基于 NoSQL 的 RDF 数据存储与查询技术综述*

王林彬[1,2]，黎建辉[2]，沈志宏[2]
(1. 中国科学院大学，北京 100190；2. 中国科学院计算机网络信息中心 科学数据中心，北京 100190)

摘 要： 随着语义网的发展和 RDF(resource description framework，资源描述框架) 数据量的快速增长，利用 NoSQL 数据库存储和管理大规模 RDF 数据已经成为了当前的研究热点。介绍了 NoSQL 数据库的种类划分和各类型特点，阐述了 RDF 数据在各类 NoSQL 数据库中存储结构设计和并行查询算法的研究现状，分析比较了不同方法的优缺点。最后，讨论了利用 NoSQL 数据库管理 RDF 的优势，总结了现有研究的不足之处，并展望了未来的研究方向。

关键词： 资源描述框架；NoSQL 数据库；数据模型；存储结构设计；RDF 并行查询算法
中图分类号： TP311.13 **文献标志码：** A **文章编号：** 1001-3695(2015)05-1281-06
doi:10.3969/j.issn.1001-3695.2015.05.001

Overview of NoSQL databases for large scaled RDF data management

WANG Lin-bin[1,2], LI Jian-hui[2], SHEN Zhi-hong[2]
(1. *University of Chinese Academy of Sciences, Beijing* 100190, *China*; 2. *Science Data Center, Computer Network Information Center, Chinese Academy of Sciences, Beijing* 100190, *China*)

Abstract: With the development of semantic Web and rapid growth of RDF (resource description framework) data, it has become a research hotspot to make use of NoSQL databases to store and manage massive RDF data. This paper introduced characteristics of different types of NoSQL databases, elaborated storage organization of RDF data, parallel query algorithms in different types of NoSQL respectively and compared strength and weakness of different strategies. At last, this paper discussed advantages of managing RDF data in NoSQL database, summarized disadvantages of current researches and proposed the future directions.

Key words: resource description framework; NoSQL database; data model; storage organization; RDF parallel query algorithm

语义网的快速发展和互联网数据的急剧增长，对大规模关联数据的存储和处理提出了新的挑战。传统集中式的 RDF 存储系统已经越来越难以满足海量数据应用，人们开始将目光投向分布式数据库系统。目前，这方面的研究可以划分为两类[1]：a) 专门开发分布式 RDF 存储系统，如 4store[2]、Bigdata[3]、YARS[4] 等，这类系统采用传统的分布式计算架构并对 RDF 存储和查询进行了专门的优化；b) 采用基于云计算环境的、支持分布式计算的非传统数据库，即 NoSQL 数据库[5] 来存储和管理大规模 RDF 数据。

NoSQL 数据库种类繁多，逻辑模型和物理实现上千差万别，但是一个共同的特点是都去掉了 RDMBS 的关系型特征，结构简单，因此具备大数据量下优秀的读写能力。除了高性能外，NoSQL 灵活的数据模型、高可用的架构和能够部署在廉价的 PC 集群上的特点，都使得它被众多研究开发人员所青睐。基于上述优点，一些研究人员开始考虑采用基于 NoSQL 数据库的 RDF 存储方案，以期解决大规模 RDF 的存储和查询问题。尽管目前这方面研究还处于初始阶段，但是已经有一些研究成果，并有许多系统被开发出来。本文在充分调研和深入分析的基础上，对现有的基于 NoSQL 数据库的 RDF 数据存储和查询技术进行了分析和综述。

1 RDF 和 SPARQL

RDF 是由 W3C(World Wide Web Consortium，万维网联盟组织) 提出的一种数据模型，已经成为语义网领域存储关联数据的推荐标准。RDF 最初是为了描述及管理 Web 资源和使用这些资源的元数据信息，但现在 RDF 中的资源已经不再局限于 Web，而泛化为关于任何可识别事物的信息。总的来说，RDF 提供了一种用于描述信息、使得信息能够在应用程序间不失语义地交换的通用框架[6]。在这个框架下，数据被描述为主体(subject)、谓词(或属性 predicate) 和客体(或属性值 object) 三元组。一系列的三元组可以组成 RDF 图，RDF 图中的节点可以是资源描述符(URI reference)、文字(literal，字符串或其他数据类型) 或空节点。

SPARQL(simple protocol and RDF query language) 是 W3C 提出的一种 RDF 数据的标准查询语言。SPARQL 包含四种类型的查询：SELECT、CONSTRUCT、ASK 和 DESCRIBE。其中最常使用的是 SELECT 查询，它和 SQL 的语法类似，用来返回满

收稿日期：2014-08-19；修回日期：2014-10-09 基金项目：中国科学院计算机网络信息中心“一三五”规划重点培育方向专项基金资助项目(CNIC_PY_1405)；主任基金资助项目(CNIC_ZR_201304)

作者简介：王林彬(1990-)，女，山西大同人，硕士研究生，主要研究方向为科学数据的组织、集成和管理(wanglinbin@cnic.cn)；黎建辉(1973-)，男，研究员，博士，主要研究方向为云计算、大数据处理、大规模数据的组织和集成；沈志宏(1977-)，男，高级工程师，博士，主要研究方向为科学数据的组织、集成和管理.

足查询条件的数据。SPARQL 查询是通过图模式(graph pattern) 匹配实现的,简单而言就是在 RDF 图中找到匹配 SPARQL 查询图的子图位置。图模式中最常见的是基本图模式(basic graph pattern) ,它是三元组模式(triple pattern) 的集合。三元组模式和 RDF 三元组结构类似,只是主体、谓词和客体可以部分或者全部被查询变量代替。

2 NoSQL 数据库简介

NoSQL 意指 not only SQL,即对关系型数据库的补充。长久以来,关系型数据库就是存储数据的首选方案。随着互联网的发展,必须拥有更多的计算资源才能应对急剧增长的数据。计算资源的扩展包括纵向扩展(scale up) 和横向扩展(scale out) 两种方式。纵向扩展的一般方式是更换 CPU 性能更高、内存更大、功能更加强劲的计算机,显然这种方式的成本较高,且扩展能力有限; 另外一种方式则是利用小型机组成的集群来承担大规模的计算任务。由于关系型数据库并不适合构建在集群之上,研究者们开始考虑其他存储数据的方法,NoSQL 数据库应运而生。相对于 RDMBS,NoSQL 数据库具备一些共同特征,包括抛弃关系模型、不使用“模式”、不采用 SQL(部分数据库为了方便使用提供类 SQL 查询) 、项目开源、适合构建在集群上等。

数据库结构的基础是数据模型,依据数据模型的不同,可以将 NoSQL 数据库划分为以下几类[7,8]:

a) 键值存储。键值模型是 NoSQL 数据库中最基本的模型,即将数据按照键值对的形式进行组织、索引和存储。这类数据库具有极高的并发读写性能,但是通常只能通过键的完全一致来查询获取数据。常见的基于键值模型的数据库有 Redis[9] 、DynamoDB[10] 等。

b) 列式存储。区别于普通数据库中以行为单位存储数据的方式,在面向列的数据库中,数据存储是围绕“列”进行的。这类数据库还支持“列族”的特性,将多个需要经常一起访问的列组织在一起以提高查询效率。面向列的数据库具有高扩展性,数据量的增加不会降低数据库的处理速度(尤其是写入速度) ,因此非常适合大数据应用。最早的列式数据库是 Google 提出的 BigTable[11] ,Apache HBase[12] 被认为是 BigTable 的开源实现,另外一个普遍应用的列式数据库是 Cassandra[13] 。

c) 文档存储。文档数据库既具备键值存储的高性能、可伸缩性特点,又实现了关系型数据库中丰富的功能。文档数据库限制了存放内容,并定义了允许的结构和数据类型,因此能够通过复杂的查询灵活地访问数据。这类数据库主要用来管理半结构化数据,数据经常以 JSON 格式来组织存储。常用的文档型数据库包括 MongoDB[14] 、CouchDB[15] 等。

d) 图形存储。图数据库在 NoSQL 领域中独树一帜。不同于上述三类数据库,图数据库的流行并非由于它能够解决大数据背景下的集群扩展问题,而是因为它可以更加有效地存储和管理数据间的关联。图模型非常简单,包括若干节点和节点之间的边,但是它特别适合存储文件树这样的递归结构和社交图这样的网络结构数据。图数据库通常运行在单一的服务器上,如 Neo4j[16] 和 GraphDB[17] ,但也有一些分布式图数据库被开发出来,如 Titan[18] 。

在 NoSQL 数据存储模型中,列式存储和文档存储可以被理解为键值存储模型的一种扩展。文档模型可以认为是把键值中的值设定为结构化的文档; 而列式数据库中的存储模型可以这样理解: 将行 ID、列簇号以及时间戳组合在一起形成一个键,然后将值依照键顺序存储。在很多 RDF 相关研究[19~22]中,研究者将 HBase 和 Cassandra 这些列式数据库泛称为键值数据库。而最基本的键值存储模型则太过简单,难以进行 RDF 模型的转换。因此在本文中,键值存储和列式存储被归为同一类,并以 HBase 作为典型应用进行讨论。

3 基于 NoSQL 的 RDF 数据存储和查询

大部分集中式 RDF 存储系统,如 Jena SDB[23] 和 Sesame[24] ,都是采用 RDBMS 作为底层存储。由于 RDF 本身是一种图数据,而 RDBMS 的基础是关系代数理论,将 RDF 数据交于 RDBMS 管理存在着很多问题,包括存储效率、查询性能和推理支持等。另外,随着语义网中 RDF 数据的快速增长,RDBMS 的弱扩展性逐渐成为制约 RDF 数据管理系统的瓶颈。基于云计算环境的 NoSQL 数据库提供了一种行之有效的解决方案。

3.1 基于列式模型的 RDF 存储和查询

本文以 HBase 作为代表探讨列式模型下的 RDF 数据存储和查询技术。Apache HBase 使用 HDFS[25] 作为底层存储,并支持 MapReduce[26] 开源计算架构。HBase 中采用了与关系型数据库概念相近的数据表 HTable,行键是 HTable 中每一行的唯一标志符,HBase 对行键提供默认的类似 B + 树的高效索引。因此在 HBase 中存储数据,一定要充分利用行键索引的特点。

3.1.1 基于列式模型的存储设计

HTable 与关系型数据表概念相似,利用关系型数据库存储 RDF 数据已经有很多较为成熟的方案。因此在设计 RDF 在 HBase 中的存储模式时,可以借鉴关系型数据库中的存储结构。常见的主要包括三元组存储结构、属性表存储结构、垂直划分存储结构和多索引存储结构。

三元组存储结构是将 RDF 数据映射到数据表的一种最简单的方式。构建一张包含主体、谓词和客体的三列表,将所有的 RDF 三元组放在这个表中。这种方式简单通用,对简单查询的执行效率较高; 而进行复杂查询需要对信息进行多次的遍历和连接操作,效率较低。

Wilkinson 等人提出了属性表存储结构,Jena2 中把属性表作为该框架的一种标准存储组织方式[27,28] 。属性表包括两种不同的组织方式,一种是聚类属性表(clustered property table) ,一种是类别属性表(property-class table) 。属性表减少了查询中的主体—主体自连接操作,提高了查询效率。但是这种方法增加了数据存储的复杂性,并且对多值属性支持不好。

在垂直划分存储结构[29] 中,RDF 数据集中的每一个属性都对应一张数据表,每一个数据表包含两列,存储拥有该属性的三元组的主体和客体。这种数据组织方式的优点是结构简单,解决了多值属性存储和数据稀疏的问题。缺点是数据表过多,针对不同属性的查询时不得不进行大量表连接操作; 当碰到谓词不固定的查询时,需要遍历所有表。

Harth 和 Wood 等人[30,31] 提出了多索引存储结构,在此基础上又进一步产生了 RDF-3X[32] 和 Hexastore[33] 等代表性方案。多索引结构列举了三列表的所有排列组合的可能性,创建了六个索引,覆盖了全部的访问模式。多索引结构能够对所有的三元组快速返回结果,因此在简单查询下的效率很高,但是对于复杂查询还需要进一步的优化。

在设计基于 HBase 的 RDF 存储结构时,研究人员充分借

鉴了上述的几种存储方案。金强[34]采用了一种基于 Hexastore 的存储结构，为了减少六个索引表造成的数据冗余，他将索引表减少至三个，仅保留 SP_O、PO_S 和 OS_P。Khadilkar 等人[35]对 HBase 中 RDF 数据的存储模式进行了详细的研究对比。他们在属性表、垂直划分表和多索引结构等基础上设计了六种存储布局，包括：a) 简单布局，创建三张 HTable，以主体/谓词/客体为行键，其他两个成分组合为列名；b) 垂直切分布局，每个谓词对应两张 HTable，以包含该谓词的三元组主体/客体为行键，另一成分存于列名中；c) 全索引布局，创建六张 HTable 对应六个索引，所有成分组合构成行键；d) 垂直切分—索引布局，保留垂直切分中所有表，并添加 SPO、OSP 和 OS 三张索引表；e) 混合布局（简单—垂直切分），除垂直切分中所有表外，增加简单布局中以主体/客体为行键的两张表；f) 哈希布局，将混合布局中的行键、列名等进行哈希映射。通过对六种不同布局下的数据装载速度和查询效率的对比，实验证实了混合存储结构在各个方面都更胜一筹。

本文对比了不同存储布局的优缺点，如表 1 所示，并总结了在设计基于 HBase 的 RDF 数据存储模式时应遵循的一些基本规律。

表 1　基于 HBase 的 RDF 存储布局优缺点比较

布局方案	优点	缺点
简单布局	对(s,?,?)、(?,p,?)和(?,?,o)这三种三元组模式的匹配速度很快；数据重复存储三次，空间效率中等	另外几种三元组模式的匹配速度较慢，整体查询效率较低
垂直切分	对于谓词固定的三元组查询能够快速匹配；数据只重复存储两次，空间效率较高	对于谓词不固定的三元组，需要遍历所有的数据表，匹配速度慢
全索引	对全部三元组模式都能快速匹配	数据重复存储六次，空间效率很低
垂直切分—全索引	具备垂直切分中较高的空间效率，通过增加几个全索引结构来快速响应谓词不固定的三元组匹配	设计实现复杂
混合布局	具备垂直切分中较高的空间效率，通过增加简单布局中的表来快速响应谓词不固定的三元组匹配	设计实现复杂
哈希布局	将长字符串或者 URI 通过哈希函数映射为 8 Byte 整数，节省存储空间	实现复杂，并且需要额外增加映射表

a) 尽量将三元组成分存储在行键中。因为 HBase 中默认地对行键构建 B + 索引，行键的匹配比较要比列名的匹配快得多。另外，一般不在 HBase 的 cell 中（即 value）存储数据，因为 HBase 中的 value 匹配是在返回数据后才过滤实现。

b) 垂直切分能显著提高查询效率。在三元组的三个成分中，谓词的数量较少，区分度比较差。如果不切分谓词，容易导致以谓词为行键的表过宽，列名和 value 匹配耗时，查询效率会很低。

c) 好的存储模式往往是一些基础模式的组合，将垂直切分与其他方式混合，取长补短，可以获取最优的空间和时间性能。

3.1.2　基于列式模型的 RDF 数据分布式查询

基于云计算环境的数据管理系统不仅需要可扩展的数据存储能力，还要具备处理大数据的并行计算能力。本节中继续以 HBase 作为代表，讨论 RDF 数据分布式查询的方法。由于 HBase 支持 Hadoop MapReduce 并行计算框架，将 RDF 分布式查询方法划分为基于 HBase 原生的 Java API 和采用了 MapReduce 计算框架两大类。

HBase 本身不支持查询语言，但是它提供了 Java API 对其进行插入删除和查询等基本的数据操作。在 HBase 上执行 SPARQL 查询时，需要自己构建查询引擎，将复杂的 SPARQL 查询翻译成 HBase 支持的简单操作。本文根据对 Khadilkar 和 Guéret 等人[35,36]工作的总结，给出了一个基于 HBase Java API 的 RDF 并行查询系统架构图，如图 1 所示。

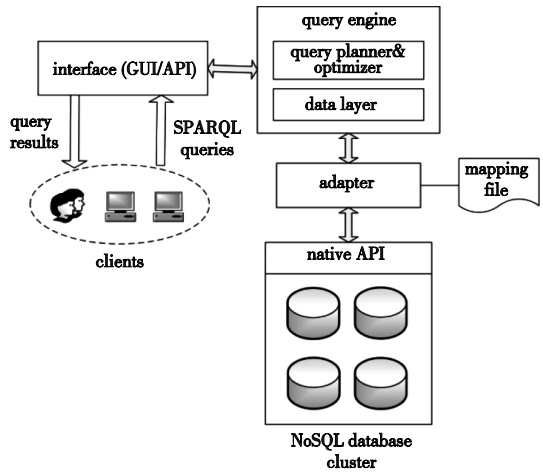

图1　RDF并行查询系统架构设计

该架构主要包含以下主要模块：

a) 数据库集群。系统中用于存储 RDF 数据的 HBase 集群。

b) 查询引擎。整个系统中最主要的部分，基本功能是将 SPARQL 查询中的图模式解析为嵌套的三元组模式，并根据一定的规则优化三元组模式的查询顺序。查询引擎还要负责将三元组模式的匹配结果合并处理，得到最终的 SPARQL 查询结果。进一步地，该模块可以根据功能划分为两个子模块：查询规划和优化器负责查询的解析、优化和结果合并；数据层是搭建在不同数据源之上的抽象层，提供透明地访问 RDF 数据的能力，使得查询引擎不必关心 RDF 数据在 HBase 中的具体存储结构。

c) 适配器。根据 RDF 数据在 HBase 中存储结构的不同，增加适配器来提供查询引擎中的操作原语到 HBase 基本操作的映射。mapping file 中存储了详细的映射机制，可以将其认为是一些配置文件，这种方式提高了适配器的灵活性。

d) 用户接口。提供向上的 Web 服务或者其他应用的接口。

上述模型是一个通用的设计，在不同的实现方案中，查询引擎的模块划分和具体功能又有很多的变化。Guéret 等人[36]采用了一个名为 eRDF 的引擎来执行分布式数据源上的 SPARQL 查询，通过采用不同的适配器，eRDF 能够透明地应用在不同的数据库上。在 TripleCloud[22]中，他们将 eRDF 分别搭建在 HBase 和 Google App Engine[37]上，验证了复杂 SPARQL 查询可以通过这种方式执行在大规模 RDF 数据集上。Khadilkar 等人[35]将 Jena 框架搭建在 HBase 集群上，既吸收了 HBase 作为存储层的模式自由、容错性强、扩展性好等优点，又充分利用了 Jena 内置的 RDF API、ontology API 和 ARQ 查询引擎的功能，系统的搭建难度小。为了使 Jena 支持 HBase 作为后台数据库，需要扩展 ARQ 引擎，具体方法是重写一个名为 Graph 的 SPI，这个接口可以封装不同的底层存储，使得 Jena 透明地操作非 RDF 数据库。作者比对了 Jena TDB 单机存储 Jena HBase 集群存储两种方案下的 SP2[38]和 LUBM[39]两种基准测

试结果，发现在小数据集下 Jena TDB 查询效率高于 Jena HBase 方案，但当数据量增大单机环境无法运行时，Jena HBase 仍然有效地能够执行标准测试中的查询，从而证明了利用 HBase 集群存储管理大规模 RDF 数据的可行性和有效性。另外一些研究关注在 HBase 背景下的查询算法具体实现，文献[40]提出了三个查询算法，分别实现了 triple pattern 与三元组的匹配、在 HBase 数据表中查询和 triple pattern 匹配的三元组、在 HBase 数据库中查询与 SPARQL BGP 匹配的三元组。文献[41]除了实现上述三个功能外，还实现部分推理功能和 RDF 图更新操作。

另一类 RDF 数据分布式查询方法利用了 MapReduce 框架。在 SPARQL 中，即使是一个简单的查询也会被翻译为多个三元组模式的连接操作，三元组模式之间是通过它们共有的未绑定变量而关联起来的。在利用 MapReduce 处理 BGP 查询时，最主要的问题是规划多个三元组模式连接操作执行的顺序，减少中间结果和任务迭代的数量。Myung 等人[42]提出了一种多路连接选择策略来处理 SPARQL BGP。他们提出了两种连接变量选择策略：贪心策略优先选择出现次数最多的查询变量；多变量策略在一次迭代过程中选择覆盖的三元组模式有交集的多个变量。这两种策略能够尽可能地减少 MR 任务迭代次数。在该研究中，RDF 数据以文件的形式存储在 HDFS 系统中，金强[34]在此基础上进行了改进，将数据存储在 HBase 中，充分利用 HBase 中的行键索引，可以提高 MR 中数据选择操作的效率。Papailiou 等人[43]在其开发的 H_2RDF + 系统中采用了全索引结构存储数据，采用了 Snappy[44]等压缩算法提高空间效率，实现了多路 merge 连接算法和 sort-merge 连接算法。他们引入了一个代价模型来选择连接操作的执行顺序和平台资源的分配策略。该系统在执行简单操作时性能接近目前最优的 RDF-3X，在执行大规模数据量下的复杂操作时要比 RDF-3X 快几个数量级。

3.2 基于文档模型的 RDF 存储和查询

3.2.1 基于文档模型的 RDF 存储设计

面向文档的数据库将关系数据库中行换成更加灵活的文档模型，这种方式更适合存储半结构化数据。MongoDB 和 CouchDB 等面向文档型数据库支持 JSON 格式的数据存储。JSON 是一种轻量级的数据交换格式，采用完全独立于编程语言的文本格式，易于人阅读和书写，也易于机器生成和解析。JSON 能够灵活方便地用于不同的系统之间的数据交换。

利用文档型数据库存储 RDF 数据的第一步工作就是实现 RDF 数据的 JSON 格式序列化。RDF 三元组中包含主体、谓词和客体三个基本成分，而 JSON 的对象结构仅包含 key 和 value，因此如何实现 RDF 到 JSON 的映射，并在此基础上实现高效的查询算法是一个技术难点。RDF 数据的 JSON 格式序列化方法主要有以下几种：

a) 扁平三元组方式。这种方式将一个 RDF 图中的所有三元组存储在一个命名为“triple”的根文档对象中，“triple”的值是一个包含多个对象的数组，数组中的每个对象存储了一个三元组的全部信息。

b) 主体中心方式。Alexander[45]提出了以主体为中心的序列化方法，RDF 数据中的每一个主体作为一个 JSON 对象的 key，对应的 value 是一些内嵌对象，这些内嵌对象的 key 是与主体相关的谓词，value 是与之对应的客体数组。

c) JSON-LD 方式。这种方式是 W3C RDF 工作组于 2014 年列入推荐状态的一种关联数据存储标准，旨在向 JSON 中加入关联数据(linked-data)语义。不同于前边两种被动地将 RDF 数据进行 JSON 序列化的方式，JSON-LD 意图作为关联数据的一种基本存储方式，与 RDF/XML、Turtle 等格式并存，并在某些方面表现出更出众的特点。JSON-LD 主要设计目的是提供一种在 Web 编程环境下使用关联数据的方式，能够与现有的 Web 服务交互，并支持关联数据在 MongoDB、CouchDB 等基于 JSON 格式的文档型数据库中的存储[46]。

```
{
    "@context": "http://json-ld.org/context/person.jsonld",
    "name": "Manu Sporny",
    "homepage": "http://manu.sporny.org/",
    "image": "http://manu.sporny.org/images/manu.png"
}
}
```

上面的编码是一个非常简单的 JSON-LD 格式的信息描述，这个文档表示一个人，很容易推断出这里“name”表示人的名字，“homepage”是其主页，“image”是相关照片。但是计算机不理解“name”和“image”这样的术语的语义。在语义网中通过使用国际化资源标志符(international resource identifiers，IRIs)来提供一种标准的 Web 资源定义，使得机器可以知道这些字符背后的语义。为了兼顾到书写的简洁性，JSON-LD 中引入上下文(context)用于短语和 IRIs 的映射。

JSON-LD 是一种混合语法，一个 JSON-LD 文档可以代表一个 RDF 数据模型实例。通过对 RDF 数据模型的扩展，JSON-LD 可以序列化一般的 RDF 数据集。JSON-LD 处理算法和 API 规范中提供了 RDF 序列化为 JSON-LD 和反序列化的算法工具。

本文对比了三种序列化方式的优缺点，如表 2 所示，由于 JSON-LD 已经成为一种推荐标准，具有比较全面的技术支持，本文认为这种格式更具优势。

表 2 RDF 的 JSON 序列化方式优缺点比较

序列化方式	优点	缺点
扁平三元组方式	易于理解、容易实现	没有 RDF 模型很难获取数据，有时需要遍历根文档对象才能取出所需的属性
主体中心方式	格式接近 RDF/XML 形式，高效地支持以主体为中心的星型查询	相对于扁平三元组方式处理较为复杂
JSON-LD 方式	W3C 推荐标准，语义规范完整，有 RDF 序列化和反序列化算法工具	语义细节较多，实现较为复杂

3.2.2 基于文档模型的 RDF 数据分布式查询

本节以 MongoDB 为代表，讨论文档模型下的 RDF 查询技术。MongoDB 不支持 SPARQL 查询，它有自己的查询语言。MongoDB 的查询语言是一种面向对象的命令式语言，而 SPARQL 是一种面向领域的描述性语言，因此将 SPARQL 转换为 MongoDB 查询语言是很困难的。

Tomaszuk[47]采用上文提到的扁平化方式将每一个 RDF 三元组序列化成一个单独的 JSON 文档，他提出了三个关于 SPARQL 查询到 MongoDB 查询的转换算法，分别实现了 SPARQL 基本查询、DISTINCT 查询和 UNION 查询到 MongoDB 查询的映射。另外，作者对比了 MongoDB 和 Seasame、4store、Jena SDB 等三元组数据库的 RDF 装载和查询速度。实验证明，MongoDB 和 4Store 是数据装载最快的两个数据库；在基于 Ber-

lin SPARQL benchmark [48] query 6 的查询性能测试中，MongoDB 查询速度要比其他数据库低。本文对 MongoDB 作为三元组数据库的可行性进行了验证，但是由于实验采用的数据量较小（1000313 个三元组）且没有采用集群部署，并未体现出 MongoDB 的可扩展性优势和应对大数据能力。

Mutharaju 等人 [49] 提出的 D-SPARQ 分布式 RDF 查询引擎将 MapReduce 并行处理框架和 MongoDB 结合起来。系统的主节点主要包含图切分、三元组放置和查询协调三个模块，在数据节点中存放数据并执行 MongoDB 查询。本研究借鉴了 Lee 等人 [50] 的研究，从语义的角度切分数据将主体相同的三元组组织在一起，这样的数据组织方式对星型查询响应很快；另外适当复制了跨越不同分区的三元组，提高了复杂查询的查询效率。实验证明对 SP2 基准测试中的某些查询语句，该系统的查询速度要快于当前性能最佳的集中式三元组数据库 RDF-3X。但是本文没有提及 SPARQL 查询转换成 MongoDB 查询的具体实现算法，MapReduce 框架在本系统中的应用也鲜有提及。

3.3　基于图模型的 RDF 存储和查询

在数据模型的发展过程中，关系数据库中采用的关系模型和语义网中采用的图模型是两个主要流派。Codd [51] 提出的关系型模型在后来的发展中逐渐成为了数据库研究的主流，特别是它的标准查询语言 SQL 已经成为了查询语言的一种范式。但是由于关系模型的实现特点，一些具有网络结构的数据在执行查询时会进行多次表连接操作，这种操作非常缓慢，而且可伸缩性较差，而语义网使用的图模型则能够有效地解决上述问题。一个图结构主要包含节点（顶点）、关系（边）和属性等要素，在图数据库中可以应用许多基于图论的算法。尽管 RDF 三元组可以存储在关系型数据库中，但是从根本上说，RDF 模型可以看成是图模型的一种特例，因此采用图形数据库将会更加合适。

Neo4j 是当下流行的一种开源图数据库，它重点解决了传统关系型数据库在查询时需要进行大量表连接而导致的性能衰退。Neo4j 能够以相同的速度遍历节点和边，其遍历速度与构成图的数据量没有任何关系。Neo4j 可以支持大规模的扩展性，单台机器能够支持 10 亿级别的节点或边的大图，另外它的商业版还支持多台机器的并行运行。Thinkerpop 团队开发了 LinkedDataSail [52]，它提供了在图数据库中处理 RDF 数据的接口。利用这个接口，Neo4j 能够支持 SPARQL 查询，可以作为一个三元组数据库来使用。

现有的利用 Neo4j 存储 RDF 并支持 SPARQL 查询的研究并不多见，仅限于一些工程应用上的尝试。Martella 将 DBpedia 数据集存入 Neo4j [53]，在此基础上展开 SPARQL 查询和其他图算法。笔者注意到，他们选择将 RDF 数据存入 Neo4j 而不是普通三元组数据库的主要原因是图数据库对一些基于图的查询支持更好。Angles 等人 [54] 在其研究中也表明大部分 RDF 查询语言对于一些基本的图查询的支持较低，甚至 SPARQL 也不支持图结构中路径查询和节点距离查询，而这些查询在实际应用中却很重要。比如警用数据库中经常需要查询两个嫌疑人之间是否有关联（路径查询），文献数据库中要查找被引用次数最多的文献（求最大度节点）等。因此，研究者们呼吁在 RDF 查询语言中提高对图算法的支持。

4　结束语

基于以上研究分析，本文总结出利用 NoSQL 数据库存储管理 RDF 数据集有以下优势：a）NoSQL 数据库的高扩展性、可伸缩性以及可部署在低端 PC 机集群的特点能够支持 RDF 大数据应用；b）NoSQL 数据库灵活的数据结构和自由的存储模式非常适合管理 RDF 这种半结构化数据；c）大部分 NoSQL 数据库支持 MapReduce、Storm [55] 等并行计算框架，可以用来进行 RDF 并行查询和推理；显然 NoSQL 中的图数据库更适合存储 RDF 图数据。

但是可以看到，目前基于 NoSQL 的 RDF 存储和查询研究还处于初级阶段，有许多问题需要进行更加深入的探讨，未来的研究方向主要包括以下几点：

a）更加高效的并行化查询方法。在数据规模可处理情况下，基于 NoSQL 的 SPARQL 查询效率远低于本地三元组数据库。这是由于本地数据库对 RDF 数据索引和查询等作了很多方面的优化，而基于 NoSQL 的并行查询算法还处于研究的初始阶段，算法效率较低。未来可以从以下几个方面来提高：（a）研究更加合理的分布式数据布局和索引方案，提高三元组模式匹配的速度；（b）进一步优化算法中连接操作的执行策略，减少中间数据量和 MapReduce 的迭代次数。

b）更加全面的语义网技术支持。现有的多数研究中，NoSQL 数据库仅支持 SPARQL 四类查询格式中的 SELECT 格式，并不支持 COSNTRUCT、ASK 和 DESCRIBE 类型的查询。另外，语义网的核心功能之一的逻辑推理还没有在当前的研究中涉及到，未来应提供 OWL 接口和全面的 SPARQL 功能。

c）更具适应性的查询语言。JSON-LD 作为一种新型的关联数据格式标准，以其简单不复杂和面向一般开发人员的特点正逐渐被接受。JSON-LD 在实现时没有选择包括 SPARQL 查询的大部分 Web 语义技术栈，这就使得当在文档型数据库中存储这种格式的数据时，不得不暂时采用将 SPARQL 查询转换成 MongoDB 查询的方法。显然，要想高效地管理这类数据，需要提出一种适合 JSON 格式的关联数据查询语言。

参考文献：

[1] 何少鹏，黎建辉，沈志宏，等. 大规模的 RDF 数据存储技术综述[J]. 网络新媒体技术，2013，2(1)：8-16.

[2] HARRIS S，LAMB N，SHADBOLT N. 4store：the design and implementation of a clustered RDF store[C]//Proc of the 5th International Workshop on Scalable Semantic Web Knowledge Base Systems. 2009：94-109.

[3] SYSTAP. Bigdata RDF database[EB/OL]. http://www.systap.com.

[4] HARTH A，UMBRICH J，HOGAN A，*et al*. YARS2：a federated repository for querying graph structured data from the Web[C]//Proc of the 6th International Conference on Semantic Web. Berlin：Springer，2007：211-224.

[5] STRAUCH C，SITES U L S，KRIHA W. NoSQL databases[EB/OL]. [2012-07-11]. http://www.christof-strauch.de/nosqldbs.pdf.

[6] KLYNE G，CARROLL J J. Resource description framework（RDF）：concepts and abstract syntax[EB/OL].（2004-02-10）[2009-03-20]. http://www.w3.org/TR/rdf-concepts-20040210/.

[7] POPESCU A. NoSQL at CodeMash：an interesting NoSQL categorization[EB/OL]. 2010. http://nosql.mypopescu.com/post/396337069/presentation-nosql-codemash-an-interesting.

[8] SCOFIELD B. NoSQL：death to relational databases[EB/OL].（2010-04-14）. http://www.softdevtube.com/2010/04/14/nosql-death-to-relational-databases.

[9] Redis[EB/OL]. [2011-03-19]. http://redis.io.

[10] VOGELS W. Amazon DynamoDB：a fast and scalable NoSQL database service designed for Internet scale applications[EB/OL].（2012-01-

18). http://www.allthingsdistributed.com/2012/01/amazon-dyna-modb.html.
[11] CHANG F, DEAN J, GHEMAWAT S, *et al*. Bigtable: a distributed storage system for structured data [J]. ACM Trans on Computer Systems, 2008, 26(2): 4.
[12] Apache HBase [EB/OL]. https://hbase.apache.org/.
[13] Apache Cassandra [EB/OL]. http://cassandra.apache.org/.
[14] MongoDB [EB/OL]. http://www.mongodb.org/.
[15] Apache CouchDB [EB/OL]. http://couchdb.apache.org/.
[16] Neo4J [EB/OL]. http://www.neo4j.org.cn.
[17] GÜTING R H. GraphDB: modeling and querying graphs in databases [C]//Proc of the 20th International Conference on VLDB. 1994: 12-15.
[18] Titan [EB/OL]. http://thinkaurelius.github.io/titan/.
[19] LADWIG G, HARTH A. An RDF storage scheme on key-value stores for linked data publishing [R]. [S.l.]: Karlsruhe Institute of Technology, 2010.
[20] LADWIG G, HARTH A. CumulusRDF: linked data management on nested key-value stores [C]//Proc of the 7th International Workshop on Scalable Semantic Web Knowledge Base Systems. 2011: 30-42.
[21] CHOKSUCHAT C, CHANTRAPORNCHAI C. Experimental framework for searching large RDF on GPUs based on key-value storage [C]//Proc of the 10th International Joint Conference on Computer Science and Software Engineering. 2013: 171-176.
[22] GUERET C, KOTOULAS S, GROTH P. TripleCloud: an infrastructure for exploratory querying over Web-scale RDF data [C]//Proc of IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. Washington DC: IEEE Computer Society, 2011: 245-248.
[23] CARROLL J J, DICKINSON I, DOLLIN C, *et al*. Jena: implementing the semantic Web recommendations [C]//Proc of the 13th International World Wide Web Conference on Alternate Track Papers & Posters. New York: ACM Press, 2004: 74-83.
[24] Sesame [EB/OL]. http://rdf4j.org.
[25] SHVACHKO K, KUANG H, RADIA S, *et al*. The Hadoop distributed file system [C]//Proc of the 26th IEEE Symposium on Mass Storage Systems and Technologies. [S.l.]: IEEE Press, 2010: 1-10.
[26] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters [J]. Communications of the ACM, 2008, 51(1): 107-113.
[27] WILKINSON K, SAYERS C, KUNO H A, *et al*. Efficient RDF storage and retrieval in Jena2 [C]//Proc of International Semantic Web Conference. 2003: 131-150.
[28] WILKINSON K, WILKINSON K. Jena property table implementation [C]//Proc of the 2nd International Workshop on Scalable Semantic Web Knowledge Base. 2006.
[29] ABADI D J, MARCUS A, MADDEN S R, *et al*. Scalable semantic Web data management using vertical partitioning [C]//Proc of the 33rd International Conference on Very Large Data Bases. 2007: 411-422.
[30] HARTH A, DECKER S. Optimized index structures for querying RDF from the Web [C]//Proc of the 3rd Latin American Web Congress. 2005: 71-80.
[31] WOOD D, GEARON P, ADAMS T. Kowari: a platform for semantic Web storage and analysis [C]//Proc of XTech Conference. 2005: 05-0402.
[32] NEUMANN T, WEIKUM G. RDF-3X: a RISC-style engine for RDF [J]. Proceedings of the VLDB Endowment, 2008, 1(1): 647-659.
[33] WEISS C, KARRAS P, BERNSTEIN A. Hexastore: sextuple indexing for semantic Web data management [J]. Proceedings of the VLDB Endowment, 2008, 1(1): 1008-1019.
[34] 金强. 基于 HBase 的 RDF 存储系统的研究与设计 [D]. 杭州: 浙江大学, 2011.
[35] KHADILKAR V, KANTARCIOGLU M, THURAISINGHAM B M, *et al*. Jena-HBase: a distributed, scalable and effcient RDF triple store [C]//Proc of International Semantic Web Conference. 2012.
[36] GUÉRET C, GROTH P, OREN E, *et al*. eRDF: a scalable framework for querying the Web of data [EB/OL]. (2009-10-28). http://www.few.vu.nl/~pgroth/btc-2009submit.pdf.
[37] ZAHARIEV A. Google app engine [D]. Helsinki: Helsinki University of Technology, 2009.
[38] SCHMIDT M, HORNUNG T, MICHAEL M, *et al*. SP^{2}Bench: a SPARQL performance benchmark [C]//Proc of the 25th IEEE International Conference on Data Engineering. Berlin: Springer, 2010: 371-393.
[39] GUO Yuan-bo, PAN Zheng-xinag, HEFLIN J. LUBM: a benchmark for OWL knowledge base systems [J]. Web Semantics: Science, Services and Agents on World Wide Web, 2005, 3(2): 158-182.
[40] ABRAHAM J, BRAZIER P, CHEBOTKO A, *et al*. Distributed storage and querying techniques for a semantic Web of scientific workflow provenance [C]//Proc of IEEE International Conference on Services Computing. 2010: 178-185.
[41] 朱敏. 基于 HBase 的 RDF 数据存储与查询研究 [D]. 南京: 南京大学, 2013.
[42] MYUNG J, YEON J, LEE S. SPARQL basic graph pattern processing with iterative MapReduce [C]//Proc of Workshop on Massive Data Analytics on the Cloud. New York: ACM Press, 2010: 1-6.
[43] PAPAILIOU N, TSOUMAKOS D, KONSTANTINOU I, *et al*. H_2RDF+: an efficient data management system for big RDF graphs [C]//Proc of SIGMOD International Conference on Management of Data. 2014.
[44] Snappy [EB/OL]. https://code.google.com/p/snappy/.
[45] ALEXANDER K. RDF in JSON: a specification for serialising RDF in JSON [C]//Proc of Workshop on Scripting for Semantic Web. 2008.
[46] W3C. JSON-LD [EB/OL]. http://www.w3.org/TR/json-ld/.
[47] TOMASZUK D. Document-oriented triplestore based on RDF/JSON [J]. Studies in Logic, Grammar and Rhetoric, 2010, 22(35): 125-140.
[48] BIZER C, SCHULTZ A. The Berlin SPARQL benchmark [J]. International Journal on Semantic Web and Information Systems, 2009, 5(2): 1-24.
[49] MUTHARAJU R, SAKR S, SALA A, *et al*. D-SPARQ: distributed, scalable and efficient RDF query engine [C]//Proc of the 12th International Semantic Web Conference and the 1st Australasian Semantic Web Conference. 2013.
[50] LEE K, LIU Lin. Scaling queries over big RDF graphs with semantic hash partitioning [J]. Proceedings of the VLDB Endowment, 2013, 6(14): 1894-1905.
[51] CODD E F. A relational model of data for large shared data banks [J]. Communications of the ACM, 1970, 13(6): 377-387.
[52] LinkedData-Sail [EB/OL]. (2013-06-30). https://github.com/tinkerpop/gremlin/wiki/linkeddata-sail.
[53] Dbpedia4neo [EB/OL]. https://github.com/claudiomartella/dbpedia4neo.
[54] ANGLES R, GUTIERREZ C. Querying RDF data from a graph database perspective [M]//The Semantic Web: Research and Applications. Berlin: Springer, 2005: 346-360.
[55] Storm [EB/OL]. http://storm.incubator.apache.org/.