OWL本体到关系数据库模式的映射*)

朱姬凤　马宗民　吕艳辉

(东北大学信息科学与工程学院　沈阳 110004)

摘　要　通过对现有本体存储模式的分析,以及对OWL本体和关系数据库模式之间的概念对应关系的分析,定义了从OWL本体到关系数据库模式的转换规则,给出了OWL本体存储模式的设计原则,并基于该原则提出了一种新的本体存储模式。针对本体描述语言OWL的构词特点,该模式提出将本体中的语义信息存放在不同的表中,以达到方便理解、结构稳定和提高效率的目的,适应于多数中小型本体的存储。

关键词　语义Web,OWL,本体,存储模式,关系数据库模式

Mapping OWL Ontologies to Relational Database Schema

ZHU Ji-feng　MA Zong-min　LU Yan-hui

(College of Information Science and Engineering, Northeastern University, Shenyang 110004, China)

Abstract　By analyzing existing ontology storage schemas and relations between the OWL ontologies and relational database schema, the formal rules of mapping OWL ontology to relational database schema are developed. On the basis, the design priciples of ontology's storing sechma in relation databases are given. A new storage schema is hereby proposed based on the principles. According to the characteristics of OWL ontology description language, the semantic informations of the ontology are stored in multiple relational tables. As a result, the schema is more understandable and stable. Furthermore the schema can increase the query efficiency and be suitble for storing middle scale ontology.

Keywords　Semantic Web, OWL, Ontology, Storage schema, Relational database model

语义Web[1]被看成是未来的智能网络,此类网络中的数据含有丰富语义,不仅人类能方便地阅读和访问这些数据,更重要的是这些数据更适合于机器理解和自动处理,因此也就能更有效地提高网络服务速度和质量。本体(Ontology)[2]作为语义Web的核心,从提出语义Web这一概念开始,它就受到了广泛关注。简单地说,本体就是关于可共享概念化体系的明确的形式化规范说明,从某种角度来看就是一个领域内的概念模式。对本体的研究包括对知识表示和组织的研究,目前许多学者致力于语义Web的研究工作。而要实现这一目标,最为重要的就是要构建丰富的本体。有两种常用的本体构建方法:(1)以现有网络为基础并基于本体构造语义Web网络站点[3],从而逐步将现有网络改造成语义Web,这称为自底向上构造方法;(2)以先建立的领域本体为基础设计站点底层数据库模式及相关网络服务,这称为自顶向下构造方法,而这种方法中极为关键的一步就是如何将本体的类、属性以及类之间的关系、属性之间的关系、属性约束等内容表示成为关系数据模式。

本体描述语言经历了RDF[4],RDFS[5],OIL[6],DAML+ONT[7],DAML+OIL[8],OWL[9]等发展阶段,2004年OWL成为W3C推荐标准。OWL由OWL Lite,OWL DL和OWL Full三个表达能力递增的子语言组成,这三个子语言分别针对不同的需求。整个OWL语言称为OWL Full,它最大可能地兼容了RDF和RDFS,包括语法和语义。OWL Full表达能力如此强大以致于可能无法判定,难以有推理软件能对OWL FULL所有成分提供完全有效的推理支持。OWL DL可看作OWL Full的一个子语言,包括了OWL语言所有成分,但有一定限制。它以描述逻辑为基础,在不失计算完全性和可判定性的条件下提供尽可能大的表示能力。OWL Lite是将OWL DL约束限制在一个更为严格的语言构造子集上。相比于OWL Full而言,OWL Lite容易掌握,也更容易实现,但表达能力受限。OWL Full表达的语义非常丰富,而与OWL DL相比,OWL Lite还具有更低形式复杂度[10,11]。为了简化、体现最基本的语义映射规则,在本文中主要探讨以OWL Lite为基础的映射规则。

目前,本体存储方法可以概括为以下几种。

(1)纯文本:纯文本方式可以直观显示本体语言的语法,易于理解,适用于本体的直接表示和存储。如Protégé[12],OntoEdit[13]等工具,可以直接生成扩展名为.OWL的文件并可以对本体进行一些开发、维护、推理、查询。这种形式比较简单,但是在本体较大、概念和实例比较多的情况下,效率会有所降低,不适合中大规模本体的存储和查询管理服务。

(2)专门管理工具:还有一些软件,如OMM[14]等,支持对RDF,OWL的存储管理,并能提供各种接口,还可使用查询语言对本体进行查询。但这些技术仍不够成熟,存储效率也不及关系数据库。Kowari[15]是一个本体存储管理系统,它在存储设计中利用数据冗余提高查询效率,存储容量也比较大。

(3)关系数据库:数据库并不是表达本体的最好方式,因为本体含有丰富的语义信息,其内在逻辑性比关系模式复杂得多。但是,数据库开发研究历史较长,相关工具软件较多,相应技术比较成熟,适合大规模数据存储,效率高、易管理、便于查找[16-24]。目前支持语义信息存储的技术尚未成熟,对于

*)教育部新世纪优秀人才支持计划(NCET20520288)。朱姬凤　硕士研究生;马宗民　教授,博士生导师;吕艳辉　博士研究生。

海量数据本体的存储和管理，数据库是较好的选择。protégé和Jena[25]等都支持将本体导入数据库，但仍存在少量语义方面的问题。在尽可能保持语义的前提下，基于关系数据库的本体存储技术是本文目标所在。

1 现有基于关系数据库的本体存储模式

基于关系数据库存储本体的方法主要有水平模式、垂直模式和分解模式等。

(1) 水平模式

这种模式下只在数据库中创建一张表，表的一条记录就是本体的一个实例，表的列对应本体的属性[26]。这种方法直接将所有属性转换为表的属性列，算法简单、易于实现，但受限于表的列数限制而无法存储较大的本体，并且在OWL本体中，对于个体的描述往往是没有将它所有性质都明确表示出来或者没有说明是否有值或为何值，因而本体映射时会产生大量空值，不仅浪费存储空间，而且增加了系统维护索引的代价；另外，因本体发生变化而需要频繁地进行删除、添加表中的列及修改表的模式等操作，所付出的代价很大。

(2) 垂直模式

这种模式只包含一张三元组表，表中每个记录对应一个RDF三元组[25,27]，即用三元组的形式来描述本体全部信息。这种模式简单、结构稳定，要修改本体只需修改表中相应元组。但是这种模式可读性差，难以在此基础上做进一步的应用工作(比如查询)，所以极不利于本体的应用推广。

(3) 分解模式

这种模式的基本思想是将数据库进行模式分解[16]。根据分解对象的不同，此种模式又可分为两种：

基于类的分解模式，即分别为本体的每个类创建一张表，表名为类名，列为类的属性。这种模式下表结构清晰，但当本体中类或属性发生变化时，表结构要随之变化，所以在数据库维护方面有一定难度。

基于属性的分解模式，即分别为本体的每个属性创建一张表，表名为属性名，每个表都包含两列，分别代表RDF三元组中的Subject和Object。在该模式中，查询类的隐含实例的代价很大，效率较低[26]。

上述两种分解模式随着本体的变化都要不断地创建和删除表，这在数据库系统中需要付出很大的代价。实际应用中可以将上述几种模式进行混合使用。例如，可以将两种分解模式存储方法并用，即在本体中定义一个类就为该类创建一个表(基于类分解)，在本体中定义一个属性就为该属性创建一个表(基于属性分解)[26]。这种方法的缺点是当处理大规模数据时效率较低。

文献[28]提出的映射方法是一种基于类的分解模式，该方法在某些相关语义的保留方面有所欠缺。文献[29]和[30]分别提出了基于关系数据库的大规模本体存储模式，此模式基于OWL Lite的构词规则对整个本体进行分解，设计了多个表存储OWL的资源和关系，将较常用的本体查询信息抽取出来单独存放，减少了查询时进行表连接的代价，提高了操作效率和本体存储的可扩展性。但这种方法的设计结构不够清晰，查询、修改等操作花费代价大，考察资源之间的关系时需要连接多个表，最重要的是本体语义的保留也不够充分。

参照上述存储模式，通过对OWL构词特征的分析，结合本体映射研究的需要，本文提出一种基于关系数据库的存储OWL本体的方法。这种方法根据OWL的构词特征，用相应关系表来保存本体中的资源及资源之间的关系，描述它们的特征和约束，刻画出本体的层次和等价关系。

2 OWL Lite

语义Web的本体语言OWL有三个子语言OWL Full，OWL DL和OWL Lite。OWL Lite是OWL DL的一个子语言，其主要特点在于它针对工具的开发者给出相对简单的语言特征集。OWL Lite遵守与OWL DL一样的语义约束，允许推理机进行推理以保证可靠的性能。OWL Lite是OWL DL的简单版本，对于RDF词汇(如类、属性的不相交等)的使用有一定的约束和限制[10]。

OWL Lite的构词[10]包括：

(1) 类(Class)

owl:Class用于定义类。OWL提供六种声明类的方式：①通过一个类标识符(引用URI)；②通过枚举类的个体；③通过属性约束；④用两个类的交集操作；⑤通过两个类的并集操作；⑥通过一个类的补集操作。后五种声明方式声明的一般是匿名类。OWL Lite只支持第①、③、④种方式声明的类。

owl:subClassOf描述类之间的层次关系。

owl:equivalentClass表示一个类等同于另一个类，前者必须是一个命名类，而后者可以是命名类或匿名类。

owl:intersectionOf表示通过两个类的交来定义一个新的类。

(2) 属性(Property)

owl:ObjectProperty表示对象属性，用于描述类与类之间的关系。

owl:DatatypeProperty表示数据类型属性，用于描述对象与数值型值之间的关系。

owl:subProperty表示属性之间的层次关系，表明一个属性是另一个属性的子属性。

rdfs:domain表示属性的定义域，它限定了属性的作用范围，即属性是用于描述哪个类。

rdfs:range表示属性的值域，它限制了属性值的可取值范围。

owl:inverseOf描述的也是两个属性之间的关系，表示两个属性互为逆属性。

owl:equivalentProperty表示属性间的等价。

owl:FunctionalProperty表示函数属性，对每一个个体在该属性上只能有唯一不同的值。

owl:InverseFunctionalProperty表示反函数属性，正好和函数属性相反。

owl:TransitiveProperty，owl:SymmetricProp-erty分别表示属性具有传递性、对称性。

owl:allValuesFrom属性的值约束，它将一个约束类与一个类描述或一个数据范围相联系，也可认为一个包括owl:allValuesFrom的约束描述了一个匿名类，其所有个体的属性值都是来自于类描述的类外延或者是指定数据范围之内的数据值。

owl:someValuesFrom属性的值约束，它将一个约束类与一个类描述或一个数据范围联系在一起。一个包含owl:someValuesFrom的约束描述了一个匿名类，其所有个体的相关属性值至少有一个值是类描述的一个实例或者是数据值域内的一个数据值。

owl:hasValue属性的值约束，它将一个约束类与值v相

联系，而 v 既可以是一个个体也可以是一个数据值。一个包括 owl:hasValue 的约束描述了一个匿名类，这个类中的所有个体的相关属性值至少有一个值在语义上是与 v 等价的(它也可以是其他的值)。

owl:maxCardinality，owl:minCardinality，owl:cardinality 分别是属性的最大、最小基数约束和基数约束，在 OWL Lite 中其取值只能为 0，1。

(3) 个体(Individual)

owl:Thing 可以用于定义个体。

rdf:type 可以用于说明某个个体属于哪个类。

owl:AllDifferent 指定多个个体两两不同。

owl:differentFrom 声明一个个体与其它的个体不相同。

owl:sameAs 声明两个个体是相同的。

此外，OWL Lite 还包括一些用于表示本体版本信息、头信息的构词。

3 基于关系数据库存储 OWL 本体的方法

3.1 形式化描述 OWL 和 RDBS

关系数据库模式由一组关系模式组成，包括数据库的基本表结构和完整性约束两部分[16]。基本表定义了关系表的结构、属性列及其数据类型等。完整性约束定义了语义施加在数据上的约束，包括主键约束、外键约束、取值约束等。关系数据模式有多种表达形式，本文使用的是用 SQL DDL(数据定义语言)定义的关系数据模式。采用这种形式表达关系数据库模式的好处是通用性强，所有的关系数据库都支持 SQL 语言，而且 SQL 语言早已标准化。

OWL 本体由类、属性、个体以及一组公理组成，其中类可以形式化表示领域内的概念，属性反映了类与类、类与数据类型的关系，个体是类的某个具体的实例。相对于 RDBS，OWL 更加关注“关系”，即属性。在 OWL 中除了表示关系的存在，还提供了描述“关系”的语言成分，比如基数约束、属性的函数性、属性的逻辑性、子属性和等价属性等。

为便于后面表述 OWL 本体到关系数据库模式的映射规则，现给出两者的形式化描述。

定义 1 关系数据库模式(relational database schema，RDBS)[16,28] R 是一个六元组 $R = \{Name, col, records, PK, FK, Check\}$，其中 $Name$ 是名字集，col 是表中列的集合，$records$ 是表中的记录集合，PK 是表的主键，FK 是表的外键，$Check$ 是约束集合。

Name：名字集 $Name$ 是一个有限集合，由两两不相交的三部分组成：$Name = ET \cup RT \cup DT$。$ET$ 是实体表名的集合，每一个实体表包含一些实例数据行(记录)，这些行描述的是现实世界中的实体；RT 是联系表名的集合，每一个联系表也包含一些实例数据行(记录)，这些行描述了实体间的联系；DT 是数据类型名集合，每个数据类型名是 DBMS 预定义的类型名，它规定数据的取值范围。

col：表的列集，$\forall t \in ET \cup RT$，都有一个有限的非空列集合 $col(t)$，且每个列 $c \in col(t)$ 有一个相关的预定义数据类型 $datatype(t) \in DT$ 作为它的取值范围。

records：表的所有记录的集合，$\forall t \in ET \cup RT$，都有 n ($n \geqslant 0$)个 $record \in t$，用 $records \in (t)$ 表示表 t 的所有记录。

PK：表的主键，$\forall t \in ET \cup RT$，t 有且仅有一个主键 $PK(t)$，其值唯一决定了 t 中每行实例数据，且要么 $PK(t) \in col(t)$(此时称它为单主键，只有实体表有且仅有单主键)，要么 $PK(t) \subseteq col(t)$(此时称它为复合主键，只有联系表有且仅有复合主键)。

FK：外键，$\forall t \in ET \cup RT$，t 可能有 m ($m \geqslant 0$)个外键；每个外键 $FK(t,r) \in col(t)$ 引用其他实体表 $r \in ET$ 单主键的外键，$FK(t,r) \in col(t)$ 满足 $value(FK(t,r)) \subseteq value(PK(r)) \cup \{null\}$，$PK(r) \in col(r)$。value(*)表示“*”的值域。

Check：*Check* 是一组值约束集合，可以看作强制类型限制。*Check*(*P*)表示表 t ($t \in ET \cup RT$)中所有记录 records(t)必须满足谓词 P。

定义 2 一个 OWL 本体 O 是一个二元组 $O = (ID, Axiom)$，其中

ID 是一个有限的 OWL 标识符集，由两两不相交的四个子集组成：$ID = CID \cup PID \cup ITID \cup DTID$。

CID 是类标识符集，包含本体所定义的全体类标识符，含有两个预定义类，owl:Thing 和 owl:Nothing，前者是所有类的父类，后者是所有类的子，即不包含任何个体，是 owl:Thing 的补。

PID 是属性标识符集，包含本体所定义的全部属性标识符。它可分为两个子集：数据类型属性标识符集 $DPID$ 和对象属性标识符集 $OPID$。$DPID$ 包含本体定义的所有数据类型属性标识符；$OPID$ 包含本体定义的所有对象属性标识符。

$ITID$ 是个体标识符集，它包含本体所定义的全部个体的标识符。

$DTID$ 是数据类型标识符集，包含本体所用的 XML Schema 的全部预定义数据类型标识符。

$Axiom$ 是一个有限的 OWL 公理集。它由三个两两不相交的子集组成：$Axiom = CAxiom \cup PAxiom \cup IAxiom$。

$CAxiom$ 是类公理集，包含本体所定义的全体类公理。$PAxiom$ 是属性公理集，包含本体所定义的全部属性公理。$IAxiom$ 是个体公理集，包含本体中所定义的全部个体公理。

例如，子类约束 subClassOf $(c,d) \in CAxiom$ 表示 c 是 d 的子类；equivalentClass $(c,d) \in CAxiom$ 表示类 c 的外延正好与另一个类 d 的外延相同；disjointWith $(c,d) \in CAxiom$ 表示类 c 的外延与另一个类 d 的外延没有相同成员；subPrpertyOf $(c,d) \in PAxiom$ 表示属性 c 是属性 d 的子属性；equivalentProperty $(c,d) \in PAxiom$ 表示属性 c 与属性 d 等同；属性的定义域约束 domain $(p) \in PAxiom$ 表示若 $P \in DPID \cup OPID$ 且定义域为 $c \in CID$，则 domain $(p) = c$；属性的值域约束 range $(p) \in PAxiom$ 表示若 $p \in DPID$ 的值域为 $dt \in DTID$，则 range $(p) = dt$，若 $P \in OPID$ 的值域是 $c \in CID$，则 range $(p) = c$；函数约束 functional $(p) \in PAxiom$ 表示 $P \in DPID$ 是函数属性，即 p 在 range (p) 中取值唯一；属性互逆约束 inverse $(p,q) \in PAxiom$ 表示 $p, q \in OPID$，domain (p) = range (q) 且 range (p) = domain (q)；sameAs $(c,d) \in$ Iaxiom 表示个体 c 与个体 d 是同一个体；differentFrom $(c,d) \in Iaxiom$ 表示个体 c，d 不同[4]。

3.2 OWL 与 RDBS 的概念对应关系和映射规则

OWL 本体使用 XSD (XML Schema Datatype)基本数据类型[31]，可分为时间日期型、字符型、布尔型、基本数值型和约束数值型等五类，主要包括：data，datatime，decimal，integer，time，boolean，float，int，negativeInteger，short，long，nonNegativeIteger，string，nonPositiveInteger，positiveInteger 等。前四种类型都可以直接和 SQL 标准中的基本数据类型对应，

但第五种类型不能直接对应到SQL标准数据类型，不过若在映射时增加一个取值约束，那么在SQL中就能够表达这些数据类型了。例如一个属性 A 它具有 positiveInterger 取值范围，在映射到SQL数据定义语言时除了定义一个 int 型的属性 A 之外再对 A 增加一个约束 check（A > 0）。表1列出了OWL语言与SQL语言的数据类型的对应关系。

表1 OWL数据类型与SQL中的数据类型的对应关系表

数据类型	OWL数据类型属性的值域	SQL中的数据类型
时间日期型	xsd:date	DATE
	xsd:time	TIME
	xsd:dateTime	DATETIME
布尔型	xsd:boolean	BOOLEAN
字符型	xsd:string	CHAR/VARCHAR/TEXT
基本数值型	xsd:decimal	DECIMAL
	xsd:integer	INT
	xsd:short	SMALLINT
	xsd:long	LONG
	xsd:float	FLOAT
	xsd:double	DOUBLE PRECISION
约束数值型	xsd:negativeInteger	INT + check 约束
	xsd:nonNegativeInteger	INT + check 约束
	xsd:positiveInteger	INT + check 约束
	xsd:nonPositiveInteger	INT + check 约束

经过对OWL本体的分析以及对OWL本体和关系数据库模式[32,33]的形式化定义，可得出OWL本体与关系数据库模式各元素之间的对应关系，如表2所示。

表2 OWL本体中的元素与RDB Schema中的元素对应关系表

OWL本体中的元素	RDB Schema中的元素
类	基本表
以某个类为定义域的数据类型属性及其XML模式数据类型	表的非外键属性列及其相应的基本数据类型
以某个类为定义域的数据类型属性，并且是函数属性	表的主键(若已有主键，则不设为主键)
类A的对象属性，其range为类B	类A对应的表有一个外键引用类B对应表的主键
类A与类B间的父子关系，即类A是类A对应的表增加主键列，同时这个主键列设为引用类B对应表的主键的一个外键	
类B的子类	所属类所对应的表的记录行(相关的列的值)
个体(及相关的属性值)	

除了上述的对应关系，为了尽可能保持本体中的丰富语义，还需添加一些辅助表，最后得出OWL本体到关系数据库模式的映射规则。为了方便表示映射规则，引入以下辅助函数：

(1) createtable (name, $p1$, $p2$, …, pn)：创建一个名为name的表，属性列有 $p1$, $p2$, …, pn。

(2) getName (id)：取OWL本体O中标识符的名字，为字符型的值。

(3) getType (t)：取与XML Schema数据类型相对应的SQL数据类型。

(4) getTable (c)：取OWL本体O中类 c 对应的关系数据库模式 R 中的表，即若 $c \in CID$ 对应的表是 $t \in ET \cup RT$，则 getTable (c) = t。

针对OWL中的类、属性和个体，下面给出它们到关系数据库模式的映射规则。

3.2.1 类映射规则

(1) $\forall c \in CID \rightarrow$ (create (Thing, className, cURInamespace, cURIlocalname) $\in ET \cup RT \wedge PK$ (Thing) = "className" $\in col$ (Thing))

对一个本体建立表Thing，该表中包含的属性列有：className, URInamespace, URIlocal-name, subclassOf，其中className为主键，将所有类映射成此表的记录，subclassOf是一个引用本表的外键，其值为当前类的直接超类，以此表明类之间的层次关系。此表相当于全部类的总表。

(2) $\forall c \in CID \rightarrow t$ = getName (c) $\wedge$ create (t, individual-name) $\in ET \cup RT$。

将OWL本体中的每个类映射为关系数据库模式中的一个表，且表名与类名相同，表中添加一个个体名列。

(3) $\forall c1, c2 \in$ CID $\wedge$ subClassOf ($c1$, $c2$) $\rightarrow t1$ = getTable ($c1$) $\in$ ET $\wedge t2$ = getTable ($c2$) $\in ET \wedge dp =$ getName (PK ($t2$)) $\in col$ ($t2$) $\wedge PK$ ($t1$) = $dp \wedge FK$ ($t1$, $t2$) = dp

若OWL本体中类 $c1$ 是类 $c2$ 的子类，则在类 $c1$ 对应表中添加一个与类 $c2$ 对应表的主键同名的列，并将此列作为此表的主键同时也作为一个外键来引用类 $c2$ 对应表的主键。

因为在OWL中所有类都被看成是owl:Thing的子类，所以类映射所得的表中都应该添加一个列(类名)作为外键引用表Thing的主键。

(4) $\forall C = A \cap B \in CID \rightarrow t3$ = getTable (C) $\in ET \wedge t1$ = getTable (A) $\in ET \wedge t2$ = getTable (B) $\in ET \wedge dp1 =$ getName (PK ($t2$)) $\in col$ ($t2$) $\wedge dp2 =$ getName (PK ($t3$)) $\in col$ ($t3$) $\wedge FK$ ($t1$, $t2$) = $dp1 \wedge FK$ ($t1$, $t3$) = $dp2$

对于以交集方式定义的类映射成的表，比如类 $C = A \cap B$，则在类 C 所对应的表中添加两个属性列，这两个属性列作为外键分别引用类 A 对应表和类 B 对应表的主键。同时由于类 C 是类 A 与类 B 的交集，所以必然类 C 也是类 A, B 的子集，这时也需要参考上面的映射规则(3)进行相应的处理。

3.2.2 属性映射规则

(1) ($\forall dp \in$ DPID $\rightarrow a$ = getName (dp) $\in col$ (getTable (domain (dp))) $\wedge$ datatype (a) = getType (range (dp)) $\in DT$

将本体的每一个数据类型属性及其值域映射为其定义域所对应表的列及其相应数据类型。

(2) ($\forall dp \in$ DPID $\wedge$ functional (dp) $\wedge$ t = getTable (domain (dp)) $\in ET \cup RT \wedge \neg \exists PK$ (t) $\in U$ (t) $\rightarrow t \in ET \wedge PK$ (t) = getName (dp) $\in U$ (t)) $\vee$ ($\forall dp \in$ DPID $\wedge$ functional (dp) $\wedge$ t = getTable (domain (dp)) $\in ET \wedge \exists PK$ (t) $\in col$ (t) $\rightarrow$ Unique (t) = getName (dp) $\in$ col (t)

将本体中函数属性映射为其定义域所对应的表的主键，此表为实体表；若该表已经具有主键，则不再需要添加新的主键。

(3) $\forall op \in$ OPID $\wedge t1$ = getTable (domain (op)) $\in ET \wedge t2$ = getTable (range (op)) $\in ET \rightarrow FK$ ($t1$, $t2$) = $op \in col$ ($t1$)

将本体中每个对象属性映射为其定义域所对应的表外键以便引用其值域对应表的主键。

(4) Create (PropertyTable, propertyName, propertyType, domain, range) $\in ET \wedge$ PKPropertyTable) = propertyName

建立一张属性表PropertyTable，将OWL本体中所有属性映射成为此表的记录，该表包含的列有：属性名◊属性类型(即表明此属性是对象属性还是数据类型属性)、定义域domain、值域range，其中属性名为此表的主键。

(5) Create (Property_relation, property1, property2, relationtype) $\in RT \rightarrow FK$ (Property_relation, PropertyTable) = property2

建立一张描述属性结构关系的表 Property_relation。该表包含的列 property1, property2, relationtype 分别表示属性名 1、属性名 2、关系类型。目前的属性关系类型有 sub-PropertyOf、equivalentProperty、inverseOf,分别表示属性 property2 是 property1 的父属性、等价属性和逆属性。property2 作为此表的外键,该外键引用上一规则所建立的属性表中的属性名,并将全体属性中具有上述关系的属性映射成此表中的记录。

(6) Create (Property_Characteristics, propertyName, transitive, symmetric, Functional, inverseFunctional) $\in RT \rightarrow PK$ (Property_Characteristics) = propertyName

建立一张属性特征表 Property_Characteristics,此表所包含的列为属性名,transitive, symmetric, Functional, inverseFunctional。将全体属性中具有这些特征的内容全部映射成为此表对应的元组。

(7) Create (Property_restriction, className, propertyNa-me, restrictionType, value) $\rightarrow FK$ (Property_restriction, Thing) = className $\wedge FK$ (Property_restriction, PropertyTable) = propertyName $\wedge PK$ (Property_restriction) = (className, propertyName, restrictionType)

为全体属性建立一张属性约束表,表名为 Property_restriction,该表包含的属性列有类名,属性名,约束类型以及相关约束值。此表可以作为类表与属性表之间的联系表。其中,类名为外键,引用 Thing 表的主键,属性名也为一个外键,引用 PropertyTable 的主键,这两个外键再加上约束类型合并起来成为此表的一个复合主键。

3.2.3 个体映射规则

(1) 个体映射成为所属类所对应表的记录。

(2) Create (Individual_relation, relationType, class1, individal1, class2, individual2) $\in RT$

个体之间的关系用表"Individual_relation"描述,此表中包括的列有:关系类型、类名 1、个体名 1、类名 2、个体名 2。目前,关系类型有:differentFrom 和 sameAs,differentFrom 表明两个个体不同(AllDifferent 表示一定数量的个体之间两两不同,但由于关系模式只能表示二元关系,故需要将多元关系分解成为多个二元关系,用 differentFrom 表示);sameAs 说明 IndvId1 与 IndvId2 相同。将全体个体中具有上述关系的个体映射成为此表的记录。

4 实验

本实验基于 Java 平台,使用惠普实验室开发的 jena[34] 开发包及最初由 OTI, IBM 公司的 IDE 产品开发组创建的 Eclipse[35,36] 集成开发环境。

Eclipse 是一个开放源代码的、基于 Java 的可扩展开发平台。就其本身而言,它只是一个框架和一组服务,用于通过插件组件构建开发环境。尽管 Eclipse 是使用 Java 语言开发的,但它的用途并不限于 Java 语言。Eclipse 框架还可用作与软件开发无关的其他应用程序类型的基础,比如内容管理系统。

由于手动创建一个本体是很复杂和繁琐的一件事,因此解析用的实例本体采用的是用 Protégé 直接生成本体。

遵循前面的映射规则设计一个从 OWL 本体到关系数据库模式的自动转化工具 OWL2RDBS。它以 OWL 文档作为输入,将 OWL 本体映射为关系数据库模式之后输出为 SQL 脚本。整个系统由 4 个主要模块组成:

(1) OWL 文档解析模块 OWL Parser,负责读取和解析 OWL 文档,将所获得信息存储于系统内部模型中,为映射工作做准备。

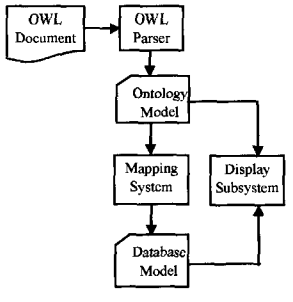
(2) 映射转化模块 Mapping System,从 OWL 本体到关系数据库模式的六条映射规则,将 OWL 本体模型转化为系统内部的关系数据库模式,这是整个系统的核心。

(3) SQL 脚本生成模块,包含一个系统内部存放关系数据库模式的模型,含经转化模块所得到的全部信息,并能生成标准的 SQL 脚本。

(4) 显示模块 Display Subsystem,负责将系统内部的 OWL 本体模型和关系数据库模式模型以树形结构显示出来。

OWL2RDBS 的各个模块之间的关系和系统处理流程如图 1 所示。

将 Protégé 直接生成的本体文档作为 OWL2RDBS 的输入,运行结果如图 2 所示,左侧树所显示内容为 OWL 本体中的类、数据类型属性和对象属性等信息,右侧则为映射后的关系数据库模式。



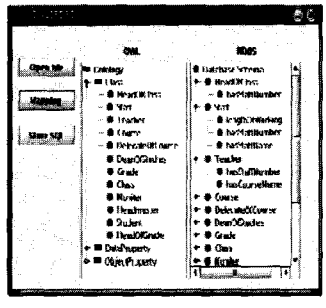


图 1 OWL2RDBS 系统流程图 图 2 OWL 结构和 RDBS 的结构

结束语 虽然上述映射规则在将 OWL 本体模型映射为关系数据库模式时并不能把 OWL 本体中蕴含的所有语义信息完全映射到关系数据库模式,但仍最大限度地保持了 OWL 最为重要和基本的语义,除此之外还尽可能地将一些细微语义保留了下来。本文只是暂定讨论范围限于 OWL Lite,如果是其它子语言所写的本体,则需要在此规则基础之上再做进一步扩展。

参 考 文 献

[1] Berners Lee T, Hendler J, LassilA O. The Semantic Web[J]. Scientific American, 2001, 284(5): 34-43

[2] Hendler J. Ontologies on the Semantic Web[J]. IEEE Intelligent Systems, 2002, 17(2): 73-74

[3] Stojanovic L, Stojanovic N, Volz R. Migrating data-intensive Web sites into the Semantic Web[C]// Proc. of the 17th ACM Symposium on Applied Computing. New York: ACM Press, 2002: 1100-1107

[4] Lassila O, Swick R R. Resource Description Framework (RDF): Model and Syntax Specication. ecommendation, World Wide Web Consortium, February 1999. http://www.w3.org/TR/REC-rdf-syntax/

[5] RDF Schema. http://www.w3.org/TR/rdf-schema/

[6] Horrocks I, Fensel D, Broekstra J, et al. The Ontology Inference Layer OIL[EB/OL]. http://www.Ontoknow-ledge.org/oil/, 2000-09-10

(下转第 205 页)

指标及影响流行的因素指标确定疟疾疫情影响因素，建立统计学模型，研究各影响因素与疟疾发病水平间的关系，将World Wind客户端作为专题应用模型运行的载体，对疟疾高发原因进行空间化表达，生成病例分布图、人蚊接触机率图、土地利用图、NDVI图、地形地貌图、气温、气湿、降雨量分布图，以及干预控制图。最终目的为综合利用流行病学、统计学、3S(GIS,GPS,RS)等技术建立风险预测模型，分析评价疟疾高发的主要危险因素，为当地疟疾防控工作的提供科学决策依据。

结束语 数字地球原型系统中的三维表现和虚拟现实技术，需要与业务系统结合才体现出其价值和生命力，既能够作为吸引眼球的手段，又能成为模型运行的载体，具有强大的分析、计算功能。为此，将强大的GIS服务器与三维数字地球原型系统相结合是很好的选择，数字地球原型系统不仅具有良好的三维展示能力，将GIS Server作为服务内容的后台支撑，又可以很好地解决三维数字地球系统GIS分析不足的问题。

参 考 文 献

[1] 王慧，申家双，陈冬阳. 一种高性能的大区域遥感影像管理模型[J]. 海洋测绘，2006，26(3)：71-74

[2] Bell D G, Kuehnel F, Maxwell C. NASA World Wind: Opensource GIS for Mission Operations. IEEEAC paper #1048, Version 2, Updated October 2006

[3] Tooth S. Virtual globes: A catalyst for the re-enchantment of geomorphology [J]. Earth Surface Processes and Landforms 2006, 31: 92-94

[4] 张慧杰，孙吉贵，刘雪洁. 大规模三维地形可视化算法研究进展[J]. 计算机科学，2007，34 (3)：10-16

[5] 张立强. 构建三维数字地球的关键技术研究. 博士学位论文[D]. 北京：中国科学院遥感应用研究所，2004

(上接第169页)

[7] DAML-ONT. http://www.daml.org/2000/10/ damlont.html

[8] DAML+OIL(March 2001) Reference Description. http://www.w3.org/TR/daml+oil-reference, 2003-06-10

[9] Smith M K, Welty M, McGuinness D. Web Ontology Language (OWL) Guide. W3C Working Draft, Mar 2003. http://www.w3.org/TR/2003/WD-owl-guide-20030331/

[10] Bechhofer S, van Harmelen F, Hendler J, et al. OWL Web Ontology Language Reference, W3C Recommendation. http://www.w3.org/TR/owl-ref/

[11] Mcguinness D L, Harmelen F V. OWL Web OntologyLanguage Overview (W3C Recommendation). http://www.w3.org/TR/2004/REC-owl-features-20040210/. February 2004

[12] The protege ontology editor and knowledge acquisition system. http://protege.stanford.edu/

[13] OTK tool repository: Ontoedit. http://www.ontoknowledge.org/tools/ontoedit.shtml

[14] Ontology Middleware System Documentation. http://www.ontotext.com/omm

[15] Wood D, Gearon P, Adams T. Kowari: A Platform for Semantic Web Storage and Analysis // Proc. of WWW 2005. Chiba, Japan, May 2005

[16] 萨师煊，王珊. 数据库系统概论. 第三版. 北京：高等教育出版社，2004

[17] Broekstra J, Kampman A, Harmelen F. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema[A] // Proc. of the 1st International Seman-tic Web Conference[C]. Sardinia, Italy, June 2002: 54-68

[18] Ma Li, Su Zhong, Pan Yue, et al. RStar: An RDF Storage and Query System for Enterprise Resource Management[A] // Proc. of CIKM 2004[C]. New York, NY, USA, 2004: 484-491

[19] Carroll J J, Reynolds D, Dickinson I, et al. Jena: Implementing the Semantic Web Recommendations[A] // Proc. of WWW 2004 [C]. May 2004: 17-22

[20] Harris S, Gibbins N. 3store: Efficient Bulk RDF Storage[A] // Proc. of the 1st International Workshop on Practical and Scalable Semantic Systems[C]. Sanibel Island, Florida, USA, 2003: 1-15

[21] Wang E, Kim Y, et al. Ontology Modeling and S-torage System for Robot Context Understanding. KES, 2005(3): 922-929

[22] Zhou Jian, Ma Li, et al. Minerva: A Scalable OWL Ontology Storage and Inference System. ASWC, 2006: 429-443

[23] Das S, Chong E I, Eadon G, et al. Supporting ontology-based semantic matching in RDBMS[A] // Proc. of VLDB[C]. Toronto, Canada, August 2004: 1054-1065

[24] Roldan-Garcia M M, Montes J F A. A Tool for Storing OWL Using Database Technology. http://www.mindswap.org/2005/OWLWorkshop/sub1.pdf

[25] McBride B. Jena: implementing the RDF model and syntax specification [R]. Technical Report. Hewlett Packard Laboratories, Bristol, 2000

[26] Agrawal R, Somani A, Xu Y. Storage and querying of e-comm-erce data [A] // Peter M G A, Paolo A, eds. Proc of the 27th VLDB[C]. Roma: Morgan Kaufman Piblishers Inc, 2001: 149-158

[27] Alexaki S, Christophides V, Karvounarakis G, et al. On storing voluminous RDF description: the case of Web portal catalogs [A] // Mecca G, Simeon J, eds. Proc. of the 4th WebDB in conjunction with ACM SIGMOD'01 [C]. Bristol: Hewlett Packard Laboratories, 2001: 43-48

[28] 许卓明，黄永菁. 从OWL本体到关系数据库模式的转换[J]. 河海大学学报(自然科学版)，2006，34(1)：95-99

[29] 李曼，王琰，赵益宇，等. 基于关系数据库的大规模本体的存储模式研究. 华中科技大学学报(自然科学版)，2005，33(12. 增刊)：217-220

[30] 周扬. 基于关系数据库的本体映射方法研究[M]. 长春：吉林大学，2006：31-35

[31] Biron P V, Permanente K, Malhotra A. XML Schema Part 2: Datatypes Second Edition (W3C Recommen-dation). http://www.w3.org/TR/ xmlschema22/, 28 October 2004

[32] 王能斌. 数据库系统教程：上册[M]. 北京：电子工业出版社，2002：220-238

[33] XU Zhuoming, CAO Xiao, et al. Formal approach and automated tool for translating ER schemata into OWL ontologies [C] // Dai H, Srikant R, Zhang C, eds. Advances in Knowledge Discovery and Data Mining (PAKDD2004 Proceedings, LNAI Vol. 3056). Berlin: Springer, 2004: 464-475

[34] Carrol J, McBride B. The Jena Semantic Web Toolkit. Public api, HP-Labs, Bristol, 2001. Avalable at: http://www.hpl.hp.com/semweb/jena-top.html

[35] Object Technology International, Inc. Eclipse Platform Technical Overview [EB/OL]. http://www.eclipse.org/ whitepapers/ eclipse-overview.pdf, February 2003

[36] 赵勇. Eclipse简介和插件开发[EB/OL]. http://www-900.cn.ibm.com/developerWorks/cn/java/I-eclipse-plugin, 2004-11