基于 Neo4j 的海量石油领域本体数据存储研究

宫法明 李翛然
(中国石油大学(华东)计算机与通信工程学院 山东 青岛 266580)

摘 要 语义网技术的发展促进了石油领域中多学科本体之间的整合技术的发展。随着数据的规模的增大,传统的基于关系型数据库的数据存储和信息检索等存在较多问题。对此,提出了一个基于 Neo4j 数据库的领域本体构建过程,专注于改进数据存储和信息检索两个方面。首先,提出了一种基于图形数据库 Neo4j 的大规模本体数据存储问题的解决方案,通过设计一种基于 Neo4j 的存储模型配合分布式存储机制,实现存储空间的高效利用。其次,在 Neo4j 数据模型的基础上,设计了一种两层索引结构的检索算法。实验评估表明,提出的方法与基于关系数据库的方法相比,在数据存储方面可以节省 10%以上的存储空间,在信息检索方面将检索效率提高了 30 多倍。

关键词 石油领域本体,RDF 数据,Neo4j 数据库,大数据存储,运动目标判断

中图法分类号 TP391 **文献标识码** A

Research on Ontology Data Storage of Massive Oil Field Based on Neo4j

GONG Fa-ming LI Xiao-ran
(College of Computer & Communication Engineering, China University of Petroleum, Qingdao, Shandong 266580, China)

Abstract The development of semantic web technology has promoted the development of integrated technology between multidisciplinary ontology in the oil field. As the scale of data increases, the traditional data storage and information retrieval based on relational database have encountered a lot of problems. In view of this problem, this paper proposed a domain ontology construction process based on Neo4j database to improve data storage and information retrieval. Firstly, this paper proposed a solution of large-scale ontology data storage problem based on Neo4j graphics database. By designing a distributed storage mechanism based on Neo4j storage model, the efficient use of storage space was realized . Secondly, based on the Neo4j data model, this paper designed a two-tier index architecture retrieval algorithm. In the light of experimental evaluation, compared with the method based on the relational database, the method proposed in this paper can save more than 10% storage space, and improve the search efficiency by more than 30 times.

Keywords Ontology in oil field, RDF data, Neo4j database, Big data storage, Moving object judgment

1 引言

本体是一种概念框架,可以对某一领域中知识的概念以及这些概念之间的关系进行形式化表示[1]。领域本体作为本体的一种,同时也是知识工程的重要组成部分,是对特定学科概念的描述[2]。一般来说,领域本体由 4 部分组成:领域学科、概念属性、概念关系集和属性关系约束,通常使用资源描述框架(RDF)、网络本体语言(OWL)和其他本体描述语言表示领域中的特定知识[3]。

石油领域本体描述了石油领域中的各种知识概念,以及概念、领域活动和领域特点之间的相互关系。在石油领域中,本体可以实现石油领域多学科的知识集成和信息集成,阐明术语与术语之间的关系及其领域公理,并对它们进行了形式化描述。近年来,"互联网+"与各行业进行了深入融合,随着石油开采的深化以及科学水平在大数据时代的迅速发展,越来越多的石油产业选择领域本体进行知识管理。尽管在技术上取得了一定的进展,数据的快速增长也给本体库带来了极大的挑战。许多领域本体数据集规模庞大,即使在本体数据管理方面取得了进展,对存储和搜索的成本要求仍然很高[4]。

基于传统关系型数据库的存储方法根据其存储结构的不同可以分为:三重表、水平分区和垂直分区等。三重表法将整个 RDF 数据存储到一个三列表中,每一行都分别对应 RDF 数据的主体资源、对应关系和客体资源。三重表法对于小型规模的数据有着十分优越的性能,但是随着数据规模的增加,会产生大量的自连接,导致数据处理效率大幅度降低。水平分区法是将所有 RDF 数据存储到一个表中,该表为 RDF 数据的每个谓词值指定一个专用列,且该表支持多值属性,但由于稀疏属性导致大量空单元格,因此该存储方法不适用于大规模数据的存储。垂直分区方法将三重表重写为 n 个两列表,其中 n 是数据中唯一属性的数量,对指定谓词值的查询的执行效率高,但随着数据规模的增加,信息的检索时间将呈指数增长。

本体的构建方法[5-7]可以分为手动构建和半自动化构建。

本文受科技部创新方法工作专项基金资助。

宫法明(1970-),硕士,副教授,主要研究方向为自然语言处理、大数据、深度学习、计算机视觉,E-mail:gfaming@163.com;李翛然(1993-),男,硕士,主要研究方向为自然语言处理,E-mail:lixiaoran93@163.com(通信作者)。

研究人员已经提出了诸如 IDEF-5、Skeletal Methodolody、TOVE 企业建模法、Methontology 和循环采集等多种构建方法。图 1 给出了本文提出的石油领域本体的构建流程。该方法的提出受到了经典的本体构建法——Skeletal Methodolody (SM)的启发。然而,除 SM 方法之外,我们还综合考虑了石油领域的多学科特征以及后期本体扩展的需要,因此制定了特定的流程,具体步骤有:知识提取、本体表示、本体存储、本体检索、本体扩展。

图 1 本体构建流程图

本文特别关注存储和检索步骤。我们设计了一个本体文件(RDF 数据)到 Neo4j 数据库的映射规则。通过在 RDF 图的数据结构和 Neo4j 数据库的存储结构之间建立映射关系,实现了 Neo4j 中本体数据的有效转储。为了应对海量数据的存储问题,我们提出了基于 Neo4j 数据库的分布式存储机制。同时,采用双层索引架构(包括对象索引和三元组索引)来减少加载时间,并匹配不同的匹配模式以便精确检索。此外,还提出了一种基于此架构的检索方法。鉴于评估,与使用关系数据库的方法相比,该方法可以节省 10%以上的存储空间,使检索效率提高 30 多倍。

本文第 2 节介绍了基于图形数据库 Neo4j 的 RDF 数据的存储和检索的相关方法;第 3 节介绍了 RDF 数据与 Neo4j 数据模型之间的映射关系,并提出了数据存储的方法和两层索引的查询处理机制;第 4 节将本文提出的方法与传统基于关系型数据库的方法进行了实验对比,并分析了其优劣;最后总结本文,并提出了未来的研究方向。

2 准备工作

对于领域本体,优秀的数据存储模型可以提高知识的检索效率,同时也便于本体的扩展需求。现阶段,较多学者选择使用关系型数据库进行本体的存储。WANG 等[8]设计了一个基于 OWL 文件的关系数据模型,并开发了一种从 OWL 文档转换为关系模式的本体文件存储系统。Khalid 等[9]提出了一系列如何将 OWL 文件传输到数据库的规则,并提出了一个"OntRel"关系模型。Thaker 等[10]提出了一种基于 XPath 结构的存储模型(XPOS),该模型使用类和属性的层次信息作为 XPath 结构,并且可以进行直观有效的信息检索。陶皖等[11]针对 OWL 本体和属性的特点,通过建立关系表来改进现有的模式,并添加关系约束表 Trestrict,使其更容易实现类、属性和复杂关系等信息在 OWL 本体中的存储。本体数据的基本单位是三元组,包含概念以及概念之间的关系,其数据结构是一种有向图结构,由于关系型数据库的数据存储结构是二维表格,将本体数据转换存储在关系型数据库中会出现"阻抗不匹配"的问题[12]。Elbauah[13]选择了用于本体存储的面向对象的数据库来解决这一问题。虽然基于面向对象的数据库的本体存储模型已经取得了初步的进展,但该技术仍不足以支持大数据环境下的海量石油领域本体数据库。

资源描述框架(Resource Description Framework,RDF)是一个使用 XML 语法来表示的资料模型,用于描述 Web 资源的特性,以及资源与资源之间的关系[14]。RDF 数据中的每个基本结构都是一个包含主体资源、对应关系、客体资源的三元组。RDF 数据表示为(S,P,O)三元组[15]:

$$t=\langle s,p,o\rangle\in(I\cup B)\times I\times(I\times B\times L) \tag{1}$$

其中,t 代表 RDF 三元组,即 I 表示统一资源标识符(URL),B 表示空节点,L 表示文字。多个相关联的 RDF 三元组可以组成 RDF 有向图,RDF 有向图由被标记的节点和边缘表示,描述了主体资源与其所指向的客体资源之间的属性关系。RDF 有向图是其包含的所有三元组的主体资源,并且边的方向总是指向其对应的客体资源。通常,一组 RDF 数据可以形成一个 RDF 有向图。如图 2 所示,它是一个 RDF 有向图和它对应的一组 RDF 数据。

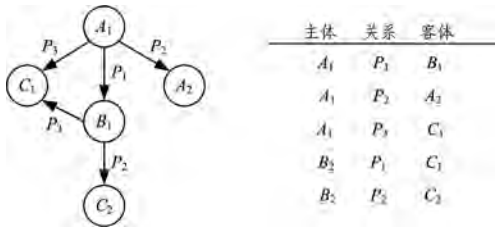

图 2 一个 RDF 图和其对应的三元组

石油勘探开发领域包含了地质、勘探、钻井、机械、地下作业、石油生产和储油运输等 20 多个分支领域。其领域本体选用 OWL-DL 语言进行描述,由 OWL 文件进行存储。通过解析 OWL 文件,可以将本体数据表示为 RDF 数据,进而表示为 RDF 有向图。因此,本体数据的存储实质是指 RDF 有向图数据的存储,RDF 有向图数据的存储的基本单位为 RDF 三元组。

Neo4j 是一种 NoSQL 类型的数据库,与传统 SQL 类型的数据库相比,它的存储结构不是二维列表结构,而是一种图结构。图结构作为一种通用的数据结构,可以有效地对数据建模,从而使 Neo4j 数据模型在表达能力方面表现出众。例如链接列表、树和哈希表以及其他数据结构也可以在图结构中得以兼容。图 3 给出了 Neo4j 数据模型的结构图,它由节点、属性以及节点间的对应关系 3 个要素组成,3 个要素完整地描述了任何用户的情况。该存储模型的优势在于,可以灵活扩展网络模式、节点属性可以随时添加或删除,从而有效解决半结构化、非结构化数据的存储、内存的浪费等问题。同时,得益于其独特的数据模型,Neo4j 数据库通过深度遍历等方式可以快速查询节点的相关信息。

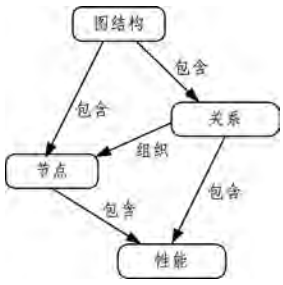

图 3 Neo4j 数据模型结构图

3 基于 Neo4j 的存储检索方法

本文受到了经典的本体构建方法 Skeletal Methodolody (SM)[16]的启发,并结合石油领域知识概念的特点,提出了基于"五步法"的石油领域本体的生命周期构建法。

3.1　石油领域本体的结构

石油领域本体的结构是一个五元组 $O:=\{C,R,Hc,Rel,Ao\}$[17]。通过石油领域不同科目之间沟通的语义基础，石油领域本体由描述油田某种情况的特定术语组合以及相应表示的假设组成。领域本体不仅可以描述概念的层次结构，还可以表达概念之间的其他关系，并通过添加一组适当的关系、公理、规则来约束概念的内涵解释。完整的本体包括 5 个要素：类(C)、关系(R)、属性(At)、公理(Rel)和实例(Ao)。图 4 给出了石油领域本体的 RDF 有向标记图的一个例子。其中，实线椭圆表示本体的类、公理和实例，虚线椭圆表示对应的属性，有向箭头表示语义之间的对应关系。

1)类(C)。除了概念的一般意义外，还可以将 RDF 三元组中的任务、动作和事件等名称表示为主体资源和客体资源。例如，“油气勘探开发”是一个类，使用三元组形式表示为(油气勘探开发，rdfs:type，Owl:class)。

2)关系(R)。通过定义和约束概念，描述了领域中概念之间的关系。其中，定义域由概念集合中的概念组成，而值域可以由概念、数值等数据类型组成。领域本体之间的主要关系包括子类关系(subClassOf)以及实例与概念之间的关系(edf:type)。例如：subClassOf(geological object，oil and gas field)，就是一个子类关系的代表，其意义为油气田是地质对象的子类。

3)属性(At)。领域本体的概念包含两个属性，即对象属性和数据属性。对象属性是指对象之间彼此相关联，数据属性是指对象与数据类型值之间相关联。

4)公理(Rel)。其是对永恒真理的描述，在任何情况下都是真实的。

5)实例(Ao)。其是类的具体实例。在领域本体的范围内，实例继承相关类的属性和关系。例如，塔中一井(rdfs:type，Owl:钻孔)表示塔中一井是钻孔型油井的一个实例。

由此可见，石油领域本体是由海量 RDF 数据组成的。RDF 数据的元素之间相互联系，组成了一张石油领域的知识网络，其中每个 RDF 数据中的主体资源和客体资源都被多次引用。因此，在存储该知识网络时，以使用关系型数据库进行本体存储为例[18]，由于关系型数据库是基于表进行存储的，会采用多张表的形式对同一本体进行存储，导致相同的 RDF 数据会在多张表中重复存储，这样极大地浪费了数据存储空间，也不利于数据的操作和维护。针对该问题，本文参考 No-Sql 数据库中的图数据库 Neo4j 的存储原理，提出了 RDF 数据到 Neo4j 数据库的映射规则解，有效地解决了此类问题。

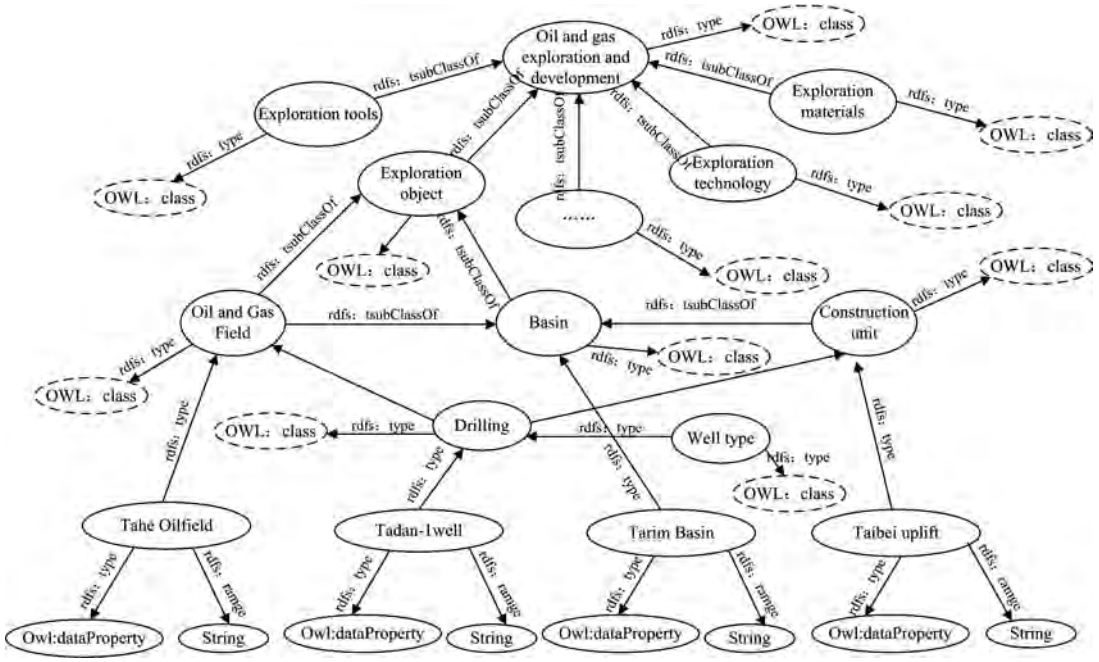

图 4　石油本体 RDF 方向标记图的一个例子

3.2　Neo4j 的存储功能

Neo4j 图也称为属性图(PG)[19]，PG 的重要组成方式是节点和关系。其中，每一个关系都需要连接对应的起始节点，节点和关系都可以拥有自己的属性，可以为每一个节点赋予多个类型的标签。

Neo4j 的底层会通过图的形式将节点以及关系存储起来，通过这种方式可以实现从某个节点开始，通过节点和节点之间的关系找出两个节点之间的联系。Neo4j 的基本结构由节点、关系和属性组成[20]。节点可以分为起始节点和终止节点，并且两个节点通过关系连接。属性是对不同节点的补充。Neo4j 的每个节点都有一个标签，分为 iri，literal 和 bnode。iri 节点有两个属性，即 kind 和 IRI；literal 节点有 4 个属性，即 kind，value，datatype，language；bnode 只有一个属性。将 4 种不同的具有不同属性的相连节点以链表的形式存储，如图 5 中的 Neo4j 数据库的存储过程所示。而概念之间也根据其相互联系的不同分为 4 种不同的关系(见表 1)，为概念间的关系分类。

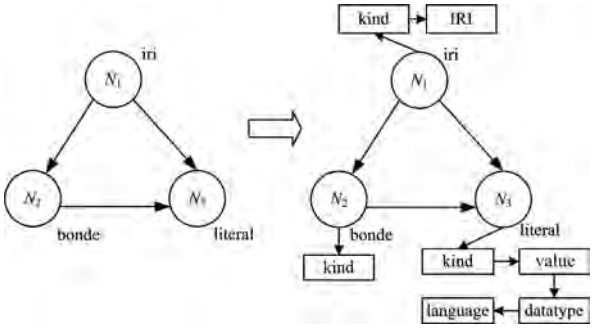

图 5　Neo4j 数据库的存储过程

表 1 概念间的关系分类

关系名称	表示	含义	区别
包含	Kind-of	表示概念之间的继承关系，与面向对象思想中的父、子类关系类似	上下位概念的属性存在传递关系
概念与实例	Instance-of	表示概念与实例之间的关系，与面向对象思想中的类和对象之间的关系相类似	概念可以被进一步地细分，实例则不可
部分与整体	Part-of	表示概念之间整体与部分的关系	整体和部分的属性不存在传递的关系
概念与属性	Attribute-of	表示概念是另一个概念的属性	没有很严格的界限，某些属性既可以是概念也可以是属性

3.3 RDF 数据到 Neo4j 的映射规则

RDF 有向图由主体资源、对应关系和客体资源表示，主体资源可以对类和概念进行表达，对应关系负责表示主体资源和客体资源之间的关系[21]，客体资源除了可以表达类和概念之外，还可以表示类的定义和属性。Neo4j 数据模型由节点、关系、节点和关系的属性组成，与图 4 中石油领域本体部分信息的 RDF 有向图在结构模型上存在相似性。本文通过建立系列的映射规则，实现了 RDF 有向图到 Neo4j 数据结构的映射，如图 6 所示。

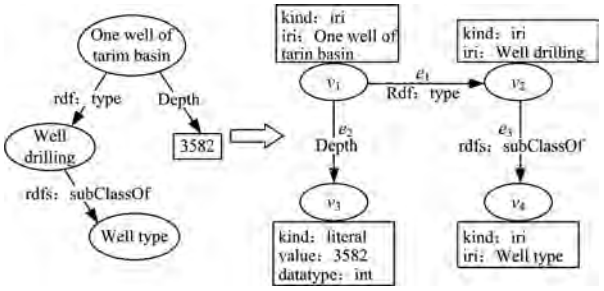

图 6 将 RDF 有向图映射到 Neo4j 的数据结构

以下是从 RDF 有向图到 Neo4j 数据库的映射步骤：

1)遍历 RDF 有向图中的每个属性值。在 Neo4j 中，RDF 有向图中的每个属性值都由相应的节点生成。每个节点可以与多个节点建立多个关系，单个节点可以设置多个属性。例如，$V=\{v_1,v_2,v_3,v_4\}$ 是 Neo4j 数据库中 RDF 有向图被映射的一组节点。

2)对于节点集合 V 中的每个空节点(bnode)$v(b)$，获取属性集，表示该节点除了类型标签之外没有额外的属性。

3)对于节点集合 V 中的每个资源标识符节点(iri)$v(u)$，分别得到节点类型和“IRI”标签的属性集合，例如：

$$\phi(v(u_1))=\{\langle\text{“kind”},\text{“IRI”}\rangle,\langle\text{“IRI”},\text{“One well of tarin basin”}\rangle\}$$

$$\phi(v(u_2))=\{\langle\text{“kind”},\text{“IRI”}\rangle,\langle\text{“IRI”},\text{“Well drilling”}\rangle\} \tag{2}$$

$$\phi(v(u_4))=\{\langle\text{“kind”},\text{“IRI”}\rangle,\langle\text{“IRI”},\text{“Well type”}\rangle\}$$

4)对于节点集 V 中的每个文字节点(literal)$v(l)$，获取属性集：

$$\varphi(v(l))=\{\langle\text{“kind”},\text{“literal”}\rangle,\langle\text{“value”},\text{vm}^{-1}(\text{l})\rangle \langle\text{“datatype”},\text{im}(\text{dtype}(\text{l}))\rangle\}\bigcup\text{language} \tag{3}$$

分别获取节点类型和“值”“数据类型”“language”属性，其中“language”属性可以为 null。例如：

$$\phi(v(u_3))=\{\langle\text{“kind”},\text{“literal”}\rangle,\langle\text{“literal”},3582\rangle,\langle\text{“datatype”},\text{int}\rangle\} \tag{4}$$

5)Neo4j 数据库中的每条边都代表不同的 RDF 三元组。例如，$E=\{e_1,e_2,e_3\}$ 是 Neo4j 数据库中的 RDF 有向图的边集。

6)对于每个三元组 $t=\langle s,p,o\rangle$，边缘的标签对应于 $lbl(p)$，起始节点和结束节点是 $src(p)$和 $tgt(p)$：

$$src(e1)=v_1,tgt(e_1)=v_2,lbl(e_1)=\text{“rdf:type”}$$

$$src(e_2)=v_1,tgt(e_2)=v_3,lbl(e_2)=\text{“Depth”}$$

$$src(e_3)=v_2,tgt(e_3)=v_4,lbl(e_3)=\text{“rdfs:subClassOf”} \tag{5}$$

本文主要基于 java jena API 方法在 eclipse 环境下实现 RDF 文件到 Neo4j 数据存储模型的映射转换。

3.4 基于 Neo4j 数据库的分布式存储

基于 Neo4j 数据库的数据存储方法与关系型数据库的存储方法相比，在相同大小的存储空间中，虽然基于 Neo4j 数据库的数据存储方法可以存储更多本体数据，但是相比于数据的增量，单一硬盘的容量是远远不够的。为了适应海量数据的存储需求，分布式的存储方式是需要解决的主要问题。

本文特别提出了一个基于 Neo4j 数据库的分布式存储框架来实现 Neo4j 数据库的分布式存储功能。如图 7 所示，该分布式存储框架由存储层和逻辑层组成。存储层由多个独立的存储节点组成，每个存储节点运行 Neo4j 图数据库，通过 Neo4j 自带的 API 接口进行访问。逻辑层由一个键值(Key-Value)型数据库和许多功能模块组成。其中，“数据管理”模块将对 RDF 数据集进行分割处理，并决定 RDF 数据的存储位置；“数据更新”模块负责数据的存储更新，将 RDF 数据存储到指定的 Neo4j 存储节点上；“数据查询”模块依靠双层索引机制定位查询数据所在存储节点，实现高效查询。

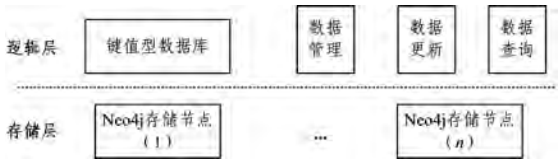

图 7 Neo4j 数据库分布式存储框架

为了方便数据的分布式存储，对 RDF 数据集进行分割，并确定每个 RDF 数据的存储位置。结合 RDF 数据之间疏密程度不同的关联度，在进行 RDF 数据集分割时，根据数据的被引用和查询的热门度对 RDF 数据进行定义，分为热门数据、一般数据和冷门数据 3 种类型。在执行数据分割时，应当遵循以下 3 条原则[22]：

1)按照数据的不同类型进行存储，将每种类型数据中关联度高的数据尽量存储在同一个存储节点，以减少跨界点通信带来的资源消耗；

2)对于与其他数据关联度高的热门数据，可以将其备份存储在不同的存储节点，以便同一存储节点的数据调用；

3)考虑机器的存储容量和计算能力，每个存储结点的数据量应大致相同，防止有的节点负载过重而有的节点负载过轻。

在逻辑层，我们使用了一个键值型数据库来存储 RDF 数据的存储位置信息。键值型数据是一种 NoSQL 数据库，其

特点是模式资源、高查询性能和支持持久化存储，适合大规模非结构化数据分析的应用场景。对于一个 RDF 三元组而言，其对应于 RDF 图中的两个顶点、一条有向边，我们依次将这两个顶点、有向边的存储位置信息存入键值型数据库，将顶点 ID、边 ID 作为 Key，存储位置作为 Value。

3.5 基于两层索引架构的检索方法

Neo4j 图形数据存储结构与传统的关系型数据库的存储结构有着极大不同，为了适用 Neo4j 数据库图形化的存储结构，优化本体数据的检索算法，我们使用 Neo4j 数据库匹配的 CYPHER[23] 搜索语言和 Apache Solr[24] 索引技术并结合石油领域中多学科领域的特点，提出了适用于石油领域本体的检索算法。

首先应该创建两个索引机制：1）对象索引；2）三元组索引。表 2 和表 3 分别列出了对象索引和三元组索引。其中，所有对象都被索引机制分配了一个 id 数字以建立索引表，方便查询时快速查询对象。

表 2　对象索引

意义	标签
编号	id
名称	label
其他名称	altLabel
实体类型	entityType

表 3　三元组索引

意义	主体资源	对应关系	客体资源
编号	sID	pID	oID
名称	sLabel	pLabel	oLabel
别名	sType	pType	oType
实体类型	sTypeValue	pTypeValue	oTypeValue

搜索引擎分析用户的搜索语句，并根据检索要求构建以下几种不同的检索表达式，并分别执行检索任务。

1）对象匹配检索。对象匹配通过查询精确匹配或模糊匹配的结果来检索对象索引，并对结果集进行排序得出结果集的相关性。例如名为“塔里木盆地”，其搜索范围为：

Query=(label:“Tarim Basin”) or (altLabel:“Tarim Basin”)

2）关系匹配检索。关系匹配搜索主要用于检索方法的三元组索引。在指定的三元组(s,p,o)中，需要检索 s 和 o 之间的关系。例如，需要检索“探索技术”对象，搜索引擎首先在 id 索引表中找到“探索技术”对应的 id，访问相应的 id 为“9672”，其结构关系搜索为：

Query=(sID:‘9672’) or (oID:‘9672 ’)

如果知道对象及其指定的熟悉程度，可以通过三元组(s,p,o)已知的 s 和 p 或 o 和 p 来检索另一个相应的对象，需要使用对象 o 或 s 来检索另一个匹配项。例如，需要检索“石油”的“成分”。结合相互规则体系，给定的搜索结构如下：

Query=((sLabel:‘oil’) and (pLabel:‘composition’)) or ((pLabel:‘composition’) and (sLabel:‘oil’))

3）关系度检索：关系度检索是根据石油领域本体中三元组集合形成的图形结构特征，在已知两个对象不同的情况下，通过查询两个对象所对应的实例节点之间的路径，将路径根据距离定义为“近”“较远”“远”等选项，并且可以自由选择将路径长度设置为不超过 2，3 或 4，并返回节点的路径。实施 Neo4j 数据库关联度检索与地图遍历机制，通过 CYPHER 查询 Neo4j 数据库来灵活访问查询结果。例如，查询“胜利油田”与“塔里木油田”之间的关系。CYPHER 查询语句是：

```
start a=node(*),b=node(*) match p=a-[*0.2]-b
where was a.label=“Shengli Oilfield”
and b.label=“Tarim Oilfield”
return p order by length(p);
```

4 实验

4.1 实验设计

实验评估的目的是：

1）将所提方法与传统基于关系型数据库的方法进行比较，以验证所提方法的优劣；

2）用不同方法验证不同规模的数据集的查询响应时间；

3）用不同方法验证不同规模的数据集所占用的存储空间。

4.2 实验设置

为了进一步验证所提方法的优点，用柏林 SPARQL 基准(BSBM)[25] 标准数据集进行了测试。首先，将数据集按照规模大小分为 5 个不同的子数据集，并分别将它们命名为 a，b，c，d，e，如表 4 所列。其次，分别使用本文提出的存储方法和检索方法以及传统基于关系型数据库的存储方法和检索方法对不同规模的数据集进行对比实验。所有的实验均是在带有 Intel Core i5-5200U 处理器、4 GB RAM 以及 230 GB(SSD)的 Windows 7 设备上进行的。

表 4　5 组不同规模的数据集

编号	a	b	c	d	e
RDF 三元组数量	50000	250000	1000000	5000000	25000000

4.3 实验执行

在 RDF 数据存储实验中，将 RDF 数据三元组放入 Neo4j 数据库中，并将其存储在 Java 中，将实验所用数据集按照不同规模划分为 5 个大小不同的数据集，并分别命名为 Na，Nb，Nc，Nd，Ne。同时，通过创建表等多种形式将相同规模的数据集在关系型数据库中进行存储，并将其命名为 Ra，Rb，Rc，Rd，Re。

在检索效率的比较实验中，基于 Na，Nb，Nc，Nd，Ne 5 个数据集进行检索，分别记录本文提出的双层索引的检索方法所消耗的时间。同时，使用传统的 SQL 查询方法在 Ra，Rb，Rc，Rd，Re 中检索相同的问题，并记录检索时间。实验使用相同的方法比较了 5 种不同大小的数据集查询效率的实验结果，并对所提存储方法与传统的存储方法在相同大小的数据集下的检索结果进行了比较。

4.4 结果

如图 8 和图 9 所示，虚线表示本文提出的存储方法和检索方法，实线表示传统关系数据库的存储方法和检索方法，它比较了不同规模大小的数据集使用不同方法占用的存储空间的大小和检索相同问题所花费的时间。由结果可以得出以下结论：

1）在规模相同的数据中，所提存储方法可以比传统的存储方法节省 10％的存储空间；

2）在规模相同的数据中，所提搜索方法的效率比传统的 sql 查询方法高 30 倍。

实验结果表明，随着数据集的增加，所提存储映射方法和检索方法相比传统方法更具适应性。

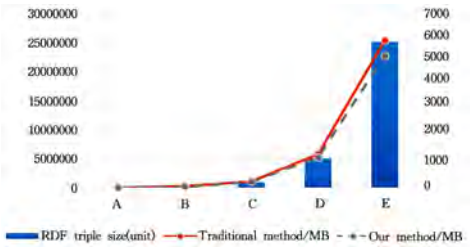

图 8 存储实验结果图

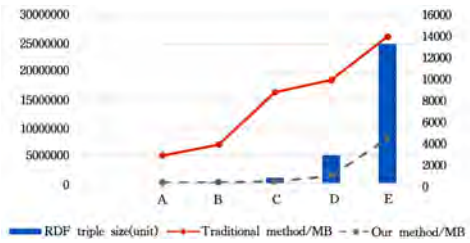

图 9 查询实验结果图

结束语 本文提出了一种基于 Neo4j 图形数据库的领域本体构建过程和基于双层索引架构的检索方法。评估结果表明，所提方法可以节省 10%的存储空间，其检索效率是关系型数据库的 30 倍，这些方法是构建大规模领域本体论的主要步骤。

这项工作是更大研究项目的第一步，它大大改进了大规模的 RDF 数据管理，所提方法只关注建筑领域本体过程的 5 个阶段中的 2 个。在未来，我们的目标是支持本体扩展和多本体整合的后续阶段[26]，根据需要使不同的本体集成为一个新的本体，对于此问题，可以通过将 RDF 数据映射到现有的内部信息结构，并建立私有和公共数据之间的共同演进，这涉及来自 RDF 源的连续更新传播，同时保留对这些数据集的先前版本的修改。

参 考 文 献

[1] ISOTANI S,IBERT BITTENCOURT I,BARBOSA E F,et al. Ontology Driven Software Engineering:A Review of Challenges and Opportunities[J]. IEEE Latin America Transactions,2015,13(3):863-869.

[2] JONES M V,COVIELLO N,TANG Y K. International Entrepreneurship research (1989-2009):A domain ontology and thematic analysis[J]. Journal of Business Venturing,2011,26(6):632-659.

[3] SEQUEDA J F,ARENAS M,MIRANKER D P. On Directly Mapping Relational Databases to RDF and OWL[C]// International Conference on World Wide Web. ACM,2012:649-658.

[4] LIU B,HUANG K,LI J,et al. An incremental and distributed inference method for large-scale ontologies based on MapReduce paradigm[J]. IEEE Transactions on Cybernetics,2015,45(1):53.

[5] 段炼. 基于文本分析的石油领域本体自动构建方法的研究[D]. 大庆:东北石油大学,2015.

[6] 文必龙,段炼,汪志群,等. 基于语料库和规则库的石油本体自动构建研究[J]. 计算机技术与发展,2015(9):209-212.

[7] 王月. 基于本体的油田开发知识库构建研究[D]. 大庆:东北石油大学,2016.

[8] WANG E,YONG S K,KIM H S,et al. Ontology Modeling and Storage System for Robot Context Understanding[M]// Knowledge-Based Intelligent Information and Engineering Systems. Springer Berlin Heidelberg,2005:922-929.

[9] KHALID A,SHAH S A H,QADIR M A. OntRel:An Ontology Indexer to store OWL-DL Ontologies and its Instances[C]// Soft Computing and Pattern Recognition. 2009:478-483.

[10] THAKER R,GOEL A. Domain Specific Ontology based Query processing System for Urdu Language[J]. International Journal of Computer Applications,2015,121(3):20-23.

[11] 陶皖,姚红燕. OWL 本体关系数据库存储模式设计[J]. 计算机技术与发展,2007,17(2):111-114.

[12] PINKEL C,BINNIG C,JIMÉNEZRUIZ E,et al. RODI:A Benchmark for Automatic Mapping Generation in Relational-to-Ontology Data Integration[M]// The Semantic Web. Latest Advances and New Domains. Springer International Publishing,2015:21-37.

[13] ELBATTAH M,ROUSHDY M,AREF M,et al. Large-scale ontology storage and query using graph database-oriented approach:The case of Freebase[C]// IEEE Seventh International Conference on Intelligent Computing and Information Systems. IEEE,2016:39-43.

[14] HUANG J,ABADI D J,REN K. Scalable SPARQL querying of large RDF graphs[J]. Proceedings of the Vldb Endowment,2011,4:1123-1134.

[15] MALEWICZ G,AUSTERN M H,BIK A J C,et al. Pregel:a system for large-scale graph processing[C]// ACM SIGMOD International Conference on Management of Data. ACM,2010:135-146.

[16] KHAN L,LUO F. Ontology construction for information selection[C]// IEEE International Conference on TOOLS with Artificial Intelligence. IEEE Xplore,2002:122-127.

[17] DOMBAYCI C,FARRERES J,RODRíGUEZ H,et al. On the Process of Building a Process Systems Engineering Ontology Using a Semi-Automatic Construction Approach[J]. Computer Aided Chemical Engineering,2016,37:941-946.

[18] 张晓冉,舒眷. 基于关系数据库的油田领域数据质量本体构建[J]. 微型电脑应用,2016,32(7):71-73.

[19] HARTIG O. Reconciliation of RDF * and Property Graphs[R]. University of Waterloo,2014.

[20] 张前进. 基于 Neo4j 的智能学习系统语义链接图式存储研究[J]. 佳木斯大学学报(自然科学版),2017,35(2):299-301.

[21] 彭安琪. 分布式溯源信息存储系统的研究与实现[D]. 成都:电子科技大学,2016.

[22] 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.

[23] HOLZSCHUHER F. Performance of graph query languages:comparison of cypher,gremlin and native access in Neo4j[C]// Joint EDBT/ICDT 2013 Workshop GraphQ. 2013:195-204.

[24] KU R F. Apache Solr 4 Cookbook[M]. Packt Publishing,2013.

[25] ASSOCIATION I R M. International journal on Semantic Web and information systems[J]. Journal of Polymer Science Polymer Chemistry Edition,2013,22(10):2625-2640.

[26] RAJITH A,NISHIMURA S,YOKOTA H. JARS:Join-Aware Distributed RDF Storage[C]// International Database Engineering & Applications Symposium. ACM,2016:264-271.