基于 XML 的 Web 数据模型*)

秦 杰[1] 赵淑梅[1] 杨树强[2] 窦文华[2]

(河南工业大学信息科学与工程学院 郑州 450052)[1] (国防科技大学计算机学院 长沙 410073)[2]

摘 要 在对现有半结构化数据模型分析的基础上,针对这些模型作为 Web 数据模型的不足,提出一种新的基于 XML 的 Web 数据模型——XWDM,它主要解决了 Web 数据名称异构问题和查询回路问题。

关键词 XML,半结构化数据,Web 数据模型

XML-based Web Data Model

QIN Jie[1] ZHAO Shu-Mei[1] YANG Shu-Qiang[2] DOU Wen-Hua[2]

(School of Information Science and Engineering, Henan University of Technology, Zhengzhou 450052)[1]

(School of Computer, National University of Deference Technology, Changsha 410073)[2]

Abstract Some semi-structured data models have been analyzed, and some shortcomings of these models as Web data model are point out. Based on XQuery 1.0 and XPath 2.0 data model, a new XML-based Web Data Model(XWDM) is presented. XWDM gives a solution to the problem of name heterogeneous, which is resulted from the name difference of the same data element in different Web pages, and solves the problem of infinite circulation which is common during Web data querying.

Keywords XML, Semi-structured data, Web data model

Web 数据模型是 Web 数据管理的基础[1]。要实现 Web 数据的有效查询,必须通过适当的模型清晰地表示 Web 数据。针对 Web 数据半结构化(Semi-structured)特点,寻找一个适合 Web 数据表达的半结构化数据模型是解决问题的关键所在[1]。目前,得到 W3C 推荐的半结构化数据模型有 4 种:XPath 数据模型[2]、DOM 模型[3]、Infoset 模型[4]、XQuery 1.0 和 XPath 2.0 数据模型[5]。这些模型都可以作为 Web 数据模型[5]。使用这些数据模型描述 Web 数据时,普遍存在以下问题:(1)数据异构。在不同 Web 页面中,对于同一数据对象的描述使用的名称或者描述方式很可能不同,因而存在数据异构问题;(2)查询回路。由于上述模型允许在 Web 页面的不同层次上的数据元素名称可以相同,在对这种 Web 数据查询时会出现循环嵌套(查询回路)问题[5]。为了解决这两个问题,本文提出了基于 XML 的 Web 数据模型(XML-based Web Data Model,简称 XWDM),XWDM 主要解决了 Web 数据名称异构问题和查询回路问题。

本文内容安排如下:首先简单介绍本文需要用到的 XML 相关基础知识;之后对比较有代表性的半结构化数据模型——对象交换模型(OEM)、XQuery 1.0 和 XPath 2.0 数据模型进行介绍;最后提出基于 XML 的 Web 数据模型(XWDM)。

1 XML 基础

1.1 XML 文档构成

XML 是一种标识语言,是一组用来创建描述数据的语法标记的规则集[6]。XML 描述的是一类被称为 XML 文档的数据对象,并且部分地描述了处理这些数据的计算机程序的行为。每个 XML 文档都具有逻辑和物理结构。物理上而言,XML 文档由按照一定的层次顺序排列的若干实体单元组成。从逻辑上讲,XML 文档由声明(declaration)、元素(element)、注释(comment)、字符引用(character reference)以及处理指令(processing instruction,简称 PI)等成分组成,这些成分在文档中都用不同的标签(tag)加以表示。

XML 文档以一个 XML 声明开始,用来说明关于文档的基本信息。

元素(element):是 XML 文档的基本单元。一个 XML 元素由开始标签、结束标签以及标签之间的数据构成,元素标签的格式与 HTML 中的标签格式相同。XML 文档数据必须包含在一个单一元素中。这个单一元素称为根(root)元素,或文档实体。文档实体代表整个文档的内容。

属性(attribute):一个元素可以包含一个或多个属性,属性是一个在元素的开始标签中的属性名称-属性值对。属性有两个规则:属性必须有值,值必须用引号括起。XML 有 3 种类型的属性:字符串类型(StringType),记号类型(TokenizedType)和枚举类型(EnumeratedType)。字符串类型可以以任意的常量字符串作为值,各个记号类型有不同的词法和语义约束[1]。

注释(comment):可在文档中其它标记之外的任何位置出现,起注解作用。

处理指令(PI):在 XML 文档中允许包含由应用来处理的指令,这些指令不属于文档数据内容。

字符引用:一个字符引用可以引用 ISO/IEC10646 字符集中的一个字符。

实体引用(entity reference):可以引用一个命名实体的内

*)863 课题:网络环境的新一代中间件核心技术及运行平台(No. 2004AA112020)。秦 杰 博士生,研究方向:Web 数据库技术;杨树强 教授,研究方向:数据库技术;窦文华 教授,研究方向:网络技术。

容,表达元素之间的引用信息。

CDATA 段:出现在字符数据可以出现的任何地方。CDATA 标记的内容被当作字符数据看待。

XML 文档通过文档类型声明(DOCTYPE)来声明对 XML 文档逻辑结构的定义。DOCTYPE 必须位于文档根元素之前。存在两种定义 XML 文档逻辑结构的方式:文档类型定义(Document Type Define,简称 DTD)和 XML 模式(XML Schema)。

DTD[7]是最初的 XML 规范的一部分,是一套关于标记符的语法规则,规定可以在文档中使用的标记、标记之间的顺序关系、标记嵌套关系,以及元素可以使用的属性等。XML 本身并没有一个通用的 DTD,用户可以自行定义 DTD。DTD 可以是一份单独的文件,也可以直接包含在 XML 文档中。

XML Schema[8]是 W3C 于 2001 年 5 月提出的,其目的是替代 DTD 来描述 XML 文档的逻辑结构。XML Schema 除了具有 DTD 的所有功能之外,还可以定义数据类型,具有比 DTD 更复杂的规则。

XML 文档中元素可以是有序的,属性是无序的;串是 XML 文档唯一的数据类型,其它复杂的数据类型可以通过 XML Schema 定义。XML 文档的逻辑结构定义是可选的,即一个 XML 文档可以没有逻辑结构的定义。

一个标准 XML 文档由序言(Prolog)和一个根元素两大部分组成。序言必须以一个 XML 文档声明开始,还可以包括其它一些声明语句,如用来指定 DTD(或者 XML Schema)的文档类型声明(DOCTYPE)语句;根元素是文档内容的主体,根元素中包含文档的数据内容。

通常所说的 XML 文档指的是 XML 文档的物理结构中根元素所包括的内容。

1.2 XML 相关技术规范

本文涉及以下 XML 技术标准和规范。

XML 名称空间规范(XML namespace)[9]是 XML 规范的另一个重要部分,XML 名称空间是一组 XML 文档的元素或属性名称的集合。名称空间中的成员由 URI(Universal Resource Identifier)识别,它们在 Internet 上是唯一的。为了避免 XML 文档中元素或属性之间出现相同名称的冲突,每个名称空间中的成员唯一对应一个元素或者属性名称,这样就避免了文档中同名元素或者属性名称之间的冲突。

XML 路径语言(XML Path Language,简称 XPath)[2]描述 XML 文档中元素和属性位置的语法。

XQuery 是 W3C 关于 XML 查询语言的推荐标准。

XML 链接语言(XML Linking Language,简称 XLink)[10]是关于 XML 链接的标准,定义了将不同资源链接在一起的各种方法。

XML 指针语言(XML Pointer Language,简称 XPointer)[11]是关于 XML 引用的标准,它使用 XPath 作为引用其它资源的方法,还包括对 XPath 的一些扩展。

文档对象模型(Document Object Model,简称 DOM)定义了如何将 XML 文档转换为驻留内存的树结构,以及如何使用 DOM 接口来操作这个树结构[3]。DOM 是目前处理 XML 文档的一种典型方式。

1.3 XML 查询表达式

查询表达式是 XQuery[17]查询语言结构中最重要的部分,表达式具有循环构造特性,由以下 8 种基本模式构成:

a. 路径表达式(Path expressions);

b. 元素构造符(Element constructors);

c. FLWR 表达式(FLWR expressions);

d. 算子和函数表达式(Expressions involving operators and functions);

e. 条件表达式(Conditional expressions);

f. 限定表达式(Quantified expressions);

g. 列表构造符(List constructors);

h. 数据类型表达式(Expressions that test or modify data-types)。

其中,路径表达式是 XQuery 查询表达式的核心。W3C 专门推出了 XPath[2],用于对路径表达式进行规范。XPath 是本身也是一种对节点操作的查询语言,它通过路径表达式来选择 XML 文档树的节点,从而实现对 XML 数据的查询。

下面介绍查询表达式中两个最重要的成分:路径表达式、FLWR 表达式。

(1)路径表达式

XPath 定义的路径表达式的核心是正则路径表达式(regular path expression,简称 RPE),RPE 的构成语法定义为:

r::=//r|r1/r2|r*|r+|r?|(r1|r2)|#|name|ε

其中,r 是包含?,*,+,?,#,∈的路径表达式。//r 表示任意的到达 r 的路径,r1/r2 表示从 r1 到 r2 的路径,r*、r+和 r? 分别表示 r 的 0 或多、1 或多,以及 0 或 1 次重复,(r1|r2)表示选择关系,# 表示任意的路径,name 表示节点名称,ε 表示空路径。

(2)FLWR 表达式

FLWR 表达式是 XQuery 最为重要的语法类型之一。FLWR 是 For-Let-Where-Return 的缩略词,类似于 SQL 中的 SELECT-FROM-WHERE 结构,构成了 XQuery 表达式的主干。

可以认为 FLWR 表达式是 XQuery 查询语句的标准形式,而 XPath 路径表达式是对查询的简洁描述方式。

2 半结构化数据模型

目前,大多数研究直接采用半结构化数据模型作为 Web 数据模型[2~5,13],其中最有代表性的是 OEM 模型以及 XQuery 1.0 和 XPath 2.0 数据模型。

2.1 对象交换模型

对象交换模型(Object Exchange Model,简称 OEM)是一种典型的 Web 数据模型[1]。该模型产生于 1997 年,是为了专门表示半结构化数据而提出的,并应用在斯坦福大学 Abiteboul 等人开发的半结构化数据库管理系统 Lore(light-weight object repository)[13]中。在 OEM 模型中,所有实体都是对象(object),一个 OEM 对象是一个四元组:

(label,oid,type,value)

其中,label 是对象标记,表示对象的名称,标记有两个作用:用于区分不同的对象、表达对象的含义;oid 是对象标识符,不同对象的 oid 是唯一的;type 表示对象类型,OEM 定义了两种类型的对象:复杂类型、原子类型;value 是对象的取值。当 type 是原子类型时,称对象为原子对象(atomic object),此时 value 是该类型的一个原子值,如 integer、real、string、gif 等;当 type 是复杂类型时,对象称为复杂对象(complex object),此时 value 是对象标识的集合(或列表),即复杂对象由若干

子对象构成，子对象可以是复杂对象或者原子对象。

XML 推出后，OEM 数据模型又专门进行了基于 XML 的扩展[14]，在基于 XML 的 OEM 数据模型中，一个 XML 元素用一个二元组〈eid，value〉表示。eid 是该元素唯一的标识符；value 是元素的值，该值可以是一个文本串形式的原子值，也可以是包含以下 4 种成分的复杂值：(1)用字符串表示的该元素标记；(2)属性名-原子值对列表，其中属性名是一个字符串，代表属性的名称，原子值是该属性的取值，原子值的类型可以是 integer、real、string、ID、IDREF、或者 IDREFS 等；(3)一个有序的形如(label，eid)的链接子元素列表，其中 label 是一个字符串，表示子元素的标记，eid 为链接元素的标识符，链接子元素通过 IDREF 或者 IDREFS 属性类型加以说明；(4)一个有序的形如(label，eid)的一般子元素列表，表示非链接子元素，其中 label 是子元素的标记，eid 为子元素的标识符。对一般子元素与链接子元素加以区分是为了使模型可以同时支持文法模式和语义模式。

OEM 数据模型可以用有向边标记图模型表示，称为 OEM 图模型。在 OEM 图模型中，节点代表对象或者原子值，边对应元素名称。图 1 显示的是一个简单的 XML 文档和它对应的 OEM 图模型[1]。从图中可以直观地看出所描述数据的内容和结构特征：元素标识符(eid)以 &1、&2…的形式出现在节点中；指向该节点的边上的标记表示节点的名称(元素名称)；元素值以及属性名称和对应的属性值显示在相应节点的旁边。图中元素之间以及元素与对应的原子值之间的边用实线表示，IDREF 属性用斜体表示，交叉链接边用虚线表示。子元素的顺序按照自左至右排列。

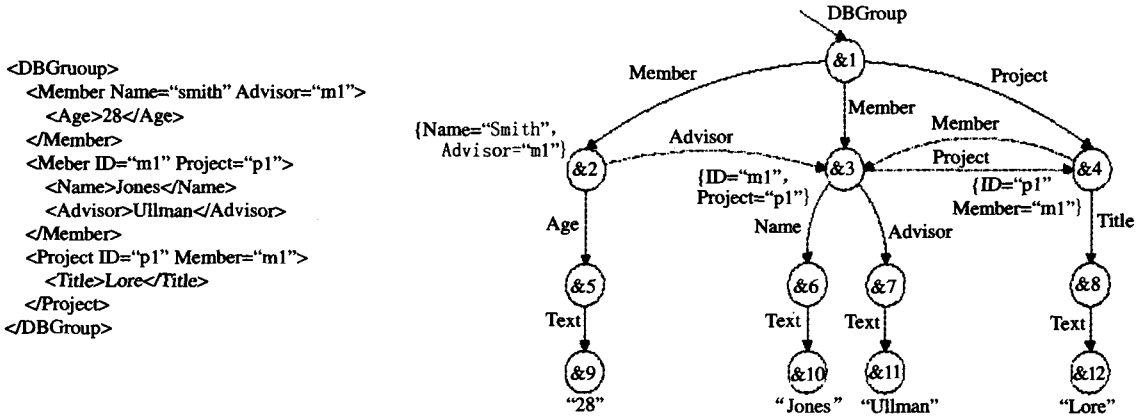

图 1　一个 XML 文档及其对应的 OEM 图模型

2.2　XML 数据模型

Infoset 模型、XPath 数据模型、DOM 模型、XQuery 1.0 和 XPath 2.0 数据模型都是 W3C 推荐的 XML 数据模型，这些模型都是 OEM 的变体。其中，XQuery 1.0 和 XPath 2.0 数据模型是 W3C 的候选推荐标准，该模型来源于 Infoset 模型，目前由 XSLT 2.0、XQuery 1.0 和 XPath 2.0 三部分组成。模型定义了 XML 文档的表示形式，描述了实现查询必须理解的数据项和形式语义的基础，对 XSLT 和 XQuery 处理器所能够处理的信息进行了说明，为 XSLT、XQuery、XPath 语言中的表达式所允许的值进行了定义。该模型目前仍然处于草案阶段。

在该模型中，把一个 XML 文档看成是由若干(数据)项(item)组成。一个 XML 文档可以表示成一棵树(XML 文档树)，文档中所有的 item 在 XML 文档树中用节点表示(node)，一个 item 是树的一个节点。模型定义了 7 种节点类型：文档节点、元素节点、属性节点、名称空间节点、处理指令节点、注释节点、文本节点。文档节点是 XML 文档的根节点，代表整个文档。元素(节点)和属性(节点)的取值为原子类型(atomic type)[6]值域中的值，包括 XML Schema 和 xdt:untypedAtomic 定义的所有类型。每个节点有一个唯一的节点标识(node identity)。另外，元素节点和属性节点还可以有类型值(typed values)、字符串值(string values)和名称，这些内容可以从节点中抽取出来。

3　基于 XML 的 Web 数据模型——XWDM

虽然 OEM 模型和 XML 数据模型都可以作为 Web 数据模型，但现有的这些 Web 数据模型普遍存在的问题是：(1)名称异构。在不同的 Web 页面中表示同一数据对象的数据元素可以有不同的命名方式，比如：软件与软体、数据库与资料库、信息与资讯等表达的含义是一样的，但名称不同，这种名称异构现象妨碍了对 Web 数据的查询，严重影响了信息查询的完全性[1]；(2)超级链接导致的查询回路问题。例如有两个 Web 页面 A 和 B，A 通过自己页面的超级链接可以访问 B，同样 B 通过自己页面的超级链接也可以访问 A。当查询涉及到与 A、B 两个页面内容都相关的信息时，就会出现 A 调用 B，而 B 又调用 A 的情况。这种在两个或者多个 Web 页面之间的超级链接所形成的回路，在查询时会形成查询回路，严重影响查询的效率，同时造成查询结果大量重复。目前这种现象非常普遍，如使用 Google、百度等查询引擎执行查询，经常会得到大量的重复指向同一 Web 页面的结果。

针对现有 Web 数据模型的不足，本文提出了一种新的基于 XML 的 Web 数据模型(XWDM)。XWDM 建立在 XQuery 1.0 和 XPath 2.0 数据模型[6]基础之上，并根据 Web 数据的实际特点，加以裁减和补充，去掉了在 Web 数据中所没有的注释(comment)和处理指令(PI)，增加了元素的别名属性，并专门为根节点增加了针对超级链接的约束。为元素设置别名的目的是解决信息查询中的名称异构问题，对超级链接进行约束可以解决查询中的循环回路问题。下面对 XWDM 中的主要成分进行说明。

在 XWDM 中，规定元素是 Web 数据的基本构成单位，根据元素描述内容的差异将它们划分为 4 种：根元素(doc-elem)、基元素(base-elem)、内部元素(inside-elem)、链接元素(link-elem)。

一个根元素代表一个 Web 页面，根元素的定义与 XML

文档根(root)元素的定义一致。

基元素、内部元素、链接元素都可以是根元素的子元素。内部元素是指元素的值包含子元素,且该元素自身没有指向其它根元素的指针。如果一个元素的属性中包括指向其它根元素的指针,则称该元素为链接元素。内部元素、链接元素统称为复杂元素(complex element)。基元素是不包含其它元素的元素,它的取值为所有原子类型的值。

根元素表示为六元组:

Doc-elem(Eid,Name,Alias,Value,Linklist,Attributes)

其中,Eid 代表该元素在 Web 页面中唯一的标识符;Name 是该元素的名称;Alias 是 Name 的别名属性,该属性说明除了元素名称之外,该元素可以使用的其它命名,Alias 是一个集合,内部包含对应元素可能的别名。逻辑上,对于一个元素来说有 Name∈Alias,在本模型中用来解决信息查询中的名称异构问题;Value 表示元素可能的取值,它可以表示的数据类型包括所有原子类型,还可以是其它子元素,子元素可以是基元素、内部元素、链接元素,并且子元素允许嵌套;Linklist 表示所有指向该元素(包括它的子元素)的 Eid;Attributes 是属性-值对列表,表示元素属性的取值。

基元素、内部元素、链接元素均表示为五元组:

(Eid,Name,Alias,Value,Attributes)

其中,基元素的 Value 取值只能是原子类型,Attributes 不能为 IDREF 或者 IDREFS。

3.1 XWDM 图

XWDM 使用图模型对数据进行描述,XWDM 图是带根有向图,节点是模型的基本构成单元,下面给出 XWDM 图相关定义。

定义 1(图) 一个图 G 包含一个节点集合 V 和一个边集合 E,记为 $G(V,E)$。如果 $e\in E$,称 e 为 E 的一条边,则必有两个节点 $i,j\in V$,使得 i,j 为 e 的两个顶点,记为 $e=\langle i,j\rangle$,称 e 为连接节点 i 和节点 j 的边。如果 e 为有向边,则$\langle i,j\rangle$为有序节点序偶,称节点 i 为 e 的源节点,记作 Source(e),节点 j 为 e 的目标节点,记作 Target(e)。如果 E 中有一条以上的有向边,则图 G 为有向图。

定义 2(路径) 图 G 中的一条路径 P 是边的有序序列 $e_1e_2\cdots e_k$,且满足条件 Target(e_i)= Source(e_{i+1}),$1\leqslant i\leqslant k-1$。称 P 是一条从源节点 Source(e_1)到目标节点 Target(e_k)的路径,路径 P 中边的数目 k 称为该路径的长度,用 $d(P)$ 表示路径 P 的长度。如果一条路径的起点和终点相同,则这种路径称为圈。不含圈的图称为无圈图。

定义 3(节点的度) 在一个图中,与一个节点连接的边数称为该节点的度。在有向图中,对于任意节点 v,以 v 为目标节点的边数,称为节点 v 的入度;以 v 为源节点的边数,称为节点 v 的出度。

定义 4(带根连通有向图) 给定有向图 $G=(V,E)$,设 $v_0,v_1,\cdots,v_i,\cdots,v_n\in V,e_0,e_1,\cdots,e_n\in E$,称 G 为带根连通有向图,表示为 $G(V,E,r)$是指,若存在一个节点 r 同时满足以下条件:(1)该点的入度为 0;(2)$d(e_r\cdots e_i)>0$,其中 Source(e_r)$=r$,Target(e_i)$=i,r\in V,v_i\in V-\{r\},i=0,1,\cdots,n$。其中节点 r 称为根节点。除根节点之外,度数为 1 的节点称为叶节点,度数大于 1 的节点称为分枝节点或内部节点。

定义 5(树) 带根连通有向无圈图称为树,用 $T(V,E,r)$ 表示,其中 V 表示树中节点集合,E 表示树中边的集合,r 表示树的根(节点)。$T(V,E,r)$可以简单表示为树 T。在树 T 中,对于与任意一个节点 v 直接连接的边 e 来说,如果 Source(e)$=u$,Target(e)$=v$,则称 u 是 v 的父节点,称 v 是 u 的孩子节点或者子节点;从根节点到 v 的父节点的路径上经过的所有节点,统称为 v 的祖先节点;从 v 的所有子节点出发,到达叶节点经过的路径上的所有节点,统称称为 v 的子孙节点或者后继节点。如果某个节点的子节点之间有顺序,则称树 T 为有序树。

定义 6(XWDM 图) 一个 Web 页面可以用一个 XWDM 图表示,一个 XWDM 图是一个带标记的树 T_{xwdm}:

$$T_{xwdm}=(V_T,E_T,root,V_{leaf},V_{in},V_{out},L)$$

其中,V_T 是树中所有节点的集合,节点包括:根节点($root$)、内部节点集合 V_{in}、链接节点集合 V_{out}、值节点集合 V_{leaf};即 $V_T=\{root\}\cup V_{leaf}\cup V_{in}\cup V_{out}$,$V_{leaf}$ 包括 *Web* 页面中的所有原子值,即 *Base-elem* 的值和属性值,每个节点都根据其在数据中出现的顺序被赋予一个唯一的节点标记符 nid;E_T 是图中边的集合,每条边上都有标记,表示边所指向的节点的名称:元素名或者属性名。对于任意的 $e\in E_T$,用 $lab(e)$表示边 e 的标记。

为便于 XWDM 图对 XML 数据的表达,在上述定义的基础上,对 XWDM 图的表示进行如下规定:

规定 1 为了将属性名称与元素名称区分开来,在 XWDM 图中,在所有连接属性节点的边标记前面附加标记@。

规定 2 在 XWDM 图中,根节点有唯一的一条入边,该边的标记表示根节点的名称。

规定 3 在 XWDM 图中用有向虚边表示元素的链接属性与链接的目标元素间的引用关系。

规定 4 在 XWDM 图中,规定同层元素节点的顺序为从左到右的顺序依次排列。

根据上述定义和规定,有以下结论:

结论 1 在 XWDM 图中,内部节点可以有多条出边,但只能有唯一的入边,表明该节点仅能有一个父节点,反映了 XML 文档严格的层次嵌套关系。

结论 2 在 XWDM 图中,链接节点有一个用虚线表示的出边,表明该节点的父节点为链接元素。

3.2 XWDM 与 XML 文档的匹配

根据上述定义和结论,可得出 XWDM 与 XML 文档匹配的关系:

(1)XML 文档根元素对应 XWDM 的根元素;XML 文档中的元素可以用 XWDM 的内部或者链接元素表达,XWDM 的内部元素与子元素的关系反映了 XML 的子元素嵌套关系。

(2)XML 文档元素的多个属性-值的表达:在 XWDM 图中用带标记@的有向边和该边所指向的节点表示,有向边的标识名代表属性名,节点旁边的文本代表属性的值。

(3)XML 元素之间的顺序:在 XWDM 图中规定各节点按照从左到右的顺序作为相应的 XML 元素之间的顺序。

(4)在 XWDM 图中,通过标识元素名称的边所指向的节点表达 XML 元素的内容。

(5)XML 的链接引用存在两种方法:一种是内部指针引用,即通过 ID 与 IDREF(S)实现;另一种是外部链接,即通过 Xlink、Xpointer 实现。在 XWDM 中,为了描述简便进行了简化,统一采用 ID 与 IDREF(S)实现,链接引用的属性名均为 IDREF(S)类型,在 XWDM 图中通过有向虚边标识链接引

用。

下面是对 XWDM 核心成分采用 EBNF 的描述：

```
Node::=Element | Attribute | Text
Element::=Doc-elem | Base-elem | Inside-elem | Link-elem
Doc-elem::=(Eid,Name,Alias,Value,Linklist,Attribute)
Eid::='&'[0-9]+
Name::=QName
Alias::="Alias="Enumeration
Value::=Base-elem | Inside-elem | Link-elem | Text
Text::=anyAtomicType | untypedAtomic
Linklist::="("[Source_ID";"]+ ")"
Source_ID::=Eid
Attributes::=[AttType "="Attributevalue]+
AttType::=StringType|TokenizedType|EnumeratedType
StringType::="CDATA"
TokenizedType::="ID"|"IDREF"|"IDREFS"|"ENTITY"| "ENTITIES"| "NMTOKEN"| "NMTOKENS"
EnumeratedType::=NotationType| Enumeration
NotationType::="NOTATION"S"("S? Name(S? "|"S? Name )* S? ")"
Enumeration::="("S? Nmtoken(S? "|"S? Nmtoken)* S? ")"
Attributevalue::=anyAtomicType | untypedAtomic
Base-elem::=(Eid,Name,Alias,Value,Attributes)
Inside-elem::=(Eid,Name,Alias,Value,Attributes)
Link-elem::=(Eid,Name,Alias,Value,Attributes)
```

说明：

上述描述中 Base-elem 与 Inside-elem、Link-elem 定义方式相同，但所包含的 value、attributes 存在差异：Base-elem 的 value、attributes 取值为原子类型；Inside-elem 的 value 取值可以为 Base-elem、Inside-elem 或者 Link-elem，attributes 取值为 IDREF、IDREFS 类型以外的原子类型；Link-elem 的 value 取值为原子类型，attributes 类型只能为 IDREF 或者 IDREFS。QName 的详细定义参见文[17]。为了描述简便，对内部链接和外部链接不做区分，统一采用 IDREF 和 IDREFS 加以标识。

在上述定义中，将 Inside-elem、Link-elem 加以区分是为了支持 XML 所特有的两种模式：语义模式和文法模式。采用上述元素划分方法可以根据实际应用需求把 Web 数据看成是两种模式之一。当希望把 Web 数据看成是一个相互连接的图时使用语义模式，在表示语义模式的图中可以省略 IDREF 和 IDREFS 类型，此时 Inside-elem 与 Link-elem 的边之间没有区别；当希望把 Web 数据看成是一个 XML 文档时，可以使用文法模式。在文法模式中 Web 数据被描述成一棵树。IDREF 和 IDREFS 属性作为文本串是可见的，而它们对应的链接边不可见。

在上述定义的基础上，采用如下方式避免查询回路问题：在基于路径表达式的查询中，如果遇到从 A 页面到 B 页面的查询，即存在从 A 页面中的元素 x 到 B 页面的超级链接，则在 A 页面的根元素的 Linklist 中查找是否有与 B 页面根元素的 eid 相同的值。如果没有，则说明查询没有构成循环回路。先将 B 页面(根元素)的 eid 存放在 A 页面的根元素的 Linklist 中，之后执行对 B 页面的查询。如果在 A 页面的根元素的 Linklist 中有与 B 页面根元素的 eid 相同的值存在，说明已经对 B 页面执行过查询。查询终止，避免重复查询，这样就避免了循环回路。

在对元素名称进行查询时，不仅比较元素的 name，如果查到的元素 name 与查询的名称不符，则进一步对该元素的别名列表进行比较，看是否有与查询相符的名称。

为了实现上述对元素名称的判别和基于路径的查询中回路的判别，本模型在 XQuery 1.0[17] 所提供的查询功能基础上，增加了如下两个测试函数：

ElementNameTest::= Boolean (1)

ReflinkTest::= Boolean (2)

函数(1)在查询时对访问的元素名称以及该元素的别名列表进行比较，判断是否与查询的名称一致。如果访问的元素名称或者别名列表元素与查询的名称相同，则返回真值，查询继续。否则，返回假值，查询停止。该函数与元素的别名属性配合使用来解决名称异构问题。

函数(2)用于执行带链接的路径查询过程中。当查询的当前元素的下一个元素为链接元素时，则调用此函数。查看当前元素的根元素的 Linklist 中是否包含有链接元素的 eid；有，则说明查询已经构成循环，查询终止；没有，则在当前元素的根元素的 Linklist 中添加链接元素的 eid，继续执行查询。

实际上，上述 EBNF 的描述相当于 XWDM 所表达的 XML 文档的 DTD，从而保证了在该模型基础上所表达的 XML 文档的一致性和有效性。

4 XWDM 应用举例

下面通过两个简单的例子说明 XWDM 与其它模型的区别。

例 2 假设要从 Web 数据中查找作者 Author1 出版的书籍名称。已知有两个 Web 页面分别包含下面两段用 XML 描述的与 Author1 有关的数据片段。

```
<! --doc1.xml-->
<Publications>
  <Book Year="2001">
    <Title>Title1</Title>
    <Author>Author1</Author>
    <Author>Author2</ Author>
      ……
  </Book>
  ……
</ Publications>
<! --doc2.xml-->
<Publications>
  <Book>
    <Title>Title2</Title>
    <Writer>Author1</Writer >
    ……
  </Book>
  ……
</Publications>
```

上述问题所对应的 XQuery 查询语句为：

```
for $b_title in document("*")//Publications/Book/Title where
document("*")//Publications/Book/ Author/ text()
    ="Author1"
return $b_title
```

由于两个文档中对作者采用了不同的名称标识(Author 和 Writer)，采用 XQuery 1.0 和 XPath 2.0 数据模型执行上述查询得到的结果为：

<Title>Title1</Title>

显然查询结果不完全。

在 XWDM 中上述两个文档有如下的形式：

```
<! --doc1.xml-->
<Publications>
  <Book Year="2001">
    <Title>Title1</Title>
    <Author Alias="Writer">Author1</Author>
    <Author Alias="Writer">Author2</Author>
      ……
  </Book>
  ……
</ Publications>
<! --doc2.xml-->
<Publications>
  <Book>
    <Title>Title2</Title>
    <Writer Alias="Author">Author1</Writer >
    ……
  </Book>
  ……
</Publications>
```

上述两个文档片段对应的 XWDM 图如图 2 所示。从图 2(b)中可以清楚看到<Writer>的别名为<Author>，即两个文

档对应的元素名称〈Author〉和〈Writer〉是等价的。

说明:为了使图示更加清晰,在图 2(a)中省略了 Author 的别名标识。

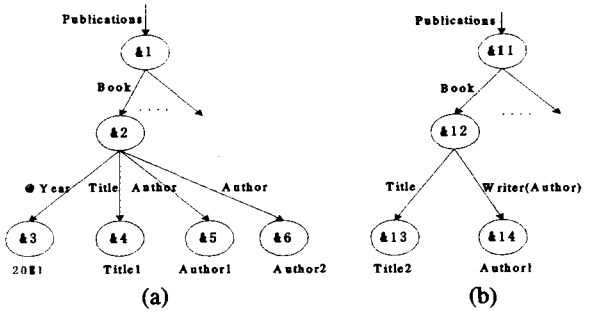

图 2　例 2 中文档对应的 XWDM 图

实际上,在 XWDM 中所对应的 XQuery 查询语句相当于执行了两个查询:

```
for $b_title in document("*")//Publications/Book/Title where
document("*")//Publications/Book/Author/text()
    ="Author1"
return $b_title
```

和:

```
for $b_title in document("*")//Publications/Book/Title where
document("*")//Publications/Book/Writer /text()
    ="Author1"
return $b_title
```

采用 XWDM 执行上述查询得到的结果为:

〈Title〉Title1〈/Title〉

〈Title〉Title2〈/Title〉

这个例子说明 XWDM 如何解决名称异构问题。

例 3　执行路径查询 select //B/C/E/F/B,已知与该查询相关的文档片段如下:

```
〈! –doc3. xml–〉〈P〉
    〈B id="b1"〉
      〈C idref="e1"〉
        ……
      〈 /C〉
      ……
    〈 /B〉
    〈/P〉
      〈! –doc4. xml–〉
      〈 E id="e1"〉
          〈F idref="b1"〉
          ……
          〈/F〉
      ……
    〈/E〉
```

文档中 id 的属性类型为 ID,idref 的属性类型为 IDREF,这两个文档之间通过链接构成了循环回路 B→C→E→F→B。对上述查询采用 XQuery 1. 0 和 XPath 2. 0 数据模型可以得到查询结果,对应 doc3. xml 中的元素:

〈B id="b1"〉……〈 /B〉

显然,这个结果不符合用户的实际需要。

根据 XWDM,上述查询过程如下:查询首先从 doc3 开始,采用基于路径的查找方式找到第一个节点 B,接着找到 B 的子元素 C,通过 C 的链接找到 doc4 中的元素 E,此时调用 ReflinkTest()函数,为 doc4 的根元素 E 的 Linklist 属性添加链接源的根节点标识 &1,这里假定 doc3 的根元素 P 的节点标识为 &1;之后从 doc4 中的元素 E 继续查找,找到元素 F,元素 F 存在一个到 doc3 中元素 B 的链接,此时再次调用 ReflinkTest()函数,查看元素 F 的链接目标 B 所在的根节点标识 &1 是否已经在 doc4 的根元素 E 的 Linklist 中。由于 E 的 Linklist 中已经存在 B 的根节点标识 &1,说明查询形成回路,此时 ReflinkTest()函数返回值为假,查询结束,即根据 XWDM 上述查询将返回空值。

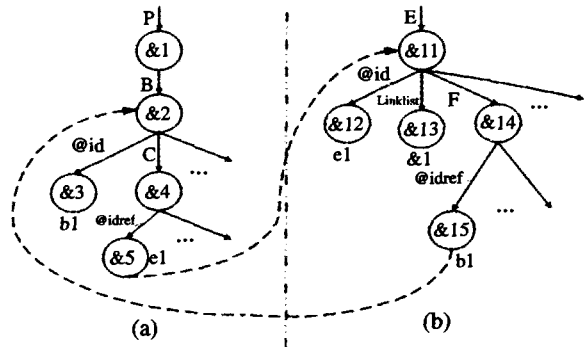

图 3　例 3 中文档对应的 XWDM 图

这个例子说明 XWDM 可以有效地处理循环回路问题。这两个文档对应的 XWDM 图如图 3 所示。图中虚线左侧对应 doc3,虚线右侧对应 doc4,图中带箭头的有向虚边与边 C(&2,&4)、@idref(&4,&5)、F(&11,&14)、@idref(&14,&15)共同组成了查询所形成的回路。

结束语　作为新一代的 Web 信息描述语言,XML 具备强大的描述能力,也使传统的 Web 技术发生了巨大变化。本文针对现有 Web 数据模型存在的名称异构和查询回路问题,在 XQuery 1. 0 and XPath 2. 0 数据模型基础上,提出了基于 XML 的 Web 数据模型——XWDM,对基于 Web 的多数据源集成中存在的名称异构问题,以及基于路径查询的回路问题,提出了解决方案。XWDM 在描述 XML 表达的 Web 数据的有效性和一致性的同时,还具备了可扩展性,为基于 XML 表达的 Web 数据查询、信息集成奠定了基础。

参 考 文 献

1　Abiteboul S, Buneman P, Suciu D. Data on the Web: from relations to semistructured data and XML. CA 94022, USA: Morgan Kauffman Publishers, 1999

2　Berglund A, Fernández M, Boag S, et al. XML Path Language (XPath) 2. 0. Editors. W3C, 4 Apr 2005. http://www. w3. org/TR/xpath20

3　Le Hors A, Le Hégaret P, Wood L, et al. Document Object Model (DOM) Level 2 Core Specification Version1. 0. W3C Recommendation 13 November, 2000. http://www. w3. org/TR/2000/REC-DOM-Level-2-Core-20001113

4　Cowan J, Tobin R. XML Information Set, W3C Working Draft 2, February 2001. http://www. w3. org/TR/xml-infoset/

5　Nagy M, Walsh N, et al. XQuery 1. 0 and XPath 2. 0 Data Model. Editors. W3C. 4, Apr 2005. http://www. w3. org/TR/xpath-datamodel/

6　World Wide Consortium. Extensible Markup Language(XML)1. 0(Third Edition). 04 Feb 2004. http://www. w3. org/TR/REC-xml/

7　OASIS. XML DTD-An Introduction to XML Document Type Definitions. http://www. xmlfiles. com/dtd/

8　Fallside D C, Walmsley P. XML Schema Part 0: Primer Second Edition. Editors. W3C, 28 Oct 2004. http://www. w3. org/TR/xmlschema-0/

9　Bray T, Hollander D, Layman A. Namespaces in XML. 14 Jan 1999. W3C Recommendation. http://www. w3. org/TR/1999/REC-xml-names-/

10　DeRose S, Maler E, Orchard D. XML Linking Language(XLink) Version 1. 0. 27 June 2001. W3C Recommendation. http://www. w3. org/TR/2001/REC-xlink/

11　Grosso P, Maler E, Marsh J, et al. XPointer Framework. 25 March 2003, W3C Recommendation. http://www. w3. org/TR/2003/REC-xptr-framework-20030325

12　Goldman R, McHugh J, Widom J. From Semistructured Data to XML: Migrating the Lore Data Model and Query Language. Stanford University, 1999

13　Boag S, Chamberlin D, Fernández F M, et. al. Query 1. 0: An XML Query Language. Editors. W3C. 4 Apr 2005. http://www. w3. org/TR/xquery