计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200800006

对象关系数据库到 RDF(S)的映射方法

鲁佳文　严　丽
南京航空航天大学计算机科学与技术学院　南京 211106
(lu_jiawen@163.com)

摘　要　随着智能信息技术的发展,知识图谱已被广泛应用于智能搜索等各个领域。知识图谱中的信息一般采取 RDF(S)的数据模型来表示。知识图谱的构建需要从大量的数据源抽取信息,而数据库是不可忽视的重要数据源。近几年,对象关系数据库得到了广泛的应用,且其中存储着丰富的语义信息,而基于对象关系数据库自动构建 RDF(S)的研究却较少。因此,文中给出了对象关系数据库与 RDF(S)的形式化定义,根据形式化定义将对象关系数据库中的语义信息进行抽取,提出了构建 RDF(S)数据的映射规则。该映射规则不仅考虑了数据库的面向对象的语义,还考虑了数据库的约束,可以充分抽取数据库中包含的语义信息。最后实现了一个名为 ORDB2RDF 的映射工具,验证了该映射规则的正确性与映射结果的语义完整性。

关键词:对象关系数据库;RDF(S);知识抽取;PostgreSQL
中图法分类号　TP311.131

Mapping Method from Object-relational Database to RDF(S)

LU Jia-wen and YAN Li
College of Computer Science and Technology,Nanjing University of Aeronautics and Astronautics,Nanjing 211106,China

Abstract　With the development of intelligent information technology,knowledge graph has been widely used in intelligent search and other research area. The information in the knowledge map is generally represented by the data model of RDF(S). The construction of knowledge graph needs to extract information from different data sources and database is an important data source that cannot be ignored. Nowadays,object-relational databases are widely used and contain rich semantic information,but research on constructing RDF(S) from object-relational databases is few. This paper puts forward formal definitions of object-relational databases and RDF(S) data and proposes mapping rules for constructing RDF(S) data from object-relational databases. The mapping rules not only consider the object-oriented semantics of the database,but also consider constraints,which can fully extract semantic information contained in the database. Finally,a mapping tool named ORDB2RDF is implemented to verify the correctness of the mapping rules and the semantic integrity of the mapping results.

Keywords　Object-relational database,RDF(S),Information extraction,PostgreSQL

1　引言

随着智能网络的发展,知识图谱在智能搜索、推荐系统和智能医疗等各个领域得到了广泛的应用。知识图谱在语义网的基础上得以更好地表示实体之间的关系。语义网是一种智能网络,它通过添加元数据使得机器能够理解数据的概念以及不同数据之间的逻辑关系。资源描述框架(RDF(S))包含 RDF 和 RDF Schema,是万维网联盟(W3C)提出的模型框架,用于对语义网的内容进行规范化描述[1]。RDF(S)语言的规则性和对称性使得机器能够理解数据的概念和数据间的推理关系[2]。若现有的海量数据转换为 RDF(S)数据将具有巨大的价值。现如今,包括 XML[3]、UML[4]、文本[5]、数据库等在内的多样性数据源已被用于 RDF(S)的自动或半自动构建。其中,数据库中数量巨大的高质量数据成为构建 RDF(S)不可忽视的重要信息来源。

关系数据库是当前使用最为广泛的一种数据库,它采用二维表的结构存储数据,数据表易于设计且具有规范的 SQL 标准。但关系数据库只能支持特定的数据类型,如整型、浮点数等,无法自定义数据类型,这使得关系数据库无法存储复杂的数据结构,在存储空间信息、地理信息等非结构化数据时有很大的局限性。为了解决这一问题,面向对象数据库应运而生。面向对象数据库是根据面向对象范式设计和构建的,将面向对象的功能集合到了数据库中[6]。真实世界的任何实体都可以采用对象的形式来表示。但面向对象数据库完全失去

到稿日期:2020-08-01　返修日期:2020-11-27　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:江苏省基础研究计划(BK20191274)
This work was supported by the Basic Research Program of Jiangsu Province,China(BK20191274).
通信作者:严丽(yanli@nuaa.edu.cn)

了 SQL 语言的优势，应用领域受到了一定的限制。而对象关系数据库结合了关系数据库和面向对象数据库的优势，不仅包含关系模型的所有元素，允许用户自定义数据类型，还支持继承、聚合等面向对象的特征，并且支持 SQL 语言，可以进行高效存储与检索[7]。对象关系数据库的许多思想已经并入了 SQL:1999 标准，实际上，遵循 SQL:1999 面向对象的产品都可以被认为是对象关系数据库产品，如 IBM 的 DB2、Oracle 数据库等[8]。近些年来，对象关系数据库已在软件工程[9]和多媒体[10]等很多领域得到广泛应用，其也因此成为知识抽取中不可忽视的数据源。

目前，基于关系数据库构建 RDF(S)数据的研究较为完善。2012 年，万维网联盟(W3C)提出了直接映射和 R2RML 两个标准，用于从关系数据库中抽取语义信息构建 RDF(S)。直接映射基于数据库的模式和内容直接生成 RDF 三元组[11]；而 R2RML 是一种映射语言，用户需要手动制定映射规则完成映射[12]。文献[13]给定关系数据库模式定义及其完整性约束，利用直接映射生成本体。该方法考虑了映射过程的信息保存性和查询保存性。信息保存性表示直接映射的结果具备重建原始数据库的能力，查询保存性表示对关系数据库的查询可以转换为对直接映射结果的等价查询。文献[14]通过将本体查询语言 SPARQL 直接转换为数据库查询的 SQL 语言来对关系数据库实现语义查询。一些有效的转换工具(如 D2RQ[15]，Ontop[16]等)也被广泛用于将关系数据库的数据转换为 RDF(S)数据。针对面向对象数据库，文献[17]提出了一种基于面向对象数据库构建本体的方法。据了解，目前基于对象关系数据库构建 RDF(S)数据的研究较少，文献[18]提出了一种将对象关系数据库转换为面向文档数据库的方法，数据最终以 JSON 格式进行保存，但是其未对理论方法进行实验验证。

由于对象关系数据库整合了关系数据库和面向对象数据库的特点，本文基于现有的对于关系数据库和面向对象数据库构建 RDF(S)数据的研究，分别提出了对象关系数据库和 RDF(S)的形式化定义。根据形式化定义，提出了一种基于对象关系数据库构建 RDF(S)的新方法。该方法不仅考虑了数据库的面向对象的语义，还考虑了数据库的约束，可以充分抽取数据库中包含的语义信息。基于该方法实现的 OR-DB2RDF 映射工具可以实现基于数据库的自动化映射，无需用户进行大量的交互操作。

2 对象关系数据库及 RDF(S)的形式化定义

本节根据对象关系数据库和 RDF(S)的特点，分别提出了对象关系数据库和 RDF(S)的形式化定义，有利于更好地抽取完整的语义信息，完成映射过程。

2.1 对象关系数据库的形式化定义

对象关系模型对存储的数据进行了更复杂的建模与封装，支持用户通过自定义数据类型来扩展数据模型，同时支持使用 SQL 语句对数据库进行操作[19]。基于对象关系数据库的特征，对象关系数据库的形式化定义如下。

定义 1 对象关系数据库模型可以用一个五元组 $ORDB=(B,RE,I,R,RO)$ 来表示。

B 代表对象关系数据库中基本关系的有限集，$B=T\cup C\cup D$。其中 T 代表所有数据表的有限集，$T=\{t\mid t\in T\}$。C 代表数据表中字段的有限集，$c(t)=\{c_1,c_2,c_3,\cdots\}\subseteq C$，其中 $c(t)$ 表示数据库表 t 中所有的字段。D 代表所有的基本数据类型，$d(c)\in D$ 代表字段 c 的数据类型。

RE 代表对象关系数据库中的约束关系，$RE=P\cup F$。P 表示主键约束的有限集，$PK(t,c)$ 表示数据库表 t 的主键是 c。F 表示外键约束的有限集，$FK(t_1,c_1,PK(t_2,c_2))$ 表示表 t_1 的外键是 c_1，同时该外键指向表 t_2 的主键。

I 是对象关系数据库中继承关系的有限集，$I=I_S\cup I_M$，其中 I_S 表示所有单继承关系的有限集，$I_S=I_{SD}\cup I_{SF}$。对象关系数据库中有两种方式可以表达单继承语义。第一种是用户在创建表时可以直接定义表与表之间的继承关系。$I_{SD}=\{(t_1,t_2)\mid t_1\in T,t_2\in T,t_1 \text{ is the subclass of } t_2\}$，表示表 t_1 是表 t_2 的子类。第二种是当表 t_1 的主键也是该表的外键，同时该外键指向表 t_2 的主键时，那么也可以认为表 t_1 是表 t_2 的子类。$I_{SF}=\{(t_1,t_2)\mid t_1\in T,t_2\in T,PK(t_1,c_{t1}),FK(t_1,c_{t1},PK(t_2,c_{t2}))\}$。

对象关系数据库还具备多继承属性，一个表可以从多个表继承属性。$I_M=I_{MD}\cup I_{MF}$，其中 $I_{MD}=\{(t_1,t_2,t_3,\cdots)\mid t_1\in T,t_2\in T,\cdots,t_1 \text{ is the subclass of } t_2,t_3,\cdots\}$，表示表 t_1 继承了多个表。$I_{MF}=\{(t_1,t_2,t_3)\mid t_1\in T,t_2\in T,t_3\in T,PK(t_1,c_{t11},c_{t12}),FK(t_1,c_{t11},PK(t_2,c_{t21})),FK(t_1,c_{t12},PK(t_3,c_{t31}))\}$，表示表 t_1 的主键是联合主键 $PK(t_1,c_{t11},c_{t12})$，主键 c_{t11} 是表 t_1 的外键，指向表 t_2 的主键，主键 c_{t12} 也是表 t_1 的外键，指向表 t_3 的主键，此时可以认为表 t_1 同时继承表 t_2，t_3。

R 是对象关系数据库中的复合数据类型，$R=R_S\cup R_F$。对象关系数据库允许用户自定义复合类型并像使用简单数据类型一样使用复合类型。R_S 表示所有用户自定义的复合数据类型，$R_S=\{(r,t)\mid r \text{ is defined by user},t\in T\}$。如果表 t_1 的非主键作为外键指向表 t_2 的主键，那么可以认为表 t_2 是复合数据类型。$R_F=\{(c_{t12},t_1)\mid FK(t_1,c_{t12},PK(t_2,c_{t21})),t_1\in T\}$。

RO 代表数据库中的所有实例数据。$RO(t)$ 表示表 t 的所有实例的集合。

可以用一个简单的例子来解释对象关系数据库的形式化定义。例如，对象关系数据库中存储的数据表结构如图 1 所示。图 1 给出了 7 张数据库表，PK 表示该表的主键，FK 表示该表的外键。表 graduateStudent 的 stuId 和 staffId 是该表的联合主键，分别作为外键指向表 student 的主键和表 staff 的主键，此时可以认为表 graduateStudent 同时继承表 student 和 staff。表 professor 表示由用户定义的表 staff 的子类表。表 student 的 major 作为外键指向表 department 的主键，此时可以认为 major 的属性类型是 department。表 student 的 per 字段的数据类型是由用户自定义的 Person 复合类型。Person 复合类型包含 name 和 age 两个属性。对象关系数据库允许数据表中的列定义为多维数据，表 class 中的 classes 字段的数据类型是字符数组类型。

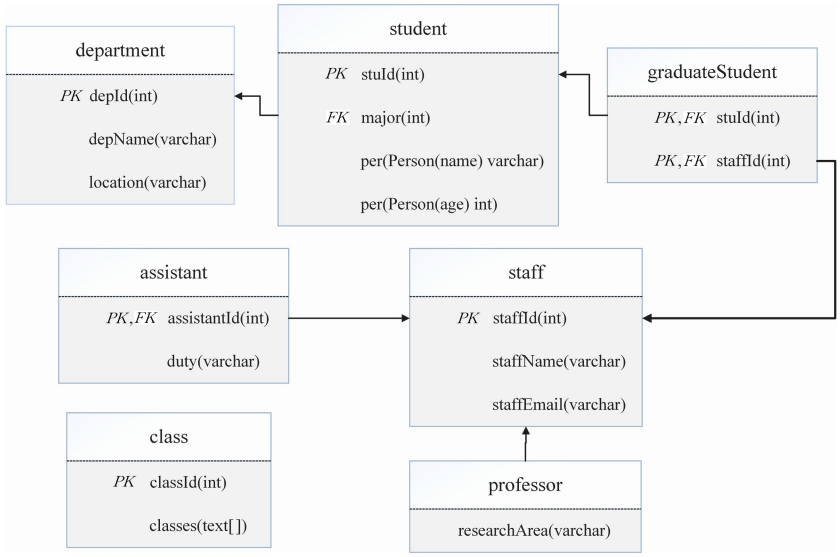

图 1 对象关系数据库的数据表结构

Fig. 1 Data structure of an object-relational database

图 2 给出了图 1 的数据表结构完整的形式化表示。

$B=T\cup C\cup D$

$T=\{staff, department, professor, assistant, student, graduateStudent, class\}$

$C=\{staffId, staffName, StaffEmail, depID, depName, location, researchArea, assistantId, duty, stuID, per, major, classId, classes\}$

$D=\{varchar, int, text[\ ]\}$

$I=I_S\cup I_M$

$I_{SD}=\{(professor, satff)\}$, $I_{SF}=\{(assistant, staff)\}$

$I_{MF}=\{(graduateStudent, staff, student)\}$

$R_S=\{(person, student)\}$

$R_F=\{(major, student)\}$

$RE=P\cup F$

$P=\{PK(department, depId), PK(student, stuId), PK(graduateStudent, stuId, staffId), PK(assistant, assistantId), PK(staff, staffId), PK(class, classId)\}$

$F=\{FK(student, major, PK(department, depId)), FK(graduate, stuId, PK(stuId, stuId)), FK(graduate, staffId, PK(staff, staffId)), FK(assistant, assistantId, PK(staff, staffId))\}$

图 2 数据表结构的形式化表示

Fig. 2 Formal representation of data table structure

2.2 RDF(S)模型的形式化定义

RDF 语句是由 RDF 陈述构成的，每个 RDF 陈述可以由主语、谓语、宾语三元组的形式来表示。网络中任何一个资源都可以当作三元组的主语和宾语。谓语用来表示主语和宾语之间的关系。宾语是谓语的属性值，可以是字面变量或者其他资源。三元组的表达方式使得 RDF 可以用来表示网络上任何被标志的信息。但是如果想要表达更复杂的语义关系，则需要使用 RDF Schema 。RDF Schema 是一种轻量级的本体语言，提供了一组具有固定含义的建模元语，包括类、子类、属性、子属性、定义域以及值域等约束。RDF(S)的形式化定义如下。

定义 2 RDF(S)模型可以由一个三元组 $RS=(RB, RA, RI)$来表示。

RB 表示所有 RDF(S)基本属性的有限集，$RB=RC\cup RD\cup RP$。其中，RC 表示 RDF(S)中所有类的有限集，RD 表示所有基本数据类型的有限集，RP 表示所有属性的有限集。

RA 表示 RDF(S)中的公理的有限集。$RA=CAxiom\cup PAxiom$，$CAxiom_{c1}=\{c\mid c$ is the super class of $c_1, c\in RC, c_1\in RC\}$，$CAxiom_{c1}$代表所有类$c_1$的父类的有限集，$CAxiom_{c1}\subseteq Caxiom$。$PAxiom$ 代表所有属性公理的有限集，$PAxiom=Dxiom\cup Rxiom$。$Dxiom_p=\{c\mid c$ is the domain of property p where $p\in RP$ and $c\in RC\}$，$Dxiom_p$表示属性 p 的定义域是类 c，$Dxiom_p\subseteq Dxiom$。$Rxiom_p$表示属性 P 的值域是属性 x，$Rxiom_p=\{x\mid x$ is range of property p where $p\in RP$ and $x\in RC\cup RD\}$，$Rxiom_p\subseteq Rxiom$。

RI 表示所有实例的有限集。

图 3 给出了一份 RDF(S)数据。在该数据中存在 staff，student，department 和 professor 类，以及 researchArea，major 等属性，其中 professor 是 staff 的子类，该数据用形式化定义表示的结果如图 4 所示。

```
〈rdfs:Class rdf:ID="staff"/〉
〈rdfs:Class rdf:ID="student"/〉
〈rdfs:Class rdf:ID="department"/〉
〈rdf:Property rdf:ID="researchArea"〉
〈rdfs:domain rdf:resource="#professor"/〉
〈rdfs:range rdf:resource="&xsd:string"/〉
〈/rdf:Property〉
〈rdf:Property rdf:ID="major"〉
〈rdfs:domain rdf:resource="#student"/〉
〈rdfs:range rdf:resource="#department"/〉
〈/rdf:Property〉
〈rdf:Property rdf:ID="depName"〉
〈rdfs:domain rdf:resource="#department"/〉
〈rdfs:range rdf:resource="#xsd:string"/〉
〈/rdf:Property〉
〈rdf:Property rdf:ID="staffId"〉
〈rdfs:domain rdf:resource="#staff"/〉
〈rdfs:range rdf:resource="&xsd:integer"/〉
〈/rdf:Property〉
〈rdfs:Class rdf:ID="professor"〉
〈rdfs:subClassOf rdf:resource="#staff"/〉
〈/rdfs:Class〉
```

图 3 RDF(S)数据

Fig. 3 RDF(S) data

$RB=RC\cup RD\cup RP$
$RC=\{\text{staff},\text{student},\text{department},\text{professor}\}$
$RD=\{\text{xsd:string},\text{xsd:integer}\}$
$RP=\{\text{researchArea},\text{staffId},\text{depName},\text{major}\}$
$RAxiom=CAxiom\cup PAxiom$
$Caxiom=\{CAxiom_{professor}=\{staff\}\}$
$PAxiom=Dxiom\cup Rxiom$
$Dxiom=\{Dxiom_{researchArea}=\{professor\},Dxiom_{staffId}=\{staff\},$
$Dxiom_{depName}=\{department\},Dxiom_{major}=\{student\}\}$
$Rxiom=\{Rxiom_{researchArea}=\{xsd:string\},$
$Rxiom_{staffId}=\{xsd:integer\},Rxiom_{depName}=\{xsd:string\},$
$Rxiom_{major}=\{department\}\}$

图 4 RDF(S)数据的形式化表示

Fig. 4 Formal representations of RDF(S) data

3 对象关系数据库到 RDF(S)的映射规则

根据对象关系数据库和 RDF(S)的形式化定义，本文的映射规则将从对象关系数据库类层次结构映射为 RDF Schema（以下简称 RDFS），以及对象关系数据库实例属性映射为 RDF 数据两个层面进行。

3.1 将对象关系数据库类结构映射为 RDF Schema 数据

使用定义 1 中的五元组 $ORDB=(B,RE,I,R,RO)$ 来表示对象关系数据库模型，使用定义 2 中的三元组 $RS=(RB,RA,RI)$ 来表示 RDF(S)模型，定义符号 φ 表示映射的过程。

规则 1 $\forall t\in T$ THEN $\varphi(t)\in RC$。

数据库中的任意一张表都可以直接映射为 RDFS 中的类。例如，数据库中存在一张名为 staff 的表，那么映射成为 RDFS 的结果为：

〈rdfs:Class rdf:ID="staff"/〉

规则 2 $\forall i=\{(t_1,t_2)\}\in I$ THEN $\varphi(t_1)\in RC$ AND $\varphi(t_2)\in RC$ AND $\varphi(i)\in CAxiom_{\varphi(t1)}=\{\varphi(t_2)\}$。

数据库中表的继承关系可以映射为 RDFS 中的继承关系。根据对象关系数据库的形式化定义，数据库存在单继承与多继承关系。

例如，数据库中存在表 professor 单继承表 staff，在映射时，先将父表 staff 按照规则 1 映射成类，再将 professor 表映射为 staff 表的子类。映射结果如下：

〈rdfs:Class rdf:ID="staff"/〉
〈rdfs:Class rdf:ID="professor"〉
〈rdfs:subClassOf rdf:resource="http://ordb#staff"/〉
〈/rdfs:Class〉

数据库中表 graduateStudent 的 stuId 与 staffId 为联合主键，其中 stuId 为外键指向表 student 的主键，同时 staffId 也作为外键指向 staff 表的主键，这时表 graduateStudent 继承了两张表。在映射时，先将表 student 和表 staff 映射成类，再将表 graduateStudent 映射为子类。映射结果如下：

〈rdfs:Class rdf:ID="staff"/〉
〈rdfs:Class rdf:ID="student"/〉
〈rdfs:Class rdf:ID="graduateStudent"〉
〈rdfs:subClassOf rdf:resource="http://ordb#staff"/〉
〈rdfs:subClassOf rdf:resource="http://ordb#student"/〉
〈/rdfs:Class〉

规则 3 $\forall c\in C$ AND $d(c)\in D$ THEN $\varphi(c)\in RP$ AND $\varphi(d)\in RD$。

如果数据库中表的字段属性是基础数据类型，那么该字段可以映射为类的属性，数据库表映射得到的类是该属性的定义域，该字段的数据类型可以映射为属性的值域。

例如，staffName 属性属于 staff 表，并且该属性的数据类型是基本数据类型，那么映射结果为：

〈rdfs:Class rdf:ID="staff"/〉
〈rdf:Property rdf:ID="staffName"〉
〈rdfs:domain rdf:resource="http://ordb#staff"/〉
〈rdfs:range rdf:resource="&xsd:string"/〉
〈/rdf:Property〉

RDF(S)没有内置的数据类型，因此选用 XML 模式数据类型(XSD)。对象关系数据库的基本数据类型和 RDF(S)的基本数据类型的映射表如表 1 所列。

表 1 对象关系数据库到 RDF(S)的基本数据类型映射

Table 1 Basic data type mapping of object-relational databases to RDF(S)

DataType	ORDB	RDF(S)
Numerical type	integer	xsd:integer
	bigint	xsd:long
	decimal/numeric	xsd:decimal
	smallint	xsd:short
	real	xsd:float
	double precison	xsd:double
Character type	text	xsd:string
	character	xsd:character
	varchar	xsd:character
Date and time type	timestamp	xsd:datetime
	date	xsd:date
	time	xsd:time
Boolean type	boolean	xsd:boolean

规则 4 $\forall c\in C$ AND $d(c)\in R$ THEN $\varphi(c)\in RC$ AND $\varphi(d)\in Dxiom_{\varphi(c)}$。

根据对象关系数据库的形式化定义 $R=R_S\cup R_F$，存在两种形式的复合属性。R_S 是用户自定义的复合类型，对象关系数据库允许用户自定义复合数据类型并像使用简单数据类型那样使用复合类型。

例如，数据库中表 student 存在一个复合数据类型 person，在映射时，先将该数据类型映射为类，再按照基础数据类型的映射规则进行映射。映射结果如下：

〈rdfs:Class rdf:ID="Person"/〉
〈rdf:Property rdf:ID="name"〉
〈rdfs:domain rdf:resource="http://ordb#person"/〉
〈rdfs:range rdf:resource="&xsd:string"/〉
〈/rdf:Property〉
〈rdf:Property rdf:ID="age"〉
〈rdfs:domain rdf:resource="http://ordb#person"/〉
〈rdfs:range rdf:resource="&xsd:integer"/〉
〈/rdf:Property〉
…
〈rdf:Property rdf:ID="per"〉
〈rdfs:domain rdf:resource="http://ordb#student"/〉

〈rdfs:range rdf:resource="http://ordb #person"/〉

〈/rdf:Property〉

R_F是外键表示的复合属性。对于$R_F=\{r|PK(t_1,c_1),FK(t_1,c_2,PK(t_2,c_3))\}$,如果表$t_1$的非主键作为该表的外键指向表$t_2$的主键,可以认为两张表之间存在聚合关系。如数据库中表 student 的 major 属性是该表的非主键,指向表 department 的主键,可以将 department 看作复合数据类型,映射结果如下:

〈rdfs:Class rdf:ID="student"/〉

〈rdfs:Class rdf:ID="department"/〉

〈rdf:Property rdf:ID="major"〉

〈rdfs:domain rdf:resource=" http://ordb #student"/〉

〈rdfs:range rdf:resource="http://ordb #department"/〉

〈/rdf:Property〉

3.2 将对象关系数据库实例数据映射为 RDF 数据

对象关系数据库实例数据的映射需要符合数据库中的类结构,映射得到的 RDF 数据也应该符合 3.1 节映射得到的 RDF Schema 结构。

规则 5 $\forall OID\in RO$ THEN $\varphi(OID)\in RI$。

在对象关系数据库中可以利用主键来区分不同的数据记录,如果没有设置主键,系统会默认生成一个隐藏的唯一标识。RDF 中,可以利用命名空间作为 RDF 数据的唯一标识。将数据库中的数据映射为 RDF 数据时,可以随机选取一个唯一值映射为 RDF 的命名空间。

例如,数据库中表 staff 记录的一条数据如下[staffId:789365,staffName:Mike,staffEmail:mike@gmail.com],映射时,选取随机数 78932 作为 RDF 的命名空间,其映射结果如下:

〈rdf:Description rdf:about="http://www.oodb.org/uni-ns/78932"〉

……

〈/rdf:Descrition〉

规则 6 $\forall ro\in RO$ THEN $\varphi(ro)\in RI$。

数据库中表的数据可以根据生成的 RDF Schema 的结构来进行映射。例如,数据库中存在表 staff 的数据[staffId:789365,staffName:Mike,staffEmail:mike@gmail.com]。该数据会根据生成的 RDFS 结构进行映射,映射结果如下:

〈rdf:Description rdf:ID="78932"〉

〈rdf:type rdf:resource="#staff"/〉

〈staffId rdf:datatype="&xsd:integer"〉789 365〈/staffId〉

〈staffName rdf:datatype="&xsd:string"〉Mike〈/staffName〉

〈staffEmail rdf:datatype="&xsd:string"〉mike@gmail.com〈/staffEmail〉

〈/rdf:Description〉

规则 7 $\forall ro(t_1)\in RO$ AND $i=\{(t_1,t_2)\}\in I$ THEN $\varphi(ro(t_1))\in RI$ AND $\varphi(ro(t_2))\in RI$。

如果实例数据所在的表存在继承关系,那么映射时会将父类表中的数据一起映射。

例如,表 assistant 的主键 assistantId 同时也是该表的外键,指向 staff 表的主键,此时 assistant 表和 staff 表之间存在继承关系。当表 assistant 中存在一条数据[assistantId:89654,duty:evaluation],表 staff 中相对应存在数据[staffId:89654,staffName:Joe,staffEmail:joe@gmail.com],则映射结果为:

〈rdf:Description rdf:ID="78933"〉

〈rdf:type rdf:resource="#assistant"/〉

〈assistantId rdf:datatype="&xsd:integer"〉89 654〈/assistantId〉

〈duty:rdf:datatype="&xsd:string"〉evaluation〈/duty〉

〈staffName rdf:datatype="&xsd:string"〉Joe〈/staffName〉

〈staffEmail rdf:datatype="&xsd:string"〉joe@gmail.com〈/staffEmail〉

〈/rdf:Description〉

规则 8 $\forall ro\in RO$ AND $r\in R$ THEN $\varphi(ro)\in RI$ AND $\varphi(r)\in RI$。

如果数据表的记录中包含复合类型的数据,即$r\in R_S$,映射时会将该复合类型中的所有数据一起进行映射。如果是用户自定义的复合类型,则直接按照数据类型进行映射。如果表中的非主键作为外键,指向了另一张表的主键,即$r\in R_F$,则需要将另一张表的实例一起进行映射。

例如,表 student 中的数据为[stuId:78953,major:89653,per(name:monica,age:24)],其中 per 属性是用户自定义的 person 数据类型;而 major 代表 department 表中的数据,即数据为[depId:89653,depName:computer science,location:building A]。映射时直接将 major 指向表 department 的实例,映射结果如下:

〈rdf:Description rdf:ID="78934"〉

〈rdf:type rdf:resource="#student"/〉

〈stuId:rdf:resource="&xsd:integer"〉78953〈/stuId〉

〈major:rdf:resource="#department"〉#78935〈/major〉

〈per rdf:datatype="#person"〉(monica,24)〈/per〉

〈/rdf:Description〉

〈rdf:Description rdf:ID="78935"〉

〈rdf:type rdf:resource="#department"/〉

〈depId:rdf:datatype="&xsd:integer"〉89653〈/depId〉

〈depName:rdf:datatype="&xsd:string"〉computer science〈/depName〉

〈location rdf:datatype="&xsd:string"〉building A〈/location〉

〈/rdf:Description〉

规则 9 $\forall ro\in RO$ AND $c\in C$ AND $d(c)\in D$ THEN $\varphi(ro)\in RI$ AND $\varphi(d)\in RD$。

对象关系数据库支持定义数组数据类型。对于数据库中的数组数据可以用 RDF 中的容器元素来表示。

例如,数据库中存在 class 表中的数据[classname:firstclass,classes:("A01","A02")],则映射结果为:

〈rdf:Description rdf:about="http://ordb#78936"〉

〈rdf:type rdf:resource=#class/〉

〈classname:rdf:datatype="&xsd:string"〉first-class〈/classname〉
〈j.0:classes〉
〈rdf:Bag〉
〈rdf:li〉"A01"〈/rdf:li〉
〈rdf:li〉"A02"〈/rdf:li〉
〈/rdf:Bag〉
〈/j.0:classes〉
〈/rdf:Description〉
〈/rdf:RDF〉

4 映射工具的实现与实验分析

为了检验映射规则的正确性以及可实现性,本文开发了一个名为 ORDB2RDF 的映射工具,可以实现将对象关系数据库中的数据自动构建为 RDF(S) 数据的功能。本文选取 PostgreSQL 数据库作为实验数据库。PostgreSQL 数据库是由加州大学开发的一款特性齐全的对象关系型数据库管理系统,在商业上逐渐得到了广泛的应用。ORDB2RDF 在一台 Intel Core i5 2.50GHz CPU 和 8GB RAM 的 PC 上运行。使用的编程语言是 Java,运行环境是 JDK1.8.0。

4.1 系统设计与实现

该工具主要分为 4 个模块,系统设计如图 5 所示。系统输入为对象关系数据库,ORDB2RDF 可以通过连接数据库获取数据,系统输出为 RDF(S)文档。数据库解析模块可以根据对象关系数据库的形式化定义分析数据存在的语义关系和数据信息。RDF Schema 生成模块可以根据得到的数据库类层次关系生成 RDF Schema 文档。RDF 生成模块根据已经生成的 RDF Schema 结构和数据库中的数据信息生成 RDF 文档。经数据库解析,RDF 生成模块得到的数据可以传输给显示模块进行显示。

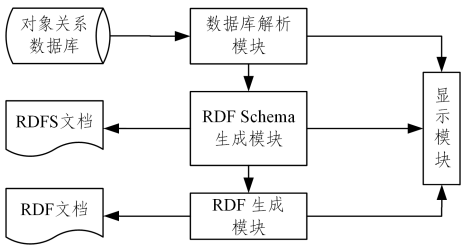

图 5 ORDB2RDF 的系统设计结构

Fig. 5 ORDB2RDF's system design structure

该工具主要分为登录界面和映射界面。登录界面需要用户手动输入需要转换的数据库以及 PostgreSQL 数据库的连接用户名和密码。对象关系数据库到 RDF(S)的映射界面如图 6 所示。用户可以在映射界面选择生成 RDF(S)文档的存储位置。该界面左侧对数据库解析模块得到的数据库类和属性进行显示;该界面右侧对系统生成的 RDF(S)文档进行显示。

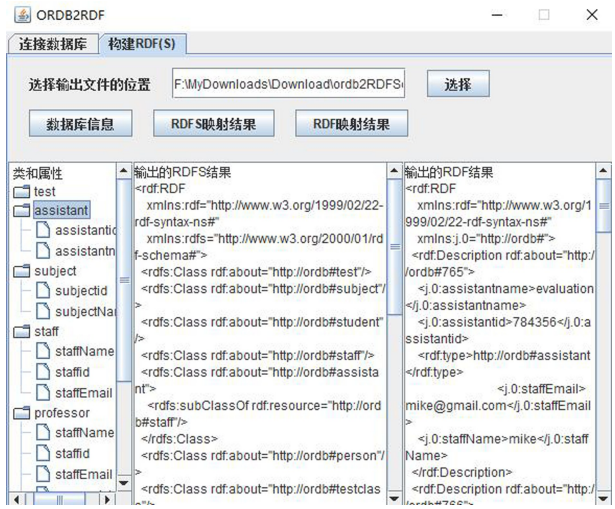

图 6 对象关系数据库到 RDF(S)的映射界面

Fig. 6 Object-relational database to RDF(S) mapping interface

4.2 实验结果分析

本文选取图 1 中的数据库结构,利用 SPARQL 查询语言,检验在映射过程中是否存在语义丢失的情况。2013 年 W3C 官方发布了 SPARQL1.1 标准对 SPARQL 语言进行了标准化,可以用于查询和操作 RDF 数据[20]。

首先对对象关系数据库中的继承语义进行检验。在图 1 的数据库结构中,存在表 assistant 和表 professor,这两个表都可被认为是表 staff 的子类表。对于子类表中的所有数据,映射时会将从父类表中继承的属性与属性值一同映射。本文利用 SPARQL 语言对生成的 RDF 文档中 assistant 类和 professor 类的实例数据进行查询,查询结果如图 7 所示。查询结果显示,表 assistant 和表 professor 中的数据在映射过程中将从 staff 表中继承得到的属性全部映射成功。这证明了该工具可以不失语义地将对象关系数据库中的继承关系映射到 RDF 数据中。

URI	type	staffEmail	staffName	duty
<http://ordb#765>	"http://ordb#assistant"	"mike@gmail.com"	"mike"	"evaluation"

URI	type	staffEmail	staffName	researchArea
<http://ordb#772>	"http://ordb#professor"	"harry@gmail.com"	"harry"	"knowledge graph"

图 7 RDF 文档中 staff 子类的实例数据

Fig. 7 Instance data for the staff subclass in the RDF document

接下来对对象关系数据库中存在的复合数据类型进行检验。在图 1 的数据表结构中 student 表存在两种复合数据类型。一种是 student 表中的 per 属性,该属性是由用户在创建数据表时自定义的 person 数据类型;另一种是 student 表中的 major 属性,major 作为非主键由外键约束指向了表 department,可以认为该 major 的数据类型是 department 类型。利用 SPARQL 语句对生成的 RDF 文档中的 student 类的实例数据进行查询,得到的结果如图 8 所示。查询结果显示,在生成的 RDF 文档中,student 类的 per 属性数据是完全按照用户自定义的 person 数据结构得到的。而 major 属性则根据映射规则指向 RDF 文

档中相对应的 department 实例。该结果表明,ORDB2RDF 工具可以不失语义地映射对象关系数据库中的复合类型数据。

URI	type	stuid	major	per
<http://ordb#775>	"http://ordb#student"	"1816002"	"http://ordb#778"	"(monica,24)"

URI	type	depid	depname	location
<http://ordb#778>	"http://ordb#department"	"89653"	"computer science"	"building A"

图 8 RDF 文档中 student 类的实例数据

Fig. 8 Instance data for the student class in the RDF document

结束语 本文根据对象关系数据库和 RDF(S)数据的特点分别给出了相应的形式化定义,可以更好地表达数据库和 RDF(S)文档中存在的语义信息。基于形式化定义,本文在对象关系数据库的类层次结构和实例结构上给出了将对象关系数据库中的数据映射成为 RDF(S)数据的新方法。该映射方法不仅保留了数据库约束表达的语义信息,还成功映射了对象关系数据库特有的数据结构。本文还利用 PostgreSQL 数据库实现了一个自动化映射工具,并对方法的可行性以及映射结果的正确性进行检验。实验结果表明,经由提出的规则转换得到的 RDF(S)文档具备语法正确性以及语义完整性。

本文的未来工作有以下方面:1)利用 ORDB2RDF 工具对对象关系数据库中存储的包含复杂语义信息的大规模数据进行映射;2)尝试优化映射工具的算法结构,进一步提升映射的效率。

参 考 文 献

[1] Resource Description Framework (RDF):Concepts and Abstract Syntax [OL]. [2020-06-24]. https://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.

[2] TOMASZUK D,HYLAND-WOOD D. RDF 1.1:Knowledge Representation and Data Integration Language for the Web [J]. Symmetry,2020,12(1):1-33.

[3] JUNIOR A C,ORLANDI F,O'SULLIVAN D,et al. Using Mapping Languages for Building Legal Knowledge Graphs from XML Files[C]// Proceedings of the Contextualized Knowledge Graphs (CKG) Workshop co-located with the 18th International Semantic Web Conference. 2019:1-12.

[4] TONG Q,ZHANG F,CHENG J. Construction of RDF(S) from UML Class Diagrams [J]. Journal of Computing and Information Technology,2015,22(4):237-250.

[5] MARTINEZ-RODRIGUEZ J L,LOPEZ-AREVALO I,RIOS-ALVARADO A B,et al. Extraction of RDF Statements from Text [M]// Knowledge Graphs and Semantic Web. Cham:Springer,2019:87-101.

[6] ALZAHRANI H. Evolution of Object-Oriented Database Systems [J]. Global Journal of Computer Science and Technology,2016,16(3):33-36.

[7] BARONI A L,CALERO C,ABREU F B E,et al. Object-Relational Database Metrics Formalization[C]// International Conference on Quality Software. IEEE,2006:30-37.

[8] Object-relational database[OL]. [2020-06-25]. http://tagged-wiki.zubiaga.org/new_content/135b89a028bb29a3c3654fbe3a821bcc.

[9] GREGORY V. Lessons in Persisting Object Data Using Object-Relational Mapping [J]. IEEE Software,2019,36(6):43-52.

[10] KHANDUJA V,CHAKRAVERTY S. A generic watermarking model for object relational databases [J]. Multimedia. Tools and Applications,2019,78(19):28111-28135.

[11] A Direct Mapping of Relational Data to RDF[OL]. [2020-06-25]. https://www.w3.org/TR/2012/REC-rdb-direct-mapping-20120927/.

[12] R2RML:RDB to RDF Mapping Language [OL]. [2020-06-25]. https://www.w3.org/2001/sw/rdb2rdf/r2rml/Overview.html.

[13] SEQUEDA J F,ARENAS M,MIRANKER D P. On directly mapping relational databases to RDF and OWL[C]// Proceedings of the 21st International Conference on World Wide Web. Association for Computing Machinery,2012:649-658.

[14] UNBEHAUEN J,STADLER C,AUER S. Optimizing SPARQL-to-SQL rewriting[C]// IEEE International Conference on Semantic Computing. 2013:324-330.

[15] EISENBERG V,KANZA Y. D2RQ/Update:Updating Relational Data via Virtual RDF [C]// Proceedings of the 21st International Conference on World Wide Web. 2012:497-498.

[16] CALVANESE D,COGREL B,KOMLAEBRI S,et al. Ontop:answering sparql queries over relational databases [J]. Semantic Web,2017,8(3):471-487.

[17] ZHANG F,MA Z M,YAN L. Construction of Ontologies from Object-Oriented Database Models [J]. Integrated Computer-Aided Engineering,2011,18(4):327-347.

[18] AGGOUNE A,NAMOUNE M S. A Method for Transforming Object-relational to Document-oriented Databases[C]// 2020 2nd International Conference on Mathematics and Information Technology (ICMIT). IEEE,2020:154-158.

[19] AUZINS A,EIDUKS J,VASILEVSKA A,et al. Object-Relational Database Structure Model and Structure Optimisation [J]. Applied Computer Systems,2018,23(1):28-36.

[20] PEREZ J,ARENAS M,GUTIERREZ C. Semantics and complexity of SPARQL [J]. ACM Transactions on Database Systems,2009,34(3):1-45.

LU Jia-wen, born in 1996, postgraduate. Her main research interests include data and knowledge engineering.

YAN Li, born in 1964, Ph.D, professor, is a member of China Computer Federation. Her main research interests include data and knowledge engineering.