基于实体描述属性技术的 XML 重复对象检测方法

李亚坤 王宏志 高 宏 李建中
(哈尔滨工业大学计算机学院 哈尔滨 150001)

摘 要 由于 XML 文档越来越广泛地被用于信息交换与集成,其数据质量问题引起了人们的关注.解决由数据质量引发的问题,实体识别技术非常关键.当实体识别被应用于 XML 数据中时,最为关键的操作是实体数据对象的匹配.为了克服现有方法的不足,在海量 XML 数据上进行高效的重复对象检测,文中提出一种基于实体描述属性技术的高效 XML 重复数据对象检测方法.它将所有标签属性与结点统称为属性,用实体来描述属性,通过属性的属性结点表的构建,快速地找到在某个属性上相同的所有实体对象,然后比较它们是否重复.此方法的优势体现在无需比较所有实体对象,只需要比较在属性结点表中同一位置的结点,大大节省了时间.此外,我们提出的 MaxMerge 算法,在兼顾相似对象传递性与独立性的基础之上,将所有相似对象进行聚类,大大提高了算法的精确率与召回率.

关键词 XML;数据集成;数据质量;实体识别
中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2011.02131

Efficient Entity Resolution on XML Data Based on Entity-Describe-Attribute

LI Ya-Kun WANG Hong-Zhi GAO Hong LI Jian-Zhong
(*School of Computer Science, Harbin Institute of Technology, Harbin* 150001)

Abstract As being more and more widely used for data exchange and integration, the XML data quality issues cause for concern. In order to overcome the problems caused by data quality, Entity Resolution (ER) is critical. When ER is applied to XML data, the crucial operator is Object Matching (OM). To overcome the drawbacks of current methods and perform entity resolution efficiently and effectively on massive XML data set, an entity-describe-attribute (EDA) rule based object matching method is presented in this paper. Our EDA method uses entities to describe their attributes. By the construction of attribute-node table, we can compare the objects which have one or several common attributes. Then the MaxMerge algorithm is proposed. It clusters the duplication efficiently and effectively.

Keywords XML; data integration; data quality; entity resolution

1 引 言

XML 已经成为 Web 上数据交换与数据集成的主要标准,其主要原因是其良好的可扩展性,以及其结构的灵活性.由于使用日趋广泛,XML 文档的数据质量问题引起了人们的关注,如输入错误、各数据源中数据的结构差异、表达方式不同等因素令 XML

收稿日期:2011-08-29;最终修改稿收到日期:2011-09-20.本课题得到国家自然科学基金(61003046,61111130189,60933001,61033015,61133002)、国家"九七三"重点基础研究发展规划项目基金(2012CB316200)、国家博士后基金(20090450126,201003447)、教育部博士点基金(20102302120054)、哈尔滨工业大学优秀青年教师培养计划(HITQNJS.2009.052)资助.**李亚坤**,男,1988 年生,硕士研究生,主要研究方向为数据质量.E-mail: liyakun@hit.edu.cn. **王宏志**,男,1978 年生,博士,副教授,主要研究方向为数据管理、数据质量管理、XML 数据管理等. **高 宏**,女,1966 年生,博士,教授,博士生导师,主要研究领域为无线传感器网络、物联网、海量数据管理和数据挖掘. **李建中**,男,1950 年生,教授,博士生导师,主要研究领域为物联网、无线传感器网络、数据库和海量数据处理.

文档中存在着大量的数据冗余,从而对数据发布、数据交互、数据集成及数据挖掘等操作产生不良影响[1].为了克服由数据质量引发的问题,研究人员已经提出许多技术[2-5],其中实体识别起着重要作用.

实体识别是找到那些指向相同实体的数据对象.当实体识别被应用于 XML 数据时,最为关键的操作是实体数据对象的匹配.实体数据对象匹配是指判断两个数据对象是否指向同一真实世界的实体,如当两家书店合并以后,需要合并所有图书资料,可是有些图书可能分别存在于原来的两个数据源中,数据表现形式可能不同.

例 1. 图 1 是一个集成后的脏数据片段,在这个数据源中,4 个元素都指向同一实体,但它们的表现形式有很大不同.不仅数据内容不尽相同,数据组织结构也有很大差异.如:元素 1 的书名中加入了版本号;元素 2 中的作者被写到了一起,publishdate 以属性的形式给出;元素 3 中增加了版本属性、作者不全且多了一个 publish 结点;元素 4 中多了 language、page、rank 子结点.各元素的 price 中的数值也各不相同,并且还有些简单的拼写错误存在于各属性及属性值当中.

```
〈book title="Introduction to Algorithms, Third Edition"〉
        〈author〉Thomas H. Cormen〈/author〉
        〈author〉Charles E. Leiserson〈/author〉
        〈author〉Ronald L. Rivest〈/author〉
        〈author〉Clifford Stein〈/author〉
        〈ISBN〉0262033844〈/ISBN〉
        〈price〉63.8〈/price〉
        〈publisher〉The MIT Press〈/publisher〉
        〈publishdate〉2009-9-30〈/publishdate〉
〈/book〉
```

(a) 元素 1

```
〈book title="Introduction to Algorithms"〉
        〈authors〉Thomas H. Cormen, Charles E. Leiserson,
                Ronald L. Rivest〈/authors〉
        〈price〉85〈/price〉
        〈publisher publishdate="2009-9-30"〉The MIT Press
                        〈/publisher〉
        〈edition〉Third Edition〈/edition〉
〈/book〉
```

(b) 元素 2

```
〈book title="Introduction to Algorithms" edition="Third Edition"〉
        〈author〉Charles E. Leiserson〈/author〉
        〈author〉Ronald L. Rivest〈/author〉
        〈ISBN〉0262033844〈/ISBN〉
        〈price〉70〈/price〉
        〈publish〉
                〈publisher〉The MIT Press〈/publisher〉
                〈publishdate〉200909〈/publishdate〉
        〈/publish〉
〈/book〉
```

(c) 元素 3

```
〈book〉
    〈title〉Introduction to Algorithms〈/title〉
    〈author〉Thomas H. Cormen〈/author〉
    〈author〉Charles E. Leiserson〈/author〉
    〈author〉Ronald L. Rivest〈/author〉
    〈author〉Clifford Stein〈/author〉
    〈ISBN〉0262033844〈/ISBN〉
    〈price〉63.8〈/price〉
    〈publisher〉The MIT Press〈/publisher〉
    〈publishdate〉2009-9-30〈/publishdate〉
    〈language〉English〈/language〉
    〈page〉1312〈/page〉
    〈rank〉136663〈/rank〉
〈/book〉
```

(d) 元素 4

图 1 一个脏数据片段

实体识别技术的许多方法已经在关系数据上提出并使用[6-9],但是由于 XML 数据结构多样,致使这些方法并不适用.近年来,一些在 XML 数据上识别实体的方法被提出来.

第 1 种方法[10]为了解决元素多样性问题,先通过采用 XQuery 语言将结构不同的 XML 数据元素变换成统一的结构,再将同一层次的内容合并为一个元素来处理.但这种方法使原本具有不同标签的数据之间也进行相似性计算,因此得到的精度不高.

第 2 种方法[1,11]将 XML 数据抽象形成为一个树,通过计算树之间的编辑距离得出对象相似度.这种方法的时间复杂度在 $O(|T(e)|\times n^2)$ 数量级上,其中 e 为实体对象元素,n 为实体数量.若实体的描述信息庞大,这种方法的效率将会非常低.

第 3 种方法[12]将具有相似结构的 XML 元素进行合并,提出了三类启发式聚类算法,分别使用全部比较聚类、选择比较聚类、M 树聚类方法来实现重复元素的合并.主要不足在于没有解决 XML 数据结构多样性的问题.

总而言之,现有方法的缺点可以归纳为三点:第一,这些方法需要将所有实体对象一一比较,效率不高;第二,这些方法不能区别对待各个属性及结点,从而产生的精确度不高;第三,现有方法可扩展性差,不适用于海量数据.

为了克服现有方法的不足,在海量 XML 数据上进行高效的重复对象检测,我们提出一种基于实体描述属性的高效 XML 重复数据对象检测方法.这种方法将待识别元素定义为实体对象,将标签、属性和子元素都看作实体属性,用实体属性来辨识实体.通过属性结点表的构建,可以快速地找到在某个

属性上相同的所有实体对象，从而达到发现描述同一实体的不同数据对象的目的. 此方法的优势体现在仅需比较在某些属性上值相同或相似的元素，而无需比较所有的实体对象，从而提高了效率. 这种方法还通过抽取属性与结点值降低了 XML 数据中描述实体对象结构的复杂性. 在本文的方法中，各个属性的权值根据其区分实体对象的能力确定，当得到两个候选对象后，将所有相似的属性的权值累加，若大于阈值则以二元组的形式输出. 再通过使用 MaxMerge 算法，在兼顾相似对象的传递性与独立性的基础之上，将所有相似的对象进行聚类.

本文的贡献如下：

(1) 提出了基于实体描述属性技术的高效 XML 重复对象检测算法. 两个数据对象如果描述同一个真实世界的实体，那么它们很难在所有相应属性上都不同，因此我们重点考虑在所有不同属性中有若干属性相同或相似的情况. 通过实体对象的改造、插入属性结点表、查询属性结点表进行重复对象的检测. 使用这种算法进行 XML 数据的重复对象检测架构对于删除冗余是有效且实际的.

(2) 提出了有效的实体对象改造方法. 经过改造，其结果适用于具有多种结构的 XML 文档上的实体识别.

(3) 提出了高效的用于实体识别的聚类算法 MaxMerge. 它兼顾了各个实体对象之间相似的传递性与独立性，是一种快速有效的聚类方法.

(4) 通过实验，我们证实了本文提出算法的效率和有效性，并对其参数影响进行了验证.

本文第 2 节介绍为了使输入数据满足算法要求而对数据进行的改造的方法. 基于实体描述属性的重复对象检测算法将在第 3 节提出. 为了将输出的重复对象二元组进行聚类，我们将在第 4 节提出 MaxMerge 算法，并对算法进行分析. 结论将在第 5 节给出.

2　实体对象改造

在实体识别过程中，XML 数据不规则可能会引发一系列问题. 例如，没有主键便无法直接定位到对象，两个不同的 XML 数据源中对某个属性的属性名表述可能不同，而两个相同的属性名在不同的数据源中可能代表不同的意义，对象的父级及以上的结点降低了对象的检索速度. 如在图 2 中的两个实体对象中，两个 book 路径不一致，难以唯一定位到实体对象；结点 1 中作者这一属性的属性名为 name，而在结点 2 中为 author；结点 1 中的属性名 name 与结点 2 中的属性名 name 分别代表对象的作者和书名这两个不同的属性；结点 2 中属性结点 edition 与子结点 edition 的属性值完全相同，它们对于对象的描述意义冗余；结点 1 中的父结点与结点 2 同处于一个层上，检索对象结点时不属于同一个结果集，降低了检索的效率.

```
〈books〉
    〈book title="Introduction to Algorithms "〉
        〈name〉Thomas H. Cormen〈/name〉
        〈name〉Charles E. Leiserson〈/name〉
        〈name〉Ronald L. Rivest〈/name〉
        〈name〉Clifford Stein〈/name〉
        〈ISBN〉0262033844〈/ISBN〉
        〈price〉63.8〈/price〉
        〈publisher〉The MIT Press〈/publisher〉
        〈publishdate〉2009-9-30〈/publishdate〉
    〈/book〉
〈/books〉

〈book name="Introduction to Algorithms" edition="Third Edition"〉
        〈author〉Charles E. Leiserson〈/author〉
        〈author〉Ronald L. Rivest〈/author〉
        〈edition〉 Third Edition 〈/edition〉
        〈ISBN〉0262033844〈/ISBN〉
        〈price〉70〈/price〉
        〈publish〉
                〈publisher〉The MIT Press〈/publisher〉
                〈publishdate〉200909〈/publishdate〉
        〈/publish〉
〈/book〉
```

图 2　两个不规则的对象结点

为了支持有效的实体识别，需要对 XML 数据进行预处理，使其具有适用于实体识别的结构. 我们首先基于 XML 数据的特征提出了 4 条数据源改造规则. 规则 1 使对象结点检索具有唯一性；规则 2 统一了不同数据源中的属性名；规则 3 消除了冗余的属性对；规则 4 令所有实体对象结点处在数据源中的同一层上，极大地方便了对象的检索. 以下是这 4 条规则的详细描述，其中对于实体对象 a 和 b，$path(a)$ 表示对象 a 的 XQuery 路径，$height(a)$ 表示 a 到根结点的高度，$path(a)=path(b)$ 表示 a 与 b 的路径完全相同，$a=b$ 表示是同一个对象. 对于属性名 $name_a$，$mean(name_a)$ 代表 $name_a$ 所代表属性的语义，$mean(name_a)=mean(name_b)$ 表示两个属性名所代表的意义相同.

规则 1.　对于实体对象 a 和 b，$path(a)=path(b)\Rightarrow a=b$.

规则 2. 对于属性名 $name_a \in a, name_b \in b$, $name_a = name_b \Leftrightarrow mean(name_a) = mean(name_b)$.

规则 3. 对于实体对象 a, 属性对 $\langle name, value1\rangle \in a \wedge \langle name, value2\rangle \in a \Rightarrow value1 \neq value2$.

规则 4. 对于实体对象 a, $height(a) = 1$.

为了使待处理 XML 数据满足上述条件,需要对其进行预处理,该处理主要包含如下 4 个步骤.

第 1 步. 使对象具有唯一性.

由于 XML 数据中没有主键这一概念,可以存在部分甚至完全相同的实体对象,因此实体对象具有不唯一性. 而我们的算法需要将对象的唯一标识插入属性结点表,并通过这一标识来检索对象,因此需要将实体对象改造为具有唯一标识的元素. 改造的方法可以为把实体对象的标签改成字母+数字的形式,通过遍历各个实体对象结点,将各对象的标签名字中的数字部分逐一增加.

第 2 步. 统一属性名.

两个不同的 XML 数据源中对某个属性的属性名表述可能不同,而两个相同的属性名在不同的数据源中可能代表不同的意义. 我们希望通过相应属性上的属性结点表来定位实体对象,因此有必要在开始阶段将对应的属性名进行统一. 令所有代表相同意义的属性名相同,所有代表不同意义的属性名不同.

第 3 步. 消除冗余的属性对

冗余的属性对是指在一个实体对象当中属性名和属性值都完全相同的属性对. 如果存在冗余的属性对,它们对实体的描述价值可以由其中之一替代,因此我们可以直接删除冗余的属性对.

第 4 步. 使所有对象结点处于同一层上

实体对象的具有的复杂层次结构降低了对象的处理效率,我们可以将其祖先和父亲改造成为对象的子结点,表示其对对象的描述. 如果事先已经得知对象的部分或全部父级上的结点对其并无描述的意义,那么我们可以直接删去这些结点,把对象结点上提到根结点的直接子结点的位置上.

经过以上的 4 步改造,XML 数据中的对象具备了标识唯一性和属性统一性,消除了冗余属性对,且使得属性处在了同一层,从而满足了上面所提出的 4 条规则.

例 2. 下面我们使用这 4 步对图 2 中的两个结点进行预处理.

第 1 步. 将两个结点的标签分别改为 book1 和 book2,使其具有唯一性.

第 2 步. 将结点 1 中的属性名 name 改为 author,将结点 2 中的属性名 name 改为 title.

第 3 步. 删除结点 2 中的属性结点 edition.

第 4 步. 删除结点 1 的父结点,将结点 1 上提到与结点 2 同一层的位置上.

经过 4 步预处理后,改造后的两个结点为图 3 中的两个结点.

```
〈book1 title ="Introduction to Algorithms "〉
        〈author 〉Thomas H. Cormen〈/author〉
        〈author 〉Charles E. Leiserson〈/author〉
        〈author 〉Ronald L. Rivest〈/author〉
        〈author 〉Clifford Stein〈/author〉
        〈ISBN〉0262033844〈/ISBN〉
        〈price〉63.8〈/price〉
        〈publisher〉The MIT Press〈/publisher〉
        〈publishdate〉2009－9－30〈/publishdate〉
〈/book1〉
〈book2 title="Introduction to Algorithms"〉
        〈author〉Charles E. Leiserson〈/author〉
        〈author〉Ronald L. Rivest〈/author〉
        〈edition〉 Third Edition 〈/edition〉
        〈ISBN〉0262033844〈/ISBN〉
        〈price〉70〈/price〉
        〈publish〉
                〈publisher〉The MIT Press〈/publisher〉
                〈publishdate〉200909〈/publishdate〉
        〈/publish〉
〈/book2〉
```

图 3 两个规则的对象结点

3 重复对象检测

两个数据对象如果描述同一个真实世界实体,尽管它们很可能在某些属性的表现形式上有所不同,但是它们的某些属性值可能相似. 因此,我们提出了重复对象模型检测算法. 该算法按照属性组织对象,把在某个属性上相同的对象放到同一个属性结点表中的相同位置上,这样就可以通过一次对数据源的遍历操作,把各个对象的唯一标识插入到其各属性结点表中的相应位置上,然后只比较在对应属性上相同的对象. 本小节首先给出算法的基本模型,接着介绍将实体对象插入到其各个属性的属性结点表的方法(3.1 节),然后再利用已得的属性结点表进行重复对象的检测工作,得到描述同一实体的对象对(3.2 节),其中用于利用对象对求得最终实体识别结果的 MaxMerge 算法将在下一部分详细介绍.

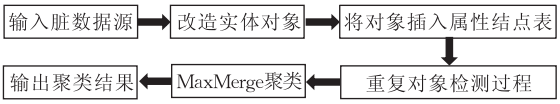

图 4　重复对象检测模型

3.1　将对象插入属性结点表

我们的方法需要按照属性组织实体对象. 因此，我们提出了属性结点表用于实现 XML 数据上的实体识别. 本小节将介绍该数据结构与其维护方法.

属性节点表分为两个部分，分别是属性名表与对象表. 首先介绍属性名表，由于我们的算法把描述实体对象的所有〈文本元素标签，文本元素值〉及〈元素属性名，属性值〉都看成属性对，从而得到的属性名的个数较多，且事先无法预知其具体数目，因此需要为所有属性名建立一个属性名表，这样我们可以通过查询此表而得知某属性是否已经出现过，并能得到它的唯一序号. 如图 5 所示，属性名表中的每个位置上的链表中的每一项包含属性名、属性序号、属性出现的次数与指向下一项的指针.

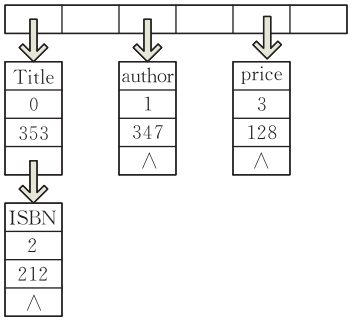

图 5　属性名表

当我们遍历到实体对象的某个属性对时，为了确定将实体对象插入到哪个对象表中，需要查询属性名表获得这一属性的序号. 首先检查一下这个属性名是否已经存在于属性名表当中，如果存在，那么返回这个属性的属性序号；如果不存在，就要把这个属性名插入到属性名表当中，并得到唯一的属性序号. 然后我们将实体对象的唯一标识插入到得到的属性序号的属性结点表中. 算法如下.

算法 1.　属性序号计算算法.

输入：所有属性名 *attrName*
输出：属性序号

```
1. while get attrName and attrValue;
2.    hash attrName into a number;
3.    number=number % attrNameIndexSize;
4.    if attrName in attrNameIndex[number]
5.       attrName.times++;
6.       get attrOrder;
7. else
8.       add attrName into attrNameIndex [number];
9.       attrName.times=1;
10. get attrOrder;
11. end if;
12. end while;
```

当我们得到属性名的属性序号后，我们就可以把当前对象的唯一标识插入对象表中相应位置上. 首先通过一个字符串散列函数把当前属性值散列成一个数值，再用这一数值模上对象表的尺寸，然后把当前对象的唯一标识插入到对象表中的这一数值的位置上. 插入算法如下.

算法 2.　实体对象唯一标识插入算法.

输入：所有属性名序号 *attrOrder* 与属性值 *attrValue*
输出：对象表

```
1. get attrOrder and attrValue;
2. if attrValue has not been inserted;
3.    hash attrValue into a number;
4.    number=number % attrValueIndexSize;
5.    add uniqueIdentifier to
6. attrValueIndex[attrOrder][number];
7. end if;
```

将所有对象的标识插入到对象表后，最后得到的对象表如图 6 所示.

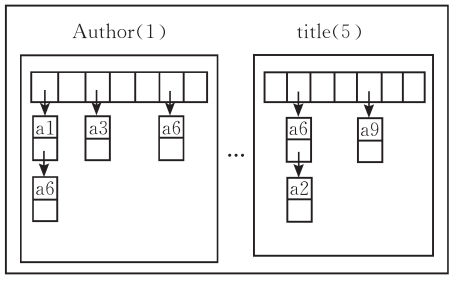

图 6　对象表

3.2　重复对象检测过程

在遍历完所有实体对象，把它们的唯一标识分别插入到其相应对象表中后，基于实体对象表进行重复对象的检测，获得可能描述同一实体的对象对. 该过程分为如下两个步骤.

3.2.1　计算各属性权值

由于不同的属性对于对象的描述重要性可能不同，为了更加精确地识别实体对象，需要根据属性描述对象的重要性赋予不同的权值. 在将所有实体对象分别插入到各个属性结点表中后，每个属性出现的次数已经被记录下来，我们认为在所有对象中出现次数较多的属性更能够描述对象. 因此我们选择出现次数最多的属性的次数 T 作为计算各个属性权值的基值，各个属性的权值(Weight)定义为其出

现次数除以基值所得的商，可表示为 $Weight_i=\frac{T_i}{T_{MAX}}$，其中 T_i 是各个属性自己出现的次数，T_{MAX} 是出现次数最多的属性的次数.

3.2.2 得到候选对象及其相似性比较

首先我们介绍一下用到的符号和缩写，如表 1 所示.

表 1 符号、缩写及其意义

符号、缩写	意义
$Similarity$	相似度
$\wedge$	求合
O_i	实体对象 i
$attr_i$	第 i 个属性
$Weight_i$	第 i 个属性的权值
K	对应相同的属性值总数
γ_i	第 i 个属性的权值累加值
$Size(O_1[attr_i])$	O_1 中 $attr_i$ 中属性值的个数
$Similar$	字符串相似性比较
M	输入的相似度阈值

为了找出数据源中的重复对象对，检测的过程如下：首先遍历第 1 个属性的属性结点表，若表中某一位置的标识数为 0 或 1，则跳过该位置，继续遍历下一位置；若此位置的标识数大于等于 2，则取出所有标识. 通过第 1 个标识，取出其所代表的节点，创建一个包含所有出现过的属性名的链表数组，将此节点的所有属性值分别插入到对应位置的链表中. 然后创建一个相同大小的链表数组，将第 2 个标识代表的节点的所有属性分别插入到此链表数组中. 开始比较之前，相似度(Similarity)为 0. 从第 1 个属性进行比较，将第 1 个对象的此属性名内的所有属性值分别与第 2 个对象的此属性名内的所有属性值进行字符串相似性(Similar)比较，由于在 XML 数据中同一属性名可能对应多个不同的属性值，而我们的算法认为只有当两个对象在同一属性对应所有属性值相等情况下，才能达到此属性原本的权值，若只有部分属性值对应相同，设此对应相同的属性值总数为 K，对象 O_1、O_2 中此属性名 $attr_i$ 下的属性值个数分别为 $Size(O_1[attr_i])$、$Size(O_2[attr_i])$，则此属性权值的累加值 γ_i 应为

$$\gamma_i=Weight_i\times\frac{K}{Size(O_1[attr_1]+Size(O_2[attr_1]))}$$

若两个对象此属性内所有值都不相同，则累加值 $\gamma=0$. 将所有 h 个属性的累加值加到一起，便可得到两个对象的相似度(Similarity)

$$Similarity=\wedge_{j\in[1,h]}\gamma$$

若两个对象的相似度大于我们输入的阈值 M，则认为它们是重复的，输出这两个对象. 该算法依次扫描所有对象表，得到所有描述同一实体的数据对象对.

例 3. 使用该算法对已经改造并插入到属性结点表后的图 1 中的数据集合进行重复对象检测过程如下.

先检测 title 属性的对象表，发现 O_2、O_3、O_4 处在此表中的同一位置上，根据其插入的唯一标识取出这 3 个结点，对 O_2 与 O_3 的所有对应属性进行相似性比较，得到其相似度大于 M，输出 (O_2,O_3)；然后以同样的方法比较 O_3 与 O_4、O_2 与 O_4，输出相似度大于 M 的对象对. 然后检测 author 属性的对象表，发现 O_1 与 O_4 同时出现在 4 个位置上，其中两个位置上还有 O_2，依次进行各个对象的两两比较，输出相似的对象对. 再检测下一个对象表直至结束，得到了所有的相似对象对.

3.3 算法分析

实体对象改造与插入属性结点表两步可以在扫描数据一次之内完成，因此在预处理时只需要扫描数据 1 次.

设数据源中实体结点的个数为 n 个，所有实体对象共有 g 个不同的属性，其中 i 个用于生成属性结点表，每个属性结点表包含 s 个元素，在遍历属性结点表的某一位置时，得到的候选对象的平均个数为 m，考虑到在真实数据中冗余的概率很低，若选取建立属性结点表的属性恰当、Hash 函数的冲突率足够低、属性结点表的尺寸足够大的话，属性结点表中的很多位置的候选对象数都为 0 或 1，考虑到在对象数为 0 或 1 的位置上不存在可以比较的候选对象对，因此计算平均候选对象时只考虑对象个数大于等于 2 的候选组. 设候选对象数大于等于 2 的位置的平均数量为 q，故有 $iqm\leq ng$，且 $m\geq 2$，$q\leq s$.

每得到一组候选对象，就要对其每个结点的所有信息进行一次扫描，抽取出其各属性值，一共需要访问 iqm 个结点. 每两个候选结点的每个对应的属性之间都需要进行一次比较操作，共需要比较的次数为 $giqC_m^2$. 设访问一个实体结点所需要的平均时间为 t_1，比较两个属性值所需要的平均时间为 t_2，则整个算法所需要的时间 T 为

$$T=nt_1+tgmt_1+giqC_m^2t_2 \quad (1)$$

由此可得在属性结点表尺寸 s 不变的条件下，候选对象数大于等于 2 的位置数量为 q 越小、平均候选对象数 m 越小，我们算法运行所需要的时间越少. 即当对象在属性结点表的冗余度越小，我们的算法的效率越高. 由 $tqm\leq ng$ 可以得到此算法运行所

需要的最大时间为

$$T_{MAX}=nt_1+ngt_1+\left(\frac{n^2g^3}{2iq}-\frac{ng^2}{2}\right)t_2$$

当结点属性表中所有位置均为 0 或 1，即没有冗余或冗余很少时 $q=0$，此时达到算法的最高效率，代入式(1)得到算法运行所需要的最小时间为 $T_{MIN}=nt_i$. 考虑到真实数据中冗余通常较低，插入到属性结点表后候选对象大于等于 2 的位置较少且平均候选对象很少，因而我们的算法在大多数情况下效率较高，算法时间复杂度是 $O(n)$.

4　MaxMerge 算法

重复对象检测工作完成以后，我们只输出了两两重复的实体对象对，而我们希望得到的结果是元素的聚类，每一类中的元素描述同一实体，因此我们需要对重复对象检测工作输出的结果进行合并，让所有相似的元素聚类到一起. 为了实现快速地合并，我们提出了 MaxMerge 算法，它兼顾了各个实体对象之间相似的传递性与独立性，是一种快速有效的聚类方法.

4.1　MaxMerge 算法模型

在该算法中，我们首先用一个实体对象信息表(infoBook)来记录所有实体对象的基本信息(info). 根据对象的唯一标识可以得到其在对象信息表中的位置(position)，如对象 a246 的信息应当插入到实体对象信息表中的第 246 的位置上. 所需要插入的对象信息如下图所示，其中含有对象的名字(name)、对象是否已经出现过(show)，对象被归并到的所有的聚类的编号(clusterNO)，及其在此聚类中出现的次数(time)，聚类的编号与其出现的次数合称为聚类信息 ClusterInfo.

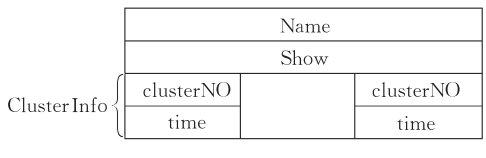

图 7　一个实体对象的信息

当算法扫描到一个二元组的实体对象(a,b)时，首先判断它们是否已经出现过. 若两个对象都未出现过，我们新建一个聚类，并把此聚类的编号存入两个对象的信息当中，此聚类出现的次数都为 1，两个对象是否出现过的标记改为 true；若两个对象中只有一个出现过，那么取出出现过的对象中出现次数最多的聚类编号，然后把它存到未出现过的对象的信息中，设置其次数为 1，把它的标记改为 true；若两个对象都出现过，则检测这两个对象中的聚类编号中是否有重复，如有重复，则增加此重复的聚类编号次数，如果没有重复的，那么把两个对象中各自出现次数最多的聚类编号增加到对方信息中，并设置其次数为 1.

在扫描完所有二元组的实体对象后，新建一个聚类表，开始遍历每个实体对象的信息，将对象插入到聚类表中自己所有聚类编号中出现次数最多的编号的位置上. 在插入聚类表完成后，输出整个聚类表作为聚类结果.

算法 3.　MaxMerge 算法.

输入：所有相似对象二元组

输出：聚类结果

```
1. while a pair of objects
2.   get nums and names from objects;
3.   if neither showed then
4.     add new cluster to item1 and item2;
5.     make both of them showed;
6.   else if just one showed then
7.     add the maxCluster of the showed to the unshowed;
8.     make the unshowed showed;
9.   else if both of them have a same cluster then
10.    the time of the same cluster++;
11.  else
12.    add the maxCluster to each other;
13.  end if;
14. end while;
15. while traverse the info_book
16.   if nameBook[i] showed then
17.     maxCluster=the MaxCluster of item;
18.     add the name to cluster_table[maxCluster];
19.   end if;
20. end while;
```

4.2　MaxMerge 算法分析

文献[4]中使用了传递闭包的方法来进行重复元素对的聚类，它仅仅考虑到了相似的传递性，如果元素 1 与元素 2、元素 2 与元素 3 重复，那么元素 1 与元素 3 必重复. 但是这种方法忽略了各相似对的判断是独立的，并且有可能由于一个相似的元素对，而导致两个指向不同实体的元素聚类聚到一起. 当重复的元素规模增大时，会导致重复检测方法的精确率变低.

在 MaxMerge 算法中，同时兼顾了相似的传递性与独立性，主要体现在如下两个方面：元素由于相似次数的增加而不断增加自己某个聚类编号的出现次数，一个元素必然与自己所属聚类中的多个元素相似，而这些相似的元素会不断增加此元素中相应

的聚类编号出现的次数，这是 MaxMerge 算法中传递性的体现；而最后把元素分别插入到各个聚类中时，仅考虑它的所有聚类编号当中出现次数最多的那个，对于元素中出现次数不多的聚类编号直接忽略，把元素放到自己最可能所属的聚类中，这是 MaxMerge 算法中独立性的体现.

在文献[4]中的传递闭包方法中，当检测到两个元素重复时，就需要将所有之前与这两个元素重复的元素聚类，所以检测是非线性的，而在 MaxMerge 算法中，当检测到两个元素重复时，只对两个元素的结构信息进行改变，无需修改之前与它们重复的元素，因此检测过程是线性的. 因此 MaxMerge 算法运行所需要的时间仅为扫描一次二元组文件与扫描一次内存中对象信息表所需要的时间.

5 实验验证

本文所提出的基于实体描述属性技术的算法可快速地找到可能相似的候选对象，然后再在候选对象中进行任意一种核心的实体识别算法即可. 本实验内采用的核心实体识别算法为一般的基于规则的实体识别算法. 为了验证本算法的有效性并测试影响效率的主要因素，我们进行了大量的实验，并对实验结果进行相关的分析. 为了验证 MaxMerge 算法的运行效率，还进行了一系列的对比实验. 实验采用的硬件环境为 Intel Pentium 4 CPU，内存 1GHz，操作系统为 Windows XP，系统代码用 Java 实现，软件开发环境为 Eclipse 3.4.1.

5.1 算法效率实验

在本实验中，我们利用数据生成系统生成 6 个含有不同数量实体对象的 XML 数据集合. 前 3 个数据集合分别为，S1 含有 4k 实体对象，S2 含有 8k 实体对象，S3 含有 16k 实体对象，这 3 个数据集合中冗余程度相近且适中. S4、S5、S6 也分别含有 4k、8k、16k 个实体对象，但数据中的冗余程度较低. 数据集合中的每个实体对象的结构中不存在孙级以下的结点. 子级、孙级结点共由 8～11 个属性组成，这些属性或以结点属性的形式给出，或以结点的子、孙结点的形式给出，其中 cluster 属性的值代表其实体对象真实归属的聚类，不参与重复对象检测过程，但用于统计精确率与召回率. 其它 7～10 个属性用来参与实体对象的重复对象检测，每个属性由 6～12 个英文字母组成，如图 8 中的两个对象结点.

```
〈book cluster="11"〉
        〈attr0〉wdwdmzgj〈/attr0〉
        〈attrs〉
            〈attr2〉swjswmqh〈/attr2〉
            〈attr3〉errztafr〈/attr3〉
            〈attr4〉tcnlnmar〈/attr4〉
            〈attr5〉kuwgxthuj〈/attr5〉
            〈attr6〉janvqq〈/attr6〉
        〈/attrs〉
        〈attr8〉pemvtkyf〈/attr8〉
        〈attr9〉ydoahbkgu〈/attr9〉
〈/book〉
〈book cluster="89"〉
        〈attr1〉kqwwpea〈/attr1〉
        〈attr2〉nexqwiog〈/attr2〉
        〈attr3〉ggnsqirl〈/attr3〉
        〈attr4〉svkmguk〈/attr4〉
        〈attr5〉bmtxlelx〈/attr5〉
        〈attrs〉
            〈attr7〉sbbfgyx〈/attr7〉
            〈attr8〉qmeahjbj〈/attr8〉
        〈/attrs〉
〈/book〉
```

图 8 数据集合中实体对象的表现形式

我们的算法运行于 4k、8k、16k 个结点的数据集合上进行重复对象检查时，所需要的时间如图 9 所示，分别为 2.58 min、11.54 min、61.55 min. 可见，当数据中冗余的对象比率大致相同时，数据集合的尺寸增大，重复检测所需要的时间显著增加. 运行结果如图 10 所示，精确率都接近 100%，召回率也都在 90%以上，这验证了算法的有效性.

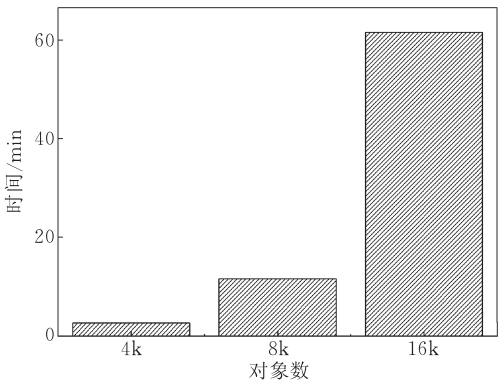

图 9 效率

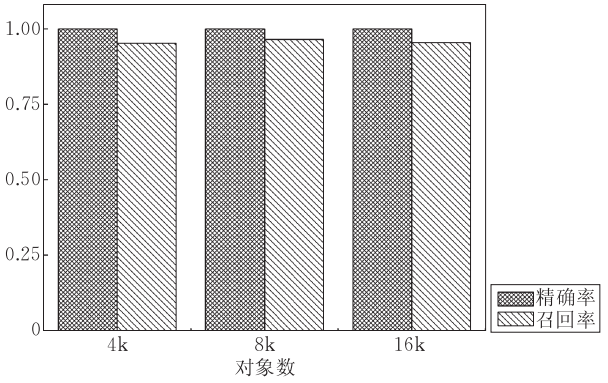

图 10 精确率与召回率

由图 11 可得，当数据的冗余度较低时，处理数据所需要的时间显著减少. 并且在低冗余度时，当处理的数据量为原来的 2 倍时，所需要的时间呈线性增长. 由此可得，算法在低冗余时的时间复杂度小于 $O(n^2)$，接近 $O(n)$，这验证了我们之前对算法的高效性分析.

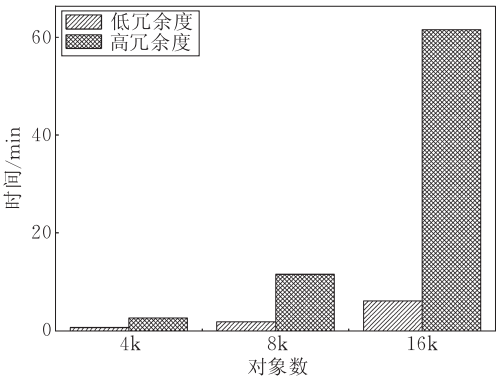

图 11　冗余度的影响

5.2　数据冗余程度的影响

为了测试数据冗余程度对算法的影响，我们生成了 10 个数据集合，其中每个数据集合都含有 4096 个对象，每个数据集合中指向相同实体的对象的集合叫做一个聚类，数据集合 $D1$ 中的所有实体对象共可以划分为 100 个聚类，$D2$ 可划分为 200 个聚类，依次类推，$D10$ 可划分为 1000 个聚类，并且在数据集合当中所有的实体对象平均分配到各个聚类当中.

算法在冗余程度不同的数据集合当中的运行效率如图 12 所示. 可以看出，算法在运行 10 个冗余程度不同但实体对象总数同为 4k 的数据集合时，聚类数量最少的 $D1$ 所需要运行的时间最多，其它数据集合所需要运行的时间随着聚类数量的增多而减少. 由于在数据集合当中所有的实体对象是平均分配到各个聚类当中的，在实体对象总数不变的前提下，聚类数量越多，算法的冗余度越低. 所以可以得数据集合中的实体对象的冗余程度越低，算法的运行效率越高. 这也验证了我们之前对算法进行的分析.

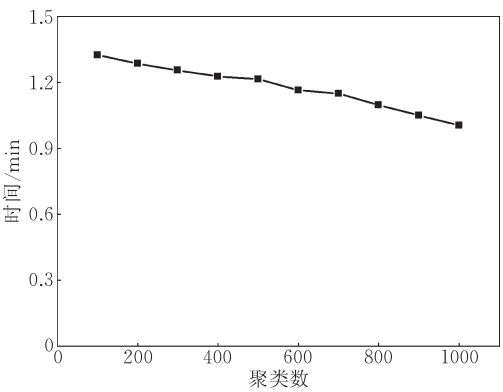

图 12　不同冗余程度下的效率图

图 13 是算法在冗余程度不同的数据集合当中运行后的精确率与召回率. 易看出，算法的精确率一直稳定在将近 100%，而召回率也一直浮动于 80% 以上，召回率的浮动与数据集合中数据本身有关. 召回率与精确率都很高，这验证了算法的有效性.

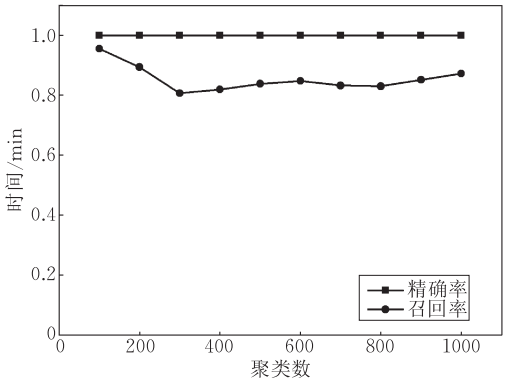

图 13　不同冗余程度下的精确率与召回率

5.3　数据的结构复杂度的影响

我们所测试的是 3 个除平均深度不同外，叶结点与属性结点互相完全相同的含有 4k 个结点的数据集合. 也就是说这 3 个数据集合 $S1$、$S2$、$S3$ 只是从对象结点到叶结点的路径长度不同，其中 $S1$、$S2$、$S3$ 到叶结点的平均路径长度分别为 1、10、20.

由图 14 可以看出，当实体对象的结构变得复杂时，算法运行的时间明显增多，这表明算法执行过程中的相当大的一部分时间被用来遍历描述实体的结点与属性，对于结构越简单的实体，算法的执行效率越高.

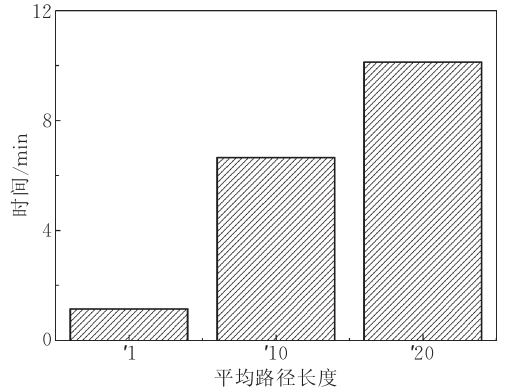

图 14　数据的结构对算法的影响

5.4　MaxMerge 算法的对比实验

表 2 是 MaxMerge 算法与文献[4]中的传递闭

包算法的对比实验结果.分别使用两个算法处理数据总量为 1k、2k、4k 的数据集合,得到的结果如下.容易看出,MaxMerge 算法所需要的时间都略低于传递闭包方法,并且精确率与召回率都有显著的差距.由于兼顾了相似的传递性与独立性,MaxMerge 算法是一种高效且有效的方法.

表 2 对比实验结果

数据量	MaxMerge 算法			传递闭包方法		
	时间/s	精确率	召回率	召回率	时间/s	精确率
1K	0.281	1	0.94573	0.328	0.85110	0.69994
2K	0.471	1	0.90221	0.473	0.71701	0.48510
4K	0.620	1	0.94614	0.631	0.83303	0.69661

实验结果总结:本文提出的算法在 XML 数据集合当中可以快速有效地检测重复实体,且当数据集合中冗余的程度很低时,算法的时间复杂度接近于 $O(n)$.算法运行中的大部分时间被用来遍历对象的结点与属性,当描述实体的 XML 对象的数据结构很简单时,算法的运行效率会很高.用于对重复的数据对象进行聚类的 MaxMerge 算法,由于其扫描时是线性的,所花费的时间要略小于传递闭包方法.由于其兼顾了相似的传递性与独立性,其精确率与召回率都显著的优于传递闭包方法,是一种高效精确的聚类方法.

6 结 论

本文提出的使用实体描述属性的 XML 数据对象实体识别方法,通过使用属性结点表快速定位在某个属性上相同的实体对象,大大减少了所需要的比较次数,再通过使用 MaxMerge 算法对输出的重复对象二元组进行聚类,同时考虑了相似的传递性与独立性,显著提升了算法的精确率与召回率.大量的实验验证了我们的方法是高效且有效的.

参 考 文 献

[1] Hernandez M A, Stolfo S J. Real-world data is dirty: data cleansing and the merge/purge problem. Data Mining and Knowledge Discovery, 1988, 2(1): 9-37

[2] Hassanzadeh O, Sadoghi M, Miller R J. Accuracy of approximate string joins using grams//Proceedings of the International Workshop on Quality in Databases (QDB). Vienna, Austria, 2007: 11-18

[3] Hassanzadeh O. Benchmarking declarative approximate selection predicates[Ph. D. dissertation]. University of Toronto, Canada, 2007

[4] Whang Steven Euijong, Menestrina David, Koutrika Georgia et al. Entity resolution with iterative blocking//Proceedings of the 35th SIGMOD International Conference on Management of Data. Rhode Island, USA, 2009: 219-231

[5] Weis M, Naumann F. Detecting duplicate objects in XML documents//Proceedings of the IQIS. Paris, France, 2004: 10-19

[6] Weis Georgia, Naumann Felix. DogmatiX tracks down duplicates in XML//Proceedings of the ACM SIGMOD 2005. New York, USA, 2005: 431-442

[7] Pluempitiwiriyawej C, Hammer J. Element matching across data-oriented XML sources using a multi-strategy clustering model. Data & Knowledge Engineering, 2004, 48(3): 297-333

[8] Wang Tian-Liang, Chen Gang, Xu Hong-Bing. Duplicates detection in XML based on object tree similarly match. Computer Science, 2006, 33(11): 162-166(in Chinese)
(王天亮,陈刚,徐宏炳. 基于对象树相似匹配的 XML 重复对象检测. 计算机科学, 2006, 33(11): 162-166)

[9] Karr A F. Exploratory data mining and data cleaning. Journal of the American Statistical Association, 2006, 101(473): 399-399

[10] Low Wai Lup, Lee Mong Li, Ling Tok Wang. A knowledge-based approach for duplicate elimination in data cleaning. Information Systems, 2001, 26(8): 585-606

[11] Marcel W, Roberto R. Efficient topology-aware overlay network. ACM SIGCOMM Computer Communication Review, 2003, 33(1): 101-106

[12] Aebi D, Perrochon L. Towards improving data quality//Proceedings of the International Conference on Information Systems and Management of Data. Delhi, India, 1993: 273-281

LI Ya-Kun, born in 1988, M. S. candidate. His research interests focus on data quality.

WANG Hong-Zhi, born in 1978, associate professor, Ph. D.. His research interests include data quality, XML data management.

GAO Hong, born in 1966, Ph. D., professor, Ph. D. supervisor. Her research interests include wireless sensor networks, cyber-physical systems, massive data management and data mining.

LI Jian-Zhong, born in 1950, professor, Ph. D. supervisor. His research interests include wireless sensor networks, cyper-physical systems, database, massive data processing etc.

Background

XML has become the main criteria of the Web data exchange and integration, mainly because of its good extensibility as well as the spiritual nature of its structure. As more and more widely the XML data has been used, the data quality issues, such as input errors, the structural diversity, and the differences in expression, cause for concern. Thus the data operations, such as data release, data exchange, data integration and data mining, will be profoundly affected, even misleading. It is necessary to process the dirty XML data and many techniques have been proposed. Among them, entity resolution is very important. It is useful in inconsistency and inaccuracy discovery as well as duplication detection.

Entity resolution (ER) is to find the data objects referring to the same real-world entity. When ER is performed on XML data, the crucial operator is object matching (OM), which is to judge whether two objects refer to the same real-world entity.

To overcome the drawbacks of current methods and perform entity resolution efficiently and effectively on massive XML data set, an entity-describe-attribute (EDA) rule based object matching method is presented in this paper. Our EDA method use entities to describe their attributes, by the construction of attribute-node table, we can compare the objects which have some specific attributes alike. And by the Max-Merge algorithm, the recall rate and precision of the identification are both greatly improved.

This research is partially supported by National Natural Science Foundation of China under Grant Nos. 61003046, 61111130189. Key Program of the National Natural Science Foundation of China under Grant Nos. 60933001, 61033015, 61133002. The National Basic Research Program (973 Program) of China under Grant No. 2012CB316200. National Postdoctoral Foundation of China under Grant Nos. 20090450126, 201003447, Doctoral Fund of Ministry of Education of China under Grant No. 20102302120054. Development Program for Outstanding Young Teachers in Harbin Institute of Technology under Grant No. HITQNJS. 2009. 052.

Our group has been focusing on the research of database for a long time. Many outstanding works have been published in worldwide conferences and transactions, such as SIGMOD, VLDB, ICDE, KDD, INFOCOM, TKDE, VLDB Journal et al. This paper proposes a novel way on matching objects in XML fragment. It gives a solution on entity resolution on massive XML data.