一种基于 HBase 的语义数据存储模型

翟社平 高 山 郭 琳 李兆兆
(西安邮电大学计算机学院 陕西 西安 710121)

摘 要 随着语义 Web 的快速发展与应用,如何高效地组织管理大规模语义数据成为语义 Web 研究中的一个关键问题。为了高效地执行各种查询操作,在充分利用语义数据查询特征的基础上,提出一种基于分布式数据库 HBase 的资源描述框架 RDF(resource description framework)数据存储模型。该模型将以谓语为索引的 RDF 数据存放在 P_OS 和 P_SO 两类索引表中,同时给出改进后的查询索引策略。在数据加载方面,通过自定义 Bulk Loading 的方式快速将数据上传至 HBase 存储表中。理论分析和实验结果表明,提出的存储模型能够快速响应基于谓语约束的查询服务;另一方面,Bulk loading 并行数据加载方法具有较高的加速比,能够显著缩短数据加载时间。

关键词 HBase 资源描述框架 语义网 语义数据存储

中图分类号 TP391.9 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2018.03.003

A SEMANTIC DATA STORAGE MODEL BASED ON HBASE

Zhai Sheping Gao Shan Guo Lin Li Zhaozhao
(*School of Computer Science, Xi'an University of Posts and Telecommunications, Xi'an 710121, Shaanxi, China*)

Abstract With the rapid development and application of Semantic Web, how to organize and manage large-scale semantic data becomes a key problem in Semantic Web research. In order to perform various query operations efficiently, an RDF data storage model based on distributed database HBase was proposed based on the semantic data query feature. The RDF data with predicate index were stored in P_OS and P_SO index tables, and the improved query indexing strategy was given. In terms of data loading, the data was uploaded to the HBase storage table quickly by customizing Bulk Loading. The theoretical analysis and experimental results showed that the storage model proposed in this paper could respond quickly to the query service based on predicate constraints. On the other hand, the loading method of Bulk loading parallel data had a higher speedup, which could significantly shorten the data load time.

Keywords HBase RDF Semantic web Semantic data storage

0 引 言

语义 Web 核心思想[1]是通过语义信息的分享,实现网络信息服务的智能化、自动化,从而促使互联网成为信息交换的通用媒介。

随着人工智能的发展和计算机应用需求的不断增强,语义 Web 技术已经深入到社会生活的各个领域,如舆情监控、知识图谱、穿戴计算、智能家居、智慧校园等。语义 Web 是万维网的一个扩展,其影响范围和应用范围在不断地扩大,越来越多的研究者开始高度关注和重视到语义 Web 技术的应用。

为了对 Web 资源及其属性进行规范化描述,W3C 组织提出了一个 Web 资源之间语义关系的开放元数据框架 RDF[2]。RDF 是语义 Web 建立的重要基础,通常以三元组(RDF Triple)数据模型(主语、谓语、宾语)描述 Web 资源之间存在的特定关系。简单而言,一个 RDF 语句是由资源、属性类型、属性值构成的三元组。基于语义 Web 的 RDF 三元组标签可以构造信息间逻辑推理关系的本体模型,进而使人与机器间、机器与机

收稿日期:2017-05-23。陕西省自然科学基金项目(2012JM8044);陕西省社会科学基金项目(2016N008);陕西省教育厅科学研究计划项目(12JK0733);西安市社会科学规划基金项目(17X63);西安邮电大学研究生创新基金项目(114-602080106)。**翟社平**,副教授,主研领域:语义 Web,智能 Agent。**高山**,硕士生。**郭琳**,硕士生。**李兆兆**,硕士生。

器间的交流变得像人与人之间交流一样轻松。

随着语义 Web 的迅猛发展,越来越多的互联网数据以 RDF 的形式发布出来。自 2007 年第一代关联数据发布至今,RDF 数据增长速度不断加快。据关联开放数据 LOD(Link Open Data)统计,2012 年关联数据云中已经包括 295 个数据集、316 亿个 RDF 三元组以及 5.04 亿个 RDF 链接。截止 2017 年 1 月,关联数据云中的数据集已经达到 1 146 个[3],约是 2012 年的 4 倍。如此大规模的 RDF 数据就为语义 Web 技术的发展提出了新的挑战。

现在的互联网正处在语义化蜕变过程中,智能化语义 Web 在理解人类语言的基础上逐渐实现了人机交互的网络访问。如何高效地管理大规模 RDF 数据已经成为推动语义 Web 技术发展亟待解决的重要问题。目前,RDF 数据存储需要解决以下两个关键问题:

1)如何建表才能在 RDF 数据存储空间开销和查询性能之间找到平衡点;

2)如何建立简单、有效索引才能将 RDF 数据的查询化繁为简。

针对传统的 RDF 数据存储在大数据集下表现不尽人意的现状,本文从用户行为习惯的角度出发,统计发现 90% 以上的查询都包含被约束的谓语值。通过分析现有的存储模式,充分结合语义数据查询特征,本文提出了一种基于分布式数据库 HBase 的 RDF 数据存储模型。将 RDF 数据存储到 HBase 表中以实现海量语义数据的分布式存储,并通过自定义的 Bulk Loading 数据加载方式完成数据上传,进而解决数据初始导入效率低的问题,同时实现 RDF 数据的高效查询。

1 相关研究工作

传统集中式的 RDF 数据管理大都采用关系型数据库管理系统 RDBMS(Relational Database Management System)来存储 RDF 数据。RDBMS 是一个二维映射的集中式存储数据库系统,表中每一列需预定义数据类型和长度,且表中的列数存在限制。目前基于关系型数据库的语义数据存储方案有三元组表存储、垂直存储、水平存储和模式生成存储等。

三元组表存储是最为简洁的一种 RDF 数据存储方式,其核心思想是每条三元组对应一条数据记录。垂直存储的核心思想是将同一谓语的三元组存到同一张表中,由于数据表本身已经将 RDF 的谓语属性值隐藏,因此数据表只保存主语和宾语两列数据值。水平存储的核心思想是将所有谓语作为列名存储在一张表中,通过一张表即可存储所有 RDF 数据。模式生成存储的核心思想是根据空值分布对表做垂直和水平切分,混合切分后的数据表会在很大程度上减少表中存储的空值数量。

表 1 从优势、劣势两个方面对以上四种存储方案进行简单总结。

表 1 基于关系型数据库的存储方案总结

存储方案	优势	劣势
三元组表[4]	结构简单、实现难度低	自连接过多、数据表过大、查询处理效率低
垂直存储[5]	降低存储开销,避免空值和多属性值问题	数据表过多、不支持属性未知的查询
水平存储[6]	避免多表连接或自连接问题	存储表稀疏,浪费存储空间
模式生成[7]	降低存储开销和查询连接计算开销	人工干预过多,不适合管理海量数据

面对 RDF 数据的急剧增长,传统关系型数据库已无法高效地对其进行管理,越来越多的研究者使用分布式系统和并行计算技术来管理大规模 RDF 数据。这类系统采用传统的分布式计算架构,并对 RDF 数据存储和查询进行优化。

其中,研究者曾提出一个分布式 RDF 数据查询框架 SPARQL ENGINE[8],该系统架构实现了大数据环境下 RDF 数据的分布式查询机制。但是,该框架并没有对 RDF 数据在分布式数据库中的存储策略进行研究。Bigdata 是另一种分布式存储系统,在存储上,其采用分区存储,利用 key-range 划分的 B + 树索引来分布式访问集群资源,而且这种划分是动态进行的[9]。但是该系统也存在着通信开销大、数据存取结构和安全性难以控制的缺点。

除此之外,在 RDF 数据存储领域已经有研究人员将 RDF 与 Hadoop 相结合,目前这方面的研究工作主要集中在 Hadoop 的存储设计以及基于存储模式的数据查询处理上。Hadoop 是 Apache 公司效仿 Google 云设计的开源平台,其具有可扩展、经济、高效、可靠等诸多优点,适合于对海量数据进行存储、搜索、挖掘和分析。研究学者提出了一种基于 Hadoop 的 RDF 数据编

码模型,详细介绍了词典构建的方法,通过构建好的数据词典实现 RDF 三元组的编解码操作,该方法在减少存储容量占用的同时保证了数据的安全性[10]。但是,该存储模型仍然采用 S_PO、P_OS、O_SP、PS_O、SO_P 和 PO_S 六张索引表进行存储,在数据存储设计方面并没有给出存储模式的优化方案。

2　基于 HBase 的 RDF 数据存储

2.1　存储分析

HBase 是一个高可靠性、高性能、列存储、可伸缩、支持实时读写的数据库系统,底层的 Hadoop 分布式文件系统 HDFS(Hadoop Distributed File System)具有高容错性且可以被部署在低价的硬件设备之上[11]。HBase 是一个多维映射的数据存储表,其数据存储模型为 Key-Value 类型:

(Row-Key,Column Family,Timestamp) - >Value

Row-Key 为表中唯一的行标识,动态扩展的多个列合并为一个列族(Column Family),时间戳用来区分所存数据的不同版本,进而唯一确定数据单元值 Value。

表 2 描述了百度网站分频道(百科、文库、贴吧等)关系数据在 HBase 表中的 S_PO 存储结构。该表以主语 www. baidu. com 为 Row-key,谓语 Anchor:channel 对应三个宾语值,Timestamp 表示时间戳。

表 2　表 S_PO 在 HBase 中的存储结构

Row-Key	Column + Cell	
	Anchor:channel	Timestamp
www. baidu. com	baike. baidu. com	1494950750382
	wenku. baidu. com	1494950750382
	tieba. baidu. com	1494950750382
	⋮	⋮

基于 HBase 的数据存储方案不对空值做存放操作,逻辑上为空的单元值并不占用实际的存储空间,所以 HBase 的稀疏特性非常适合存储 RDF 数据。

将 RDF 三元组 < 主语(Subject)、谓语(Predicate)、宾语(Object) > 中不同的元素和元素组合作为 Row-Key 索引,需要设计 S_PO、P_OS、O_SP、PS_O、SO_P和 PO_S 六张 HBase 索引表来对数据进行存储。实际上,S_PO、P_OS、O_SP 三种数据属性排列方式足以能够为所有的三元组查询模式提供索引服务。S_PO、P_OS、O_SP 分别表示以 S、P、O 为 Row-Key,其查询组合对应的索引如表 3 所示。

表 3　查询组合索引表

S	P	O	Index(索引)	说明
-	-	-	S_PO、P_OS、O_SP	S、P、O 均已知,查询可遍历任意索引表
?	-	-	P_OS	P_OS 快速响应(?PO)三元组
-	?	-	O_SP	O_SP 快速响应(S?O)三元组
-	-	?	S_PO	S_PO 快速响应(SP?)三元组
?	?	-	O_SP	P_OS 快速响应(?PO)三元组
-	?	?	S_PO	S_PO 快速响应(S?)三元组
?	-	?	P_OS	P_OS 快速响应(?P?)三元组
?	?	?	S_PO、P_OS、O_SP	查询可遍历任意索引表,结果返回所有三元组

注:"?"表示在三元组的位置存在一个未知值,"- -"表示该值被约束(即,固定)

2.2　存储改进策略

RDF 数据存储模型的目标是高效地执行各种查询操作,现有的 RDF 存储系统为了实现这一目标大都从数据源本身出发,对如何合理地拆分三元组数据集进行研究。但是,从数据查询的角度来讲,已有的存储方式忽略了用户查询习惯本身所带来的应用价值。

本文通过对里海大学基准 LUBM(Lehigh University Benchmark)测试基准中的查询记录进行统计分析,发现 SP?、?PO、?P? 三种约束谓语的查询模式最符合日常查询习惯,其查询请求数量约占查询总数的 90%。由此可见,谓语在大多数情况下是作为约束对象出现在查询语句中。但是在实际的 RDF 数据集中,谓语的数量通常在数十至数百范围内,远远少于主语和宾语数量,因此当谓语作为行键索引(Row-Key)时会使系统负载增大,查询效率降低。

为了适应谓语在查询中出现的高频特性,本文在 S_PO、O_SP 索引表不做处理的基础上,对 P_OS 索引表给出改进策略。将基于谓语索引的存储表细分为 P_OS、P_SO 两大类,实现谓语约束下主语或宾语查询的两种不同组合。具体实现形式表示如下:

```
P_OS:
Row Key:Predicate URI{
Column Family:Object{
        Column:(Subject)
    }
}
P_SO:
Row Key:Predicate URI{
Column Family: Subject {
        Column:(Object)
    }
```

}

2.3 模型设计

本文设计如图 1 所示的 RDF 三元组数据存储模型,该模型在 S_PO 和 O_SP 索引表不做修改的基础上对 P_OS 索引表进行改进。每条谓语分别对应 P_OS、P_SO 两类索引表,将每一谓语值作为热数据进行索引存储。完成以谓语为索引的 RDF 数据存储模型后,谓语确定的 RDF 数据查询会变得十分高效。由于谓语在整个 RDF 数据集中数量级极小,所以以谓语为索引的 RDF 数据存储表可以快速锁定数据,再加上原有 S_PO 和 O_SP 的 RDF 数据存储模型,整个存储系统就会变得很完善。从用户查询的角度讲,也能满足绝大多数数据的高效查询。

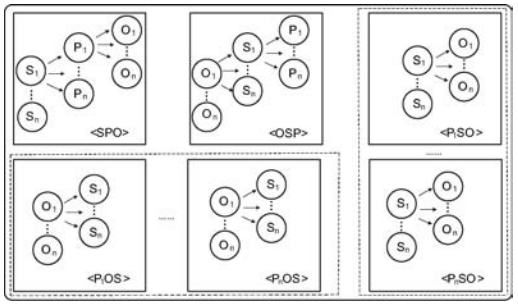

图 1 RDF 三元组存储模型

以谓语为索引的 RDF 数据存储模型旨在优化以谓语为约束条件的 RDF 数据查询。对照表 3 所示的查询组合索引,该存储模型针对不同查询索引做出改进,具体方案如下:

1) 为每种查询组合指定唯一确定的索引表;

2) 针对 SP? 高频查询组合,将优化前的 S_PO 索引表替换为 P_SO 索引表。

该存储模型充分考虑 RDF 三元组合的谓语特征,在实现系统负载均衡的同时能够提升数据的查询效率。表 4 介绍了改进后的 RDF 数据查询索引情况。

表 4 查询索引表(改进后)

S	P	O	Index(索引)
- -	- -	- -	P_SO
?	- -	- -	P_OS
- -	?	- -	O_SP
- -	- -	?	P_SO
?	?	- -	O_SP
- -	?	?	S_PO
?	- -	?	P_OS
?	?	?	S_PO

2.4 查找索引算法

查找索引算法的目标是找出与查询请求相对应的索引类型。结合表 4 的查询组合索引,给出该存储模型的查询索引算法,如算法 1 所示。

算法 1 查询索引

```
输入:查询语句
输出:索引类型
    开始
    //若谓语被约束
    if( predicate is constrained)
    {
    //且主语被约束
        if (subject is constrained)
            return P_SO
        else
            return P_OS
    }
    else
    {
    //宾语被约束
        if (object is constrained)
            return O_SP
        else
            return S_PO
    }
    结束
```

该算法给出了查询索引的详细过程。根据查询语句做出判断,若谓语被约束,可以通过判断主语是否被约束来确定相对应的索引类型;反之,可以通过判断宾语是否被约束来确定相对应的索引类型。该算法的核心思想是根据用户查询请求模型,优先匹配被约束的元素,进而快速决定使用何种对应索引。

2.5 Bulk Loading 数据加载

完成数据存储模型的设计后,接下来就要实现将 RDF 数据按照 S_PO,P_SO,P_OS,O_SP 模型加载到 HBase 中。HBase 数据加载有两种通用方法:一是通过 MapReduce 调用 TableOutpu-tFormat;二是在 client 上调用 API 写入数据。但是这两种方式都需要频繁地通过远程过程调用协议 RPC(Remote Procedure Call Protocol)方式与 HRegion 服务器进行通信,当一次性写入大量数据时会造成额外的资源开销。针对海量语义数据的存储,这两种方式显然不是最有效的。

HBase 可通过自带的用户自定义程序向 HBase 中加载数据。针对改进后的数据存储模型,本文在 HBase 自带数据加载方法的基础上,自定义了 Bulk Loading 数据加载程序。数据加载具体步骤如下:

1) HDFS 不适合存放大量小文件,所以要先把 RDF 数据处理成为与 S_PO,P_SO,P_OS,O_SP 模型对应的结构。

2）在 Hadoop 的分布式文件系统 HDFS 上为 RDF 数据创建一个专门的存储目录,然后调用 Hadoop 的 Put 接口进一步将数据导入到 HDFS 分布式文件系统的 RDF 数据存储目录下。

3）在 HBase 中建立空的 S_PO,P_SO,P_OS,O_SP 表结构,并指定它们的列族。

4）执行一个 MapReduce 任务,并行加载 RDF 数据到 HBase,任务的输入是存放 RDF 数据的文件,输出是 HBase 文件。

5）查看 HBase 中对应的表结构下是否存入了 RDF 数据。

对 Bulk Loading 数据加载核心部分的代码进行描述(向新增的 P_SO 表加载数据):

```
Method Map(Text value, Context context)
    String[ ]textStrSplit = text. toString( ). split("\t");
    //将谓语作为 Row key
    String hkey = textStrSplit[1];
    String column = textStrSplit[0];
    String hvalue = textStrSplit[2];
    rowKey = Bytes. toBytes(hkey);
    //谓语值存在
while(hkey = = ture)
    //创建 Put 对象用于存放 Put 实例
    Put HPut = new Put(rowKey);
    byte[ ] cell = Bytes. toBytes(hvalue);
    //向 Put 对象中添加单元数据
    HPut. add(Bytes. toBytes(column), cell);
context. write(HKey, HPut);
```

该数据加载算法主要包括数据分割、加载三元组和写入数据三部分。基于谓语的数据索引,需要将数据按照 P_SO 和 P_OS 模式分别完成加载。同时,对每一谓语值是否为空预先做出判断,若谓语值不存在,则不将该记录存入以谓语为索引的存储表中;反之,直接将数据写入对应的数据索引表中。S_PO 和 O_SP 存储表本文不做详细说明。

3　系统测试与分析

3.1　实验环境

实验搭建了具有 5 个节点的 Hadoop 集群,每个节点硬件配置如表 5 所示。存储结构设计是将 RDF 数据存储在 HBase 集群中,HBase 构建在 HDFS 之上,HRegionServer(域服务器)作为 HBase 的集群节点维护存储 RDF 数据的域(region),RDF 数据均衡分布在集群各节点中。实验采用 LUBM 测试集进行测试,LUBM[12]是当前使用最广泛的测试样例,该测试集可以根据指定的参数生成相应规模的数据集。例如 LUMB(10),参数 10 表示生成具有 10 个大学的语义数据集,其中指定参数越大数据集规模越大。

表 5　硬件配置

处理器	Intel Core i5 3.2 GHz
内存	8 GB
硬盘空间	500 GB
操作系统	CentOS6.5
Hadoop 版本	2.6.0
HBase 版本	0.98.8
JDK 版本	1.7.0

3.2　实验结果分析

实验采用 UBA[13](数据发生器)生成三组不同大小的 RDF 数据集进行测试,数据集大小如表 6 所示。

表 6　测试数据集

数据集规模	LUBM1	LUBM5	LUBM10
三元组个数	111 307	563 201	1 532 116

数据加载性能是评价该存储模型的一个重要指标,本文基于 MapReduce 编程模型实现了大规模 RDF 数据的并行加载功能。表 7 给出了串行加载算法和 MapReduce 并行加载算法加载三组不同规模 RDF 数据所需的时间。

表 7　数据加载消耗时间

数据集	Bulk Loading/s	串行/s	加速比
LUBM1	57	68	1.19
LUBM5	179	309	1.72
LUBM10	282	623	2.2

实验加速比可由式(1)得出:

$$S = \frac{T(1)}{T(N)} \tag{1}$$

式中:$T(1)$ 表示串行加载运行时间,$T(N)$ 表示并行加载运行时间。

由表 7 可以看出,对比串行方法,Bulk loading 并行数据加载算法在数据规模较小时优势并不明显,原因是当数据规模小时 Map 任务个数也较少,不能有效利用 MapReduce 的并行机制。随着数据规模的增大,实验表明该加载算法能够高效地完成海量 RDF 三元组数据的加载操作。

本文数据查询测试主要目的是验证谓语索引表 P_SO的高效性,设计谓语绑定,主语已知,宾语未知的查询用例:

```
SELECT ? Y
WHERE
{
```

< http://www. Department0. University0. edu > ub:subOrganizationOf ? Y

}

将该查询用例分别在 S_PO 和 P_SO 存储模式上进行测试,三组不同规模 RDF 数据所需的查询的响应时间如表 8 和图 2 所示。

表 8 SP? 查询响应时间

数据集	SP_O 索引/s	P_SO 索引/s	加速比
LUBM1	0.561	0.428	1.27
LUBM5	1.056	0.757	1.34
LUBM10	1.472	0.932	1.45

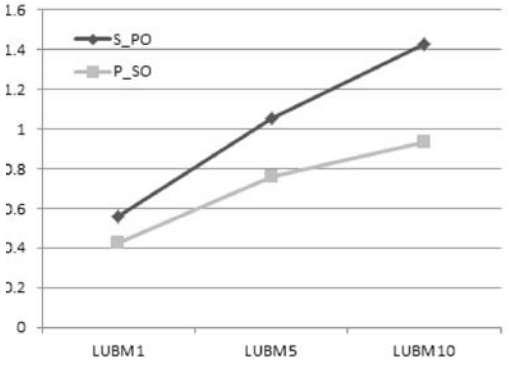

图 2 SP? 查询响应时间

由统计分析得出,基于谓语约束的查询请求数量约占整个系统查询总数的 90%。改进后整个系统的查询加速比(表 8)可由 Amdahl 定律得出:

$$S = \frac{1}{(1 - Fe) + \frac{Fe}{Se}} \tag{2}$$

本实验 $Fe = 0.9$,Fe 是执行某个任务的总时间中可被改进部分的时间所占的百分比,Se 是改进部分采用改进措施后比没有采用改进措施前性能提高倍数。

对比优化前的 S_PO 查询索引,P_SO 在查询时能够快速确定 Row-Key 范围,原因是数据源本身具有谓语数量级小的特点,根据 Row-Key 检索所有包含谓语 P 的三元组,然后根据主语 S 进行过滤操作,最终得出查询结果。随着数据规模的增长,谓语 P 确定的 Row-Key 范围也逐渐变大,因此查询时间递增。同时,MapReduce 并行机制进一步提升查询效率,查询加速比逐渐增大。

4 结 语

针对海量语义数据的管理问题,本文提出了一种基于分布式数据库 HBase 的 RDF 数据存储模型。相较于传统的三元组存储方法,该模型可以直接根据谓语的索引更加快速地查询到相应的关联数据值。在保证查询性能的同时有效地减少了存储开销,为实现大规模语义数据的高效查询和推理奠定了理论和应用基础。

本文在 RDF 数据的查询处理问题上没有做针对性的研究,接下来的工作是要对该模型上的查询算法进行优化,进一步提升数据的查询效率。

参 考 文 献

[1] 常亮,刘进,古天龙,等. 基于动态描述逻辑的语义 Web 服务组合[J]. 计算机学报,2013,36(12):2468 - 2478.

[2] 杜方,陈跃国,杜小勇. RDF 数据查询处理技术综述[J]. 软件学报,2013(06):1222 - 1242.

[3] Bizer C, Jentzsch A, Cyganiak R. State of the LOD Cloud [EB/OL]. http://lod - cloud. net/.

[4] Kaoudi Z, Manolescu I. RDF in the clouds: a survey[J]. Vldb Journal,2015,24(1):67 - 91.

[5] Harris S, Lamb N, Shadbolt N, et al. 4store: The design and implementation of a clustered rdf store[J]. Scalable Semantic Web Knowledge Base Systems,2009,4:94 - 109.

[6] 杜小勇,王琰,吕彬. 语义 Web 数据管理研究进展[J]. 软件学报,2009,20(11):2950 - 2964.

[7] Zhou Q, Hall W, Roure D D. Building a Distributed Infrastructure for Scalable Triple Stores[J]. Journal of Computer Science & Technology,2009,24(3):447 - 462.

[8] Rajapoornima M, Tamilselvan L, Priyadarshini R. Personalized semantic retrieval of information from large scale blog data[C]//IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology. IEEE,2017:1055 - 1059.

[9] Owens A, Seaborne A, Gibbins N, et al. Clustered TDB: A Clustered Triple store for Jena[C]//Proceedings of 18th Internation Conference on World Wide Web,2009.

[10] Abiri F, Kahani M, Zarinkalam F. An entity based RDF indexing schema using Hadoop and HBase[C]//International Econference on Computer and Knowledge Engineering. IEEE,2014:68 - 73.

[11] 徐德智,刘扬,Sarfraz Ahmed. 基于 Hadoop 的 RDF 数据存储及查询优化[J]. 计算机应用研究,2017,34(2):477 - 480,486.

[12] Guo Y, Pan Z, Heflin J. LUBM: A benchmark for OWL knowledge base systems[J]. Web Semantics Science Services & Agents on the World Wide Web,2005,3(2 - 3):158 - 182.

[13] Helflin J, Stuckenschmidt H. Editorial: Web-scale semantic information preprocess[J]. Web Semantic: Science, Services and Agents on the World Wide Web,2012(10):121 - 132.