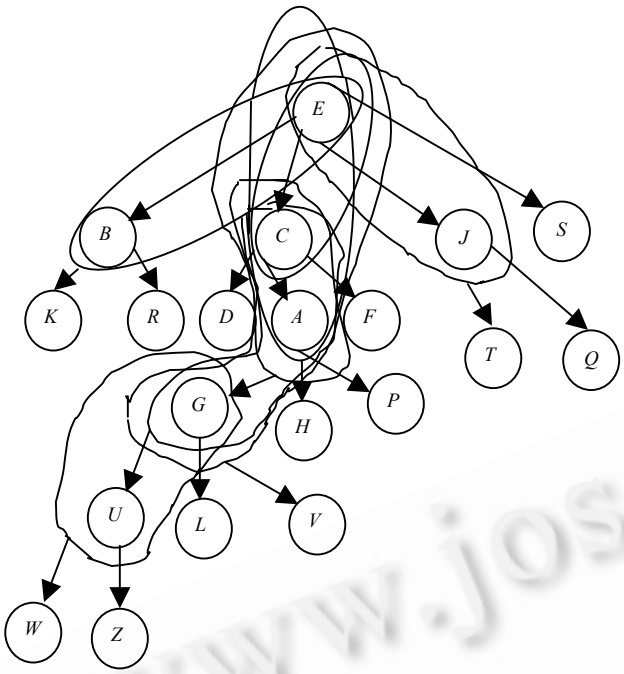

Fig.4 The GD-constraint graph
图 4 GD-约束图

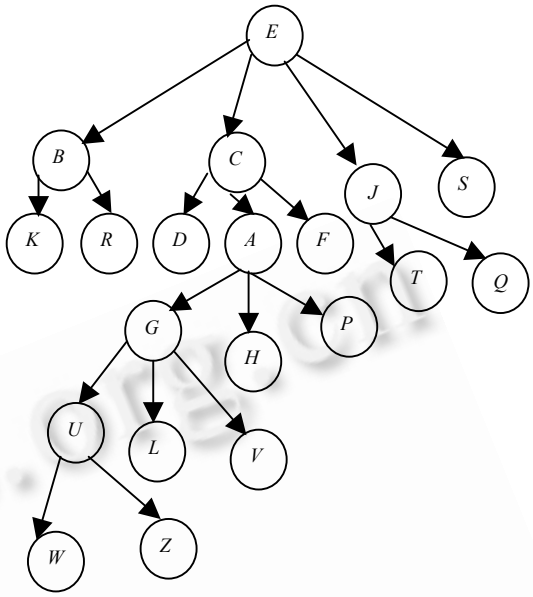

Fig.5 The normalized object structure graph
图 5 规范化对象结构图

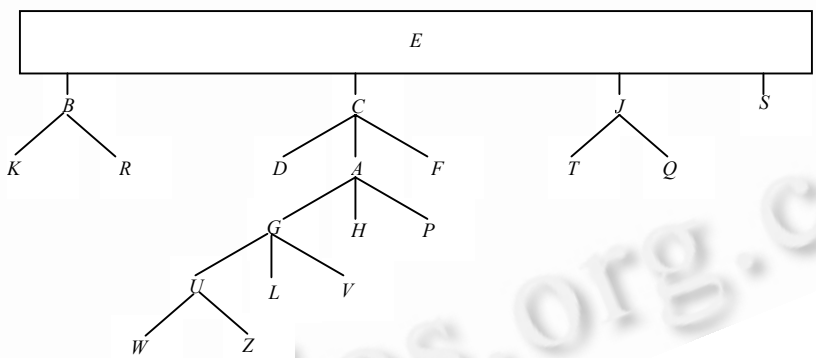

Fig.6 The object normal form *E*
图 6 对象范式 *E*

4 结束语

图论在数据库领域一直是非常有用的工具.在传统关系数据库的设计方法中,无环数据库模式的判定等价于超图中的无回路问题[7].在面向对象数据库的逻辑模型设计中,有向图被用来进行数据库模式分析及正确性检查[8,9].本文的成果为图论在面向对象数据库概念模型设计中的进一步应用奠定了基础.

References:

[1] Tari Z, Stokes J, Spaccapietra S. Object normal forms and dependency constraints for object-oriented schemata. ACM Trans. on Database Systems, 1997,22(4):513~569.

[2] Beeri C, Bernstein PA. Computational problems related to the design of normal form relational schemas. ACM Trans. on Database Systems, 1979,4(1):30~59.

[3] Mok WY. A comparative study of various nested normal forms. IEEE Trans. on Knowledge and Data Engineering, 2002,14(2): 369~385.

[4] Norman RJ. Object-Oriented System Analysis and Design. Upper Saddle River: Prentice-Hall International, Inc., 1996. 55~85.

[5] Liu GH, Chen ZJ, Ji WY, Shi BL, Huang DM. Algorithm for recognizing the user interpretation without conflicts. Chinese Journal of Computers, 2000,23(8):813~818 (in Chinese with English abstract).

[6] Liu GH, Chen ZJ, Sun XM, Shi BL. Research on the features of the user interpretation with conflicts. Journal of Software, 2001,12: 129~134 (in Chinese with English abstract).

[7] Fagin R. Degrees of acyclicity for hypergraphs and relational database schemes. Journal of the Association for Computing Machinery, 1983,30(3):514~550.

[8] Formica A, Groger HD, Missikoff M. Object-Oriented database schema analyse and inheritance processing: A graph-theoretic approach. Data and Knowledge Engineering, 1997,24(2):157~181.

[9] Formica A, Groger HD, Missikoff M. An efficient method for checking object-oriented database schemata correctness. ACM Trans. on Database Systems, 1998,23(3):333~369.

附中文参考文献:

[5] 刘国华,陈子军,季文云,施伯乐,黄冬梅.一种用于识别不含冲突用户解释的算法.计算机学报,2000,23(8):813~818.

[6] 刘国华,陈子军,孙星明,施伯乐.含冲突用户解释特征研究.软件学报,2001,12:129~134.