[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 17/30 (2006.01)

[12] 发明专利说明书

专利号 ZL 200410004379.X

[45] 授权公告日 2007 年 1 月 24 日

[11] 授权公告号 CN 1296855C

[22] 申请日 2004.2.17
[21] 申请号 200410004379.X
[73] 专利权人 北京大学
地址 100871 北京市海淀区颐和园路 5 号
[72] 发明人 杨冬青 唐世渭 刘云峰 王腾蛟 高 军
[56] 参考文献
EP1363209A1 2003.11.19 G06F17/30
审查员 李 琰

[74] 专利代理机构 北京同立钧成知识产权代理有限公司
代理人 刘 芳 刘 薇

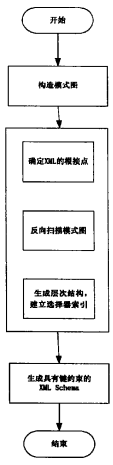
权利要求书 3 页 说明书 11 页 附图 3 页

[54] 发明名称

面向可扩展标记语言模式的键约束自动生成方法

[57] 摘要

本发明提供了一种基于关系数据库模式图 Schema Diagram 生成可扩展标记语言大纲 XML Schema 的层次描述和键约束的方法，包括：利用关系数据库的键和外键构造数据库的模式图；在扫描模式图过程中建立关系表名、候选码与 XML Schema 中键约束的选择器 XPath (Selector XPath)、域 XPath (Field XPath) 之间的对应关系，生成关系数据库键约束的 XML 选择器索引；利用该索引自动生成 XML Schema 上的键约束，从而完成基于 XML 数据发布过程中键约束的自动获得。本发明广泛应用于当前网络环境下各应用系统的基于 XML 的数据发布过程，满足了目前 Internet 上大量应用系统之间 XML 数据发布与转换的需求，具有广阔的应用前景。



1、一种面向可扩展标记语言模式的键约束自动生成方法，其特征在于：包括如下步骤：

步骤一、利用关系数据库的键和外键构造数据库的模式图；

步骤二、扫描模式图，在扫描模式图过程中建立关系表名、候选码与XML Schema中键约束的选择器XPath、域XPath之间的对应关系，生成关系数据库键约束的XML选择器索引；

步骤三、利用步骤二获得的索引生成XML Schema上的键约束，从而完成基于XML数据发布过程中键约束的自动生成。

2、根据权利要求1所述的面向可扩展标记语言模式的键约束自动生成方法，其特征在于：所述的数据库模式图为G=[V,E]，其中：V=[key attribute, non-key attribute]，即包括主属性和非主属性；E是描述表之间的参照关系，对于表s，其属性列表为X，表t，其属性列表为Y，对于每一个外键引用关系$t[\beta] \subseteq s[\alpha]$，其中$\alpha \subseteq X$，$\beta \subseteq Y$，且$\alpha$是s表的主属性，则由t表的$\beta$属性到s表的$\alpha$属性之间有一条边。

3、根据权利要求1所述的面向可扩展标记语言模式的键约束自动生成方法，其特征在于：所述的步骤一，包括如下步骤：

步骤11、对数据库中的每一个关系表用一个矩形来表示，矩形上方标注关系的名字，矩形内列出关系表的属性；

步骤12、如果有主键，用一条横线将主键属性分隔在方框上部；

步骤13、对数据库中的每一个外键，生成从参照关系的外键属性到被参照关系的主键属性之间的一个有向边。

4、根据权利要求1所述的面向可扩展标记语言模式的键约束自动生成方法，其特征在于：所述的步骤二，包括：

步骤21、根据数据库模式图，确定XML根结点及其儿子节点；

步骤22、从XML根结点出发，对模式图进行反向扫描；根据扫描的结

果，建立层次结构和 XML Schema 选择器索引表。

5、根据权利要求 4 所述的面向可扩展标记语言模式的键约束自动生成方法，其特征在于：所述的步骤 21，包括：

步骤 211：建立根结点 V_r；

步骤 212：扫描模式图，找出只有入度而没有出度的节点集合 R，并对 R 中每一个节点，相应生成由根结点 V_r 指向该节点的一条边；

步骤 213：如果没有这样的节点，表示模式图中有环，则从环中任选一节点，并生成根结点 V_r 指向该节点的一条边。

6、根据权利要求 4 所述的面向可扩展标记语言模式的键约束自动生成方法，其特征在于：所述的步骤 22 包括如下步骤：

步骤 221、将节点集合 R 压入队列，按照模式图边的相反方向，开始对模式图进行反向的宽度优先扫描；

步骤 222、记当前节点为 s(X)，其属性列表为 X；扫描到新的节点记做 t(Y)，其属性列表为 Y，由于新的结点是既有入度又有出度的结点，那么，一定存在外键连接 FK：$t[\beta] \subseteq s[\alpha]$，其中 $\alpha \subseteq X$，$\beta \subseteq Y$，

对于该外键连接，按照以下步骤生成相应的层次结构：

首先，在 t 中将已经出现在 s 节点中的属性删除；保证 s 节点的该属性只出现一次；

然后，将 t 剩余的属性作为 s 节点的子节点；这样就形成了 XML 的树的一个层次结构；

每生成一次层次结构，都在选择器索引中建立相应的索引，选择器索引的结构如下：

数据库表名 Table Name	候选键属性 Candidate Key	上下文路径 Context Path	选择器 XPath Selector XPath	域 XPath Field XPath

步骤 223、按照步骤 222 所述的方法，按反向优先扫描顺序，依次处理

所有结点，如果扫描到的节点已经被扫描过，则生成 reference 关系索引，引用已经被扫描定位的该结点，直到处理完关系数据库模式图中的所有外键约束关系。

7、根据权利要求 6 所述的面向可扩展标记语言模式的键约束自动生成方法，其特征在于：步骤 222 中，选择器索引的内容按照以下步骤生成：

步骤 a、对于根结点 V_r，将模式图中该节点记录的表名和主键属性填入选择器索引表的前两列中，其选择器 XPath 和域 XPath 为 NULL；

步骤 b、在扫描模式图过程中，每扫描到一个新节点，就将模式图中该节点记录的表名和主键属性填入选择器索引表的前两列中；其选择器 XPath 为其父节点选择器 XPath 加上当前扫描到的边，域 XPath 为目标集路径加上该节点至候选键属性的边。

8、根据权利要求 1 或 6 所述的面向可扩展标记语言模式的键约束自动生成方法，其特征在于：所述的步骤三，包括如下步骤：

步骤 30、读入步骤二生成的层次结构，建立 XML Schema 中的元素和子元素构成的模式框架；

步骤 31、逐行读入选择器索引表，将数据库表名、候选键属性，上下文路径、选择器 XPath 和域 XPath，分别设定为 T、K、C、S、F；

步骤 32、在模式框架中按照上下文路径 C 查找元素名称为 T 的元素，增加如下键约束：

```
<xsd:key name=键名>
    <xsd:selector xpath= S />
    <xsd:field xpath=F/>
</xsd:key>。
```

面向可扩展标记语言模式的键约束自动生成方法

技术领域

本发明涉及数据库描述的生成方法，尤其是一种基于关系数据库模式图Schema Diagram的自动生成可扩展标记语言模式XML Schema键约束的方法，属于计算机数据管理技术领域。

背景技术

XML（eXtensible Markup Language）已成为Internet上的数据表示和交换的事实标准，越来越多的应用系统采用XML作为标准格式来发布和交换数据。但是，由于关系数据库技术成熟，市场中占有主导地位，如何将关系数据库转换为XML文档成为当前的迫切需要解决的问题。

目前已存在一些系统和工具可以对关系数据库和XML进行数据转换，如IBM公司的XML Extender，Oracle公司的XML Enable以及XPERANTO，SilkRoute系统等，但是由于这些系统只考虑到数据结构层面，缺少对数据完整性约束的分析，而键和外键约束是关系数据模型中最重要的完整性约束，在数据库管理系统领域得到大量应用，但是现有技术中，还没有解决从关系模型到XML的键约束转换问题，造成了数据模式转换过程中的语义信息丢失的问题。具体如下：

首先，目前从关系数据库到XML文档的映射规则仅仅考虑了数据结构层面，缺少对数据完整性约束的分析。

其次，虽然目前XML Schema中已经提供了定义XML键约束的机制，但是在数据交换过程中，原来关系数据库中的键和外键语义约束，不能通过交换过程自动转换为XML上的键约束。

例如由IBM Almaden Research Center研制的XPERANTO系统，虽然该系统利用Xquery视图实现了从关系数据库到XML数据的发布和转换，但是

由该系统生成的任何 XML 文档都没有键的约束，因此丢失了原有的关系数据库中的键和外键等语义约束。

因此，目前广泛应用的基于 XML 的数据发布应用系统迫切需要提供一种基于关系数据库的键和外键约束生成 XML 数据发布键约束的方法。

发明内容

本发明所要解决的技术问题在于：提供一种面向可扩展标记语言模式的键约束自动生成方法，基于关系数据库模式图 Schema Diagram 的键约束（包括键和外键）生成可扩展标记语言模式 XML Schema 的层次描述和键约束，解决目前在关系数据库数据转换为 XML 数据进行发布的过程中结构转换时完整性约束丢失的问题。

本发明所述的技术方案，包括如下步骤：

步骤一、利用关系数据库的键和外键构造数据库的模式图；

步骤二、扫描模式图，在扫描模式图过程中建立关系表名、候选码与 XML Schema 中键约束的选择器 XPath、域 XPath 之间的对应关系，生成关系数据库键约束的 XML 选择器索引；

步骤三、利用步骤二获得的索引生成 XML Schema 上的键约束，从而完成基于 XML 数据发布过程中键约束的自动生成。

本发明通过关系数据库模式图，建立关系数据库键约束的 XML 选择器索引，从而生成 XML Schema 的键约束，解决了目前基于 XML 的数据发布过程中结构转换时完整性约束丢失的问题，可广泛应用于当前网络环境下各应用系统的基于 XML 的数据发布过程，满足了目前 Internet 上大量应用系统之间 XML 数据发布与转换的需求，具有广阔的应用前景。

附图说明

图 1 是本发明的整体流程的示意图；

图 2 是一个简化的银行系统的数据库模式图；

图 3 是图 2 所示的例子中，建立根结点和只有出度而没有入度节点的示意图；

图 4 是图 2 所示的例子中，branch 出队列后的示意图；

图 5 是图 2 所示的例子中，account 出队列后的示意图；

图 6 是图 2 所示的例子中，loan 出队列后的示意图；

图 7 是图 2 所示的例子中，R 中 customer 入队列后的示意图；

图 8 是图 2 所示的例子中，customer 出队列后的示意图。

具体实施方式

本发明是针对当前网络环境下各应用系统基于 XML 数据发布的需求而提出的一种新的完整性约束生成的技术方案，它基于关系数据库模式图（Schema Diagram）中重要的和大量存在的键和外键约束自动生成为 XML Schema 描述的层次关系和键约束，实现了关系数据库数据的 XML 发布。

本发明方法，首先利用关系数据库的键和外键构造数据库的模式图；然后在扫描模式图过程中建立关系表名、候选码与 XML Schema 中键约束的选择器 XPath（Selector XPath）、域 XPath（Field XPath）之间的对应关系，生成关系数据库键约束的 XML 选择器索引；再利用该索引自动生成 XML Schema 上的键约束，从而实现了基于 XML 数据发布过程中键约束的自动获得。

本发明所述的模式图定义为 G=(V,E)，其中：V=(key attribute, non-key attribute)，即包括主属性和非主属性；E 是描述表之间的参照关系，对于表 s，其属性列表为 X，表 t，其属性列表为 Y，对于每一个外键引用关系 $t[\beta] \subseteq s[\alpha]$，其中 $\alpha \subseteq X$，$\beta \subseteq Y$，且 α 是 s 表的主属性，则由 t 表的 β 属性到 s 表的 α 属性之间有一条边。

对于一个含有主键和外键依赖的数据库模式，可以用模式图 Schema Diagram 来表示，即所述的构建模式图，包括如下步骤：

步骤 11、对数据库中的每一个关系表用一个矩形来表示，矩形上方标注关系的名字，矩形内列出关系表的属性。

步骤 12、如果有主键，用一条横线将主键属性分隔在方框上部。

步骤 13、对数据库中的每一个外键，生成从参照关系的外键属性到被参照关系的主键属性之间的一个有向边。

下面通过一个具体的实例来说明如何构建带有内键和外键约束关系数据库的模式图:

一个简化的银行系统的数据库模式，包含如下的表结构:

branch (branch_name, branch_city, assets),

account (account_number, branch_name, balance),

depositor (customer_name, account_number, last_date),

customer (customer_name, customer_street, customer_city),

loan (loan_number, branch_name, amount)

borrower (customer_name, loan_number, last_date)

其中的下划线标出的该表的键。除此以外，各个表之间还具有外键约束联系，例如 account 表中的 branch_name 属性是一个外键，它参照 branch 表中的主键 branch_name。

如图 2 所示是根据上述关系数据库模式图构建步骤得到的数据库模式图 。本发明的获得关系数据库中模式图的方式，与现有技术基本相同，可以参考 *Paul Dorsey and Peter Koletzke, Designer/2000 Handbook, Oracle Press*。

本发明所述的扫描数据库模式图，生成索引，具体包括如下步骤:

步骤 21、根据数据库模式图，建立 XML 根结点及其儿子节点;

步骤 22、从 XML 根结点出发，对模式图进行反向扫描; 根据扫描的结果，建立层次结构和索引表。

上述的步骤 21，根据以下步骤确定 XML 根结点 V_r 以及其儿子节点，具

体包括如下步骤:

步骤 211: 建立根结点 V_r;

步骤 212: 扫描模式图，找出只有入度没有出度的节点集合 R; 并对 R 中每一个节点，相应生成由根结点 V_r 指向该节点的一条边。

步骤 213: 如果没有这样的节点，表示模式图中有环，则从环中任选一节点，并生成根结点 V_r 指向该节点的一条边。

所述的根结点 V_r 是一个空结点，只是作为 XML 树的开始结点，与具体关系数据库无关。由于 XML 数据是一个树状结构，所以找到了所有的只有入度没有出度的节点集合 R 后，从 R 出发一定可以达到所有的结点，所以步骤 212 中，完成了这一步骤。如果模式图中是环状的，则该环的结点中不存在只有入度没有出度的节点，但是从每一个节点出发都可以到达环中的所有结点，因此可以从环中任取一个节点，作为开始。

如图 3 所示是根据图 2 中的数据库模式图，其 R={branch, customer}，完成步骤 21 后的示意图。

所述的步骤 22 包括如下步骤:

步骤 221、将节点集合 R 压入队列，按照模式图边的相反方向，开始对模式图进行反向的宽度优先扫描;

步骤 222、在反向宽度优先扫描过程中，记当前节点为 s(X)，其属性列表为 X; 扫描到新的节点记做 t(Y)，其属性列表为 Y，由于新的结点是既有入度又有出度的结点，那么，一定存在外键连接 FK: $t[\beta] \subseteq s[\alpha]$，其中 $\alpha \subseteq X$，$\beta \subseteq Y$，

对于该外键连接，按照以下步骤生成相应的层次结构:

首先，在 t 中将已经出现在 s 节点中的属性删除; 保证 s 节点的该属性只出现一次;

然后，将 t 剩余的属性作为 s 节点的子节点; 这样就形成了 XML 的树的一个层次结构;

每生成一次层次结构，都在 XML Schema 选择器索引中建立相应的索引，选择器索引的结构如下：

数据库表名 Table Name	候选键属性 Candidate Key	上下文路径 Context Path	选择器 XPath Selector XPath	域 XPath Field XPath

选择器索引的内容按照以下步骤生成：

步骤 a、对于根结点 V_r，将模式图中该节点记录的表名和主键属性填入选择器索引表的前两列中，其选择器 XPath 和域 XPath 为 NULL；

步骤 b、在扫描模式图过程中，每扫描到一个新节点，就将模式图中该节点记录的表名和主键属性填入选择器索引表的前两列中；其选择器 XPath 为其父节点选择器 XPath 加上当前扫描到的边，域 XPath 为目标集路径加上该节点至候选键属性的边。

步骤 223、按照上面的方法，依次递归处理所有结点，如果扫描到的节点已经被扫描过，说明 XML 树中已经包含了该结点，则生成 reference 关系索引，引用已经被扫描定位的该结点；直到处理完关系数据库模式图中的所有外键约束关系。

本发明的步骤 223 中递归处理所有结点，采用了宽度优先的扫描方式。所述的宽度优先扫描，是指从集合 R 的每一个节点 v 开始，先访问节点 v，并将其标记为已访问节点，接着依次访问 v 的所有相邻的节点 w1, w2, ..., wn，然后再访问与 w1, w2, ..., wn 邻接的所有未被访问的节点，依此类推，直到所有的节点都被访问为止。

下面仍采用上面提到的简化银行系统的例子，来进一步说明本发明的技术方案：

步骤 A、只有出度没有入度的集合 R= {branch, customer} 中 branch 进入队列中，利用队列对关系数据库模式图进行处理，开始宽度扫描；

步骤 B、branch 出队列，branch 的属性表成为 branch 的子节点，由于通过外键约束，连接到 account 和 loan 表，所以，account 和 loan 表的属性表成为了各自的子节点；同时队列元素变成{account, loan}，生成 XML 模式如图 4 所示；

步骤 C、account 出队列，采用与步骤 B 同样的方式处理后，队列元素变为{loan, depositor}，生成 XML 模式如图 5 所示；

步骤 D、loan 出队列，队列元素变为{depositor, borrower}，生成 XML 模式如图 6 所示；

步骤 E、depositor 出队列，队列变为{borrower}；

步骤 F、borrower 出队列，队列为空；

步骤 G、R 中 customer 入队列，队列变为{customer}，生成 XML 模式如图 7 所示；

步骤 H、customer 出队，由于 depositor, borrower 都已经被扫描过，所以生成两个引用关系 reference1（depositor-customer）和 reference2（borrower-customer），处理结束，生成 XML 模式如图 8 所示。

在以上的步骤中，相应的索引表中的表项同时生成，下面是上述例子中得到的索引表：

数据库表名 Table Name	候选键属性 Candidate Key	上下文路径 Context Path	选择器 XPath Selector XPath	域 XPath Field XPath
Branch	branch_name	/	Branch	branch_name
Account	Account_number	/branch	Account	account_name
Loan	loan_number	/branch	Loan	loan_name
Depositor	NULL	NULL	NULL	NULL
Borrower	NULL	NULL	NULL	NULL
Customer	customer_name	/	Customer	customer_name

Reference 关系（反向扫描 customer 发出的边）如下：对于

depositor(customer_name) $\subseteq$ customer(customer_name)

borrower(customer_name) $\subseteq$ customer(customer_name)

得到 Reference 关系：

数据库表名 Table Name	前驱	上下文路径 Context Path	选择器 XPath Selector XPath	referencing 键路径 Keyref XPath
depositor	customer	/branch/account/	depositor	ref1
borrower	customer	/branch/loan	borrower	ref2

通过上面的步骤，本发明利用集合 R 为根的 XML 结构，得到了具有键约束的 XML Schema 的树状模式图的表示结构和便于对键约束进行检索的索引表，利用该索引集，就可以方便的生成 XML Schema 的键约束。

所述的生成具有键约束的 XML Schema，包括如下步骤：

步骤 30、读入步骤二生成的层次结构，建立 XML Schema 中的元素和子元素构成的模式框架;

步骤 31、逐行读入选择器索引表，将数据库表名、候选键属性，上下文路径、选择器 XPath 和域 XPath，分别设定为 T、K、C、S、F;

步骤 32、在模式框架中按照上下文路径 C 查找元素名称为 T 的元素，增加如下键约束:

```
<xsd:key name= 键名>
    <xsd:selector xpath= S />
    <xsd:field xpath=F/>
</xsd:key>。
```

下面上述例子中具有键约束的文字描述的 XML Schema 片断，斜体的部分表示生成的键约束:

```
<xsd:element name="customer">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="customer_name" type="xsd:string"/>
            <xsd:element name="customer_street" type="xsd:string"/>
            <xsd:element name="customer_city" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:key name="PK_c">
        <xsd:selector xpath="customer"/>
        <xsd:field xpath="customer_name"/>
    </xsd:key>
</xsd:element>
<xsd:element name="branch">
    <xsd:complexType>
        <xsd:sequence>
        <xsd: element name="branch_name" type="string" />
            <xsd: element name=" branch_city " type="string" />
```

```
<xsd: element name="assets" type="integer" />
      <xsd:element name="account" minOccurs=0 maxOccurs="unbounded">
          <xsd:complexType>
              <xsd:sequence>
           <xsd: element name="account_name" type="string" />
                    <xsd: element name="balance" type="integer" />
                   <xsd:element name="depositor" minOccurs=0 maxOccurs="unbounded" >
                       <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="last_date" type="xsd:date"/>
                        <xsd:element name="ref1" type="xsd:string"/>
                              </xsd:sequence>
                    </xsd:complexType>
                <xsd:keyref name="ref1_customer" refer="PK_c">
                       <xsd:selector xpath="branch/account/depositor"/>
                       <xsd:field xpath="ref1 "/>
                      </xsd:keyref>
                   </xsd:element>
              </xsd:sequence>
          </xsd:complexType>
        <xsd:key name="PK_a">
            <xsd:selector xpath="branch/account"/>
            <xsd:field xpath="account_name"/>
         </xsd:key>
      </xsd:element>
      <xsd:element name="loan" minOccurs=0 maxOccurs="unbounded">
          <xsd:complexType>
               <xsd:element name="borrower" minOccurs=0 maxOccurs="unbounded" >
                  <xsd:complexType>
                      <xsd:sequence>
                          <xsd:element name="last_date" type="xsd:date"/>
                      <xsd:element name="ref2 " type="xsd:string"/>
                          </xsd:sequence>
                </xsd:complexType>
                    <xsd:keyref name="ref2_customer" refer="PK_c">
                     <xsd:selector xpath="branch/loan/borrower"/>
```

```
                    <xsd:field xpath="ref2 "/>
                </xsd:keyref>
            </xsd:element>
        </xsd:complexType>
        <xsd:key name="PK_l">
            <xsd:selector xpath="branch/loan"/>
            <xsd:field xpath="loan_number"/>
        </xsd:key>
    </xsd:element>
</xsd:sequence>
</xsd:complexType>
<xsd:key name="PK_b">
    <xsd:selector xpath="branch"/>
    <xsd:field xpath="branch_name"/>
</xsd:key>
</xsd:element>
```

最后所应说明的是：以上实施例仅用以说明而非限制本发明的技术方案，尽管参照上述实施例对本发明进行了详细说明，本领域的普通技术人员应当理解：依然可以对本发明进行修改或者等同替换，而不脱离本发明的精神和范围的任何修改或局部替换，其均应涵盖在本发明的权利要求范围当中。

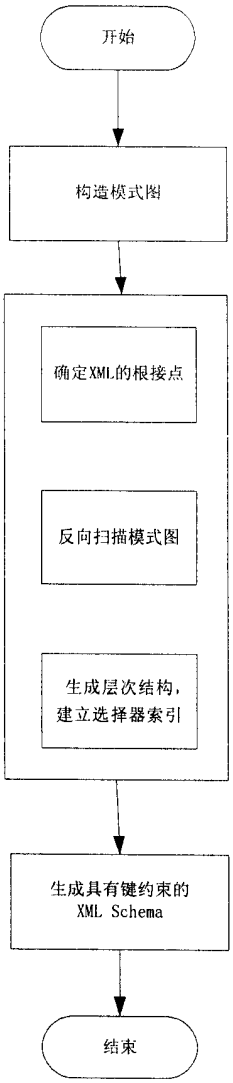

开始
构造模式图
确定XML的根接点
反向扫描模式图
生成层次结构，
建立选择器索引
生成具有键约束的
XML Schema
结束

图 1

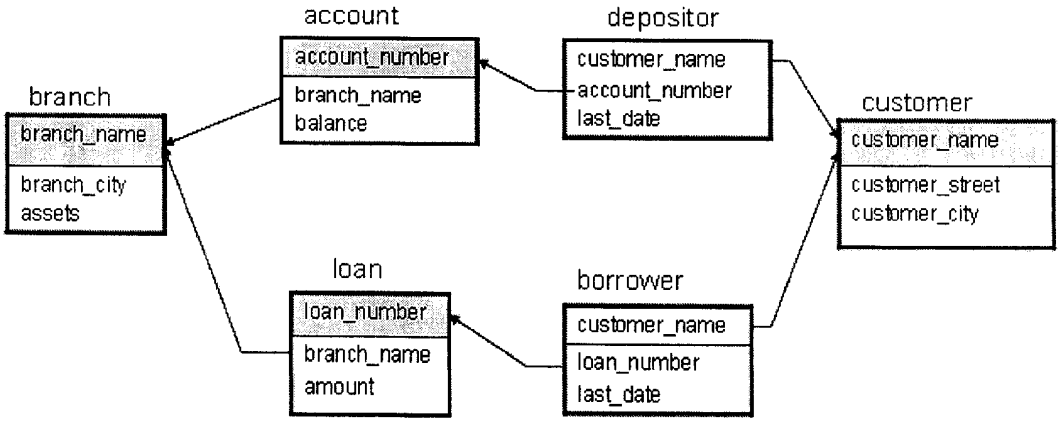

account
account_number
branch_name
balance
depositor
customer_name
account_number
last_date
branch
branch_name
branch_city
assets
customer
customer_name
customer_street
customer_city
loan
loan_number
branch_name
amount
borrower
customer_name
loan_number
last_date

图 2

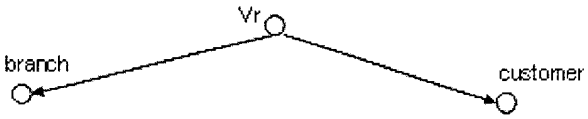

Vr
branch
customer

图 3

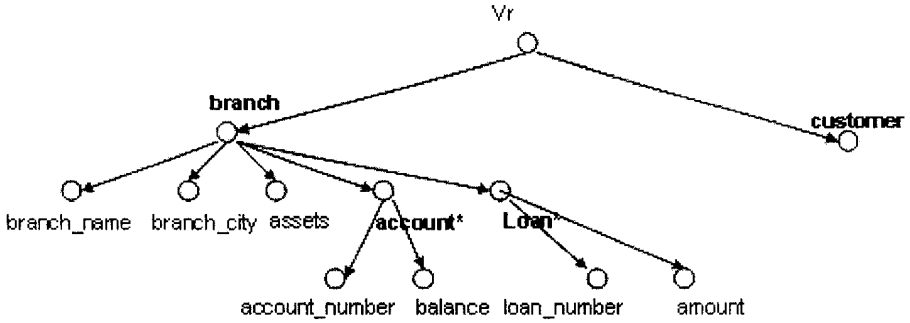

Vr
branch
customer
branch_name
branch_city
assets
account*
Loan*
account_number
balance
loan_number
amount

图 4

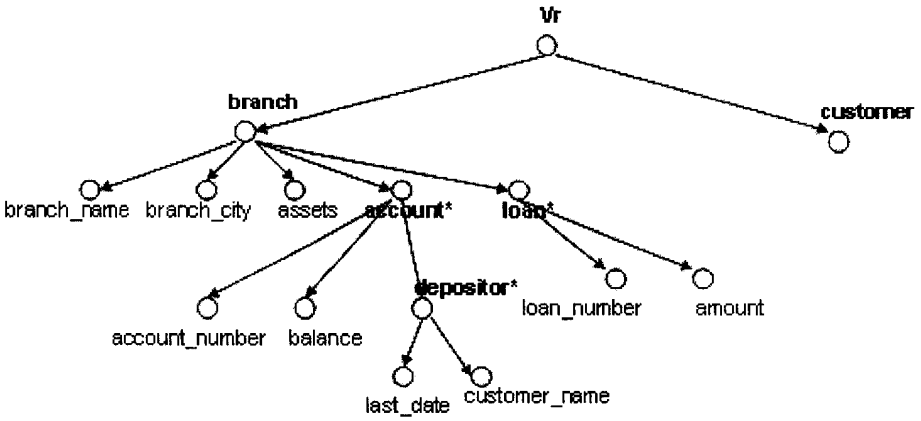

Vr
branch
customer
branch_name
branch_city
assets
account*
loan*
depositor*
loan_number
amount
account_number
balance
last_date
customer_name

图 5

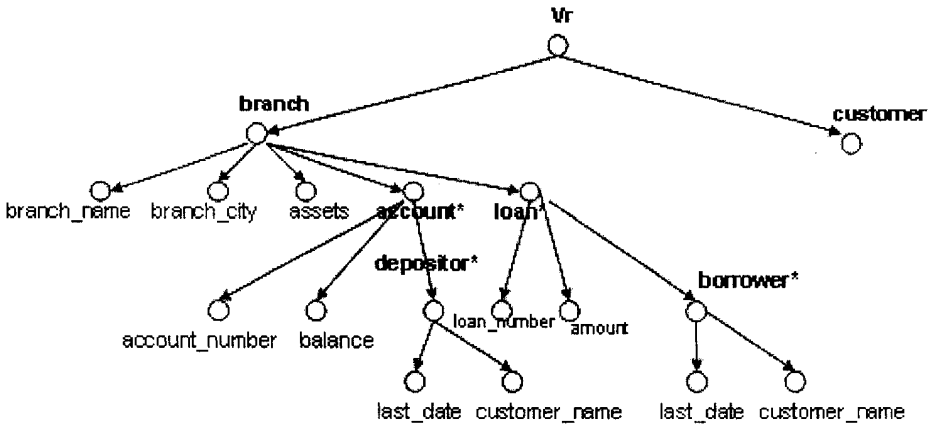

Vr
branch
customer
branch_name
branch_city
assets
account*
loan*
depositor*
borrower*
account_number
balance
loan_number
amount
last_date
customer_name
last_date
customer_name

图 6

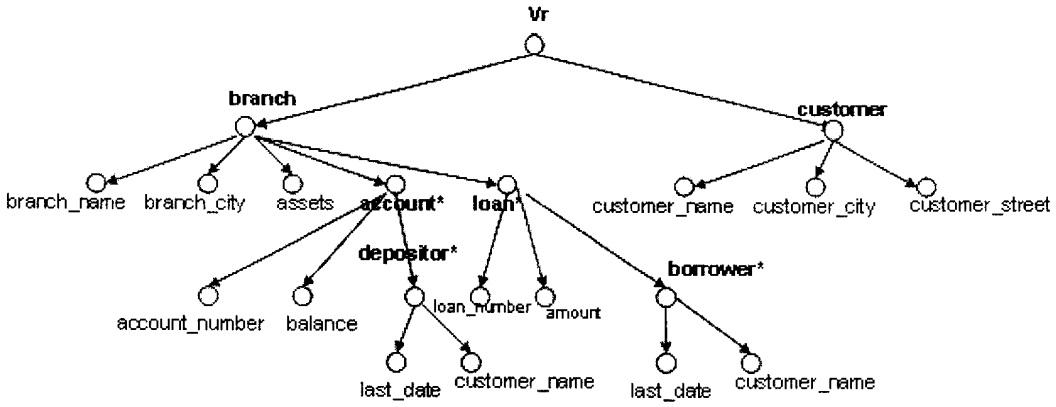

Vr
branch
customer
branch_name
branch_city
assets
account*
loan*
customer_name
customer_city
customer_street
depositor*
borrower*
account_number
balance
loan_number
amount
last_date
customer_name
last_date
customer_name

图 7

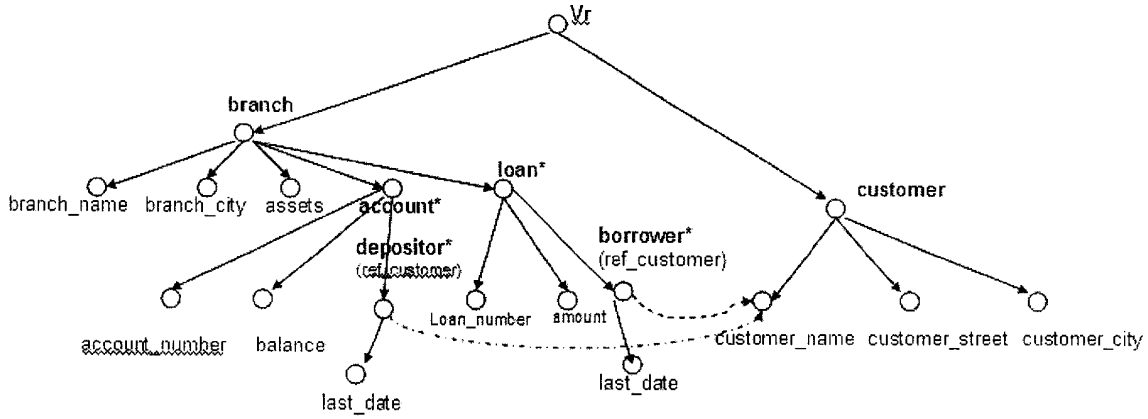

Vr
branch
loan*
customer
branch_name
branch_city
assets
account*
depositor*
(ref_customer)
borrower*
(ref_customer)
Loan_number
amount
account_number
balance
customer_name
customer_street
customer_city
last_date
last_date

图 8