ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.4, April 2006, pp.759–769
DOI: 10.1360/jos170759

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种 XML 数据库的数据模型*

何震瀛+, 李建中, 王朝坤

(哈尔滨工业大学 计算机科学与技术学院,黑龙江 哈尔滨 150001)

A Data Model for XML Database

HE Zhen-Ying+, LI Jian-Zhong, WANG Chao-Kun

(School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

+ Corresponding author: Phn: +86-451-86415872, E-mail: hzy@hit.edu.cn, http://db.cs.hit.edu.cn

He ZY, Li JZ, Wang CK. A data model for XML database. *Journal of Software*, 2006,17(4):759−769.
http://www.jos.org.cn/1000-9825/17/759.htm

Abstract: Data model is a key research field in the community of XML data management. However, the existed works fall short in their ability to model complex data structure of XML database or to support complete operations. This paper proposes a new mapping-based data model. This model aims to give the precisely defined notations for complex data structure and semantics of XML database. Along with the model, this paper also presents an associated algebra formally which includes a set of path expression operations and data modification operations. This model has already adopted in an XML-based information integration system. It shows that this model can give the precise semantics of XML database, and support the XML database applications effectively.
Key words: XML database; data model; data structure; algebra; path expression

摘 要: 数据模型是 XML 数据管理领域研究的核心问题之一.现有的数据模型在表达 XML 数据库复杂的数据结构和操作方面仍有不足.以映射为基础,提出了一种新的数据模型.该数据模型给出了 XML 数据库复杂的数据结构和语义的精确定义,并提供了数据结构上操作代数的定义,包括路径表达式操作和数据维护操作.该数据模型已应用于一个基于 XML 的信息集成系统中.事实表明,它能够有效地支持 XML 数据管理的应用.
关键词: XML 数据库;数据模型;数据结构;代数;路径表达式
中图法分类号: TP311 文献标识码: A

数据模型是 XML 数据管理研究领域的核心问题之一,用来给出 XML 数据以及数据上操作的精确语义,是 XML 数据的查询处理和优化的基础.

部分研究者利用传统数据模型(如关系模型和面向对象数据模型)来表示 XML 数据的结构和语义.其中,Stanford 的 R.Goldman 等人利用 OEM 模型来表达 XML 数据的结构和语义[1].研究者也开展了大量基于关系模型的 XML 数据管理的研究工作[2−7].这些方法的最大问题是:XML 数据上的一个操作需要用这些模型上的一系列操作来表示,因此显得力不从心.

为此,研究者给出了两个 XML 模型——树模型和图模型.这两种模型都是直观上的模型,不是严格意义上的数据模型.因此,提出一种能够有效表达 XML 数据的模型以及给出该模型上 XML 数据操作的形式化定义变得尤为重要.如何利用数学的方法严格描述 XML 数据以及数据上的操作,进而完成 XML 数据查询的代数优化,

* Supported by the National Natural Science Foundation of China under Grant No.60273082 (国家自然科学基金); the Key Program of the National Natural Science Foundation of China under Grant No.60533110 (国家自然科学基金重点项目)
Received 2005-01-14; Accepted 2005-09-19

成为 XML 数据管理领域一个十分重要的研究问题.

在 XML 数据模型和代数操作方面,现已取得了一些研究成果:

(1) 研究者用 4 种模型对 XML 数据进行描述:关系模型、面向对象模型、树模型和图模型.

面向对象模型可以方便地表达出 XML 数据的结构以及语义,但为了支持路径表达式查询,操作必须是面向过程的,需要复杂的数据导航,并不适应 XML 数据管理的需要;另外,由于 XML 数据半结构化的特点,为存储 XML 数据,将会产生大量磁盘碎片.如果用关系模型或嵌套关系模型来描述 XML 数据的结构和语义,其操作需要用大量昂贵的 join 操作来表示[2],并不适应 XML 数据管理的需要;此外,用关系和面向对象模型表示 XML 数据的结构和语义,将 XML 数据查询操作的语义映射到相应模型的开销也很大.

研究者根据 XML 数据的结构和语义,采用了两种直观的 XML 模型——树模型和图模型.其中树模型得到了广泛的应用,它把一个 XML 文档视为一棵树,把多个 XML 文档看作一个森林.因为树模型可以很容易地表达出 XML 数据的层次结构,在 XML 数据管理的研究中,人们自然地把它视为 XML 的直观数据模型.但它忽略了 XML 数据的一些特性,如 XML 数据的模式信息以及元素结点之间的引用关系.图模型把 XML 数据视为一个有向图.直观上,图模型可以表达单个 XML 文档的语义.但是,目前还没有研究给出图模型的精确定义.树模型和图模型都是对 XML 文档进行建模的,对于具有相同模式而具体数据不同的 XML 文档,这两种模型在表达上都认为它们是不同的,这并不适合数据管理的需要.为此,M.Arenas 等人利用正则表达式给出了 XML 模式和数据的精确定义[8].但该研究并未进一步给出 XML 数据上操作的定义.

(2) 在 XML 代数操作的研究方面,研究者提出了一些 XML 的代数操作.

在 Christophides 以及 Ludascher 等人关于中间件的研究中,给出了一些适合中间件数据交换的 XML 操作代数[9,10].这些研究的最主要特点是代数操作的输入和输出都是元组.这些研究是早期的 XML 数据操作代数.在 W3C 标准中给出 XML 路径表达式的查询语言 XQuery[11,12],它的输入和输出都是 XML 文档元素的列表.此外,研究者还提出了其他一些 XML 代数:Buneman 等人提出的一种半结构化数据的查询语言和代数[13];Beeri 等人提出的 SAL[14];Jagadish 等人提出的 TAX[15];Fernandez 等人于 2001 年也给出相应的代数操作[16];孟小峰等人提出了 OreintXA[17].但这些研究并没有给出 XML 代数操作的精确语义.

由于已有的 XML 模型在以下几方面各有不足之处:(1) 无法表达 XML 数据的复杂语义;(2) 没有给出完整的代数操作的定义;(3) 在模型上没有给出数据修改操作的明确定义.因此,本文对 XML 数据模型进行探讨,并提出一种 XML 数据库的数据模型.本文第 1 节给出本文使用的数据用例.第 2 节介绍模型的数据结构.第 3 节给出模型的操作代数.第 4 节对本文提出的模型以及相关研究工作进行对比和分析.最后,第 5 节总结全文.

1 数据用例

本文修改了一个现实使用的 XML 文档片段作为数据用例,用以说明数据模型和相关代数操作,如图 1 所示.

```
⟨MovieDB⟩
  ⟨actor id="a1"⟩
        ⟨name⟩ actor1 ⟨/name⟩
        ⟨age⟩39⟨/age⟩⟨/actor⟩
  ⟨actor id="a2",movie="m1"⟩
      ⟨name⟩ actor2 ⟨/name⟩⟨/actor⟩
  ⟨director id="d1"⟩
        ⟨name⟩ director1 ⟨/name⟩
        ⟨movie id="m1",director="d1"⟩
            ⟨title⟩movie2⟨/title⟩⟨/movie⟩⟨/director⟩
  ⟨director id="d2"⟩
        ⟨name⟩director2⟨/name⟩⟨/director⟩
  ⟨movie id="m2",actor="a1",director="d2"⟩
      ⟨title⟩movie1⟨/title⟩⟨/movie⟩
⟨MovieDB⟩
```

(a)

```
⟨MovieDB⟩
  ⟨actor id="a1"⟩
        ⟨name⟩actor1⟨/name⟩
        ⟨age⟩39⟨/age⟩⟨/actor⟩
  ⟨actor id="a3",movie="m1"⟩
      ⟨name⟩actor3⟨/name⟩⟨/actor⟩
  ⟨director id="d1"⟩
        ⟨name⟩director1⟨/name⟩
        ⟨movie id="m1",director="d1"⟩
            ⟨title⟩movie2⟨/title⟩⟨/movie⟩⟨/director⟩
  ⟨director id="d2"⟩
        ⟨name⟩director2⟨/name⟩⟨/director⟩
  ⟨movie id="m2",actor="a1",director="d2"⟩
      ⟨title⟩movie1⟨/title⟩⟨/movie⟩
⟨MovieDB⟩
```

(b)

Fig.1 The sample XML data fragment

图 1 XML 数据片段用例

图 1 给出的两个数据用例是用于发布 movie 信息的文档片段.在该例中,包含了一些 actors,directors 以及 movies 的信息.在这两个 XML 文档片段中,在第 5 行和第 6 行存在着一些差异,所给出的 actor 信息是不同的.

2 数据结构

词是由字母表中一系列字母构成的串,短语由若干个词构成.词分为字符型和数字型.标签是一个短语.

定义 **2.1**. 如果Φ是一个标签,Ψ是一个字符型或者数字型数据,那么$\langle\Phi\rangle\ \Psi\langle/\Phi\rangle$是一个原子特征,$\Phi$是一个原子元素.

定义 **2.2**. 一个复合特征递归定义如下:(1) 如果$\langle\Phi\rangle\ \Psi\langle/\Phi\rangle$是一个原子特征,则它是一个复合特征,或者(2) 如果Φ是一个标签,且$\delta_1,\ldots,\delta_n$是一系列的复合特征,则$\langle\Phi\rangle\delta_1,\ldots,\delta_n\langle/\Phi\rangle$是一个复合特征.其中,$\Phi$被称为一个复合元素.

原子特征和复合特征统称为特征;而原子元素和复合元素统称为元素.对于图 1(a)中给出的数据,*name* 是一个原子元素,*actor* 是复合元素.元素 *age* 的值域是数字型,而 *name* 元素的值域是字符串型.

给定一个复合体$\langle S\rangle\delta_1,\ldots,\delta_n\langle/S\rangle$,其中$\delta_1=\langle T\rangle\ \delta_1',\ldots,\delta_n'\ \langle/T\rangle$,我们称元素 S 和 T 之间有父子关系,记为 $T\leq S$ 或者 $S\geq T$.假设 S 和 T 是两个元素,S 和 T 之间有祖先-后代关系(记为 $T<S$ 或者 $S>T$)递归定义为:(1) 若 $S\geq T$,则 $T<S$;(2) 若存在元素 R,使得 $R<S$ 且 $T<R$,则 $T<S$.

在图 1(a)中,元素 *MovieDB* 和 *movie* 之间有父子关系,而元素 *MovieDB* 和 *name* 之间有祖先-后代关系.

定义 **2.3**. 设$\alpha=\{S_1,S_2,\ldots,S_n\}$是一个有限元素集合,且$\exists S\in\alpha$,使得$\forall R\in\alpha-\{S\}$,都有 $R<S$,则α为一个 S-有限元素集合,记为(α,S);S 称为α的起始结点.

定义 **2.4**. 设$\alpha=\{E_1,E_2,\ldots,E_n\}$是一个有限元素集合,$S$是$\alpha$中的奇异点,若$\forall R\in\alpha(R\neq S)$,则$R<S$或$S<R$都不满足.

性质 **1**. S-有限元素集合α中没有奇异点.

现在我们以图 1(a)为例来说明定义 2.3 和定义 2.4 中 S-有限元素集合的实际意义.在图 1(a)中共有 7 个元素,构成了如下元素集合:$E=\{MovieDB,actor,name,age,director,movie,title\}$.*MovieDB* 是这个有限元素集合的起始结点,因为$\forall R\in E(R\neq MovieDB)$,都有 $R<MovieDB$.

定义 **2.5**. 设$\alpha=\{E_1',E_2',...,E_n'\}$是一个 S_1-有限元素集合,$S_2\in\alpha$是α的终结结点,如果 S_2 可以是原子元素.$(S_1,S_2,\ldots,S_m)\subseteq\alpha$是$\alpha$的层次链,记作 $S_m\leq S_{m-1}\leq\ldots\leq S_1$ 或者$(S_1,S_2,\ldots,S_m)$,如果 S_1 是α的起始结点,S_m 是α的一个终结结点,且$\forall S_i,S_{i+1}\in\{S_1,S_2,\ldots,S_m\}$,$1\leq i\leq m-1$,满足 $S_{i+1}\leq S_i$.

定义 **2.6**. 设 $H=(S_1,S_2,\ldots,S_m)$是 S_1-有限元素集合α上的一个层次链.$p=(S_1',S_2',...,S_k')$被称为α上的一条路径,如果 p 是 H 的一个子序列,且 $lab(S_1')=lab(S_1)$. S_k' 是路径 p 的目标结点.序偶(S_i',S_{i+1}')是路径 p 的一步,如果 $S_i'\leq S_{i+1}'$,其中 $1\leq i\leq k-1$,记为 S_i'/S_{i+1}';(S_i',S_{i+1}')称为路径 p 上的若干步,如果 $S_{i+1}'<S_i'$,其中 $1\leq i\leq k-1$,记为 $S_i'//S_{i+1}'$.

设 $E=\{MovieDB,actor,name,age,director,movie,title\}$ 是图 1(a)中所有元素组成的有限元素集合,$H_1=(MovieDB,actor,name)$和 $H_2=(MovieDB,director,movie,title)$都是 E 上的层次链,对于 H_1 的子序列(*MovieDB*,*actor*)和 H_2 的子序列(*MovieDB*,*director*,*title*)则都被称为路径.

定义 **2.7**. 设 $E=\{E_1,E_2,\ldots,E_n\}$是一个 S-有限元素集合,*Rule* 是从 E 到元素类型定义的一个映射:$\tau\in E$,(1) 若τ为原子元素,$Rule(\tau)$为数字型或字符型数据;否则(2) $Rule(\tau)$为如下定义的正则表达式:$\alpha::=\varepsilon|\tau'\ |\ \alpha|\alpha\ |\ \alpha,\alpha\ |\ \alpha^*$,其中,$\varepsilon$为空,$\tau'\in E$,"|","," 和"*"分别表达或、顺序和 Kleene 闭包.τ称为规则的原象,记为 $af(\tau)$;$Rule(\tau)$称为τ的象τ,记为 $rf(\tau)$.::=简记为$\rightarrow$.

若元素 S_1 的属性 A_1 的值由元素 S_2 的属性 A_2 的值来决定,我们称属性 S_1/A_1 和 S_2/A_2,之间存在引用关系 $Ref(S_1/A_1,S_2/A_2)$.

定义 **2.8**. XML 数据集合的模式定义为 $X=((E,E_{root}),A,Rule,Ref,EAstr)$,其中:

(1) $E=\{e_1,e_2,\ldots,e_n\}$是一个 E_{root}-有限元素集合,$e_i(1\leq j\leq n)$是 XML 数据集合中的元素;

(2) $A=\{a_1,a_2,\ldots,a_m\}$是属性的集合,$a_j(1\leq j\leq m)$是 XML 数据集合中的属性;

(3) $Rule(E)$是 XML 数据集合规则的集合;

(4) $Ref(E_1/A_1,E_2/A_2)$是 XML 数据集合上的引用关系;

(5) $EAstr$ 是 E 到 A 的幂集的一个映射.

在定义 2.8 中,$\forall(e,\{A_1,A_2,\ldots,A_k\})\in EAstr$,我们称 $A_i(1\le i\le k)$是 e 的后代属性结点,记为 $A_i<e$,$\{A_1,A_2,\ldots,A_k\}$是 e 后代属性结点的集合,记为 $EA_{ext}(e)$.有限元素集合$\{E_1,E_2,\ldots,E_m\}$后代属性结点的集合为$\cup_{1\le j\le m}EA_{ext}(E_j)$.

定义 2.8 给出了一系列 XML 文档的模式信息;而根据相同模式信息写出的 XML 文档都是该模式的具体实例.下面我们给出 XML 数据集合模式实例的形式化定义.XML 数据集合模式的具体实例是一个有序的 XML 文档.假定 NE 是 XML 文档中有限元素结点的集合,我们可以用一个三元组(NE,NE,NE)来记录元素结点之间父子之间以及兄弟之间的序关系.对于$(n_p,n_l,n)\in\{(NE,NE,NE)\}$,它表达的含义是对于结点 n,它的父结点是 n_p,左兄弟是 n_l;如果 n 没有左兄弟,则 $n_l=n_p$.$[n_1,n_2,\ldots,n_k]$是 n_p 所有儿子按从左到右组成的序列,记为 $Extend(n_p)$.$Extend(n_p)\in\{(NE,NE,NE)\}^*$,是正则表达式 $Rule(n_p)$可以推导出来的一个序列,记为 $Extend(n_p)\Leftarrow Rule(n_p)$.

定义 **2.9**. 设 $X=((E,E_{root}),A,Rule,Ref,EAstr)$是一个 XML 数据集合的模式,该模式的实例定义为 $XS=(NE,NA,E_{NE},Ref_{NE},NEAstr,root)$,其中:

(1) NE 是有限元素结点的集合,使得存在一个映射 $lab:NE\to E$;

(2) NA 是有限属性结点的集合,使得存在一个映射 $att:NA\to A$;

(3) $ENE\subseteq\{(NE,NE,NE)\}$,使得对于任何 $Extend(np)\in E_{NE}^*$,都有 $Extend(n_p)\Leftarrow Rule(n_p)$;

(4) $Ref_{NE}\subseteq\{(NE/NA,NE/NA)\}$,使得$\forall(ne_1/na_1,ne_2/na_2)\in Ref_{NE}$,都有$(lab(ne_1)/att(na_1),lab(ne_2)/att(na_2))\in Ref$;

(5) $NEAstr\subseteq\{(NE,NA)\}$,使得$\forall(n_1,n_2)\in NEAstr$,都有 $att(n_2)\in EAstr(lab(n_1))$;

(6) $root\in NE$,使得 $lab(root)\in E_{root}$.

定义 **2.10**. XML 数据库由一系列具有不同模式的 XML 数据集合组成,记为$\langle((E_1,E_{root1}),A_1,Rule_1,Ref_1,EAstr_1),((E_2,E_{root2}),A_2,Rule_2,Ref_2,EAstr_2),\ldots,((E_n,E_{rootn}),A_n,Rule_n,Ref_n,EAstr_n)\rangle$.

例 2.1:设图 1(a)中 XML 数据片段的模式为$((E_1,E_{root1}),A_1,Rule_1,Ref_1,EAstr_1)$,则:

(1) $E_1=\{MovieDB,actor,name,age,director,movie,title\}$;$E_{root1}=MovieDB$;

(2) $A_1=\{id,movie,director,actor\}$;

(3) $Rule_1=\{(MovieDB\to actor^+,director^+,movie^+),(actor\to name,age^?),(director\to name,movie),(movie\to title)\}$;

(4) $Ref_1=\{Ref(acotr/movie,movie/id),Ref(movie/director,director/id),Ref(movie/actor,actor/id)\}$;

(5) $EAstr_1=\{(actor,id),(actor,movie),(director,id),(movie,id),(movie,actor),(movie,director)\}$.

3 代数操作

XML 数据库的操作代数分为 3 类:(1) 集合操作——合并、插入、差以及重构等;(2) 数据查询操作;(3) 数据修改操作——模式修改以及数据修改.

为方便起见,我们分别用 $Sch(P)$和 $Ins(P)$表示代数操作表达式 P 的模式和实例;用函数 $lab(V)$和 $value(V)$分别表示元素结点 V 的标签和值;用函数 $value(A)$表示属性结点 A 的值.

3.1 集合操作

定义 **3.1**(模式同构). 设 $S=((E_S,E_{rootS}),A_S,Rule_S,Ref_S,EAstr_S)$和 $T=((E_T,E_{rootT}),A_T,Rule_T,Ref_T,EAstr_T)$是两个 XML 数据库的模式.$T$ 和 S 同构(记为 $T\cong S$),如果:

(1) $E_T=E_S,A_T=A_S$,且 $E_{rootT}=E_{rootS}$;

(2) 存在一个一一映射 $F_{Rule}:Rule_S\to Rule_T$,使得$\forall F_{Rule}(Rule_S)=Rule_T,lab(af(Rule_S))=lab(af(Rule_T)),rf(Rule_S))=rf(Rule_T)$,且对任何 $rf(Rule_S)$中出现的元素 $E,Rule_S$ 中以 E 为原象的规则为 $Rule_S(E),Rule_T$ 中以 E 为原象的规则为 $Rule_T(E)$,映射 $Rule_S(E)\to Rule_T(E)$都满足;

(3) 存在一个一一映射 $F_{Ref}:Ref_S\to Ref_T$.使得$\forall F_{Rule}(e_{1S}/a_{1S},e_{2S}/a_{2S})=(e_{1T}/a_{1T},e_{2T}/a_{2T})$,满足 $e_{1S}=e_{1T},a_{1S}=a_{1T},e_{2S}=e_{2T},a_{2S}=a_{2T}$;

(4) 存在一个一一映射 $F_{Str}:EAstr_S\to EAstr_T$,使得$\forall F_{Str}(e_S,a_S)=(e_T,a_T)$,满足 $e_S=e_T,a_S=a_T$.

定义 **3.2**(实例相等). 设 $S=((E_S,E_{rootS}),A_S,Rule_S,Ref_S,EAstr_S)$是一个 XML 数据库模式,$XS_1(S)=(NE_1,NA_1,E_{NE_1},Ref_{NE_1},NEAstr_1,root_1)$和 $XS_2(S)=(NE_2,NA_2,E_{NE_2},Ref_{NE_2},NEAstr_2,root_2)$,$XS_1(S)$和 $XS_2(S)$相等,如果:

(1) $|NE_1|=|NE_2|$,且存在一个一一映射 $F_{NE}:NE_1\to NE_2$,使得$\forall F_{NE}(ne_1)=ne_2$,其中$(ne_1\in NE_1,ne_2\in NE_2)$,满足$lab(ne_1)=lab(ne_2)$;同时满足若 ne_1 为原子元素,ne_2 也为原子元素,且 $value(ne_1)=value(ne_2)$;

(2) $|NA_1|=|NA_2|$,且存在一个一一映射 $F_{NA}:NA_1\to NA_2$,使得$\forall F_{NA}(na_1)=na_2$,其中$(na_1\in NA_1,na_2\in NA_2)$,满足$att(na_1)=att(na_2)$;同时满足 $value(na_1)=value(na_2)$;

(3) $|E_{NE_1}|=|E_{NE_2}|$,且存在一个一一映射 F_{ENE}: $E_{NE_1}\to E_{NE_2}$,使得$\forall F_{ENE}(ene_1)=ene_2(ene_1\in E_{NE_1},ene_1\in E_{NE_2})$,其中 $ene_1=(n_{p_1},n_{l_1},n_1)$,$ene_2=(n_{p_2},n_{l_2},n_2)$,满足 $F_{NE}(np_1)=np_2$,$F_{NE}(nl_1)=nl_2$,$F_{NE}(n_1)=n_2$;

(4) $|Ref_{NE_1}|=|Ref_{NE_2}|$,且存在一个一一映射 F_{Ref}: $Ref_{NE_1}\to Ref_{NE_2}$, 使得$\forall F_{Ref}((ne_{11}/att_{11},ne_{21}/att_{21}))=(ne_{12}/att_{12},ne_{22}/att_{22})$,其中$((ne_{11}/att_{11},ne_{21}/att_{21})\in Ref_{NE_1},(ne_{12}/att_{12},ne_{22}/att_{22})\in Ref_{NE_2})$,满足 $F_{NE}(ne_{11})=ne_{12}$,$F_{NE}(ne_{21})=ne_{22}$,且 $F_{NA}(na_{11})=na_{12}$,$F_{NA}(na_{21})=na_{22}$;

(5) $|NEAstr_1|=|NEAstr_2|$,且存在一个一一映射 $F_{NEAstr}:NEAstr_1\to NEAstr_2$,使得$\forall F_{NEAstr}((ne_1,na_1))=(ne_2,na_2)$,其中$((ne_1,na_1)\in NEAstr_1,(ne_2,na_2)\in NEAstr_2)$,满足 $F_{NE}(ne_1)=lab(ne_2)$,$F_{NA}(na_1)=att(na_2)$;

(6) $F_{NE}(root_1)=root_2$.

在下边的定义 3.3~定义 3.7 中,$S=((E_S,E_{rootS}),A_S,Rule_S,Ref_S,EAstr_S)$和 $T=((E_T,E_{rootT}),A_T,Rule_T,Ref_T,EAstr_T)$是两个同构的 XML 数据集合,$Ins(S)=F_S$,$Ins(T)=F_T$.

定义 **3.3**(集合交). T 和 S 的交集 $S\cap T$ 定义为:

(1) $Sch(S\cap T)=((E_{S\cap T},E_{rootS\cap T}),A_{S\cap T},Rule_{S\cap T},Ref_{S\cap T},EAstr_{S\cap T})$,且与 S 和 T 同构;

(2) $Ins(S\cap T)\in\{e|e\in Ins(S)$且 $e\in Ins(T)\}$.

定义 **3.4**(集合并). T 和 S 的并集 $S\cup T$ 定义为:

(1) $Sch(S\cup T)=((E_{S\cup T},E_{rootS\cup T}),A_{S\cup T},Rule_{S\cup T},Ref_{S\cup T},EAstr_{S\cup T})$,且与 S 和 T 同构;

(2) $Ins(S\cup T)\in\{e|e\in Ins(S)$或 $e\in Ins(T)\}$.

定义 **3.5**(集合差). S 和 T 的差集 $S-T$ 定义为:

(1) $Sch(S-T)=((E_{S-T},E_{rootS-T}),A_{S-T},Rule_{S-T},Ref_{S-T},EAstr_{S-T})$,且与 S 和 T 同构;

(2) $Ins(S-T)\in\{e|e\in Ins(S)$且 $e\notin Ins(T)\}$.

定义 **3.6**(卡氏积). S 和 T 的卡氏积 $S\times T$ 是如下定义的二元组 $S\times T=\{(e_1,e_2)|e_1\in Ins(S)$且 $e_2\in Ins(T)\}$.

定义 **3.7**(**Skolem** 函数). S 和 T 的 Skolem 函数 $Skolem(S,T,EL)$定义为:

(1) $Sch(Skolem(S,T,EL))=((E_{Skolem(S,T,EL)},EL),A_{Skolem(S,T,EL)},Rule_{Skolem(S,T,EL)},Ref_{Skolem(S,T,EL)},EAstr_{Skolem(S,T,EL)})=(E_S\cup E_T\cup EL,A_S\cup A_T,Rule_S\cup Rule_T\cup\{EL\to E_{rootS},E_{rootT}\},Ref_S\cup Ref_T,EAstr_S\cup EAstr_T)$;

(2) 对于任何 S 的实例 $XS(S)=(NE_T,NA_T,E_{NET},Ref_{NET},NEAstr_T,ne_T)$以及 T 的实例 $XS(T)=(NE_T,NA_T,E_{NET},Ref_{NET},NEAstr_T,ne_T)$,存在一个一一映射 $F_{Skolem(S,T,EL)}:S\times T\to Ins(Skolem(S,T,EL))$,使得$\forall(ne_S,ne_T)\in S\times T$,有且仅有一个 $F_{Skolem(S,T,EL)}(ne_S,ne_T)=(NE_S\cup NE_T\cup\{ne_p\},NA_S\cup NA_T,E_{NES}\cup E_{NET}\cup(ne_p,ne_S,ne_T),Ref_{NES}\cup Ref_{NET},NEAstr_S\cup NEAstr_T,ne_p)$与之对应,且 $lab(ne_p)=EL$.

定理 **3.1**. 如果 S 和 T 是两个 XML 数据集合,那么下述结论成立:

(1) $S\cap T$ 是一个 XML 数据集合;

(2) $S\cup T$ 是一个 XML 数据集合;

(3) $S-T$ 是一个 XML 数据集合;

(4) $Skolem(S,T)$是一个 XML 数据集合.

定义 **3.8**(重构). $S_1=((E_{S_1},E_{rootS_1}),A_{S_1},Rule_{S_1},Ref_{S_1},EAstr_{S_1})$, $S_2=((E_{S_2},E_{rootS_2}),A_{S_2},Rule_{S_2},Ref_{S_2},EAstr_{S_2})$, …,$S_n=((E_{S_n},E_{rootS_n}),A_{S_n},Rule_{S_n},Ref_{S_n},EAstr_{S_n})$,是给定的 n 个 XML 数据集合,如果 $S=((E_S,E_{rootS}),A_S,Rule_S,Ref_S,EAstr_S)$是由 $S_1,S_2,\ldots,S_n$ 根据定理 3.1 规则得到,则称 S 是 $S_1,S_2,\ldots,S_n$ 的重构.

3.2 查询操作

设$\alpha=\{S_1,S_2,\ldots,S_n\}$和$\beta=\{T_1,T_2,\ldots,T_m\}$是两个有限元素集合,且$\beta\subseteq\alpha$.对于$\alpha$上的规则$S\to L_S$,$\beta$到规则$S\to L_S$的投影是从$S\to L_S$中直观去除$\alpha-\beta$元素而得到的规则.特别地,若$S\notin\beta$,则投影得到的规则为空.对于$\alpha$上的$Ref$关系,$\beta$到$Ref$的投影为$\{(S_1,S_2)|S_1\in\alpha\wedge S_2\in\alpha$,且$S_1\in\beta\wedge S_2\in\beta\}$.若$X=((\alpha,\alpha_{root}),A,Rule,Ref,EAstr)$,$\beta$到$EAstr$的投影为如下定义的一个部分映射$F_{EAstr}:(\alpha,2^A)\to\beta,\forall E\in\beta,\exists(E,2^A)\in EAstr$,使得$F_{EAstr}(E,2^A)=E$.

定义 **3.9**(路径表达式). 设$S=((E_S,E_{rootS}),A_S,Rule_S,Ref_S,EAstr_S)$是一个 XML 数据集合的模式,$p=(S_1,S_2,\ldots,S_k)$是$S$上的一条路径.路径表达式$p$操作$Exp(S,p)$定义如下:

(1) $Sch(Exp(S,p))=((E_{Sp},S_k),A_{Sp},Rule_{Sp},Ref_{Sp},EAstr_{Sp})$.其中,$E_{Sp}=\{e|e\in E_S\wedge e<S_k\}\cup S_k$,$A_{Sp}$是$E_{Sp}$的所有后代属性结点集合,$Rule_{Sp}$是$E_{Sp}$到$Rule_S$的投影,$Ref_{Sp}$是$E_{Sp}$到$Ref_S$的投影,$EAstr_{Sp}$是$E_{Sp}$到$EAstr_S$的投影.

(2) $\{Ins_1(S),Ins_2(S),\ldots,Ins_n(S)\}$是所有$S$实例的集合,$Exp(S,p)$所有实例的集合为$\cup_{1\le i\le n}\{Ins(Exp(Ins_i(S),p))\}$,其中$Ins_i(S)=(NE_i,NA_i,E_{NE_i},Ref_{NE_i},NEAstr_i,root_i)$.$Ins(Exp(Ins_i(S),p))=(NE_p,NA_p,E_{NE_p},Ref_{NE_p},NEAstr_p,root_p)$递归定义如下:

① 如果(S_1,S_2)是路径p的一步,$Ins(Exp(Ins_i(S),(S_1S_2))=(NE_{ij},NA_{ij},E_{NE_{ij}},Ref_{NE_{ij}},NEAstr_{ij},root_{ij})$,其中:

I. $NE_{ij}\subseteq NE_i$,且存在一个部分映射$F_{np12}:NE_i\to NE_{ij},\exists N\in NE_i\wedge N\le root_i\wedge lab(N)=S_2$,使得$\forall F_{np12}(N_{d_1})=N_{d_2}$,其中$(N_{d_1}\in NE_i,N_{d_2}\in NE_{ij})$,满足$lab(N_{d_1})=lab(N_{d_2})$;同时满足:若$N_{d_1}$为原子元素,$N_{d_2}$也为原子元素,且$value(N_{d_1})=value(N_{d_2})$;

II. $NA_{ij}\subseteq NA_i$,且存在一个部分映射$F_{ap_{12}}:NA_i\to NA_{ij}$,使得$\forall F_{ap_{12}}(na_1)=na_2$,其中$(na_1\in NA_i,na_{ij}\in NA_2)$,满足$att(na_1)=att(na_2),value(na_1)=value(na_2)$;

III. $E_{NE_{ij}}\subseteq E_{NE_i}$,且存在一个部分映射$F_{ENEp_{12}}:E_{NE_i}\to E_{NE_{ij}}$,使得$\forall F_{ENEp_{12}}(ene_1)=ene_2(ene_1\in E_{NE_i},ene_1\in E_{NE_{ij}})$,其中$ene_1=(n_{p_1},n_{l_1},n_1),ene_2=(n_{p_2},n_{l_2},n_2)$,满足$F_{np_{12}}(np_1)=np_2,F_{np_{12}}(n_{l_1})=nl_2,F_{np_{12}}(n_1)=n_2$;

IV. $Ref_{NE_{ij}}\subseteq Ref_{NE_i}$,且存在一个部分映射$F_{Refp_{12}}:Ref_{NE_i}\to Ref_{NE_{ij}}$,使得$\forall F_{Refp_{12}}((ne_{11},ne_{21}))=(ne_{12},ne_{22})$,其中$((ne_{11},ne_{21})\in Ref_{NE_i},(ne_{12},ne_{22})\in Ref_{NE_{ij}})$,满足$F_{np_{12}}(ne_{11})=ne_{12},F_{np_{12}}(ne_{21})=ne_{22}$;

V. $NEAstr_{ij}\subseteq NEAstr_i$,且存在一个部分映射$F_{nep_{12}}:NEAstr_i\to NEAstr_{ij}$,使得$\forall F_{nep_{12}}((ne_1,na_1))=(ne_2,na_2)$,其中$((ne_1,na_1)\in NEAstr_i,(ne_2,na_2)\in NEAstr_{ij})$,满足$F_{np_{12}}(ne_1)=ne_2,F_{ap_{12}}(na_1)=na_2$;

VI. $root_{ij}=N$.

如果(S_1,S_2)是路径p的若干步,$Ins(Exp(Ins_i(S),(S_1S_2))=(NE_{ij},NA_{ij},E_{NE_{ij}},Ref_{NE_{ij}},NEAstr_{ij},root_{ij})$,其中:

I. $NE_{ij}\subseteq NE_i$,且存在一个部分映射$F_{np_{12}}:NE_i\to NE_{ij},\exists N\in NE_i\wedge N<root_i\wedge lab(N)=S_2$,使得$\forall F_{np_{12}}(N_{d_1})=N_{d_2}$,其中$(N_{d_1}\in NE_i,N_{d_2}\in NE_{ij})$,满足$lab(N_{d_1})=lab(N_{d_2})$;同时满足若$N_{d_1}$为原子元素,$N_{d_2}$也为原子元素,且$value(N_{d_1})=value(N_{d_2})$;

II. $NA_{ij}\subseteq NA_i$,且存在一个部分映射$F_{ap_{12}}:NA_i\to NA_{ij}$,使得$\forall F_{ap_{12}}(na_1)=na_2$,其中$(na_1\in NA_i,na_{ij}\in NA_2)$,满足$att(na_1)=att(na_2),value(na_1)=value(na_2)$;

III. $E_{NE_{ij}}\subseteq E_{NE_i}$,且存在一个部分映射$F_{ENEp_{12}}:E_{NE_i}\to E_{NE_{ij}}$,使得$\forall F_{ENEp_{12}}(ene_1)=ene_2(ene_1\in E_{NE_i},ene_1\in E_{NE_{ij}})$,其中$ene_1=(n_{p_1},n_{l_1},n_1),ene_2=(n_{p_2},n_{l_2},n_2)$,满足$F_{np_{12}}(np_1)=np_2,F_{np_{12}}(n_{l_1})=nl_2,F_{np_{12}}(n_1)=n_2$;

IV. $Ref_{NE_{ij}}\subseteq Ref_{NE_i}$,且存在一个部分映射$F_{Refp_{12}}:Ref_{NE_i}\to Ref_{NE_{ij}}$,使得$\forall F_{Refp12}((ne_{11},ne_{21}))=(ne_{12},ne_{22})$,其中$((ne_{11},ne_{21})\in Ref_{NE_i},(ne_{12},ne_{22})\in Ref_{NE_{ij}})$,满足$F_{np_{12}}(ne_{11})=ne_{12},F_{np_{12}}(ne_{21})=ne_{22}$;

V. $NEAstr_{ij}\subseteq NEAstr_i$,且存在一个部分映射$F_{nep_{12}}:NEAstr_i\to NEAstr_{ij}$,使得$\forall F_{nep_{12}}((ne_1,na_1))=(ne_2,na_2)$,其中$((ne_1,na_1)\in NEAstr_i,(ne_2,na_2)\in NEAstr_{ij})$,满足$F_{np_{12}}(ne_1)=ne_2,F_{ap_{12}}(na_1)=na_2$;

VI. $root_{ij}=N$.

② $Ins(Exp(Ins_i(S),(S_1S_2\ldots S_k))=(NE_k,NA_k,E_{NE_k},Ref_{NE_k},NEAstr_k,root_k)$,如果$(S_k,S_{k+1})$是路径$p$的一步,$Ins(Exp(Ins_i(S),(S_1S_{2\ldots}S_kS_{k+1}))=(NE_{k+1},NA_{k+1},E_{NE_{k+1}},Ref_{NE_{k+1}},NEAstr_{k+1},root_{k+1})$定义为:

I. $NE_{k+1}\subseteq NE_k$,且存在一个部分映射 $F_{nep_{12}}$:$NE_k\to NE_{k+1}$,$\exists N\in NE_k\wedge N\le root_k\wedge lab(N)=S_2$,使得$\forall$ $F_{nep_{12}}$ (N_{d_1})= N_{d_2}, 其中($N_{d_1}\in NE_k$, $N_{d_2}\in NE_{k+1}$),满足 $lab(N_{d_1})=lab(N_{d_2})$;同时满足:若 N_{d_1} 为原子元素, N_{d_2} 也为原子元素,且 $value(N_{d_1})= value(N_{d_2})$;

II. $NA_{k+1}\subseteq NA_k$,且存在一个部分映射 $F_{ap_{12}}$:$NA_k\to NA_{k+1}$,使得$\forall$ $F_{ap_{12}}(na_1)=na_2$,其中($na_1\in NA_k$,$na_{k+1}\in NA_2$),满足 $att(na_1)=att(na_2)$,$value(na_1)=value(na_2)$;

III. $E_{NE_{k+1}}\subseteq E_{NE_k}$, 且存在一个部分映射 $F_{ENEp_{12}}: E_{NE_k}\to E_{NE_{k+1}}$, 使得$\forall$ $F_{ENEp_{12}}(ene_1)=ene_2$,其中($ene_1\in E_{NE_k}$, $ene_1\in E_{NE_{k+1}}$), $ene_1=(n_{p_1}, n_{l_1}, n_1)$, $ene_2=(n_{p_2}, n_{l_2}, n_2)$,满足 $F_{np_{12}}(np_1)=np_2$, $F_{np_{12}}(nl_1)=nl_2$, $F_{np_{12}}(n_1)=n_2$;

IV. $Ref_{NE_{k+1}}\subseteq Ref_{NE_k}$, 且存在一个部分映射 $F_{Refp_{12}}: Ref_{NE_k}\to Ref_{NE_{k+1}}$, 使得$\forall$ $F_{Refp_{12}}((ne_{11},ne_{21}))=(ne_{12},ne_{22})$,其中($(ne_{11},ne_{21})\in Ref_{NE_k}$, $(ne_{12},ne_{22})\in Ref_{NE_{k+1}}$), 满足 $F_{np_{12}}(ne_{11})=ne_{12}$, $F_{np_{12}}(ne_{21})=ne_{22}$;

V. $NEAstr_{k+1}\subseteq NEAstr_k$,且存在一个部分映射 $F_{nep_{12}}$:$NEAstr_k\to NEAstr_{k+1}$,使得$\forall$ $F_{nep_{12}}((ne_1,na_1))=(ne_2,na_2)$,其中($(ne_1,na_1)\in NEAstr_k$,$(ne_2,na_2)\in NEAstr_{k+1}$),满足 $F_{np_{12}}(ne_1)=ne_2$, $F_{ap_{12}}(na_1)=na_2$;

VI. $root_{k+1}=N$.

如果(S_k,S_{k+1})是路径 p 的若干步,$Ins(Exp(Ins_i(S)),(S_1S_2\ldots S_kS_{k+1}))=(NE_{k+1},NA_{k+1}, E_{NE_{k+1}}, Ref_{NE_{k+1}}, NEAstr_{k+1}, root_{k+1})$定义为:

I. $NE_{k+1}\subseteq NE_k$,且存在一个部分映射 $F_{np_{12}}$:$NE_k\to NE_{k+1}$,$\exists N\in NE_k\wedge N<root_k\wedge lab(N)=S_2$,使得$\forall$ $F_{np_{12}}$ (N_{d_1})= N_{d_2}, 其中($N_{d_1}\in NE_k$, $N_{d_2}\in NE_{k+1}$),满足 $lab(N_{d_1})=lab(N_{d_2})$;同时满足若 N_{d_1} 为原子元素, N_{d_2} 也为原子元素,且 $value(N_{d_1})= value(N_{d_2})$;

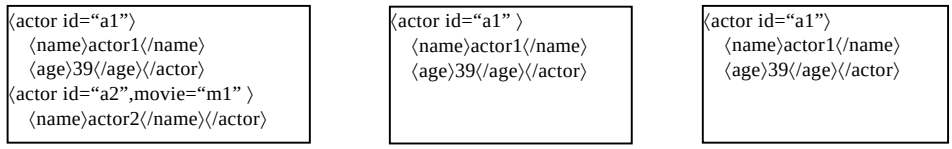
II. $NA_{k+1}\subseteq NA_k$,且存在一个部分映射 $F_{ap_{12}}$:$NA_k\to NA_{k+1}$,使得$\forall$ $F_{ap_{12}}(na_1)=na_2$,其中($na_1\in NA_k$,$na_{k+1}\in NA_2$),满足 $att(na_1)=att(na_2)$,$value(na_1)=value(na_2)$;

III. $E_{NE_{k+1}}\subseteq E_{NE_k}$, 且存在一个部分映射 $F_{ENEp_{12}}: E_{NE_k}\to E_{NE_{k+1}}$, 使得$\forall$ $F_{ENEp_{12}}(ene_1)=ene_2$,其中($ene_1\in E_{NE_k}$, $ene_1\in E_{NE_{k+1}}$), $ene_1=(n_{p_1}, n_{l_1}, n_1)$, $ene_2=(n_{p_2}, n_{l_2}, n_2)$,满足 $F_{np_{12}}(np_1)=np_2$, $F_{np_{12}}(nl_1)=nl_2$, $F_{np_{12}}(n_1)=n_2$;

IV. $Ref_{NE_{k+1}}\subseteq Ref_{NE_k}$, 且存在一个部分映射 $F_{Refp_{12}}: Ref_{NE_k}\to Ref_{NE_{k+1}}$, 使得$\forall$ $F_{Refp_{12}}((ne_{11},ne_{21}))=(ne_{12},ne_{22})$,其中($(ne_{11},ne_{21})\in Ref_{NE_k}$, $(ne_{12},ne_{22})\in Ref_{NE_{k+1}}$), 满足 $F_{np_{12}}(ne_{11})=ne_{12}$, $F_{np_{12}}(ne_{21})=ne_{22}$;

V. $NEAstr_{k+1}\subseteq NEAstr_k$,且存在一个部分映射 $F_{nep_{12}}$:$NEAstr_k\to NEAstr_{k+1}$,使得$\forall$ $F_{nep_{12}}((ne_1,na_1))=(ne_2,na_2)$,其中($(ne_1,na_1)\in NEAstr_k$,$(ne_2,na_2)\in NEAstr_{k+1}$),满足 $F_{np_{12}}(ne_1)=ne_2$, $F_{ap_{12}}(na_1)=na_2$;

VI. $root_{k+1}=N$.



(a) (b) (c)

Fig.2 Fragment for query results

图 2 查询结果片段

定义 **3.10**(路径绑定). 设 $S=((E_S,E_{rootS}),A_S,Rule_S,Ref_S,EAstr_S)$是一个 XML 数据集合的模式,$p=(S_1,S_2,\ldots,S_k)$是 S 上的一条路径.路径绑定操作 $Bind(S,p,\$b)$的结果为$\$b$ 的模式与实例同 $Exp(S,p)$.

图 2(a)给出了图 1(a)例子操作 $Path(XS_1,(MovieDB,actor))$的查询结果,其中 XS_1 是图 1(a)的模式.我们可以注意到查询结果是一系列元素的集合,而这些结果通过重构操作得到一个完整的 XML 文档.

定义 **3.11**(路径选择). 设 $S=((E_S,E_{rootS}),A_S,Rule_S,Ref_S,EAstr_S)$是一个 XML 数据集合的模式,$p=(S_1,S_2,\ldots,S_k)$是 S 上的一条路径,F 是 S_k 的终结结点的一个表达式.路径选择操作$\delta_F(S,p)$定义如下:

(1) $Sch(\delta_F(S,p))=Sch(Exp(S,p))$;

(2) $\{Ins(\delta_F(S,p))\}\subseteq\{Ins(Exp(S,p))\}$,对于每个 $Ins(\delta_F(S,p))$都满足 F.

路径选择与关系数据库中的选择操作类似,关系数据库的选择操作得到满足条件 F 的原组,在 XML 数据模

型中,选择操作得到满足条件 F 的实例.除了'<'等操作,这里增加了'IN'操作,指明字符串信息的包含关系.与关系数据库不同之处在于,这里的 F 的操作数为 S_k 的终结元素结点的 *value* 或者某个后代结点 S_i 的属性 *value*,而不是关系中的属性.图 2(b)给出了 $\delta_{actor.name='actor1'}(S,movieDB,actor)$的查询结果.

定义 **3.12**(连接). 设 $S=((E_S,E_{rootS}),A_S,Rule_S,Ref_S,EAstr_S)$和 $T=((E_T,E_{rootT}),A_T,Rule_T,Ref_T,EAstr_T)$是两个 XML 数据集合的模式,$F$ 是连接条件.S 和 T 的连接操作 θ-*join*(S,T,F)定义为:

(1) $Sch(\theta\text{-}join(S,T,F))=Sch(S\times T)$;

(2) $\{Ins(\theta\text{-}join(S,T,F))\}\subseteq\{Ins(S\times T)\}$,且满足条件 F.

图 2(c)给出了连接操作 θ-*join*$(XS_1,XS_2,XS_1//actor=XS_2//actor)$的查询结果.

定义 **3.13**(排序). 设 $S=((E_S,E_{rootS}),A_S,Rule_S,Ref_S,EAstr_S)$是一个 XML 数据集合的模式,$XS_1(S)=(NE_1,NA_1,E_{NE_1},Ref_{NE_1},NEAstr_1,root_1)$是 S 的一个实例,$p=(S_1,S_2,\ldots,S_k)$和 $sp=(S_1,S_2,\ldots,S_k,\ldots,S_n)$分别是 S 上的两条路径.路径 p 上的排序操作表示为 $Sort(XS_1(S),p,flag,sp)$,其中,$flag=DESC,ASC$,或者 *DOC*,*DESC* 表示按元素值降序排列,*ASC* 表示按元素值升序排列,*DOC* 表示按元素在文档中出现的顺序排列.定义如下:

(1) $Sch(Sort(XS_1(S),p,flag,sp))=Sch(Exp(S,p))$;

(2) $\{Ins(Sort(XS_1(S),p,flag,sp))\}=\{Ins(Exp(XS_1(S),p))\}$;

(3) $\{Ins(Sort(XS_1(S),p,flag,sp))\}$中的实例按如下规则排序:① 如果 $flag=DESC$,排序后的结果为$\{Ins_{d_1},Ins_{d_2},\ldots,Ins_{d_m}\}$,满足若 $Ins_i<Ins_j,\exists n_i\in Ins_i$ 和 $n_j\in Ins_j$,使得 $value(n_i)\geq value(n_j)$;② 如果 $flag=ASC$,排序后的结果为$\{Ins_{d_1},Ins_{d_2},\ldots,Ins_{d_m}\}$,满足若 $Ins_i<Ins_j,\exists n_i\in Ins_i$ 和 $n_j\in Ins_j$,使得 $value(n_i)\leq value(n_j)$;③ 如果 $flag=DOC$,排序后的结果为$\{Ins_{d_1},Ins_{d_2},\ldots,Ins_{d_m}\}$,满足若 $Ins_i<Ins_j$,对 $root_i\in Ins_i$ 和 $root_j\in Ins_j,\exists n_i,n_j,n_p\in XS_1(S)(root_i\leq n_i,root_j\leq n_j$ 且 $n_i\leq n_p,n_j\leq n_p)$,满足$(n_p,n_i,n_j)\in XS_1(S)$.

定义 **3.14**(聚集). 设 $S=((E_S,E_{rootS}),A_S,Rule_S,Ref_S,EAstr_S)$是一个 XML 数据集合的模式,$p=(S_1,S_2,\ldots,S_l)$是 S 上的一条路径,路径 p 上的聚集操作表示为 $Agg(S,p,flag,sp_T,DS,MS)$.其中,$DS=(sp_1,sp_2,\ldots,sp_n)$是聚集路径的集合;$FS=(Func_1(mp_1),Func_2(mp_2),\ldots,Func_k(mp_k))$是路径聚集的集合;$sp_1,sp_2,\ldots,sp_n$ 以及 $mp_1,mp_2,\ldots,mp_k$ 都是以路径 p 为起始的路径,且从 S_l 到 $sp_i(1\leq i\leq n)$的结束结点有且仅有唯一的路径;$Func_i(1\leq i\leq n)$是聚集函数,可以为 sum,count,max,min,average.聚集操作的结果为形如(D,M)的序列,其中 $D=\{sp_1,sp_2,\ldots,sp_n\}$,$M=\{Func_1(mp_1),Func_2(mp_2),\ldots,Func_k(mp_k)\}$;存在一个映射 $F_{agg}:Ins(sp_1)\times Ins(sp_2)\times\ldots\times Ins(sp_n)\rightarrow Func_1(Ins(mp_1))\times Func_2(Ins\ (mp_2))\times\ldots\times Func_k(Ins(mp_k))$.

3.3 数据维护操作

一个实用的数据库系统不但要求数据查询能力,也要对数据可以进行修改.本节将给出 XML 数据库数据维护操作的最小语义,包括模式的维护和数据维护两部分.

定义 **3.15**(规则添加). 设 $S=((E_S,E_{rootS}),A_S,Rule_S,Ref_S,EAstr_S)$是一个 XML 数据集合的模式,$R_1$ 为新添加的规则,el 为该规则的原象.规则添加 $Add\text{-}Rule(S,el,R_1)$操作后,数据集合的模式为 $Sch(Add\text{-}Rule(S,el,R_1))=((E_S\cup\{el_1,el_2,\ldots,el_n\},E_{rootS}),A_S,Rule_S\cup\{R_1\},Ref_S,EAstr_S)$,其中 $el_1,el_2,\ldots,el_n$ 为规则 R_1 中出现的元素.

定义 **3.16**(规则删除). 设 $S=((E_S,E_{rootS}),A_S,Rule_S,Ref_S,EAstr_S)$是一个 XML 数据集合的模式,$R_1$ 为将删除的规则,el 为该规则的原象.规则删除 $Del\text{-}Rule(S,el,R_1)$操作后,数据集合的模式为 $Sch(Del\text{-}Rule(S,el,pel))=((E_S-\{el_1,el_2,\ldots,el_n\},E_{rootS}),A_S-\{att_1,att_2,\ldots,att_m\},Rule_S-\{R_1\},Ref_S-\{Ref_{R_1}\},EAstr_S-\{EAstr_{R_1}\})$.其中,$el_1,el_2,\ldots,el_n$ 为规则 R_1 中出现,且没有在其他规则中出现的元素;$att_1,att_2,\ldots,att_m$ 为 $el_1,el_2,\ldots,el_n$ 的元素结点,且没有其他父亲;Ref_{R_1} 为涉及 $el_1,el_2,\ldots,el_n$ 的规则;$EAstr_{R_1}$ 为涉及 $el_1,el_2,\ldots,el_n$ 的元素.

定义 **3.17**(属性添加). 设 $S=((E_S,E_{rootS}),A_S,Rule_S,Ref_S,EAstr_S)$是一个 XML 数据集合的模式,*att* 为新加入模式的属性结点,*pel* 为 *att* 的父结点.属性添加 $Add\text{-}Att(S,att,pel)$操作后,数据集合的模式为 $Sch(Add\text{-}att(S,al,pel))=((E_S,E_{rootS}),A_S\cup\{att\},Rule_S,Ref_S,EAstr_S\cup\{(att,pel,<)\})$.

定义 **3.18**(属性删除). 设 $S=((E_S,E_{rootS}),A_S,Rule_S,Ref_S,EAstr_S)$是一个 XML 数据集合的模式,*att* 为模式中将删除的属性结点,*pel* 为 *att* 的父结点.属性添加 $Del\text{-}Att(S,att,pel)$操作后,数据集合的模式为 $Sch(del\text{-}att(S,al,pel))=$

$((E_S,E_{rootS}),A_S-\{att_1\},Rule_S,Ref_S,EAstr_S-\{(att,pel,<)\})$.其中 att_1 为∅,若 *att* 有多个父亲;$att_1=att$,若 *att* 只有一个父亲 *pel*.

定义 **3.19**(引用添加). 设 $S=((E_S,E_{rootS}),A_S,Rule_S,Ref_S,EAstr_S)$是一个 XML 数据集合的模式,$R_1$ 为新添加的引用.引用添加 $Add\text{-}Ref(S,R_1)$操作后,数据集合的模式为 $Sch\ (Add\text{-}Ref(S,R_1))=((E_S,E_{rootS}),A_S,Rule_S,Ref_S\cup R_1,EAstr_S)$.

定义 **3.20**(引用删除). 设 $S=((E_S,E_{rootS}),A_S,Rule_S,Ref_S,EAstr_S)$是一个 XML 数据集合的模式,$R_1$ 为将删除的引用.引用添加 $Del\text{-}Ref(S,R_1)$操作后,数据集合的模式为 $Sch(Del\text{-}Ref(S,R_1))=((E_S,E_{rootS}),A_S,Rule_S,Ref_S-R_1,EAstr_S)$.

定义 3.15~定义 3.20 给出了 XML 数据集合模式维护操作,这些操作必须在该模式没有实例的时候进行.

定义 3.21 和定义 3.22 中的路径 *p* 可以为通过 $Path(S,p)$操作得到的路径.

定义 **3.21**(数据更新操作). 设 $S=((E_S,E_{rootS}),A_S,Rule_S,Ref_S,EAstr_S)$是一个 XML 数据集合的模式,$p=(S_1,S_2,\ldots,S_k)$是 *S* 上的一条路径,且 S_k 是 *S* 的终结结点.XML 数据集合的更新操作 $update\text{-}value(S,p,constant_k)$表达的语义为将 *S* 上路径 *p* 的目标结点 S_k 的值更新为 $constant_k$.

定义 **3.22**(数据插入操作). 设 $S=((E_S,E_{rootS}),A_S,Rule_S,Ref_S,EAstr_S)$是一个 XML 数据集合的模式,$XS_1(S)=(NE_1,NA_1,E_{NE_1},Ref_{NE_1},NEAstr_1,root_1)$是 *S* 的一个实例,$p=(S_1,S_2,\ldots,S_k)$是 *S* 上的一条路径,且 $XS_{p_1}(S)=(NE_{p_1},NA_{p_1},E_{NE_{p_1}},Ref_{NE_{p_1}},NEAstr_{p_1},root_{p_1})$是 $exp(XS_1(S),p)$的一个实例.XML 数据集合上的数据插入操作表示为 $Insert\text{-}Ins(S,p,Ins,flag)$.其中,$Ins=(NE_{Ins},NA_{Ins},E_{NEIns},Ref_{NEIns},NEAstr_{Ins},root_{Ins})$是将插入的实例,*flag* 为'before'或者'after'.数据插入后,原数据集合的模式不变.

(1) 如果 *flag*='before',$XS_1(S)=(NE_1\cup NE_{Ins},NA_1\cup NA_{Ins},E_{NE_1}\cup E_{NEIns}\cup\{(n_p,n_l,root_{Ins}),(n_p,root_{Ins},root_{p_1})\}-\{(n_p,n_l,root_1)\},Ref_{NE_1}\cup Ref_{Ins},NEAstr_1\cup NEAstr_{Ins},root_1)$,其中,$(n_p,n_l,root_1)\in Ins_1(S).E_{NE}$;

(2) 如果 *flag*='after',$XS_1(S)=(NE_1\cup NE_{Ins},NA_1\cup NA_{Ins},E_{NE_1}\cup E_{NEIns}\cup\{(n_p,root_{p_1},root_{Ins}),(n_p,root_{p_1},n_r)\}-\{(n_p,root_1,n_r)\},Ref_{NE_1}\cup Ref_{Ins},NEAstr_1\cup NEAstr_{Ins},root_1)$,其中,$(n_p,root_1,n_r)\in Ins_1(S).E_{NE}$.

例 3.1:设例 2.1 给出的模式为 $S_1=((E_1,E_{root1}),A_1,Rule_1,Ref_1,EAstr_1)$.若 $S_2=((E_2,E_{root2}),A_2,Rule_2,Ref_2,EAstr_2)=Del\text{-}Rule(S_1,actor,(actor\rightarrow name,age^?))$,则 $E_2=\{MovieDB,actor,name,director,movie,title\};E_{root1}=MovieDB;A_2=A_1;Rule_2=\{(MovieDB\rightarrow actor^+,director^+,movie^+),(director\rightarrow name,movie),(movie\rightarrow title)\};Ref_2=Ref_1;EAstr_2=EAstr_1.S_3=Add\text{-}Rule(S_2,acotr,(actor\rightarrow name,age^?))$,则 $S_3=S_1$.若 $S_4=((E_4,E_{root4}),A_4,Rule_4,Ref_4,EAstr_4)=Add\text{-}att(S_1,nick,(nick,actor,<))$,则 $E_4=E_1;E_{root4}=E_{root1};A_4=A_1\cup\{nick\};Rule_4=Rule_1;Ref_4=Ref_1;EAstr_4=EAstr_1\cup(nick,actor,<)$.若 $S_5=((E_5,E_{root5}),A_5,Rule_5,Ref_5,EAstr_5)=Del\text{-}Ref(S_1,Ref(actor/movie,movie/id))$,则 $E_5=E_1;E_{root5}=E_{root1};A_5=A_1;Rule_5=Rule_1;Ref_5=Ref_1-\{Ref(actor/movie,movie/id)\};EAstr_5=EAstr_1$.

4 分析比较

建立 XML 数据库数据模型的目的是对 XML 数据的结构和操作语义进行描述,进而实现 XML 数据库查询优化.利用关系模型和面向对象模型对 XML 数据的结构和操作语义进行描述,本质上是将 XML 数据的结构和语义映射到这两种数据模型.因此,本文不对这两种方法进行分析比较.目前已出现的 XML 数据模型包括树模型和图模型.在这两种数据模型的基础上,研究者提出了一系列的 XML 操作代数:UnQL,SAL,TAX,OrientXA 等,其中,UnQL 和 SAL 基于图模型,TAX 和 OrientXA 基于树模型.

下面,我们对图模型、树模型以及本文提出的数据模型进行分析比较,见表 1.一个 XML 数据库的数据模型应具有如下特点:模型的表达能力(可以表达 XML 数据的结构);具有从 XML 文档选择满足给定条件信息的能力(doc selection),支持从 XML 数据库选择满足给定条件信息的能力(DB selection);支持 XML 数据上的 Join 操作(joins);能表达查询结果的语义(semantics of result),能表达路径表达式(path expressions.);具有构造新数据元素的机制(construct);支持查询结果的聚集(aggregate);支持查询结果的排序(order);支持数据更新操作(update);支持基于路径的数据更新操作(update by path);使用的方便性(convenience);形式化(形式化的模型可以直接用来进行查询优化,记为 formallized).

从表 1 可以看出,树模型只能部分支持 XML 数据的结构.树结构虽然可以表达出 XML 数据的层次结构,

但由于树自身结构的限制,使得树模型无法对 XML 数据结构的语义进行有效支持.图 3 是图 1 给出的 XML 数据的结构图,该图中存在一个从 movie/@director 到 director/@id 的引用,这使得该 XML 数据的结构无法用树模型进行表示.因此,表 1“Semantics of data structure”一项比较结果给出树模型只能对 XML 数据结构以部分支持.图 3 给出的结构可以用图模型来进行表示.图模型上的操作大多是针对对象进行的.对图 1 给出的两个 XML 文档片段,利用直观的图模型对它们分别表示,得到的图模型表示也不同,因为图模型无法表示数据库模式的概念.此外,从表 1 可以看出,图模型提供的数据更新操作和模式更新操作也不完全;也没有研究给出可以方便数据库查询优化的形式化的操作代数.因此,图模型很难适应数据库管理的需要.而本文提出的模型可以很好地解决树模型和图模型存在的这些问题.综上所述,本文提出的数据模型吸收了已有 XML 数据模型的优点,优于其他几种 XML 数据模型.

Table 1 Comparison for several XML data models

表 1 几种数据模型的比较

Data model	Graph model	Tree model	Our model
Semantics of data structure	Yes	Partially	Yes
DB selection	Partially	No	Yes
Doc selection	Partially	Yes	Yes
Joins	Yes	Yes	Yes
Semantics of result	OIDs	XML doc	XML sources
Path expression	Yes	Yes	Yes
Construct	Yes	Yes	Yes
Aggregates	Yes	Partially	Yes
Order	Yes	Yes	Yes
Tag binding	Partially	Yes	Yes
Update operations	Partially	No	Yes
Update by path	No	No	Yes
Convenience	Yes	Yes	Yes
Formallized	No	No	Yes

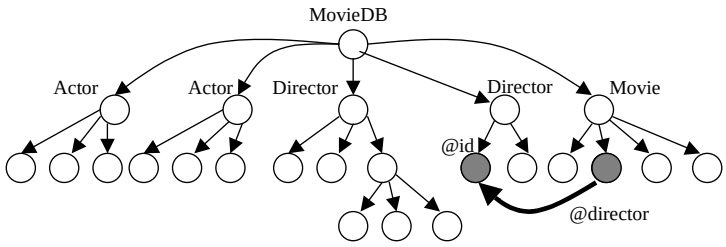


Fig.3 The structure graph for XML data

图 3 XML 数据的结构图

5 总 结

本文以映射为基础,提出了一种 XML 数据库的数据模型.该数据模型能够充分表达 XML 数据库的复杂数据结构和语义,并包含以路径表达式操作为核心的操作代数.本文同时给出了 XML 数据库上更新操作代数.与其他工作相比,它有以下优点:(1) 该模型能够表示 XML 数据的复杂语义;(2) 基于映射给出了 XML 数据操作的精确语义;(3) 给出了其他模型很少涉及的 XML 数据上更新操作的语义.

References:

[1] Quass D, Widom J, Goldman R, Haas K, Luo Q, McHugh J, Nestorov S, Rajaraman A, Rivero H, Abiteboul S, Ullman J, Wiener J. LORE: A lightwight object repository for semistructured data. In: Jagadish HV, Mumick IS, eds. Proc. of the 1996 ACM SIGMOD Int’l Conf. on Management of Data. New York: ACM Press, 1996. 549.

[2] Zhou AY, Lu HJ, Zheng SH, Liang YQ, Zhang L, Ji W, Tian ZP. VXMLR: A visual XML-relational database system. In: Apers P, Atzeni P, Ceri S, Paraboschi S, Ramaohanarao K, Snodgrass R, eds. Proc. of the 27th Int’l Conf. on Very Large Data Bases. San Fransisco: Morgan Kaufmann Publishers, 2001. 719−720.

[3] Wang Q, Zhou JM, Wu HW, Xiao JC, Zhou AY. Mapping XML documents to relations in the presence of functional dependencies. Journal of Software, 2003,14(7):1275−1281 (in English with Chinese abstract). http://www.jos.org.cn/1000-9825/14/1275.htm

[4] Florescu D, Kossmann D. Storing and querying XML data using an RDBMS. IEEE Engineering Bulletin, 1999,22(3):27−34.

[5] Tatarinov I, Viglas S, Beyer K, Shanmugasundaram J, Shekita E, Zhang C. Storing and querying ordered XML using a relational database system. In: Franklin M, Moon B, Ailamaki A, eds. Proc. of the 2002 ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 2002. 204−215.

[6] Yoshikawa M, Amagasa T, Shimura T, Uemura S. XRel: A path-based approach to storage and retrieval of XML documents using relational databases. ACM Trans. on Internet Technology, 2001,1(1):110−141.

[7] Zhang C, Naughton J, DeWitt DJ, Luo Q, Lohman G. On supporting containment queries in relational database management systems. In: Aref W, ed. Proc. of the 2001 ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 2001. 426−437.

[8] Arenas M, Libkin L. A normal form for XML documents. ACM Trans. on Database Systems, 2004,29(1):195−232.

[9] Christophides V, Cluet S, Moerkotte G, Siméon J. On wrapping query languages and efficient XML integration. In: Chen W, Naughton J, Bernstein P, eds. Proc. of the 2000 ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 2000. 141−152.

[10] Ludascher B, Papakonstantinou Y, Velikhov P. Navigation-Driven evaluation of virtual mediated views. In: Zaniolo C, Lockemann P, Scholl M, Grust T, eds. Advances in Database Technology-EDBT 2000, 7th Int'l Conf. on Extending Database Technology. Berlin, Heidelberg: Springer-Verlag, 2000. 150−165.

[11] Fankhauser P, *et al.* XQuery 1.0 and XPath 2.0 formal semantics. 2005. http://www.w3.org/TR/query-semantics/

[12] Chamberlin D, *et al.* XQuery 1.0: An XML query language. 2005. http://www.w3.org/TR/xquery/

[13] Buneman P, Fernandez M, Suciu D. UnQL: A query language and algebra for semistructured data based on structural recursion. The VLDB Journal, 2000,9(1):76−110.

[14] Beeri C, Tazban Y. SAL: An algebra for semistructured data and XML. In: Cluet S, Milo T, eds. Proc. of the 2nd ACM SIGMOD Workshop on the Web and Databases. INRIA, 1999. 37−42. http://www-rocq.inria.fr/~cluet/webdb99.html

[15] Jagadish H, Lakshmanan L, Srivastava D, Thompson K. TAX: A tree algebra for XML. In: Ghelli G, Grahne G, eds. Proc. of the 8th Int'l Workshop on Database Programming Languages. Berlin, Heidelberg: Springer-Verlag, 2001. 149−164.

[16] Fernandez M, Siméon J, Wadler P. An algebra for XML query. In: Kapoor S, Prasad S, eds. Proc. of the Foundations of Software Technology and Theoretical Computer Science, 20th Conf. Berlin, Heidelberg: Springer-Verlag, 2001. 11−45.

[17] Meng XF, Luo DF, Jiang Y, Wang Y. OrientXA: An effective XQuery algebra. Journal of Software, 2004,15(11):1648−1660 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/1648.htm

附中文参考文献:

[17] 孟小峰,罗道锋,蒋瑜,王宇.OreintXA:一种有效的 XQuery 查询代数.软件学报,2004,15(11):1648−1660. http://www.jos.org.cn/1000-9825/15/1648.htm

何震瀛(1977－),男,黑龙江牡丹江人,博士生,主要研究领域为 XML 数据管理.

王朝坤(1976－),男,博士生,主要研究领域为音频数据管理,P2P 数据管理.

李建中(1950－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据库技术,并行计算技术.