文章编号:1001 - 9081(2010)11 - 2941 - 04

数据集成中 XML Schema 到关系模式的转换方法

聂　玲,刘　波
(暨南大学 信息科学技术学院, 广州 510632)
(ddxllb@163.com)

摘　要:根据 XML Schema 中组件的定义及组件之间的嵌套关系,建立一系列从 XML Schema 转换成关系模式的结构映射规则和语义映射规则。基于这些规则实现一种转换算法,从 Schema 中提取出关系模式,并且证明映射得到的关系模式满足 4NF。结果表明得到的关系模式不仅包含了 XML Schema 中所有的结构和内容信息,还能保留大部分语义约束信息,减少存储冗余。

关键词:XML schema;嵌套关系;映射规则;关系模式;语义约束
中图分类号: TP311.13　　**文献标志码**:A

Transformable method for XML Schema to relational schemata in data integration

NIE Ling, LIU Bo
(*College of Information Science and Technology, Jinan University, Guangzhou Guangdong* 510632, *China*)

Abstract: This paper built a set of structure mapping rules and semantic mapping rules according to the definition of elements and the nested relation between elements in XML schema. A new mapping method based on these mapping rules was proposed to transform XML Schema to relational schemata. Furthermore, the relational schemata mapped from XML Schema were proved in 4NF. The results indicate that the relational schemata not only contain all the information of structure and content in XML Schema, but also preserve most of the semantic constraints and reduce storage redundancy.

Key words: XML schema; nested relation; mapping rule; relational schema; semantic constraint

0　引言

目前在各类信息系统中,大部分数据都是存储在关系数据库中,而 XML 因其具有良好的数据存储格式、可扩展性、自描述性、半结构化、便于网络传输等特性,逐渐成为了 Internet 上主要的数据表示和交换标准之一。所以,将表达半结构化信息的 XML 文档及描述其数据结构的 XML Schema 或 DTD,转换为关系数据库的结构化数据和关系模式,充分利用关系数据库的成熟技术解决 XML 数据的存储和查询问题是很有必要的。

Lee 等人在 XML Schema 的语义约束特征[1]、XML 模式描述语言比较方面[2-3]进行了大量研究,提出了面向数据结构的 XML Schema 到关系模式的映射算法[4],但该算法尚不能实现 XML Schema 语义约束的保留。由于 XML Schema 和关系模式表示结构关系、语义约束的方式不同,尤其是 XML 语言具有很强的自由度和表现力,要提取、映射、转换其语义约束信息往往比较复杂,造成现有的 XML Schema 到关系模式转换的映射算法很多仅仅考虑了数据结构信息的转换,忽略了数据语义约束的保留,保留约束条件的模式转换尚处于研究阶段[5-6],还没有一种非常完善的方法。文献[7]给出了一种满足了约束控制的转换方法,阐述了 XML Schema 中各种元素的映射规则,考虑了大部分语义约束信息的映射,但是没有考虑具有一致性约束的元素的映射,并且没有给出具体实现算法。文献[8]首先解析 XML Schema 文档,根据解析得到的 DOM 树建立节点树,然后通过判断树中的节点哪些映射为表,哪些映射为属性列,动态生成相应数据库中的表,但没有考虑 list 类型元素、mixed = "true" 的元素以及具有一致性约束的元素的映射。

针对以上问题,本文建立了一套转换 XML Schema 到关系模式的映射规则,其中对 list 简单元素和 mixed = "true" 的复杂元素都给出了比较详细的映射方法,考虑了数据类型约束、阈值约束、元素或属性默认值和固定值约束、元素出现次数约束、一致性约束等语义约束问题,考虑了元素对全局元素或全局类型的引用,转换过程中尽量保证 XML Schema 内容和结构信息的完整性,保持函数依赖,消除冗余转换,在按照元素之间的嵌套引用关系生成关系模式的整个过程中考虑关系模式的规范化,使得转换得到的关系模式满足 4NF。

1　XML Schema 相关概念

1.1　XML Schema 的构成组件[9]

XML Schema 用来描述 XML 文档的结构、内容和限制,由元素、属性、简单类型、复杂类型、属性组、一致性约束等一系列组件构成,组件可以被声明为全局的或是局部的,全局组件可以被其他同类型组件引用,局部组件则局限在包含它们的定义和声明作用域内。

XML Schema 中数据类型分为两类:一类是简单数据类型,包括内置的 44 种数据类型,还可以应用 XML Schema 的 12 个面来指定一个有效的值范围、约束值的长度和精度、枚举一系列有效值或指定有效值必须匹配的正则表达式,从而限制简单类型派生出新的简单类型。简单数据类型只包含字

收稿日期:2010 - 04 - 22;**修回日期**:2010 - 07 - 02。
基金项目:广东省科技计划项目(2010B010600026);广东省教育部产学研结合项目(2007B090400095)。
作者简介:聂玲(1986 -),女,湖南涟源人,硕士研究生,主要研究方向:高性能数据库;　刘波(1965 -),女,湖南长沙人,副教授,主要研究方向:数据库、信息集成、数据挖掘。

符数据内容,不包含子元素或属性。另一类是复杂数据类型,可以包含字符数据内容,至少要包含子元素或属性中的一种,并用 minOccurs 和 maxOccurs 表示子元素出现的次数。

1.2 XML Schema 中关系的定义[7]

XML Schema 中最明显的关系为嵌套关系,即一个元素中嵌套其他元素或属性,而属性不会嵌套其他元素或属性,另外还包括对全局组件的引用。这种嵌套关系具体表现为两种:关联依赖关系和值依赖关系。

如果存在以下几种情况,称为关联依赖关系。

1) 元素 e_i 内部定义了元素 e_j,则 e_j 函数依赖于 e_i;

2) 元素 e_i 内部定义了属性 a_i,则 a_i 函数依赖于 e_i;

3) 元素 e_i 内部的元素引用了元素 e_j 的定义,则 e_j 函数依赖于 e_i;

4) 元素 e_i 内部的属性引用了属性 a_i 的定义,则 a_i 函数依赖于 e_i;

5) 元素 e_i 的 type 属性的值是全局类型 t_i,则 t_i 中定义的元素和属性函数依赖于 e_i。

值依赖关系是描述元素中嵌套的出现次数不同的子元素或属性与元素之间的关系,分为单值依赖、多值依赖和可控依赖。

1) 元素 e_i 内部定义或引用了 maxOccurs = "1" 的元素 e_j,则 e_j 单值依赖于 e_i;

2) 元素 e_i 内部定义或引用了 maxOccurs = "unbounded" 的元素 e_j,则 e_j 多值依赖于 e_i;

3) 元素 e_i 内部定义或引用了 minOccurs = "0" 的元素 e_j 或者定义或引用了 use = "optional" 的属性 a_i,则 e_j 和 a_i 可以出现,也可以不出现,它们与元素 e_i 之间是可控关系。

2 XML Schema 转换成关系模式的基本映射规则

目前,主要存在两种基于 XML Schema 的映射算法,可分别称之为对象关系映射算法和直接映射算法。对象映射算法是应用比较广泛的一种算法,先将 XML Schema 映射为对象模式,然后再将对象模式映射为关系模式,例如开源 Java 项目 castor 可以实现从 XML Schema 到 Java 对象的映射以及 Java 对象到关系数据库表的映射。

本文采用的是直接映射算法,其主要设计思想是建立一系列映射规则,将 Schema 中定义的对象(元素、属性、元素间的嵌套关系等)映射为关系模式的相应对象(表、字段、主键、外键)。映射可分为两部分:结构映射和语义映射,结构映射主要是 XML Schema 中的元素映射为关系模式中的表,其子元素和属性映射为关系表的字段,语义映射主要包括 XML Schema 中元素之间的关系用表与表之间的外键进行关联、一致性约束(包括唯一性约束、关键字约束和关键字引用)的映射、数据类型的映射、默认值和数据长度限制的映射等[8]。

2.1 结构映射规则

结构映射主要遵循以下映射规则:

1)复杂类型元素直接生成带有主键的表。如果复杂类型元素声明中定义了一个 type = "xsd:ID" 的属性,则该属性可映射为表的主键,不用另外设置主键,否则可用自动增加的整数作为表的主键;如果定义了 type = "xsd:IDREF" 的属性,则该属性映射为引用 ID 类型字段值的外键字段;复杂类型元素声明中的其他属性或者简单类型且只出现一次的元素生成表的字段,并引用复杂类型子元素的主键字段值作为表的外键字段值。

2)复杂类型元素声明中定义或引用的,可以出现多次的简单类型元素生成带有外键字段的表,外键字段引用其父元素映射表的主键字段值。

3)复杂类型元素声明中定义或引用的 list 类型元素生成一个新表,添加一个类型为自动增加的整数的字段,并生成一个引用其父元素映射生成关系模式的主键字段值的外键字段,这两个字段一起构成该表的主键字段。定义了 type = "IDREFS"、type = "ENTITIES" 或 type = "NMTOKENS" 的属性按照相同的映射规则处理。如果父元素的 maxOccurs = "unbounded",此时父元素与 list 类型子元素之间是多对多的关系,需再生成一个表,表的字段分别引用父元素和 list 类型子元素生成表的主键字段值。

4)对于 mixed = "true" 的复杂类型元素,除该元素需要生成表以外,还需另外生成一个表来存储字符数据内容,添加一个类型为自动增加的整数的字段,用来描述字符数据内容在元素中的顺序,并生成一个引用其父元素生成关系模式的主键字段值的外键字段,这两个字段一起构成该表的主键字段,还包括一个字段对应字符数据内容[8]。如果元素的 maxOccurs = "unbounded",再生成一个新表,表的字段分别引用元素生成的表和存储字符数据内容的表的主键字段值。

5) 如果元素 e_j 单值依赖于元素 e_i,而元素 e_k 多值依赖于元素 e_j,则 e_k 生成带有外键字段的表,外键字段引用 e_i 映射表的主键字段值。这个规则主要是为了消除冗余转换。

6)复杂类型元素声明中定义或引用的复杂类型元素另外生成一个新的表,该表字段生成规则参照 1) ~5)。

2.2 语义映射规则

语义映射主要遵循以下映射规则:

1)对简单数据类型和基本自定义简单类型,映射到包装器公共类型库中相应的数据类型。对利用 restriction 派生出的简单数据类型,定义中限制元素应用 XML Schema 的 12 个面,指定一个有效的值范围、约束值的长度和精度、枚举一系列有效值或指定有效值必须匹配的正则表达式,映射为关系模式中相应字段取值的约束条件,如被 enumeration 面限制派生的简单类型映射结果需加上 check 约束。

2)default 属性指定元素或属性的默认值,映射为表字段取值约束条件 default。属性声明中包含 use = "required" 或 "fixed" 或 "default",以及简单类型元素声明中包含 use = "fixed" 或 "default" 或 minOccurs = "1",映射为表字段取值约束条件 not null。

3)在定义一致性约束的元素声明中,field 对应的元素或属性映射为关系模式中字段。若是唯一性约束,该字段具有取值约束条件 unique;若是关键性约束,该字段具有取值约束条件 not null 和 unique,即该字段可以作为表的主键,约束条件设为 primary key;若是关键字引用约束,keyref 声明中 field 指定的元素或属性映射为关系模式中字段,该字段是关系模式的外键字段,其字段值引用 schema 中具有 key 约束的元素或属性映射字段的值。

4) 外键的映射在 2.1 节中已说明。

2.3 规范化分析

按照上述映射规则生成关系模式,是以元素之间的嵌套关系为基础,一个 complexType 元素生成新的关系模式时,其声明中定义的所有属性和只出现一次的简单元素都是单值完全函数依赖于该元素的,相应地生成关系模式的字段。如果元素中定义了 key 约束的属性或子元素,或者定义了 ID 类型

的属性,说明这些属性或子元素可以唯一识别该元素,相应地生成表的主键字段,其他字段也就对应地单值完全函数依赖于该主键字段;否则就用自动增加的整数作为表的主键字段,其他所有字段都单值完全函数依赖于该主键字段。这样得出的关系模式不会存在部分依赖和传递依赖,也不会存在字段依赖于某个非主键字段,即关系模式满足 BCNF。

complexType 元素中声明的列表元素或者可以出现多次的简单元素,它们多值函数依赖于该元素,分离出去生成新的关系模式,添加一个类型为自动增加的整数的字段,并生成一个引用该元素生成关系模式的主键字段值的外键字段,这两个字段一起构成该关系模式的主键字段。最终得到的关系模式不存在非平凡且非函数依赖的多值依赖,实现了关系模式满足 4NF 的目的。

3　映射模型的处理算法与实现

3.1　映射过程的算法描述

本映射算法的逻辑框图如图 1 所示,预处理模块定义了一组函数对 JDOM 解析后的 XML Schema 进行处理,主函数是 accessElement,遍历分析树中的各个节点,对所有 element 节点构造一个 Node 类,再把 Node 类对象按照从属关系添加到 JTree 中,最后得到正确表达 element 节点层级关系的对象节点树,此时每个节点都包含了转换为关系模式所需的必要信息。模式转换模块的主函数 traverseCreateTable 遍历对象节点树,根据结构映射规则和语义映射规则生成关系模式。如图 2 是对象节点树的节点结构。

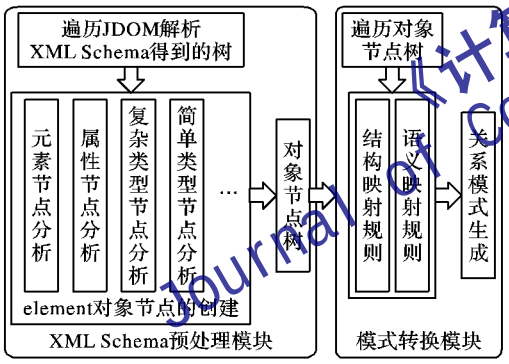


图 1　映射算法的逻辑框图

Node
nodeName: String nodeType: String primaryKey: String primaryKeyType: String attributes: Hashtable<String, String> defaultValue: String fixedValue: String isList: boolean manyOccurs: boolean includeText: boolean notNull: boolean createTable: boolean hasPrimaryKey: boolean isUnique: boolean isKey: boolean

图 2　对象节点树的节点结构

下面分模块简单地描述一下整个算法的流程。

3.1.1　预处理模块

输入　JDOM 解析 XML Schema 得到的树的根元素 parent,以根元素对应的 Node 类对象为参数构造的 DefaultMutableTreeNode 类对象 rootNode。

输出　对象节点树 elementTree。

处理步骤如下:

1)取得根节点对应的 Node 类 fatherNode,如果 parent 的名字等于 fatherNode 的 nodeName,则分别调用 getUniqueEleName(parent)和 getKeyEleParent(parent)函数得到设置了 unique 约束和 key 约束的子元素名字。

2)取得 parent 的所有子节点,随后逐个分析子节点,首先新建一个 DefaultMutableTreeNode 类对象 childNode 和一个 Node 类对象 treeNode,来存储子节点的相关信息。对于所有的子节点分下列 4 种情况进行处理。

①若子节点是属性,取得该属性的名字、类型及其约束条件。若属性类型为 ID 或者属性名字等于 keyEleName,则该属性映射为 fatherNode 生成关系模式的主键字段;若属性名字等于 uniqueEleName,则类型加上 unique 约束;最后把属性名字和类型添加进 fatherNode 的 attributes。

②若子节点是子元素,且子元素的 name 属性值等于 uniqueEleName 或 keyEleName,说明该字段有 unique 约束或可以作为主键字段。子元素的 type 属性值分以下 3 种情况处理:

a)type 属性值不为空且不等于全局类型名,取得其类型值,构造对象 treeNode = new Node(子元素名字, 子元素类型)。

b)type 属性值为空,继续取子元素的子节点。

当子节点名为“complexType”时,用子元素名和“complexType”构造 treeNode,并赋值 treeNode. createTable = true,若子节点的 mixed = “true”,则赋值 treeNode. includeText = true。

当子节点名为“simpleType”时,调用 simpleTypeNode 函数,在子元素数据类型存在阈值约束、枚举约束或者为列表类型时构造相应的 Node 类对象,返回该类对象赋给 treeNode。若子元素的 maxOccurs = “unbounded”,则 treeNode. manyOccurs = true; treeNode. createTable = true;若子元素的 minOccurs = “1”或 use = “required”或 default 属性值不为空或 fixed 属性值不为空,则 treeNode. notNull = true。

a)、b)情况下,用 treeNode 类对象构造一个 DefaultMutableTreeNode 类对象 childNode,把 childNode 添加到 rootNode 下面,最后递归调用遍历函数 accessElement(子元素, childNode)。

c)子元素类型引用了全局类型,调用 getRefTypeElement 函数得到引用的节点 refTypeElement,若引用节点名等于“complexType”或“simpleType”时,同 b)处理过程。接着用 treeNode 类对象构造一个 DefaultMutableTreeNode 类对象 childNode,并把 childNode 添加到 rootNode 下面,最后递归调用遍历函数 accessElement(refTypeElement, childNode)。

③若子节点是子元素,且子元素的 ref 属性不为空,说明该子元素引用了某个全局元素,如果 ref 属性值等于 uniqueEleName 或 keyEleName,说明该子元素对应的字段有 unique 约束或者可以作为主键字段。调用 getRefElement 函数取得被引用的元素 refElement,对 refElement 按照②所述步骤进行处理,最后递归调用遍历函数 accessElement(refElement, childNode)。

④如果子节点既不是属性也不是子元素,则递归调用遍历函数 accessElement(子节点, rootNode)。

所有的 JDOM 树节点遍历完后返回 rootNode,用 rootNode 作为输入参数构造一个 JTree 类对象 elementTree,调用

deleteRefNode 函数和 deleteRefTypeNode 函数把 elementTree 上的全局元素子树和全局类型子树删除，最后得到的 elementTree 就是预处理模块的输出。

3.1.2 模式转换模块

输入 对象节点树的根节点 root。

输出 XML Schema 对应的所有关系模式。

转换步骤如下：

1）取得 root 节点对应的 Node 类对象 rootNode，新建一个 Table 类对象 table，如果 rootNode. createTable 等于 true，逐步进行下面的处理。

2）表名和主键字段处理：nodeName 作为表名，hasPrimaryKey 等于 false 的情况下需创建一个主键字段，类型可以为自动增加的整数，添加进 table 的 column。

3）属性映射字段处理：取出节点对象 attributes 属性的值，添加进 table 的 column；

4）节点对象本身映射处理：若节点对象的 nodeType 不等于“complexType”，该节点对象信息作为表的一个字段添加进 table 的 column。在 notNull 为 true 的情况下，为字段类型加上“not null”约束。

5）列表类型节点对象或可以出现多次的节点对象映射处理：它们多值依赖于父节点，需新建一个 Table 类对象，存储它们生成的关系模式的元数据信息。

①若父节点的 manyOccurs 等于 false，且单值依赖于不为 schema 的祖父节点，则它们相应地多值依赖于祖父节点，此时为了消除模式冗余，它们的外键字段引用其祖父节点生成关系模式的主键字段值，否则外键字段引用它们父节点生成关系模式的主键字段值。

②若父节点的 manyOccurs 等于 true，则需另外生成一个新的关系模式，其字段分别为 rootNode. primaryKey +“_fk”和 fatherNode. primaryKey +“_fk”，字段值分别引用节点生成关系模式的主键字段值和父节点生成关系模式的主键字段值，把此新表添加进 Hashtable 类对象 tables。

③对于列表类型节点对象，处理到这步其所含信息都已经赋给了 table，把 table 添加进关系模式集 tables；对于可以出现多次的节点对象且该节点对象为叶节点，把 table 添加进 tables。

6）mixed = true 的节点对象映射处理：生成一个新表存储字符数据内容，该表主键字段可以为自动增加的整数，表示字符数据内容在元素中的位置。

①在节点的 manyOccurs 等于 false 的情况下，若节点单值依赖于不为 schema 的父节点，则为了消除模式冗余，该表的外键字段引用节点父节点生成关系模式的主键字段值，否则引用节点生成关系模式的主键字段值。

②若节点的 manyOccurs 等于 true，需另外生成一个新的关系模式，其字段包括 rootNode. primaryKey +“_fk”和上述存储字符数据内容的表的名字 +“_pk_fk”，字段值分别引用节点生成关系模式的主键字段值和上述存储字符数据内容的关系模式的主键字段值。

7）节点对象的子节点对象映射处理：对于所有子节点对象，分两种情况处理：

①在 createTable 等于 false 的情况下，若子节点对象的 isKey 为 true，子节点对象映射为主键字段；若子节点对象的 isUnique =“true”，则子节点对象映射的字段有 unique 约束；若子节点对象的 notNull =“true”，则子节点对象映射的字段有 not null 约束。把子节点对象映射的字段添加进 table 的 column，并把 table 添加进 tables。

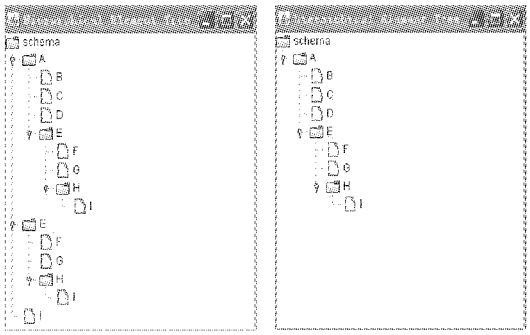
②如果 createTable 等于 true，递归调用处理映射处理函数 traverseCreateTable（子节点）。

至此，模式转换模块已经把所有生成的关系模式元数据存储在 tables 中，因为 Table 类实现了接口 Serializable，所以构造一个 ObjectOutputStream 类对象 oos，再执行 oos. writeObject（tables），就可以把关系模式元数据存储到文件中。

3.2 映射模型的实现

由于 XML Schema 是 XML 文档格式的，可以利用 XML 文档最常用的解析器 DOM 或 SAX 进行解析。本文采用 JDOM 作为解析 XML Schema 的编程接口，使用 Java 语言来实现文中描述的映射模型。具体实现过程是，递归遍历解析得到的 JDOM 树，把所有的元素按照从属关系，以层级结构构造一棵 JTree 树，JTree 树的节点是以各个元素的相关信息（包括元素名、元素类型、元素声明中定义的属性、元素基数等）定义的一个类，以方便在生成关系模式时提取对应的信息。

例如：某 XML Schema 经解析得到的 JTree 树例如图 3 所示，其中图（a）是创建一个 AnalyzeJdomTree 类，调用其递归遍历方法 accessElement 得到的结构树，这个方法考虑了元素对全局组件的引用，即遍历到引用全局组件的元素时，继续遍历被引用的全局组件，以正确反映元素间的从属关系。再调用 deleteRefNode 和 deleteRefTypeNode 这两个方法把元素引用的全局组件删除，最后得到如图（b）所示的包括 XML Schema 中所有元素并准确显示了元素之间从属关系的 JTree 树。接着创建一个 AnalyzeJTree 类，调用其递归遍历方法 traverseCreateTable 遍历图 3（b）中的 Jtree 树，按照 2.1 节和 2.2 节定义的映射规则生成关系模式，可以实现从 XML Schema 到关系模式的转换过程。



(a) 完整的schema结构树 (b) 简化的schema结构树

图3 JTree 树

4 结语

与 XML DTD 相比，XML Schema 提供了丰富的数据类型，并支持用户对数据类型的扩展，基本上满足了关系模式在数据描述上的需要，另外 XML Schema 采用 XML 语法规则，具备自描述语义的特征，所以把 XML Schema 转换为关系模式具有很好的应用意义。本文首先定义了一系列映射规则，基于这些规则编程实现了模式映射过程中数据结构和内容的正确转换以及大多数语义约束的保留，并考虑了消除关系模式冗余，能够处理元素与元素之间的多对一、一对多以及多对多的关系，符合第四范式的要求。

本文研究的映射模型尚未考虑一致性约束 field 不单一情况下的映射，以及限制或扩展派生复杂类型情况下的映射，

（下转第 2955 页）

一般情况下,前者花费的时间比后者少得多。其中:L'_{k-1} 和 C'_k 分别表示 L_{k-1} 经过定理2和推论1过滤后的$(k-1)$-维频繁项集和由 L'_{k-1} 生成的 k-项候选项集,N_{k-1} 为应用定理2和推论1过滤掉的项目。

为了测试改进后的算法的效率,用VC++6.0分别实现了 Apriori、PM-Apriori[10]58、Apriori'[11]38、GE-Apriori[13]192 和 IApriori 共5个算法,并在内存为1 GB,CPU 主频为2930 MHz,操作系统为 Windows XP SP3 的计算机上进行了实验,数据库则是文献[17]中提供的蘑菇数据(mushroom database),该数据库共有8 124条记录、22个属性,实验结果如图3所示,从图中可以看出,经过优化的算法在速度上有了较大的提高。

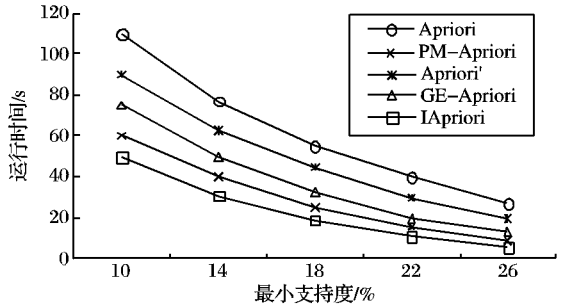


图3　不同算法在最小支持度下的运行时间

4　结语

本文通过分析 Apriori 算法及其优化算法的优劣,提出了一个改进的 IApriori 算法,在关联规则挖掘的发现频繁项集的步骤中进行了优化,并给出了优化算法的完整描述,通过利用蘑菇数据库进行实验分析评价,证明了改进后算法的有效性。

参考文献:

[1] AGRWAL R, SRIKAN R. Fast algorithms for mining association rules in large databases [C]// Proceedings of the 20th International Conference on Very Large Data Bases. San Francisco: Morgan Kaufmann Publishers, 1994: 487 – 499.

[2] 何军,刘红岩,杜小勇.多关系关联规则挖掘研究综述[J].软件学报,2007,18(11):2752 – 2765.

[3] 刘君强,孙晓莹,潘云鹤.关联规则挖掘技术研究的新进展[J].计算机科学,2004,31(1):110 – 113.

[4] PARK J S, CHEN M S, YU P S. An effective Hash based algorithm for mining association rules [C]// Proceedings of International Conference on the Special Interest Group on Management of Data. New York: ACM, 1995: 175 – 186.

[5] 尤磊,兰洋,熊炎.一种基于关系代数的 Apriori 优化方法[J].信阳师范学院学报:自然科学版,2010,23(1):156 – 160.

[6] HAN J, FU Y. Discovery of multiple-level association rules from large databases [C]// Proceedings of the 20th International Conference on Very Large Database. Zuich, Switzerland: [s. n.], 1995: 420 – 431.

[7] SAVASERE A, OMIECINSKI E, NAVATHE S. An efficient algorithm for mining association rules in large databases [C]// Proceedings of the 21st International Conference on Very Large Database. New York: ACM, 1995: 432 – 443.

[8] TOLVONEN H . Sampling large databases for association rules [C]// Proceedings of the 22nd International Conference on Very Large Database. Bombay, India [s. n.], 1996: 134 – 145.

[9] BRIN S. Dynamic itemset counting and implication rules for market basket analysis [C]// Proceedings of International Conference on the Special Interest Group on Management of Data. New York ACM, 1997: 255 – 264.

[10] 杨志刚,何月顺.基于压缩事务矩阵相乘的 Apriori 改进算法[J].中国新技术新产品,2010,30(6):57 – 58.

[11] 黄建明,赵文静,王星星.基于十字链表的 Apriori 改进算法[J].计算机工程,2009,35(2):37 – 38.

[12] 王伟勤,郑海.Apriori 算法的进一步改进[J].计算机与数字工程,2009,37(4):20 – 23.

[13] 徐章艳,刘美玲,张师超,等.Apriori 算法的三种优化方法[J].计算机工程与应用,2004,40(36):190 – 192.

[14] 陈应霞,陈艳.关联规则中的 Apriori 挖掘算法改进[J].长江大学学报:自然科学版,2008,5(4):341 – 343.

[15] 曾万聪,周绪波,戴勃,等.关联规则挖掘的矩阵其法[J].计算机工程,2006,32(02):45 – 47.

[16] 李云峰,陈建文,程代杰.关联规则挖掘的研究及对 Apriori 算法的改进[J].计算机工程与科学,2002,24(6):65 – 68.

[17] SCHLIMMER J. Mushroom data set[DB/OL]. [2010 – 04 – 30]. http://archive. ics. uci. edu/ml/machine-learning-databases/mushroom/agaricus-lepiota. data.

(上接第2944页)

对于 Schema 中的 include、redefine 和 import 也没有进行讨论和实现,接下来的工作就是完善映射模型,实现与包装器的连接以及处理包装器传递过来的查询 XML 文档数据的命令。

参考文献:

[1] MANI M, LEE D, MUNTZ R R. Semantic data modeling using XML schemas[C]// Proceedings of the 20th International Conference on Conceptual Modeling, LNCS 2224. Berlin: Springer-Verlag, 2001: 149 – 163.

[2] LEE D, CHU W W. Comparative analysis of six XML schema languages [J]. ACM SIGMOD Record, 2000, 29(3): 76 – 87.

[3] LEE D, MANI M, MURATA M. Reasoning about XML schema languages using formal language theory, RJ# 10197, Log# 95071 [R]. [S.l.]: IBM Almaden Research Center, 2002.

[4] MANI M, LEE D. XML to relational conversion using theory of regular tree grammars [C]// Proceedings of the VLDB 2002 Workshop EEXTT and CAiSE 2002 Workshop DTWeb on Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web-Revised Papers, LNCS 2590. Berlin: Springer-Verlag, 2002: 81 – 103.

[5] SUN HONGWEI, ZHANG SHUSHENG, ZHOU JINGTAO, *et al*. Constraints-preserving mapping algorithm from XML-schema to relational schema [C]// Proceedings of the First International Conference on Engineering and Deployment of Cooperative Information Systems, LNCS 2480. Berlin: Springer-Verlag, 2002: 193 – 207.

[6] SUN HONGWEI, ZHANG SHUSHENG, ZHOU JINGTAO, *et al*. Mapping XML schema to relational schema [C]// EurAsia-ICT 2002: Information and Communication Technology, LNCS 2510. Berlin: Springer-Verlag, 2002: 322 – 329.

[7] 苏宝程.从 XML 模式到关系模式的映射与规范化设计[J].内蒙古煤炭经济:工作研究与理论探讨,2006(9):29 – 33.

[8] 黄根平,郭绍忠,陈海勇,等.数据集成中 XML 模式和关系模式映射模型研究[J].信息工程大学学报,2009,10(4):527 – 531.

[9] WALMSLEY P. XML 模式权威教程[M].陈维安,乔安平,英宇,译.北京:清华大学出版社,2003:9.