

基于关系数据库的大规模网络本体存储与查询研究

邱彦章, 郭 亮

(长安大学 电子与控制工程学院, 陕西 西安 710064)

摘 要: 针对互联网中大规模本体的有效存储和无缝访问的问题, 结合 OWL 本体语言的成分与组织特点, 提出了一种易于理解且结构稳定的关系数据库(RDB)模式, 利用 RDB 对大规模的互联网本体进行存储. 在此基础上, 利用动态生成的 RDB 视图将标准的本体查询解释为 SQL 查询, 在保持标准本体查询的灵活性和丰富性的同时, 使基于 RDB 的本体数据存储相对于上层的面向本体互联网应用保持透明.

关键词: 语义网; 网络本体; 关系数据库; SQL; OWL; SPARQL

中图分类号: TP311

文献标识码: A

文章编号: 1000-7180(2011)12-0137-04

Storage and Querying of Large Scale Web Ontology in Relational Database

QIU Yan-zhang, GUO Liang

(School of Electronic and Control Engineering, Chang'an University, Xi'an 710064, China)

Abstract: In regard to the problems of storage and seamless access of large scale ontology on the Web, a comprehensive and stable relational database (RDB) schema, which is in accordance with the ingredients and organization of OWL languages, is constructed for storing Web ontology in RDB. Besides, an auto-generated RDB views based method is approached to translate the classical ontology queries to SQL queries. With this translating method, the characteristics, such as high flexibility and high richness, of classical ontology query can be maintained in RDB based ontology access, and the RDB based ontology storage becomes transparent to the upper Web applications, which are ontology oriented.

Key words: semantic web; web ontology; relational database; SQL; OWL; SPARQL

1 引言

语义网(Semantic Web)是对现有互联网的扩展, 其中数据被赋予了定义良好的含义, 更加便于计算机和人的协同^[1]. 本体(Ontology)技术是语义网的核心技术之一, 是对互联网数据的概念化、形式化和明确化的规范说明^[2]. 随着语义网中本体化数据的不断积聚, 本体的规模越来越大, 结构也越来越复杂. 鉴于关系数据库(RDB)在海量数据存储、数据备份、事务处理、安全管理等方面的优势, 如何利用 RDB 合理、高效地存储大规模的网络本体, 并无缝

支持上层的面向本体互联网应用, 成为当前语义网研究所面临的一项重要任务. 一些学者参考 OWL 词汇设计了新的基于 RDB 的本体存储模式^[3-5], 与 OWL 语言的组织结构基本一致. 但是, 这些 RDB 模式存在不区分对象属性和数值属性、对类公理支持不足、命名空间存储冗余等问题, 并且与 OWL 的三个语言版本不完全对应, 还缺少配套的本体查询支持.

本文针对应用最为广泛的 OWL Lite 语言, 提出了一种易于理解且结构稳定的基于 RDB 的本体存储模式, 对互联网中的大规模本体数据进行存储.

2 基于 RDB 中网络本体存储模式

随着本体技术的不断发展,出现了 RDF(S)、OIL、DAML、OWL 等多种本体语言. 其中,网络本体语言 OWL 是为了在互联网上发布和共享本体而提出的语义标记语言,是 W3C 的语义网框架的主要组成部分,也是目前使用最广泛的本体描述语言^[3]. 针对不同互联网应用的需要,OWL 语言分为 OWL Lite、OWL DL 和 OWL Full 三种表达能力依次递增的版本. OWL Lite 提供了单分类层次和简

单属性约束功能;OWL DL 以经典描述逻辑为基础,在保持计算完备性和可判定性的前提下,提供了尽可能强大的表达能力;OWL Full 提供了最强的表达能力和完全自由的 RDF 语法,但不保证可计算性,而且不存在完整的推理算法支持. 在三种 OWL 语言版本中,OWL Lite 表达能力最弱,但数据存储和访问效率更高,应用最为广泛. 因此,选择 OWL Lite 语言作为参考,在分析其成分和组织特征的基础上,本文提出了一种基于 RDB 的网络本体存储模式,如图 1 所示.

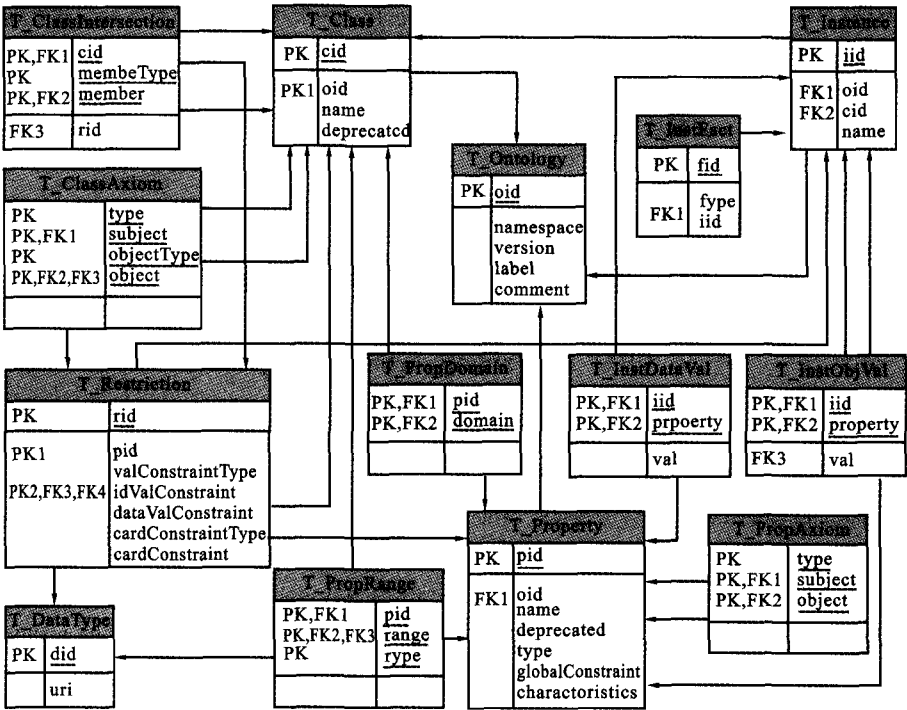


图 1 基于 RDB 的本体存储模式

- (1) T_Ontology 表存储本地本体(对应于 OWL Lite 词汇 owl: Ontology, 下同). 字段 namespace 表示本体的 URI 标识,为非空字段(图 1 中加粗字段为非空字段,否则为可空字段). 其他字段依次对应于本体的三个属性 owl: versionInfo、rdfs: label 和 rdfs: comment,均可空.
- (2) 一个本体内部可以定义多个概念(owl: Class)、属性(owl: Property)或实例,分别由表 T_Class、表 T_Porperty 和表 T_Instance 存储. 在这三个表中,使用字段 oid 关联 T_Ontology 表,使用字段 name 存储其名称(表 T_Instance 中该字段可空),而字段 deprecated 则表示该概念或属性是否不建议使用(owl: DeprecatedClass、owl: Deprecated-Property). 另外,在表 T_Property 中,字段 type 对数值属性(owl: DatatypeProperty)和对象属性(owl: ObjectProperty)进行区分,字段 globalCon-

- straint 和 characteristic 分别存储属性的全局约束(None、owl: Functional-Property、owl: Inverse-FunctionalProperty、All)和属性特性(None、owl: TransitiveProperty、owl: SymmetricProperty、All).
- (3) 表 T_DataType 记录了所有用到的数据类型,表 T_Restriction 记录了基于属性约束的匿名类(owl: Restriction, 简称为属性约束,下同). 字段 pid 指向使用的属性(owl: onProperty),字段 valConstraintType 和字段 cardConstraintType 分别记录值约束类型(owl: allValuesFrom、owl: someValues-From、owl: hasValue)和势约束类型(owl: max-Cardinality、owl: minCardinality、owl: cardin-ality),字段 idValConstraint 和 dataVal-Constraint 分别记录值约束,可以是概念、对象、数据类型或者数值,而字段 card-Constraint 记录了势约束值,因为 OWL Lite 语言的限制只能取值 0 或 1.

(4) 表 T_ClassIntersection 记录了通过概念与属性约束的交集生成新概念的关系(owl:intersectionOf). 表 T_ClassAxiom 记录了类间关系,如 owl:subClassOf、owl:equi-valentClassOf, 由字段 type 区分,而其字段 objType 则对关系目标(概念或者属性约束)进行区分.

(5) 表 T_Domain 和表 T_Range 记录属性的定义域(owl:domain)和值域(owl:range),其中值域有可能是数据类型、类或者是属性约束,由字段 type 区分. 表 T_PropAxiom 记录属性间关系,如 owl:subPropertyOf 和 owl:equivalentPropertyOf, 由字段 type 区分.

(6) 表 T_InstDataVal 和表 T_InstObjVal 记录了对象属性值三元组,其取值分别为数值或者对象. 表 T_InstFact 记录了实例间关系(owl:sameAs、owl:differentFrom 和 owl:AllDifferent), 使用字段 type 区分.

此外,图 1 中只给出了基于 RDB 的本体存储模式的核心部分,对于其他一些细节内容如 owl:Annotation、owl:imports、owl:compactableWith 等的支持由于篇幅限制予以略去. 可见,该 RDB 模式的设计与 OWL Lite 的语言成分和组织结构严密对应,易于理解,并便于本体查询的实现. 并且,与水平、分解和混合等模式不同,本体数据的变化不会引起该 RDB 模式中数据库表结构的变化,因此该模式的结构要更为稳定.

3 基于动态 RDB 视图的本体查询

标准、规范和明确化的数据表示与查询是本体的最大优势. 目前,W3C 的语义网框架使用 SPARQL 作为本体数据的标准查询语言,查询方法灵活丰富^[6]. 基于 RDB 的本体存储势必会影响到原有的本体查询,并使底层数据存储与上层面向本体的互联网应用之间数据异构. 针对这一情况,本节在图 1 所示 RDB 模式的基础上,构造出一系列自定义的 SQL 算子,利用算子动态生成必要的 RDB 视图,利用这些 RDB 视图将给定的 SPARQL 本体查询解释为面向 RDB 的 SQL 查询. 为了简便起见,本节只讨论简化的 RDB 视图生成过程,即只考虑概念间的继承关系,而不考虑概念等价关系、概念交集关系、属性关系和实例间关系.

首先,必要的 SQL 算子如表 1 所示(假设集合 C、P、D 分别表示存储的本体类、属性和数据类型的集合). 这些算子均通过一些基本的 SQL 语句实现

(需要附加一个名称匹配表实现),比如,算子对应于如下的 SQL 语句:

```
SELECT T_Class.cid
FROM T_Class AS TC, T_ClassAxiom AS TCA
WHERE TCA.type=0 AND
( TCA.subject=@clsname AND
TCA.objectType=0 AND
TCA.object=TC.cid )
```

其中,@clsname 为输入变量,TCA.type = 0 表示类关系为继承关系,而 TCA.objectType=0 表示继承目标为本体概念.

表 1 SQL 算子(部分)

算子	含义
RDBName(x), ($x \in C \cup P \cup D$)	获取本体元素对应的 RDB 元素的名称;
SupCls($c \subseteq C, (c \in C)$)	获取本体概念的直接父概念集合;
SubCls($c \subseteq C, (c \in C)$)	获取本体概念的直接子概念集合;
PropOf($c \subseteq P, (c \in C)$)	获取本体概念中直接定义的属性集合.

使用这些 SQL 算子,我们就可以通过自动化的过程构造一个本体概念对应的 RDB 视图,其过程分为如下三步.

(1) 假设给定本体概念是 $c \in C$, 获取 c 的所有上层概念(包括 c), 从而得到这些概念所共同定义的属性集合. 其算法如下:

```
//定义超类集合、属性集合、临时队列变量
var ClsSet, PropSet, TmpQueue;
//将 c 放入临时队列
TmpQueue.enqueue(c);
//临时队列非空时候循环
while( TmpQueue.notnull() ) {
    //取出一个概念分析
    var tmpCls=TmpQueue.delqueue();
    //若分析过,重新循环
    if(ClsSet.include(tmpCls))
        continue;
    //将未分析的直接父概念入队
    TmpQueue.enqueue(SupCls(tmpCls)-ClsSet);
    //该概念分析完毕,入上层概念集
    ClsSet.add(tmpCls);
    //该概念所有属性入属性集
    PropSet.union(PropOf(tmpCls));
}
```

(2) 提取概念的所有子概念(包括 c), 其算法如下:

```
var ClsSet, TmpQueue;
TmpQueue.addqueue(c);
while( TmpQueue.notnull() ) {
    var tmpCls=TmpQueue.delqueue();
    if(ClsSet.include(tmpCls))
        continue;
    TmpQueue.enqueue(SubCls(tmpCls)-ClsSet);
    ClsSet.add(tmpCls);
}
```

(3) 生成构造 RDB 视图的 SQL 语句,并直接使用表 T_Instance 的主键 iid 作为新视图的主键。

下面,通过一个 SPARQL 查询的例子,说明动态生成的 RDB 视图及与本体查询等价的 SQL 查询.假设在一个 University 本体中,概念 unv: Student 包含 unv: Bachelor、unv: Maste 和 unv: Doctor 三个子概念,包含 unv: hasSNo、unv: has-Name 和 unv: hasTutor 三个属性,这些属性详细信息如表 2 所示。

表 2 Student 类的属性详细信息

本体属性	RDB 字段名	类型	势
unv:hasSNO	SNo	数值属性	1
unv:hasName	SName	数值属性	1
unv:hasTutor	Tutor	对象属性	0.1

假设针对 unv: Student 类的一个 SPARQL 查询如下:

```
SELECT Name
WHERE {
    ? student a unv:Student
    ? student unv:hasName? Name
}
```

表示查找所有学生的姓名,包括所有本科生、硕士生和博士生.对于此查询,自动生成 unv: Sudent 概念对应 RDB 视图 V_Student 的 SQL 代码如下:

```
CREATE VIEW V_Student AS
SELECT T1.iid, T1.SNO, T1.Name, T2.Tutor
FROM
(SELECT T1.iid, TV1.hasSNO AS SNO, TV2.has-
Name AS NAME
FROM T_Instance AS T1, T_InstDataVal AS TV1, T_
InstDataVal AS TV2
WHERE (T1.cid=csid OR T1.cid= cbid OR T1.cid=
cmid OR T1.cid= cdid) AND
(TV1.iid=T1.iid) AND (TV1.pid= psid) AND
(TV2.iid=T1.iid) AND (TV2.pid= pnid) ) AS T1
```

```
LEFT JOIN
(SELECT T1.iid, TV1.hasTutor AS TUTOR
FROM T_Instance AS T1, T_InstObjVal AS TV1
WHERE (T1.cid=csid OR T1.cid= cbid OR T1.cid=
cmid OR T1.cid= cdid) AND
(TV1.iid=T1.iid) AND (TV1.pid= ptid)) AS T2
ON T1.iid=T2.iid;
```

其中,csid、cbid、cmid、cdid 及 psid、pnid、ptid 等分别表示各个概念和属性所对应的 id 值.利用视图 V_Student,最终可以构造等价的 SQL 查询如下:

```
SELECT Name
FROM V_Student
```

4 结束语

随着语义网中本体化数据的不断累积,本体的规模越来越大,结构也越来越复杂,本文以应用最为普遍的 OWL Lite 语言为对象,提出了一种易于理解且结构稳定的基于 RDB 的本体存储模式,对互联网中的大规模本体进行存储.在此基础上,利用自动生成的 RDB 视图对标准的本体查询进行解释,在保证本体存储与查询效率的同时,保持本体查询方式的灵活性和丰富性,使得底层基于 RDB 存储的本体数据与上层面向本体的互联网应用无缝集成。

参考文献:

[1] Berners Lee T, Handler J, Lassila O. The semantic web [J]. Scientific American, 2001(184):34—43.

[2] Haffner K A. Semantic web: standards, tools and ontologies [M]. New York: Nova Science Pub Inc, 2010.

[3] Shoaib M, Basharat A. ERMOS: An efficient relational mapping for ontology storage [C]// Proceedings of 2010 IEEE International Conference on Advanced Management Science. Chengdu: IEEE, 2010:399—403.

[4] 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 [J]. 华中科技大学学报:自然科学版, 2005, 33(S0): 217—220.

[5] 陈光仪. 基于关系数据库的本体存储研究 [D]. 长沙: 中南大学, 2009.

[6] E Prud'hommeaux, A Seaborne. SPARQL Query Language for RDF [EB/OL]. [2011—01—28]. W3C, http://www.w3.org/TR/rdf-sparql-query, 2008.

作者简介:

邱彦章 男,(1962—),副教授.研究方向计算机科学与技术、智能交通系统。