情报学报　2019 年 3 月　第 38 卷　第 3 期
Journal of the China Society for Scientific and Technical Information, Mar. 2019, 38(3): 310-321

DOI: 10.3772/j.issn.1000-0135.2019.03.009

基于图形数据库的OWL本体存储模型研究

黄　奇[1]，钱韵洁[2]，袁勤俭[3]，陆佳莹[4]

（1. 南京大学国家信息资源管理南京研究基地，南京　210093；2. 公安部第三研究所，上海　201204；
3. 南京大学信息管理学院，南京　210023；4. 中国建设银行股份有限公司上海大数据智慧中心，上海　200120）

摘　要　本研究从OWL本体的理论基础描述逻辑理论出发，分析本体的基本构成及语义关系，提出了基于图形数据库的OWL本体存储模型，设计了该存储模型的拓扑结构，规定了OWL本体至图形数据库存储模型的存储映射规则，从理论层面证明了该存储模型的有效性和科学性，并得出了相应的启示。

关键词　OWL本体；本体存储；图形数据库；描述逻辑

Research on the Storage Model of OWL Ontologies Based on Graph Databases

Huang Qi[1], Qian Yunjie[2], Yuan Qinjian[3] and Lu Jiaying[4]

(1. *National Information Resource Management Nanjing Research Base of Nanjing University, Nanjing*　210093;
2. *The Third Research Institute of Ministry of Public Security, Shanghai*　201204;
3. *School of Information Management, Nanjing University, Nanjing*　210023;
4. *Data Centre of China Construction Bank, Shanghai*　200120)

Abstract: Taking advantage of Description Logic, which is the theoretical basis of OWL ontologies, a storage model of OWL ontologies based on graph databases is proposed, and the topological structure of the model is designed. The mapping rules from OWL ontology to graph databases are then designed, based on the storage model. Finally, the validity and effectiveness of this storage model is proved theoretically, and a series of valuable conclusions are drawn.
Key words: OWL ontologies; ontology storage; graph databases; description logic

1　引　言

本体存储是本体理论与技术研究的重要组成部分，每当一个本体被构建成功，讨论用何种方式对其进行存储是无法回避的问题。本体存储方式的科学性和有效性直接影响了本体的后续管理与利用，因而，对本体存储技术的研究可谓是本体理论研究与应用研究衔接的桥梁。

随着本体应用领域的不断延伸，应用方向的不断拓展，本体的规模与复杂度进一步增大，用户的本体查询需求也越来越复杂，传统的采用纯文本形式存储的本体面临着推理机负荷过大而查询效率低下的问题[1]，进而形成了基于本体理论研究与实践应用的鸿沟。但与此同时，大数据时代带来的存储技术进步，又为本体的存储研究带来了新的契机，

收稿日期：2018-08-13；**修回日期**：2018-11-21
基金项目：国家社会科学基金重点项目“基于知识组织的产品分类本体研究”（13ATQ005）；南京大学双一流建设“百层次”科研项目“本体特征分析及其应用研究”。
作者简介：黄奇，男，1961年生，博士，教授，主要研究方向为网络信息资源的语义化、电子商务；钱韵洁，女，1994年生，研究实习员，主要研究方向为网络信息资源的语义化、电子商务，E-mail：649291745@qq.com；袁勤俭，男，1969年生，博士，教授，主要研究方向为情报学理论与方法、电子商务等；陆佳莹，女，1992年生，主要研究方向为网络信息资源的语义化、电子商务。

越来越多的研究者开始对本体的存储模式展开了新的探索。

2　相关研究

自本体理论与技术诞生以来，曾相继出现过基于内存、基于文本和基于数据库的多种本体存储方式。但随着主流本体构建工具，如Protégé[2]开始提供自动生成OWL文档的功能，Jena[3]、Racer[4]等主流推理机开始提供RDF、OWL等文本文档的解析与推理功能，以纯文本的形式将本体直接存储于本地的基于文本的存储方式逐渐成为主流。该方式轻便灵活，但对推理机性能依赖性高[5]，易造成本体查询复杂且更新效率不高，因此，更便于查询与应用的基于数据库的本体存储方式成为研究人员关注的重点。目前，有关本体的数据库存储的研究主要集中在基于关系型数据库的本体存储和基于非关系型数据库的本体存储两方面。

1）基于关系型数据库的本体存储研究

基于关系型数据库的本体存储利用已经非常成熟的关系型数据库技术来存储本体信息，并执行相关的事务处理[6]，其核心在于如何将N维的本体转化为二维的数据模型[7]，并通过表间的关联来尽可能地还原本体中包含的语义关系信息，因而本体向数据库转换的映射规则成为了影响该存储方式优劣的决定性因素。一些学者尝试通过RDF三元组为中介将整个本体映射为一张二维表进行存储[8]，便于理解却存在单表规模过大的缺憾；另一部分学者则提出了将本体基于类[9]或属性[10]分解为若干张二维表进行存储，提高了检索效率但涉及了更多表操作，增加了维护难度；故此后的研究者多有采长补短之意，对类、属性、实例等进行更为灵活的分解存储，提出了混合存储模型[11-13]使得本体更新更为方便。得益于关系数据库存储技术的稳定性与成熟性，基于关系型数据库的本体存储方式具有较强的实际应用性，但二维表固有的平关系模型制约了对本体中复杂语义信息的表示[14]，易造成转换过程中的语义丢失。虽则已有学者尝试通过设定特殊的规则来专门表示本体中较为复杂的属性关系与属性约束[15-16]以弥补此不足，却并未形成通用的、规范的存储模型和映射准则来适应大规模本体的存储查询。

2）基于非关系型数据库的本体存储研究

非关系型数据库（NoSQL）是一种非关系型的、分布式的并且通常不遵循传统数据库ACID原则的数据库管理系统[17]，是一种数据模型灵活、重视数据网状关系的数据库。部分学者针对以列数据为存储单位的列数据库展开了研究。Kim等[18]采用了六张表来分解存储RDF三元组的Hbase存储方案，并利用Mapreduce对其进行数据处理。Papailiou等[19]提出了通过Hbase多重索引实现分布式本体数据的合并与连接，进一步提升了Hbase存储查询效率。孙康[20]则提出了一种本体元数据与RDF实际数据分开存储的Hbase领域本体模型并将其应用于民航突发事件应急管理的本体构建，证明了分布式列式数据库存储本体信息的实践可行性。另一部分学者则将目光投向了通过网状的图结构来实现数据存储的图形数据库[21]，探讨了利用图形数据库存储本体的合理性[22-23]。康杰华等[24]证明了RDF三元组的图形数据库的存储模式的可行性；王红等[25]则详细提出了RDF图与属性图的映射规则，并构建了民航领域本体。

通过上述文献调研可以发现，基于非关系型数据库的本体存储模式相较于基于关系型数据库的存储模式显得更为灵活，也更适合于本体中语义信息的存储，尤其是与本体结构相似的图形数据库，更是显现出极高的研究价值。但相关研究目前仍处于发展阶段，存储模式的提出仍缺乏理论基础的指引，映射规则的设计多针对RDF三元组，故受到广泛认可的存储模型仍有待于进一步探索。因此，本研究即从OWL本体的理论基础描述逻辑出发，探寻OWL本体存储与图形数据库存储的映射规则，以建立具有科学性与实用性的图形数据库本体存储模型。

3　基于图形数据库的本体存储模型拓扑结构设计

图形数据库通过节点和边的形式将数据存储为网状图，这种直观而灵活的图形结构模式相较于其他数据库更契合本体的结构。但图形数据库与本体模型间终归存在一定的差异，分析存储模型的拓扑结构，是设计合理的存储模型、实现高效的本体语义信息的图形数据库存储的重要环节。

本体的表达依赖于特定的本体描述语言，目前，使用最为广泛的本体描述语言为W3C推荐的OWL语言，其凭借较强的表达能力和推理能力而备受研究者青睐。OWL语言是以描述逻辑为基础实现本体的形式化表达与推理的。当本体作为基于描述逻辑的知识表示系统时，其由Tbox和Abox两

部分构成。其中，Tbox 存储了本体中的术语断言，而 Abox 存储了本体中的实例断言。换言之，本体中的概念与角色构成的断言共同存储于 Tbox 中，是对现实世界中的对象及对象关系的抽象的说明和存储，描绘了本体中的概念结构；而实例及实例关系构成的断言则存储于 Abox 中，是现实世界中的对象的具体的描绘和存储。由此可见，概念结构的存储更加注重对概念的抽象描述以及概念间的普适关系的刻画，更为抽象精练；而实例的存储则侧重于实例特征的具体描述，更为细节具体，也涉及了更多的冗余信息的存储。因此，从本体的语义结构角度考虑，本研究在存储模型拓扑结构设计时，将本体的概念结构与实例的存储模式进行了区分，如图 1 所示。

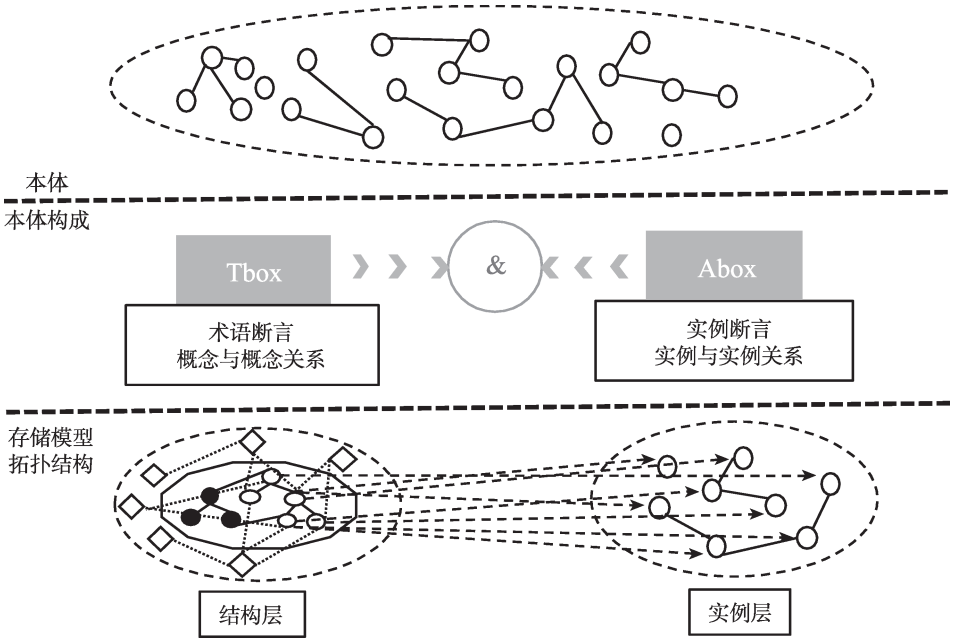

图 1 本体存储模型拓扑结构图

图 1 所展示的本体存储模型拓扑结构展现了本体中的元素的逻辑位置关系，其核心思想主要体现在两个层面：①本体存储模型分为结构层和实例层两部分。与描述逻辑中的 TBox 和 ABox 相对应，结构层存储了本体中的概念结构，其由概念集 $C=\{c_1,c_2\cdots c_n\}$ 以及概念间的关系集合 $R_c=\{r_{c_1},r_{c_2}\cdots r_{c_n}\}$ 共同构成，描述概念及概念间的关系。实例层由实例集 $I=\{i_1,i_2\cdots i_n\}$ 及实例间的关系 $R_1=\{r_{11},r_{12}\cdots r_{1n}\}$ 构成，描述实例及实例间的关联。概念层与实例层之间通过类目与实例的绑定进行关联。②结构层中概念与角色均由节点表示，概念存储核心位置，角色处于外围。概念包括类与属性，当其独立存在时，具有明确的语义，相对稳定，如概念“计算机”。角色描述了概念间的关联，当角色更新时将联动概念更新。因此，将二者都采用节点的形式表示，逻辑上分开存储，确保了其相互间的独立性，使得更新角色节点数据时不会影响到概念节点的数据，而更新概念节点数据时对角色节点的影响也降到最低。这种拓扑结构设计对本体中的对象及语义关系进行了梳理与划分，对本体的更新与衍化支持度更高，为具体的本体图形数据库存储模型的存储规则设计奠定了基础。

4 基于图形数据库的本体存储模型映射规则

基于图形数据库的本体存储模型的存储规则可以理解为本体的 OWL 存储模型到图形数据库存储模型间的映射规则，是两种存储模型间的语义转换。

定义 1 O 和 GraphO 为本体的两种不相同的存储模型。那么：

- 存储模型 O 和 GraphO 间的映射可表示为三元组 $M=(\mathrm{O},\mathrm{GraphO},\Sigma)$ 的形式，Σ 是 $<\mathrm{O},\mathrm{GraphO}>$ 上的一组逻辑公式，其中，O 称为源模型，GraphO 成为目标模型。

定义 2 令 $M=(\mathrm{O},\mathrm{GraphO},\Sigma)$ 为一个模型映射。那么：

- M中的对象为<O,GraphO>上满足Σ的实例<I,J>。
- Inst(M)表示M中所有对象的集合。
- 令I为O中的对象，J为GraphO中的对象，称满足<I,J>∈Inst(M)的对象J为I在M下的解决方案。记SOL(M,I)为I在M上的所有解决方案集合。

根据以上定义，可见映射$M=(O, GraphO, \Sigma)$本身并非标准意义上的数学映射，其描述了存储模型O和GraphO通过Σ实现关联与映射。但是，对每一个源模型中的对象I，M都会形成一个数学映射，返回所有的映射集合SOL(M,I)，其表示I在M下的解决方案。因此，对O中的实例I而言，其可能不具有解决方案或具有多个不同的解决方案。

定义3　对于一个给定的模型映射M。

- 模型映射存在性问题可转化为：对给定的源模型中的对象I，SOL(M, I)是否为Ø。
- 模型映射求解问题可转化为：对于给定的源模型中的对象I，寻找I在映射M下的解决方案J。

由此可见，对基于图形数据库本体存储模型与OWL本体间的映射模型的M的求解可转化为：为OWL本体中的每个元素I寻求基于图形数据库存储的本体中对应的元素J。

OWL本体的理论基础为描述逻辑SHOIN(D)，是在最基础的描述逻辑ALC基础上的拓展。描述逻辑ALC主要由基本构件<概念（concept），角色（role，也称关系），实例（individual，也称个体）>和基础构造算子交（∩）、并（∪）、补（¬）、存在量词（∃）和全称量词（∀）构成，而在SHOIN(D)又在此基础上新增了部分约束。由定义3可知，OWL本体的图形数据库存储映射规则的设计即为其中每个元素I计算其在图形数据库存储模型中的对应元素J。故而本文接下来将逐次探讨SHOIN(D)中的基本构件、基本构造算子和新增约束的存储模式，设计映射规则。

4.1　基本构件存储映射规则

描述逻辑的三个基本构件中，概念描述了个体集合的共同特征，表示论域中的对象；关系描述了对象之间的二元关系；实例则为概念的具体实例。基本构件并非独立存在的，它们之间的共同交互才构成了本体中的复杂语义关系。因而，在设计图形数据库中基本构件存储的映射规则时，不得不考虑它们之间的交互关系的存储模式。三个构件间的交互共有$C_3^1+C_3^2+C_3^3=7$种情况，如表1所示。

表1　基本构件交互表

自交互	Concept×Concept	结构层
	Role×Role	结构层
	Instance×Instance	实例层
两两交互	Concept×Role	结构层
	Concept×Instance	结构层×实例层
	Role×Instance	实例层
共同交互	Concept×Role×Instance	结构层×实例层

4.1.1　基本构件自交互关系的存储映射规则

自交互为在没有其他构件参与的情况下，构件内部间的交互关系。在本体中，构件间最常见的交互关系主要为等价关系（≡/≠）与包含关系（⊐/⊏），其二者为本体中最为通用且重要的二元关系。除此二者之外，不同的构件间也会有部分特殊的二元关系，如类目属性关系，这些关系在本体中常见且通用，其本身无需其他角色参与交互，也无复杂的属性约束，因而在图形数据库的存储中可映射为边的形式。

1）概念自交互（Concept×Concept）的映射规则

本体中的概念（Concept）可以分为类（Class）和属性（Property）两种。其中，类表示领域知识元，它包括了一般意义上的事物，通常具有一定的分类层级关系；属性则描述了类的性质，是一个类区别于其他类的特征。故可将概念集C看作类目集G与属性集M的并集。

$$C=\{c_1,c_2\cdots c_n\}$$
$$G=\{g_1,g_2\cdots g_n\}$$
$$M=\{m_1,m_2\cdots m_n\}$$
$$C=G\cup M=\{g_1,g_2\cdots g_n\}\cup\{m_1,m_2\cdots m_n\}$$

在利用图形数据库存储本体时，本体中的概念对应于图形数据库中的节点。提取本体构建时所得到的概念集C=G∪M，此时集合中的所有元素是独立存在的，需对他们之间的重要关联进行定义。结合OWL的语法，概念间的直接关联除了上述的等价关系与层级关系，还有一种特殊的关系，即类目概念与属性概念间的类目属性关系。因此，概念以及概念间的直接关系共同构成有向图$Graph_1=\{X_1, E_1\}$，其中，X_1为该有向图的点集，E_1为图的边集，X_1的值域为概念集C，E_1的值域为{SubClassOf, SubPropertyOf, PropertyOf, EquivalentClasses, EquivalentProperties, DisjointClasses}，如图2所示。

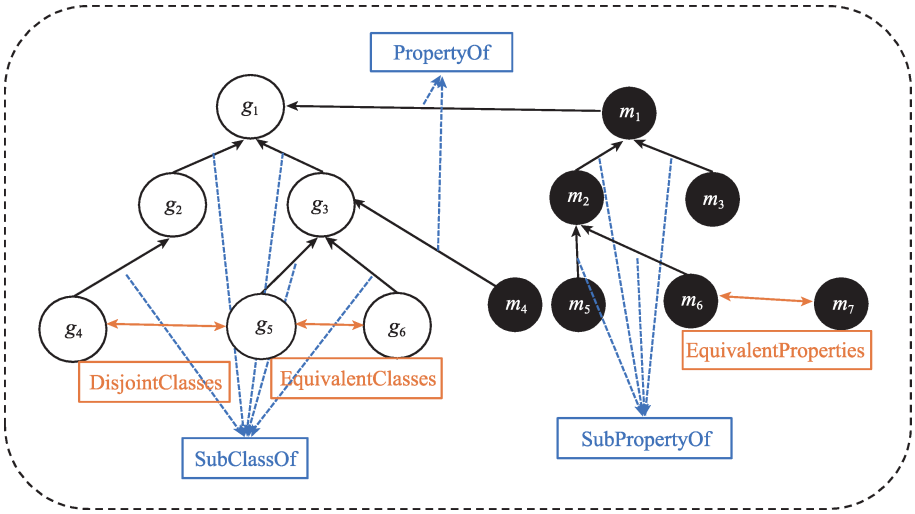

图2 概念存储示例图

在实际本体构建过程中，优质的本体在保证本体语义的正确性和完备性的同时也应当尽可能地减少冗余[26]，以降低存储成本。因而除非必要，等价的类目或属性往往不会重复定义，尤其是在一些形式化的方法，如语义分析、形式概念分析理论的指导下构建的本体，等价的类目或属性将由于具有同样的内涵而在定义时即被合并，精练了概念结构语义，降低了存储空间。同时，对于类目层级关系要求严格的本体，如产品分类本体，往往需要直观地展现类目间的层级语义关系，因而在层级关系定义时就应当保证兄弟类间的互斥关系。因此，在实际构建本体时，如产品分类本体构建时，EquivalentClasses 和 DisjointClasses 的使用频率较少。

2）角色自交互（Role×Role）的映射规则

角色（Role）也称关系，描述了本体中对象间的二元关系，类似于谓词逻辑中的二元谓词，在 OWL 中被定义为对象属性，并通过设置定义域和值域的方式与概念集 C 产生联系。角色具有关联概念的特殊性，同时角色内部也存在重要的语义关系，如层级关系，等价关系和逆关系。以角色“HasPart”为例，可定义其子角色“HasNecPart”和“HasUnnecPart”来表示必须具备的部件和非必须具备的部件。结合前文提出的本体拓扑结构模型，将角色定义为结构层的边缘位置节点，关联核心的概念节点。因此，本研究将角色集 $R_c=\{r_{c_1},r_{c_2}\cdots r_{c_n}\}$ 以点集的形式映射至图形数据库，角色节点与角色间的关系共同构成有向图 $\text{Graph}_2=\{X_2,E_2\}$。其中，$X_2$ 的值域为角色集 R。E_2 的值域则为角色间的通用二元关系，值域为 {SubRoleOf, EquivalentRoles, InverseRoles}，如图 3 所示。

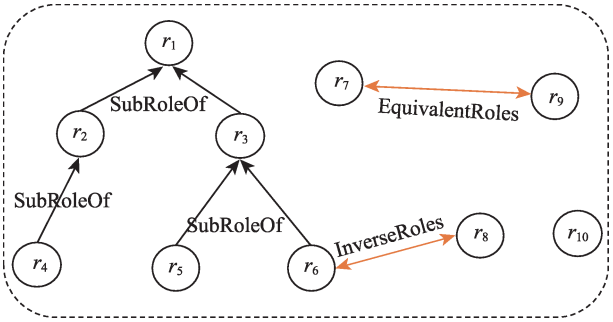

图3 角色存储示例图

和属性概念类似，角色与角色间的最重要关系仍然是上下位关系，是非对称关系，在图形数据库中表现为有向关系，而等价关系和互逆关系则具有对称性，在有向图中表现为双向的箭头，且等价和互逆的角色间有大量的重复信息，因而在规范化的本体构建过程中，对于等价的角色会进行合并以减少冗余，因而 EquivalentRoles 关系的使用频率也较少。互逆关系的存储同样涉及了语义信息的冗余问题，为了减少冗余，在实际的数据库存储过程中针对不同的图形数据库可采用不同的存储策略。

3）实例自交互（Instance×Instance）的映射规则

实例（Instance）又称个体，是类目概念的具体实例。因而，与类目概念类似实例集 $I=\{i_1,i_2,\cdots i_n\}$，可以点集的形式存储至图形数据库，形成有向图 $\text{Graph}_3=\{X_3,E_3\}$。由于实例代表的是单个的个体，其无法再进行拆分，因而，实例与实例间不存在包含关系，仅存在等价关系。因此，结合 OWL 语法的定义，X_3 的值域为实例集 I，E_3 的值域

为 {SameIndividual, DifferentIndividuals}，如图 4 所示。一般情况下在对本体添加实例时，为了减少冗余，默认实例间为相异关系，故 DifferentIndividuals 亦很少特意存储。

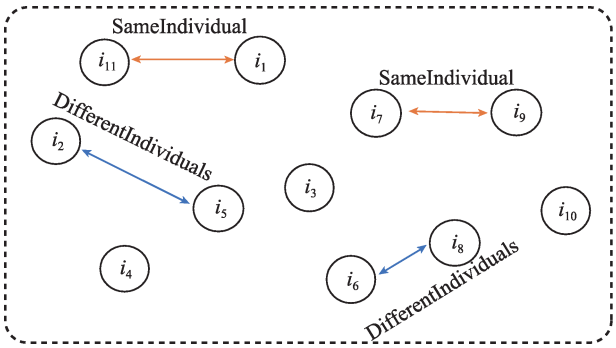

图4　实例存储示例图

4.1.2　基本构件两两交互关系的存储映射规则

基本构件间的两两交互涉及了结构层与实例层间的语义交互，相较于自交互更为复杂。尽管本体是以描述逻辑为基础的，但是其中部分语义关系具有特定的语义，使用频率高且具有通用性，其蕴含的语义为用户所熟知，因而在设计其图形数据库存储的映射规则时无需存储其复杂的逻辑表达式，而用一条边存储即可。基本构件间的两两交互关系就属于这种情况。

1）概念角色交互（Concept×Role）的映射规则

角色，即对象属性，描述了对象与对象间的二元关系。当概念集 C 与角色集 R 产生交互时，主要为 C 的子集类目概念集 $G=\{g_1,g_2\cdots g_n\}$ 与关系集 $R_c=\{r_{c_1},r_{c_2}\cdots r_{c_n}\}$ 的交互。由于类目概念与角色都通过点的形式存储，所以取点集 $X_4=G\cup R$，类目概念节点与角色间的互动关系则如前文所述，设定通用的边进行存储，本研究中取 $E_4=\{\text{RoleOf}\}$。同时，角色连接的概念关系是有向的，而由于单一的有向边难以区分特定角色连接的概念间的差异，故需对边设置相应的属性以区分角色的定义域与值域，通过对边“RoleOf”设置属性“Dereference”，其属性值可为“domain”或“range”，属性值为“domain”的边连接指向该角色的定义域，属性值为“range”的边则连接指向该角色的值域，得到图 $\text{Graph}_4=\{X_4,E_4\}$，如图 5 所示。图中对于角色 r_1 而言，它的定义域为类目 g_4，值域为类目 g_6。

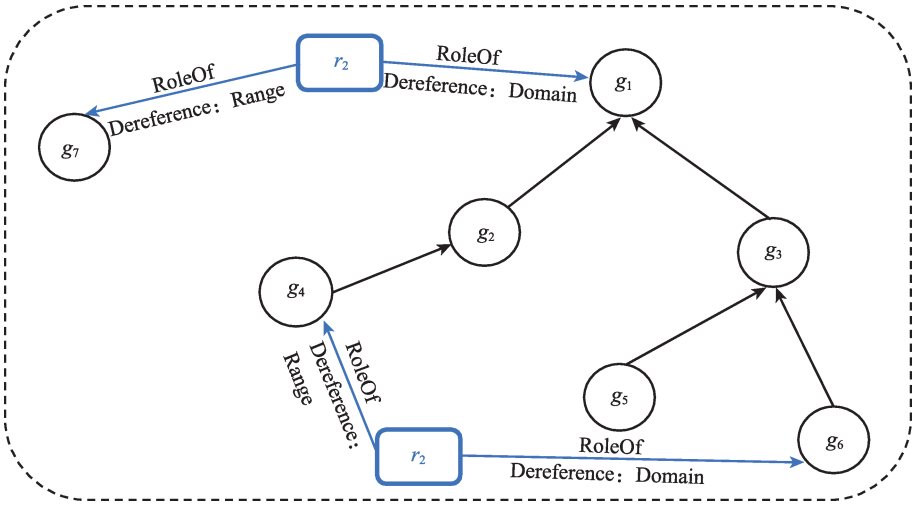

图5　概念角色存储示例图

2）概念实例交互（Concept×Instance）的映射规则

实例与概念的交互关系一方面涉及了实例层的存储规则，另一方面也实现了拓扑结构中的实例层与结构层的交互，是映射规则设计中最为复杂的部分。实例并不能脱离概念而存在，而是与其所在的类目绑定，同时其将继承类目的数据属性并拥有具体的属性值。因此，实际上每一个实例与其所在的类目概念以及类目所关联的属性概念之间都产生了交互，但由于类目概念与属性概念间的交互较大差异，因而存储方式也不尽相同。实例集 $I=\{i_1,i_2\cdots i_n\}$ 与类目概念集 G 产生交互时，设定由实例指向类目的边“IndividualOf”直接实现绑定。实例与类目绑定后意味着其将具有类目所具有的数据属性，因此具有该类目所具有的属性并可对其赋予属性值。此时产生了两种存储方式，一是建立新的节

点存储具体属性值，该方式保证了结构层与实例层属性存储方式的一致性（都采用节点方式），但该方式的存储不仅新增了节点也增加了节点间的连接边，存储空间成本较大。故本研究采用第二种方式，即将实例的属性与属性值作为实例节点的属性直接存储，在保证了语义完整性的同时节省了存储空间。最终，得到有向图 $\text{Graph}_5=\{X_5,E_5\}$，其中点集 $X_5=C\cup I=G\cup M\cup I$；$E_5=\{\text{IndividualOf}\}$，存储实例如图 6 所示。

3）角色实例交互（Concept×Role）的映射规则

实例与类目绑定后，不仅继承了类目的数据属性，也继承了其绑定的对象属性，即角色。然而，角色与类目绑定时通过定义域与值域进行表示，而具体到特定的实例，角色的语义指向更为明确具体。而且，在实例层也无需再重复存储角色之间的关系，因而可直接通过实例节点间的有向边进行表示，以减少冗余。该有向边的起点为定义域所在类目中的实例节点，而值域为值域所在类目中的实例节点。取实例集 $X_6=I$ 为点集，角色集为 E_5=R 为边集，且有向边由其在结构层的定义域类目所具有的实例指向值域类目所具有的实例，得到有向图 $\text{Graph}_6=\{X_6,E_6\}$，如图 7 所示。

4.1.3 基本构件共同交互关系的存储映射规则

当三个基本构件共同交互时，即为自交互与两两交互的汇总，并未增加新的语义关系，因而无需添加新的映射规则。当 OWL 本体中的概念、角色、实例以及它们之间的通用关系得以定义后，就构成了一个综合了本体结构与具体实例的基础本体，在

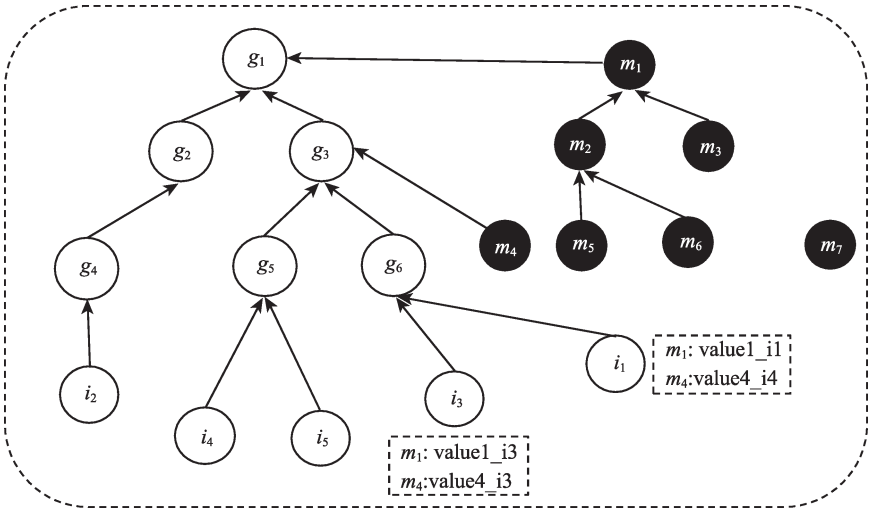

图 6 概念实例存储示例图

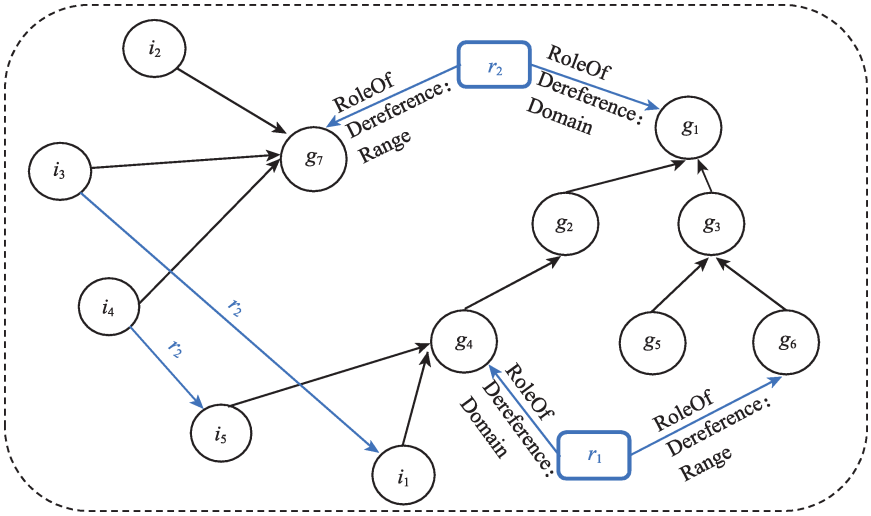

图 7 角色实例关系存储示例图

图形数据库存储时即为一张完整的有向图 GraphO = $Graph_1 \cup Graph_2 \cup Graph_3 \cup Graph_4 \cup Graph_5 \cup Graph_6 = \{C \cup R \cup I, E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5 \cup E_6\}$。因而，对本体O中属于基本构件及构件间的通用关系的任意元素I，其映射到图形数据库中的元素$J \in$GraphO。

4.2　基本构造算子存储映射规则

上述基本构件存储模型中，不仅对本体中涉及的最常用的基本元素，如类目、属性、实例、角色、通用关系，如包含关系（⊇/⊆），等价关系（≡/≠）进行了存储，也定义了其他一些重要且常用的关联的表示，类目与角色（RoleOf）、类目与实例（IndividualOf）间的特殊绑定关系，实现了最基础的本体图形数据库存储。

但面对不同的情境时，当概念与角色间的关系更为复杂且灵活性更高时，构造算子的应用将不可或缺。基础形式的描述逻辑ALC中的基本构造算子有交（⋂）、并（⋃）、补（¬）、存在量词（∃）和全称量词（∀），这些构造算子与基本概念或角色（节点）组合计算并产生新的复杂概念（节点），具有连接节点的作用，故本研究利用边对这些构造算子进行存储。

当进行复杂概念关系表示时，构造算子的组合运算具有一定的顺序，但是图形数据库中对指向同一节点的边的优先级本身不作区分。为了保证语义计算的正确性，每个节点每次仅能与构造算子进行一次运算[27]，而针对本身很复杂的概念，则通过引入中间节点，对其进行拆分直至原子概念，进行存储。如复杂概念 Mother ≡ (¬Male) ⋂ ∃hasChild.Person，则先将其拆分为¬Male和∃hasChild.Person两个中间概念，由于这两个中间概念仍然为复杂概念，故对其进行进一步分解，¬Male可看作Male通过¬运算得到，∃hasChild.Person则可拆分为∃hasChild和Person构成，其中，∃hasChild由角色hasChild计算得到，其仍然为一种角色，仍然有定义域和值域，而需要得到的复杂概念∃hasChild.Person为它的定义域，而Person概念则为值域。如图8所示，虚线框中概念都是为了定义Mother概念而增加的中间节点，属于为了表达语义而添加的冗余。

图8中涉及的构造算子有交（⋂）、补（¬）、存在量词（∃）三种，其中⋂涉及两个概念，是由两个概念组合成，故而会成双出现，如图中Mother概念有两个指向它的⋂边；同理，并（⋃）的表示方法和交（⋂）一致。其他三种构造算子则由单条有向边表示即可，但存在量词（∃）和全称量词（∀）则稍显复杂，其一般用于修饰对象属性，展示概念间的关系，因而经∃或∀运算后的角色仍然为角色节点，故其也有定义域和值域之分。

4.3　SHOIN(*D*)新增约束存储映射规则

4.1节和4.2节中对描述逻辑中的基本构件、基本构造算子及其间的关联的存储规则进行了详细的描述，已满足描述逻辑ALC的存储要求。ALC是表达能力最强的描述逻辑，其推理的复杂性自然受到一定的限制。通过在ALC的基础上增加不同的构造算子即形成了表达能力和推理能力各异的描述逻辑。目前最为常用的OWL DL本体描述语言就是以ALC拓展的描述逻辑SHOIN(*D*)，它在ALC的基础上增加了关系传递、关系层次、关系逆、概念枚举、绝对数量约束及数据类型集六种新的约束。其中，关系层次，即角色的层级关系，在4.1节的角色存储中已探讨过其存储方式与映射规则，证明是可实现存储的。其他的五种新增约束则可通过设置节点或边的属性实现，如表2所示。

（1）角色传递。角色传递即关系的传递性，对a,b,c ∈ C，若关系R满足aRb，且bRc，则aRc，那

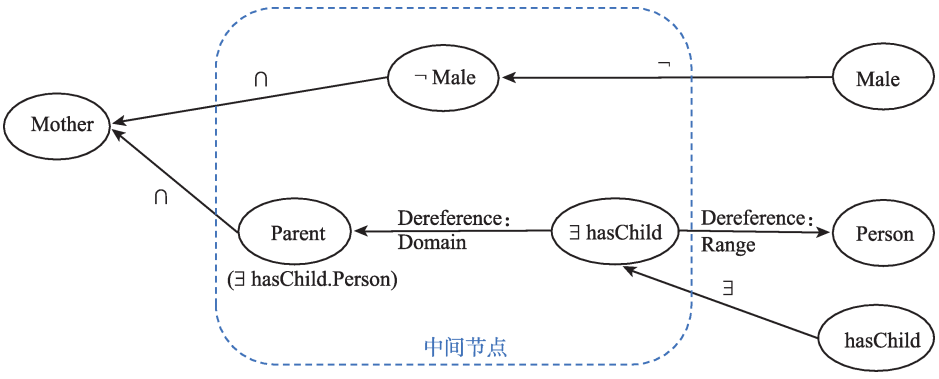

图8　复杂概念运算存储示例图

表 2 SHION(*D*)新增约束存储

SHION(*D*)新增约束	存储方式
角色传递	角色节点属性
角色逆	角色节点属性
概念枚举	类目节点属性
绝对数量约束	边属性
数据类型	属性节点属性

么关系 *R* 是具有传递性的。关系的传递性关系到本体中并无直接关联的概念节点间的语义关系，对本体查询也有重要的影响。本研究中需要本体构建者定义的角色（即关系属性）采用节点形式存储于本体中，故而可通过设置角色节点属性的方式对其进行规定与限制，设置属性 IfTransferable，属性值为 *T* 或 *F*，*T* 表示该关系可传递，*F* 表示不可传递。而对于本体中特殊的关系，层级关系，其通过边的形式存储且本身即具有传递性已是共识，因而可省略定义，在查询时默认其具有传递性。

（2）角色逆。若 *R* 是集合 *C* 上的关系，那么集合 $\{\langle c_i,c_j\rangle|\langle c_j,c_i\rangle\in R\}$ 是 *R* 的逆关系，记为 R^{-1}。本研究在通过有向图的形式存储本体时，通过定义域和值域的方式对角色的方向进行限定，鉴于角色和其逆角色仅仅为方向上的差异，因此可以通过为角色节点设置属性 InverseRole，属性值为逆关系的名称，进行标识。

（3）概念枚举。概念枚举使得 OWL 本体中可以存在枚举类，用 EnumeratedClass 表示，如 EnumeratedClass(DaysOfTheWeek Sunday Monday…Saturday)。对于枚举类，可为该节点设置属性 EnumeratedClass，属性值为枚举的实例。

（4）绝对数量约束。绝对数量约束是对本体中类与属性间关系的约束，在 OWL 中可分为 $\geqslant nR.C$、$\leqslant nR.C$、$=nR.C$、$\geqslant nR.D$、$\leqslant nR.D$、$=nR.D$，分别对类目和对象属性及数据属性的关系进行约束。对于对象属性，即角色的绝对数量约束是对角色的值域的数量的限定，以 ObjectMinCardinality(1 hasPart[Battery])为例，指该类目至少有一个电池，因而可对角色节点 hasPart 指向类目节点 Battery 的边设置属性 MinCardinality，属性值为 1。对数据属性而言，绝对数量约束是对其属性值的约束，以 DataMinCardinality(1 hasName[xsd:string])为例，其代表该类目至少有一个名字，因而可对由该类目指向数据属性 Name 的边设定属性 MinCardinality，属性值为 1。

（5）数据类型。数据类型是数据属性的重要约束，通过对属性节点增加属性 Datatype，并添加相应的数据类型作为属性值进行存储。同时，当为本体添加实例时，实例的属性值依旧要参照该约束。

5 存储模型评价

作为本体存储模型，需要保证利用该模型存储的本体能准确而完整地描述领域中的概念及概念关系，担当起领域知识共享桥梁的作用。因而，本文从模式规范性、模式结构稳定性、模式结构可理解性和语义完备性四个方面对提出的基于图形数据库的本体存储模型展开评价。

1）模式规范性

存储模式的规范性要求是为了保证数据的一致性。从图形数据库本身出发来看，图形数据库相较于关系型数据库数据结构更为灵活，更便于数据单元小型化和规范化，但现有的图形数据库出于商业考虑，大多数已经支持 ACID 性质，且可以通过设置唯一性约束、存在性约束对节点和边进行约束，结构灵活的同时不失规范。

从本研究中所提出的存储模型映射规则出发来看，本研究中提出的映射规则对于基本构造算子的存储采用每次仅能一次运算的方式进行存储，当某节点（如 Male、HasChild）经构造算子运算后的得到的新的复杂概念或角色（如¬Male、∃hasChild），仍然以节点的方式存储。前人的研究表明，在该存储模式下，若本体构建过程中未遵循规范的构建流程，本体质量无保证时可以通过计算最小不满足术语集（MUPS），对可满足性[27]与一致性[28]进行验证。因而，本研究提出的基于图形数据库的本体存储模型是一种规范的存储模型。

2）模式结构稳定性

存储模型的模式结构稳定性的目的在于增强鲁棒性，使得存储的本体能适应大数据时代信息更迭迅速的环境。从数据模型层面看，图形数据库往往提供了灵活的数据模型，其逻辑模型与物理模型间不会存在耦合，支持随时更改或添加数据类型或数据，本体更新时，仅需要按照已有规则添加、修改或删除相应的节点和边，这也更适用于本体的敏捷开发与迭代。

从拓扑结构层面看，本研究中将本体结构层与实例层采用不同的方式存储，保证了两个逻辑层的独立性，实例的更新不会影响结构层的存储，结构层概念与关系的更新也能最小限度地影响实例的存储。而在结构层中，将概念节点作为核心节点存

储，而角色节点作为边缘节点存储，连接不同的概念，也同样保证了更新角色节点数据时不会影响到概念节点的数据，而更新概念节点数据时对角色节点的影响也降到最低。

3）模式结构可理解性

模式结构的可理解性是存储模型用户友好度的重要指标，相较于其他存储模式，基于图形数据库的存储模式具有与生俱来的天然优势。当采用关系型数据库进行本体存储时，往往需要设立诸多的外键、中间表以表达复杂的语义，增加了用户的理解难度。而传统的OWL本体通过文档形式存储不利于阅读，若要形式化展现则需要通过Protégé等本体构建工具，若需要图形化展示还需导入插件的烦琐工序，且可展示的内容局限于类目的层级结构，对本体全局的理解有所不便。而本研究所提出的基于图形数据库的存储模型，在拓扑结构层面将本体分为结构层与实例层两个层面，便于用户理解本体元素间的关系。结构层的存储展示了本体中概念及概念关系的构成，通过图形数据库可直接展示其图结构，展现概念关联；实例存储具象化了每个个体，单个实例节点即可展示其所有的属性信息。

4）语义完备性

语义完备性的评估本质是对存储模式信息表达能力的评估。根据模式的表达能力强弱，可分为四层操作目标[29-30]（图9）。其中S1为原存储模式，S2为新存储模式。

图9　映射模式四层操作目标图

当新的存储模式满足第二层目标时候，新的存储模式已然可以完备地存储原存储模式中的信息，达到语义完备性。因而，当本文提出的图形数据库存储模型与OWL本体存储间的映射规则已构成单射函数时，即满足了语义完备性。

在本研究所提出的存储模式中，令本体O中的所有元素构成I(O)，根据本文提出的存储模型得到的图形数据库存储本体GraphO中的所有元素为I(GraphO)，本文提出的映射规则$\varphi:I(\mathrm{O})\rightarrow I(\mathrm{GraphO})$满足对$\xi\in I(\mathrm{O})$，则有$\varphi(\xi)\in I(\mathrm{GraphO})$。

结构层的映射规则较为直接，以类目概念为例，φ可形式化地表达为：

For$i$ = 1to$m$ //m个类
$\varphi(\xi)$ [ClassNode$_i$] ← Class$_i$

实例层的映射更为复杂，每个实例节点不仅存储实例名称和描述等基本信息，还存储了与结构层的概念绑定后定义的属性值等详细信息。因而以实例节点为例，φ可形式化地表达为：

For i = 1to m//m个类
Class$_i$ = {Prop$_i^1$,Prop$_i^2$⋯Prop$_i^n$}//类目Class$_i$有n个属性
For k = 1 to s//每个类有s个实例
For j = 1 to n
$\varphi(\xi)$[InsNode$_i^k$] ← Ins$_i^k$
//Ins$_i^k$为Class$_i$中第k个实例的名称，InsNode$_i^k$为$\varphi(\xi)$在图形数据库中实例节点名称
$\varphi(\xi)$[InsNode_Prop$_i^j$] ← Prop$_i^j$
//Prop$_i^j$为Class$_i$第j个属性,InsNode_Prop$_i^j$为$\varphi(\xi)$
$\varphi(\xi)$[InsNode_Prop_Value$_i^j$] ← ξ[Prop$_i^j$]
//InsNode_Prop_Value$_i^j$为节点$\varphi(\xi)$在图形数据库中Prop$_i^j$的属性值

（1）$\varphi(\xi)$的每个属性都为图形数据库中实例节点中属性的属性值，存在直接映射关系，映射规则φ为映射函数，满足第一层目标。

（2）假设I(O)中存在实例ξ_1和ξ_2。

$$\xi_1 = (\xi_1[\mathrm{Prop}_i^1], \xi_1[\mathrm{Prop}_i^2], \cdots, \xi_1[\mathrm{Prop}_i^n])$$
$$\xi_2 = (\xi_2[\mathrm{Prop}_i^1], \xi_2[\mathrm{Prop}_i^2], \cdots, \xi_2[\mathrm{Prop}_i^n])$$
且$\xi_1 \neq \xi_2$.

则至少存在一个$j \in \{1,2,\cdots,n\}$，使得$\xi_1[\mathrm{Prop}_i^j] \neq \xi_2[\mathrm{Prop}_i^j]$。

ξ_1和ξ_2映射到图形数据库中的信息分别为：

$$\varphi(\xi_1) = (\varphi(\xi_1)[\mathrm{InsNode}_i^1], \varphi(\xi_1)[\mathrm{InsNode\_Prop}_i^1], \varphi(\xi_1)[\mathrm{InsNode\_Prop\_Value}_i^1])$$
$$\varphi(\xi_2) = (\varphi(\xi_2)[\mathrm{InsNode}_i^2], \varphi(\xi_2)[\mathrm{InsNode\_Prop}_i^2], \varphi(\xi_2)[\mathrm{InsNode\_Prop\_Value}_i^2])$$
由于$\xi_1[\mathrm{Prop}_i^j] \neq \xi_2[\mathrm{Prop}_i^j]$

则至少存在一个$j \in \{1,2,\cdots,n\}$，

使得$\varphi(\xi_1)[\mathrm{InsNode_Prop_Value}_i^1] \neq \varphi(\xi_2)[\mathrm{InsNode_Prop_Value}_i^2]$

因此$\varphi(\xi_1) \neq \varphi(\xi_2)$

因此φ是一个单射函数，因而本文提出的模型满足模型映射的第二层目标，具备语义完备性。

由此可见，本研究所提出的图形数据库本体存储模型满足模式结构的规范性、稳定性、可理解性，且保证了语义的完备性，是一种理论上可行的规范存储模型。

6 结 语

本文从描述逻辑出发，提出了OWL本体的图形数据库存储模型的拓扑结构以及具体的映射规则，并对该模型的科学性和有效性进行了评价。在研究过程中，得出以下结论与启示：

（1）非关系型数据库的发展为本体的存储提供了新的契机。其中，图形数据库与本体的结构最为契合，适用于OWL本体的存储。在前人研究的基础上，本研究则将基于图形数据库的本体存储由RDF本体拓展至语义更为丰富的OWL本体，提出了相应的存储模型。在存储模型构建过程中，我们发现图形数据库展现出了诸多优势，如灵活的数据结构、直观的语义信息展示效果、信息存储能力丰富，再次印证了图形数据库可为本体数据存储的有效媒介。

（2）本体中的概念结构与实例具有不同的语义特征，在进行存储模型拓扑结构设计时将此二者进行区分，有利于实现低冗余的高效存储。本研究对本体中的概念、实例、关系等元素的特征展开分析，提出了从语义上对本体的概念结构和实例进行区分的双层存储模式。在概念结构层中，将类、数据属性、关系属性都作为节点单独存储，为用户提供了本体结构信息的完整理解；而在实例层中，则采用节点属性的形式存储实例属性及属性值，减少信息冗余。因而，对存储模型开发者而言，分析本体构成，根据本体的结构、内容主题、概念连通性等对本体进行拓扑结构分析或模块化划分，可作为本体存储模型构建的前期工作，辅助存储模型的设计。

（3）描述逻辑是OWL本体的理论基础，解决描述逻辑中的基本构件与构造算子的存储问题是本体的图形数据库存储模型设计的关键。OWL本体相较于RDF三元组形式更为复杂，存储模型设计容易出现缺乏理论基础指导的情况[31-33]，且少有研究对存储模型的正确性与完备性进行证明，故有可能存在语义信息存储不完备的现象。本研究从OWL DL本体的理论基础描述逻辑SHION(*D*)出发，通过分析SHION(*D*)中基本构件、构造算子与约束的存储方式，得到OWL本体图形数据库存储的映射规则，并以此为基础实现复杂语义的存储。这种存储方式的优势在于实现了本体查询和推理所需的最基础的元素的存储，可在此基础上拓展出各类复杂的语义而无需重新设计存储规范。可见，从本体的基础理论出发进行存储模型的设计是一种较为客观有效的方法；一方面从理论中的核心元素出发设计存储模型，解决了存储模型的底层逻辑问题；另一方面理论基础为模型的科学性奠定了基础，保证了模型的有效性。

参 考 文 献

[1] Horrocks I, Li L, Turi D, et al. The instance store: DL reasoning with large numbers of individuals[C]// Proceedings of the International Workshop on Description Logics. Whistler: CEUR-WS, 2004: 31-40.

[2] Protégé[EB/OL]. [2018-03-24]. http: //protege. stanford. edu.

[3] Jena Ontology API[EB/OL]. [2018-04-03]. http: //jena. apache. org/documentation/ontology/.

[4] Haarslev V. Racer: an OWL reasoning agent for the semantic web [C]// Proceedings of the International Workshop on Applications, Products and Services of Web-based Support Systems, 2003: 91-95.

[5] Bock J, Haase P, Qiu J, et al. Benchmarking OWL reasoners[C]// Proceedings of the the Workshop on Advancing Reasoning on the Web: Scalability & Commonsense, 2008: 1-15.

[6] 鲍文, 李冠宇. 本体存储技术研究[J]. 计算机技术与发展,

2008, 18(1): 146-150.

[7] 孙铭丽, 蒋婷, 傅柱 . 本体存储技术研究综述[C]// 全国知识组织与知识链接学术交流会论文集, 北京, 2012: 475-484.

[8] Agrawal R, Somani A, Xu Y. Storage and querying of e-commerce data[C]// Proceedings of the 27th VLDB Conference. San Francisco: Morgan Kaufmann Publishers, 2001: 149-158.

[9] Dehainsala H, Pierra G, Bellatreche L. Ontodb: An ontology-based database for data intensive applications[C]// Proceedings of the International Conference on Database Systems for Advanced Applications. Heidelberg: Springer, 2007: 497-508.

[10] Abadi D J, Marcus A, Madden S R, et al. Scalable semantic web data management using vertical partitioning[C]// Proceedings of the 33rd International Conference on Very Large Databases. San Francisco: Morgan Kaufmann Publishers, 2007: 411-422.

[11] Pan Z, Heflin J. DLDB: Extending relational databases to support semantic web queries[C]// Proceedings of the First International Workshop on Practical and Scalable Semantic Systems. Heidelberg: Springer, 2003: 109-113.

[12] 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(S1): 217-220.

[13] 杨炜辰, 凌海风, 武鹏, 等 . 基于关系数据库的本体存储模式[J]. 兵器装备工程学报, 2013, 34(4): 111-115.

[14] Motik B, Horrocks I, Sattler U. Bridging the gap between OWL and relational databases[J]. Journal of Web Semantics, 2009, 7 (2): 74-89.

[15] Xu Z M, Zhang S C, Dong Y S. Mapping between relational database schema and OWL ontology for deep annotation[C]// Proceedings of the IEEE / WIC / ACM International Conference on Web Intelligence. Los Alamitos: IEEE Computer Society Press, 2006: 548-552.

[16] Khalid A, Shah S A H, Qadir M A. OntRel: An ontology indexer to store OWL-DL ontologies and its instances[C]// Proceedings of the International Conference of Soft Computing and Pattern Recognition. Los Alamitos: IEEE Computer Society Press, 2009: 478-483.

[17] 刘炜, 夏翠娟, 张春景 . 大数据与关联数据: 正在到来的数据技术革命[J]. 现代图书情报技术, 2013(4): 2-9.

[18] Kim D J, Shin J H, Hong K S. Scalable RDF store based on Hbase and Mapreduce[C]// Proceedings of the International Conference on Advanced Computer Theory and Engineering. Los Alamitos: IEEE Computer Society Press, 2010: 633-636.

[19] Papailiou N, Tsoumakos D, Konstantinou I, et al. H2RDF+: An efficient data management system for big RDF graphs[C]// Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 2014: 909-912.

[20] 孙康 . 基于HBase的领域本体存储与查询方法研究[D]. 天津: 中国民航大学, 2016.

[21] 刘焘, 贾君枝 . 中文信息处理中的语义关系表示探析[J]. 现代图书情报技术, 2006(10): 25-29.

[22] Kozielski M, Stypka Ł. Gene ontology based gene analysis in graph database environment[J]. Studia Informatica, 2013, 34: 325-336.

[23] 吴鹏, 刘恒旺, 丁慧君 . 基于本体和NoSQL的机械产品方案设计的知识表示与存储研究[J]. 情报学报, 2017, 36(3): 285-296.

[24] 康杰华, 罗章璇 . 基于图形数据库Neo4j的RDF数据存储研究[J]. 信息技术, 2015(6): 115-117.

[25] 王红, 张青青, 蔡伟伟, 等 . 基于Neo4j的领域本体存储方法研究[J]. 计算机应用研究, 2017, 34(8): 2404-2407.

[26] Gómez-Pérez A. Ontology evaluation[M]//Handbook on Ontologies. Heidelberg: Springer, 2004: 251-273.

[27] Fu X F, Zhang Y, Qi G L. GrOD: Graph-based ontology debugging system[J]. Communications in Computer and Information Science, 2014, 480: 87-94.

[28] 付雪峰 . 基于图的DL-Lite本体不一致性处理方法的研究[D]. 南京: 东南大学, 2016.

[29] Miller R J, Ioannidis Y E, Ramakrishnan R. The use of information capacity in schema integration and translation[C]// Proceedings of the 19th International Conference on Very Large Data Bases. San Francisco: Morgan Kaufmann Publishers, 1993: 120-133.

[30] Kwan I, Fong J. Schema integration methodology and its verification by use of information capacity[J]. Information Systems. 1999, 24(5): 355-376.

[31] Astrova I, Korda N, Kalja A. Storing OWL ontologies in SQL relational databases[C]// Proceedings of World Academy of Science Engineering & Technology. Rhodes: World Scientific and Engineering Academy and Society, 2011: 242-247.

[32] Gorskis H, Borisov A. Storing an OWL2 Ontology in a relational database structure[C]// Proceedings of the 10th International Scientific and Practical Conference. Rezekne: Rezeknes Augstskola, 2015: 71-75.

[33] Woo E, Park M, Chung C. An efficient storage schema construction and retrieval technique for querying owl data[J]. Journal of KIISE: Databases, 2007, 34(3): 206-216.

（责任编辑　马　兰）