软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(6):1222−1242 [doi: 10.3724/SP.J.1001.2013.04387]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

RDF 数据查询处理技术综述*

杜 方[1,2], 陈跃国[1], 杜小勇[1]

[1](中国人民大学 信息学院,北京 100872)

[2](宁夏大学 数学与计算机学院,宁夏 银川 750021)

通讯作者: 杜小勇, E-mail: duyong@ruc.edu.cn

摘 要: 随着语义网以及信息抽取技术等研究的发展,Web 上涌现出越来越多的 RDF 数据,海量 RDF 数据的管理,已经成为学术界和工业界研究的热点之一.从 RDF 数据集形态及 RDF 数据组织存储两个维度以及查询表述、查询处理、查询优化等方面,深入地分析和比较了 RDF 数据查询处理方法,并在此基础上提出了未来研究的方向和挑战.

关键词: RDF;RDF 数据管理;RDF 查询处理;查询优化

中图法分类号: TP311 **文献标识码**: A

中文引用格式: 杜方,陈跃国,杜小勇.RDF 数据查询处理技术综述.软件学报,2013,24(6):1222−1242. http://www.jos.org.cn/1000-9825/4387.htm

英文引用格式: Du F, Chen YG, Du XY. Survey of RDF query processing techniques. Ruan Jian Xue Bao/Journal of Software, 2013,24(6):1222−1242 (in Chinese). http://www.jos.org.cn/1000-9825/4387.htm

Survey of RDF Query Processing Techniques

DU Fang[1,2], CHEN Yue-Guo[1], DU Xiao-Yong[1]

[1](School of Information, Renmin University of China, Beijing 100872, China)

[2](School of Mathematics and Computer Science, Ningxia University, Yinchuan 750021, China)

Corresponding author: DU Xiao-Yong, E-mail: duyong@ruc.edu.cn

Abstract: With the advance of the semantic Web and information extraction techniques, more and more RDF data emerge on the Web. Managing massive RDF data has been a hot research topic for both academic and industrial communities. In the paper, we give a thorough analysis and comparison of the existing techniques from the view of the accumulation and organization of RDF data. Topics on query representation, query processing and query optimization are extensively discussed. We finally identify challenges and future work of the area.

Key words: RDF; RDF data management; RDF data query processing; query optimization

1 引 言

RDF(resource description framework)是由 WWW 提出的对万维网(World Wide Web)上信息进行描述的一个框架,它为 Web 上的各种应用提供信息描述规范[1].RDF 用主语 s(subject)、谓词 p(predicate)、宾语 o(object)(也有一些文献,如文献[2]中写作 subject,property,value)的三元组形式来描述 Web 上的资源.其中,主语一般用统一资源标识符 URI(uniform resource identifiers)表示 Web 上的信息实体(或者概念),谓词描述实体所具有的相关属性,宾语为对应的属性值.这样的表述方式使得 RDF 可以用来表示 Web 上的任何被标识的信息[3],并且使得它可以在应用程序之间交换而不丧失语义信息.因此,RDF 成为语义数据描述的标准,被广泛应用于元数据的描述、本体及语义网中,很多机构和项目,如 Wikipedia,DBLP 等都用 RDF 表达它们的元数据;IBM 智慧地球的研究

* 基金项目: 国家自然科学基金(61170010); 核高基重大专项(2010ZX01042-002-002-03)
收稿时间: 2012-06-07; 修改时间: 2012-09-12; 定稿时间: 2013-01-25

中[4]广泛采用 RDF 数据描述和集成语义;语义网数据库技术公司 Metaweb 维护的 Freebase 知识库中,用 RDF 表示电影、体育等众多领域元信息;化学、生物、地理、生物医学等多种领域在 RDF 基础上建立领域本体.随着在语义网、非结构化数据管理、生物信息、数字图书馆等诸多领域的广泛应用,Web 上的 RDF 数据量飞速增长,特别是出现了很多大规模的 RDF 数据集,据 W3C 的 SWEO(semantic Web education and outreach)研究小组统计,截止 2012 年 3 月,互联网上 RDF 数据集(包括 Linked Open Data,YAGO,DBpedia,Freebase 等)中的 RDF 三元组数量已经达到 520 亿[5].

随着大量 RDF 数据的涌现,管理和检索大规模 RDF 数据成为棘手的问题.近几年来,工业界和学术界出现了越来越多针对 RDF 数据的研究,很多公司开发了针对 RDF 数据的管理系统和工具,如 Oracle 的 Oracle RDF、HP 公司的 Jena、IBM DB2 10 的 RDF 的管理工具等.原型系统包括德国 MPI 研究所提出的 RDF-3X、爱尔兰 DERI 研究机构的 YARS 等;实现了基于 RDF 数据的语义搜索引擎,包括 DERI 的 Sindice 以及马里兰大学巴尔的摩分校的 Swoogle 等.在这些研究中,RDF 数据的查询处理技术是这些工具和系统的核心部件之一.

1.1 RDF数据查询的特点

RDF 数据可以被表示为一个带标记的图,图中的结点对应三元组中的主语和宾语,谓词为边.因此,大规模 RDF 数据上的查询可以看做是大图上的图匹配问题.然而,由于 RDF 数据图中包含很多文本信息,结点之间关联多,图规模巨大,导致 RDF 数据查询处理复杂、效率较低.因此,如何高效地执行 RDF 查询是重点和难点.RDF 查询处理的技术挑战,归结起来有以下几点:

1) RDF 数据的组织与存储还没有确定的方案,导致查询处理解决方案的多样性

针对 RDF 数据的图结构,研究者们探索并设计了不同的数据组织与存储方案,主要包括基于关系数据库的组织存储方案、纯图存储的方案以及基本三元组的索引存储方案.对于不同的组织与存储方案,提出了不同的查询处理方法,包括基于数据库的查询方法、基于图的查询方法以及基本三元组上直接使用索引的方法等.目前还难以下结论哪种方案最佳.

2) RDF 数据的查询需求不断增加,导致查询语言的功能不断扩展

RQL,RDQL,SquishQL,MQL 以及 SPARQL 等都是针对 RDF 数据的查询语言.早期的一些 RDF 管理系统采用 RQL,RDQL 等作为查询语言,如 Sesame,Jena 等.近年来的大多数 RDF 相关研究都采用 SPARQL 作为查询语言. SPARQL[6]是 W3C 推荐的 RDF 查询语言,它与 SQL 的语法相似但不同.在文献[7,8]等研究中,提出了在 RDF 数据上实现关键词查询.针对不同语言的查询处理框架不完全相同,例如:在一些研究中,采用将 SPARQL 映射为 SQL 的方法来执行查询[2,9];还有一些研究将 SPARQL 生成查询图[10,11],在图的基础上做进一步的查询处理.

3) RDF 数据不仅是海量数据更是动态数据,导致查询处理基础平台的多样性

很多 RDF 数据集中的三元组数量都在亿级以上,这种量级的数据给查询带来许多障碍.目前的多数解决方案都是在静态数据集的基础上利用多种索引机制和一些预处理的优化策略等来提高查询效率,但随着社交网络的发展,网页上的数据经常被更新,因此在可更新的、动态的 RDF 数据集上进行查询处理是一个必然的趋势.面对海量的 RDF 数据,一些研究提出了对数据进行分布式存储,或者在目前流行的云平台上实现对 RDF 数据的查询,这种可扩展的模式使得动态数据集上的查询处理成为现实.

1.2 问题空间的划分

本文从查询的对象——静态数据集和动态数据集,以及数据组织存储方案上的查询处理方法两个维度对目前 RDF 上的查询处理技术研究方法进行了划分,图 1 为这两个维度上的总体研究框架图.

从图 1 可以看出,目前的研究工作主要集中在静态数据集上的查询处理,而可用在动态数据集的方法较少.本文从图 1 的视角出发对目前已有的技术进行综述,从 RDF 的数据组织与存储、RDF 查询表述以及查询处理技术等方面深入的分析和比较现有的查询方法.

本文第 2 节介绍不同的 RDF 查询表述方式.第 3 节分析和比较静态数据集上的 RDF 数据存储及查询处理方法.第 4 节讨论可用在动态数据集(大数据)上的 RDF 数据组织存储及查询处理技术.第 5 节介绍查询优化技

术.第 6 节进行总结,并提出未来的工作和研究挑战.

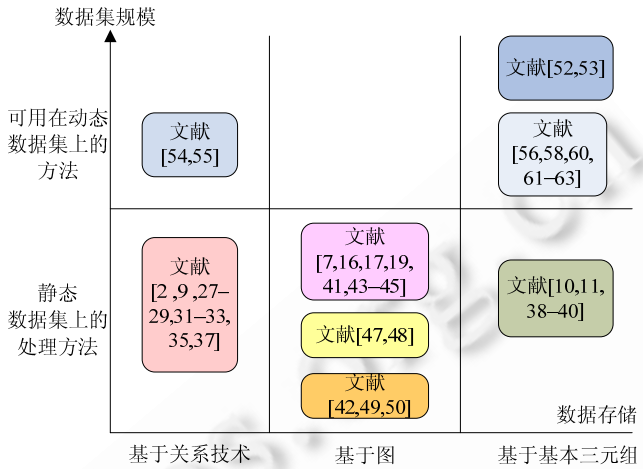


Fig.1 Framework of overall research works

图 1 总体研究框架图

1.3 预备知识

为了统一概念与符号,方便理解内容,本节对 RDF 数据查询中一些常用的概念进行形式化描述和说明.

RDF 三元组(RDF triple). 给定一个 URI 集合 R、空结点集合 B、文字描述集合 L,一个 RDF 三元组 t 是形如(s,p,o)的三元组,其中 $s\in R\cup B,p\in R,o\in R\cup B\cup L$.这里的 s 通常称为主语(subject)、资源(resource)或主体,p 称为谓词(predicate)或属性(property),o 称为宾语(object)、属性值(value)或客体.后续的章节中用 s,p,o 分别表示组成三元组的主语、谓词和宾语.

RDF 数据图(RDF data graph). RDF 数据图 G 是一个三元组(V,E,L),其中,V 为顶点集合,E 为边的集合,L 为标签集合,且 $L=L_v\cup L_p,L_v$ 为顶点的标签集合,L_p 为边的标签集合.这里的 V 对应三元组中的 s 和 o,边对应 p.即 RDF 数据图是以 s 和 o 为顶点、p 为边,并且顶点和边上都带有标签的图.

三元组模式(triple pattern). 三元组模式 $tp=\langle s,p,o\rangle$是一个三元组,VAR 代表变量集合,则有 $s\in VAR\cup R\cup B$, $p\in VAR\cup R,o\in VAR\cup R\cup B\cup L$.

三元组模式查询(triple pattern query).三元组模式查询 $Q=(q_1,q_2,\ldots,q_n)$是由三元组组成的元组,其中,每个 q_i 是一个三元组模式.本文用 tp 代表三元组模式,用 Q 代表三元组模式查询.

基本图模式(basic graph pattern,简称 BGP). 基本图模式是 SPARQL 查询中的一个基本组成部分,也称作连接查询(join query).一个基本图模式由一组三元组模式组成.图 2 中用查询图的方式表示一个基本图模式,本文用 BGP 来表示 SPARQL 查询中的基本图模式.

连接变量(join variable). 连接变量 v 是指 BGP 中的变量,通常在多个三元组模式中出现.

联合查询(conjunctive query). 联合查询的表示形式为$(x_1,\ldots,x_k).\exists x_{k+1},\ldots,x_m.A_1\wedge\ldots\wedge A_r$,其中,$x_1,\ldots,x_k$ 称为可区分的变量,即可以通过变量绑定产生查询结果的变量;$x_{k+1},\ldots,x_m$ 为存在量词;$A_1,\ldots,A_r$ 为查询原子(query atom).查询原子的形式为 $P(v_1,v_2)$,这里,P 为谓词,v_1 和 v_2 可以为变量或常量.

RDF 上的关键词查询(keyword search on RDF data). 关键词查询是形如 $W=(w_1,w_2,\ldots,w_m)$的查询表示,其中,每个 w_i 为一个关键词.

变量绑定(variable bindings). 变量绑定$\varphi(v)$是一个函数$\{v|\varphi(v)=C\}$,其中,$v\in VAR,C\in R\cup B\cup L$.即变量绑定是一个将变量映射到 URI 集合、空结点集合或文字集合的函数.

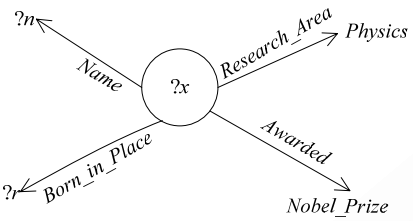


Fig.2 An example of an BGP represented by query graph

图 2 用查询图表示一个基本图模式的例子

2 RDF 查询表达

针对RDF数据,研究者提出了多个查询语言[12−15],如RQL,RDQL,SquishQL等.早期的一些RDF系统如Jena, Sesame,3Store都支持RDQL(RDF data query language)查询.RDQL将RDF看做图,其查询语句由构成图的RDF三元组组成.图3是RDQL的查询示例.

```
SELECT ?x
WHERE (?x,⟨p:x⟩,?y)
AND y=="Albert Einstein"
```

Fig.3 An example of a RDQL query

图 3 一个 RDQL 查询示例

RDQL支持路径表达,因此,实现RDQL查询可以采用图上的路径匹配方法,具体实现方法将在第3.3节中描述.

2.1 SPARQL

SPARQL(simple protocol and RDF query language)是W3C提出的RDF数据的查询标准语言,也是目前被广泛采用的一种RDF上的查询语言.当前,大多数RDF系统都支持SPARQL查询.SPARQL共有4种查询方式,分别为SELECT,CONSTRUCT,DESCRIBE和ASK.目前最常用的是SELECT查询方式,它与SQL的语法相似,用来返回满足条件的数据.图4(a)所示为一个简化的SPARQL查询的例子,图4(b)为该查询语句可能的查询结果.

```
SELECT ?name ?place
WHERE {
?x hasName ?name.
?x Research_area Physics.
?x born_in_place ?place.
?x Awarded Nobel_Prize}
```

(a) 一个简化的 SPARQL 查询的例子

name	place
Albert Einstein	Ulm German

(b) 查询结果

Fig.4. An example of a brief SPARQL query and the result

图 4 一个简化的 SPARQL 查询的例子及其查询结果

语句中SELECT子句与SQL中的SELECT子句一样,表示查询结果要投影的内容;WHERE子句为SPARQL查询的主要组成部分,用来表示涉及查询的三元组之间的连接关系、过滤条件(FILTER)等;语句中带问号的部分代表查询中的未知变量.WHERE子句中还可以使用丰富的关键词来表示联合查询、空结点信息等内容,限于篇幅,本文不再赘述,相关内容可见文献[6].

一些研究者在SPARQL的基础上进行了修改,提出了*n*SPARQL,SPARQL-DL等对SPARQL语法进行扩充,增强SPARQL的查询功能.

在基于关系数据库的RDF系统中,可以将SPARQL转换为SQL语句进行查询处理.此时,需要根据数据存

储时的模式信息补充SPARQL语句中缺少的From子句及相关信息.也就是说,利用三元组和存储模式之间的映射关系,指定查询所涉及的关系名.同时,由于SPARQL语句可以表示为图,因此也可以将SPARQL转换为查询图(query graph)来执行查询处理.根据查询图的结构,可将SPARQL查询分为星形查询和链式查询,其中,星形查询(star join)是指主语变量或宾语变量相同(s=s 或 s=o)的一组三元组模式构成的查询,链式查询(chain join)是指主语变量和宾语变量相同(s=o)的三元组模式.例如,图4(a)中的SPARQL语句可以转换为图2所示的查询图,且图2为一个星形查询的例子.具体的查询实现技术将在第3节和第4节中讨论.

2.2 关键词查询

对于普通用户来说,遵循SPARQL语法书写查询语句比较困难,因此,一些研究者提出了RDF数据上的关键词查询.根据查询处理方式的不同,RDF上的关键词查询可以分为两类:

1) 由关键词直接构造查询结果的方式[7,16−21];
2) 由关键词构造出形式化查询语句再得到查询结果[8,22−26].

2.2.1 由关键词直接构造查询结果

由关键词直接构造查询结果的方法大都采用在RDF数据图上定位子图的方法(实现方法将在第3.3节中详细讨论).这种方法通常需要借助有效的索引来定位子图并快速搜索结果.最常用的索引是基于关键词的倒排索引,通过索引可以快速定位图中包含关键词信息的结点.除此之外,根据具体解决方案的不同,研究者们还提出了摘要(summary)索引、路径索引等辅助结构来提高查询效率.

2.2.2 由关键词构造形式化查询语句

由关键词构造形式化查询语句的方法通常包括3个步骤:1) 关键词映射;2) 构建查询;3) 对查询排序.

关键词映射是指将关键词映射到数据源的概念(concept)、实例(instance)或实体(entity)、文字(literal)等对象.为了实现关键词映射,首先需要对数据进行预处理,生成数据图,并根据数据图建立关键词索引和数据图的模式索引.通过索引,可以在数据图上快速定位与关键词相关的顶点和边.

构建查询是指根据定位到的顶点和边,在数据图中找到包含这些顶点和边的子图,就得到了关键词对应的查询图.此时,再将查询图中的顶点和边分别映射为查询中的变量、常量和谓词,生成形式化查询语句.

文献[8,23,25]将关键词翻译成联合查询(conjunctive query),再得到SPARQL查询语句.例如,有关键词查询$W1$=(*Nobel_Prize*,*Physics*,*Ulm German*),通过索引这些关键词可能分别对应到数据图中标记为 *Nobel Prize*, *Physics* 和 *Ulm German* 的顶点或边,在图上定位包含这些顶点和边的子图后,就可以将子图映射为联合查询.如,$W1$ 对应的联合查询的一个例子可以是

$$?x?y.hasName(x,y)\wedge Reseach\_area(x,Physics)\wedge born\_in\_place(x,Ulm\ German)\wedge Awarded(x,Nobel\_Prize)$$

对查询排序:在关键词映射时,通常一个词会映射到图中的多个顶点或边,因此,生成的查询语句也会有多个,这就需要对得到的查询图或查询语句排序.

文献[8]根据查询图中顶点之间路径长度、顶点的流行度(popularity)以及关键词的匹配得分对查询排序.显然,顶点间的路径长度用来衡量顶点间的相关度,顶点的流行度是指在数据图中与该顶点相关的顶点数目,关键词匹配得分根据关键词和数据图中对应结点在语义和词法方面的匹配程度计算得出.文献[23]利用贝叶斯模型来计算查询语句的得分.文献[26]通过与用户的交互来选择正确表达用户意图的查询,这要求用户具有基本的领域知识.

2.2.3 对结果排序

基于关键词的查询需要对符合条件的查询结果进行排序,返回前 k 个结果.一般有基于TF/IDF,PageRank等排序方法,这些方法均为信息检索技术中常用的方法,因此本文不再赘述.

3 静态数据集上的查询处理技术

静态数据集中的数据是只读的,不考虑更新操作,目前,大多数RDF查询处理的研究都是基于这样的数据集实现查询.本节将从第2个维度——不同数据组织与存储方案上的查询处理技术,对目前静态数据集上的查询

方法进行分析比较.

3.1 基于关系的RDF查询处理技术

3.1.1 基于关系数据库的 RDF 数据组织与存储

关系数据库具有成熟、高效的数据管理和操纵技术,因此,很多 RDF 系统采用关系模型来组织数据,并将 RDF 图按三元组形式存储在关系数据库中.

三元组表:最简单、直接的方法就是按照 *s*,*p*,*o* 的三元组方式组织数据,直接将每条 RDF 三元组存储在关系数据库中.这种方式易于实现,并且可以借助成熟的关系数据库技术管理 RDF 数据,因此,早期的很多 RDF 管理系统采用这种三元组表(triple table)的组织存储方案,如 3-Store[27],Sesame[28]等.

虽然三元组表简单易实现,但将所有的数据组织成大表的方式带来了查询效率低的问题.因为一般情况下,一个实体(同一主语的三元组集合)的相关信息会出现在表的若干行中,这样即使只是针对一个实体的简单选择查询,都会被转换成含大量自连接操作,复杂查询更会导致巨大的查询时间代价.为了改善查询效率,多数系统在三元组表上使用大量索引,Oracle RDF[9]在三元组大表的基础上提取模式描述信息利用系统表单独存储,并建立规则索引(rules index,包含满足指定规则的预先计算的三元组).文献[29]在关系存储的基础上建立主索引和二级索引;也有一些研究如 Wide Table[30]等提出了水平划分的方法,将相同主语的三元组存储为表中的一行.但目前这种方法只应用于普通的 Web 文本数据.

属性表:Jena2[2]采用属性表的方式存储三元组,这种方法首先将三元组按照属性(谓词)分类,每一类存储为一张关系表,每张表的数据采用水平划分的方法.由于对属性分类,这种方法避免了水平划分中表中列过多的问题,减小了存储空间,并且在查询时可以有效减少主语间的自连接,提高查询效率.但在实际应用中,查询往往会涉及多个属性表,特别是对于属性未知的查询,需要多张表的连接或合并操作;同时,RDF 数据的结构性弱,对属性分类的策略,在很多情况下仍然无法避免大量空值的问题,因此,该存储方案只用于某些特殊应用中.

垂直存储:文献[31–33]提出用垂直存储方案存储 RDF 数据.垂直存储是指对 RDF 三元组表进行垂直划分(vertical partition),即按照谓词划分,将具有相同谓词的三元组存储在同一张表中,此时,谓词可以作为表名,只需保留 *s* 和 *o* 两列.并且,每张表按照主语排序,这样可以快速的执行关于主语的查询.文献[31,32]在列存储数据库中实现垂直存储方案,进一步加速查询——查询时只需读取相关的列.文献[34]在实验中证明,基于列数据库的垂直存储优于三元组表存储.但是由于要求主语在每个列表上都有对应的行,因此该存储方案仍然存在大量空值导致的数据稀疏问题.

3.1.2 基于关系查询技术的查询处理

基于关系数据库的 RDF 数据存储都采用关系数据查询技术来实现对 RDF 数据的查询.存储 RDF 数据时,系统将根据 RDF 模式或 RDF 图建立数据与表之间的映射.这样,对 RDF 三元组的查询将转换到相应的关系数据表上,利用关系查询技术生成结果,最后组合各部分返回内容生成最终结果.

生成关系查询语句:Jena,Sesame 等系统在关系数据库中存储 RDF 数据时,通过对数据的分析提取 RDF 的模式信息,根据模式信息将数据存储在不同的表中,并采用 SQL 语句进行查询.查询时利用模式信息生成 From 子句到相应的表中查找结果.当查询中的三元组涉及到多张表时,无法用一条 SQL 语句执行查询,这时需要对 SQL 语句进行预处理.文献[35]中,依据 RDF Schema 信息,查询包含的三元组会被分割成若干个组,同一组的谓词只在同一张表中,最后将来自不同表的返回结果组合起来得到最终结果.Oracle RDF 将类似 SPARQL 语法的查询语句嵌套在 SQL 语句中,借助函数构造出 From 子句所需的目标表,从而利用关系数据库查询引擎返回结果.Oracle RDF 的 SQL 语句格式为:

Select 〈表名|别名〉.〈目标变量名〉

From TABLE (RDF MATCH 函数)

WHERE 〈条件表达式〉,

其中,RDF MATCH 函数用来构造表,由 Oracle 数据库中的 table 函数接口支持[36].在编译时,由 table 函数接口根据函数参数及数据存储模式定位构造表所需的列集合,执行时结合 RDF MATCH 函数的各参数在定位好

的列上做自连接,生成表.

文献[27]将RDQL查询翻译成关系演算表达式,RDQL中的每个三元组对应关系演算中的存在量词,约束条件转换为相应的关系表达式,查询中的自由变量将出现在对应的查询属性位置上.例如,图2的RDQL查询会被翻译如下形式的关系演算查询表达式,然后实现对关系演算查询的处理:

$\{r.uri|resource(r)\wedge(\exists t1)(triple(t1)\wedge t1.subject=r.hash\wedge t1.predicate=hash(p{:}x)\wedge not(t1.object=hash(Albert\ Einstein))\}$

RDFBroker[37]将RDF查询图映射至关系存储中的一组表,并用关系代数操作实现查询.RDF查询图上定义了投影和投影选择组合两个操作,针对每个查询,RDFBroker 首先利用一个抽象的包含图定位相关的表,然后把这些表组合在一起,在其上生成关系代数查询运算,得到查询结果.

连接操作:在基于关系查询技术的方法中,RDF 上连接操作的处理也采用传统的关系数据库中的连接查询方法,包括嵌套循环连接和归并连接等.由于数据以三元组表的形式存储,因此在进行连接操作的同时还需要执行大量的自连接操作.为了加快查询速度,可以在表上建立索引[35],或者对一些常用的中间结果生成物化视图,见文献[31].

3.1.3 基于关系的查询处理技术比较

在关系数据库中实现查询可以借助成熟的关系查询处理技术,但由于RDF数据不具有明显的模式信息,简单的三元组存储方式带来了查询时的大量自连接操作,虽然水平划分和垂直存储的方式对查询效率有一定的改善,但空值过多等问题仍然导致无法从根本上提高查询性能.表1总结了基于关系的RDF查询处理方法.

Table 1 Comparison of relational RDF data query processing techniques

表1 基于关系的查询处理技术比较

组织存储方式	方法	索引方式	查询语言			
			SQL	SPARQL	RDQL	RQL
三元组表	3-Store[27]	主外键上的哈希索引	√		√	
	Jena[2,35]	关系表上的常用索引	√		√	
	Sesame[28]	关系表上的常用索引				√
	OracleRDF[9]	规则索引	√	√		
	SOR[29]	聚集索引(clustering index)	√	√		
垂直存储	CStore[31,32]	标识列上的聚集索引+字符串列上的非聚集索引	√			
	CODERS[33]	表上的常用索引	√			

3.2 基于基本三元组的查询处理技术

3.2.1 基于基本三元组的RDF索引存储组织

基于基本三元组的方式是指将RDF三元组以原生态方式进行组织存储,通常将三元组分别按 *s*,*p*,*o* 的不同顺序直接利用B+树、哈希等索引结构来存储RDF数据,并在该索引结构上实现对RDF数据的查询.RDF-3X[10], Hexastore[38],BitMat[39],RDF Cube[40]等均采用这种方式组织并存储数据.

由于RDF数据中包含大量的文字信息,直接用索引存储三元组需要很大的磁盘空间,因此在存储前通常需要对三元组进行整数序列化,即:将三元组中的 *s*,*p*,*o* 分别进行整数编码,并且将字符串到整数间的映射关系存储在映射表中.为了进一步节省空间,通常还需要对数据进行压缩存储.

RDF-3X将三元组的 *s*,*p*,*o* 分别整数序列化后压缩存储在B+树中,并且按照 *spo* 的所有排列的索引方式进行冗余存储,以此换来高效的查询效率.与RDF-3X类似,Hexastore也按照6种索引的方式直接存储三元组.例如,在以 *p* 优先的 *pso* 索引存储方式中,每个存储对象由一个 *p*、*p* 对应的 *s* 向量以及每个 *s* 向量下的 *o* 列表构成.BitMat采用bit cube的三维分别代表 *s*,*p* 和 *o*,cube中的每个bit表示唯一的一个 *spo* 组合.与BitMat类似,RDF Cube用hash函数将三元组中的 *s*,*p*,*o* 映射至cube的三维,cube中的每个cell代表一个RDF三元组.

基于基本三元组方式大都对数据进行冗余存储,借助完备的索引机制达到提高查询效率的目的.虽然采用压缩存储,但冗余的方式仍无法适应数据的更新和大数据的查询处理.

基于基本三元组的键值对存储:基于云平台的 RDF 数据管理技术中通常采用基本三元组的键值对组织存

储 RDF 数据,这时,三元组中的 *s*,*p*,*o* 分别作为键(key)和值(value)存储在分布式文件系统中,相关的具体存储及查询处理技术将在第 4 节讨论.

3.2.2 基于基本三元组的查询实现

由于三元组直接存储在索引中,因此这些方法可以借助完备有效的索引实现查询,这样可以在很大程度上提高查询的效率.由于事先对数据进行整数序列化及压缩存储,这种组织方式下的查询处理还包括执行查询前后在字符串和整数值之间进行转换.

SPO 索引存储实现查询:SPO 索引是指根据 *spo* 的排列顺序,依次在 *s*,*p*,*o* 上建立索引.RDF-3X 和 Hexastore 都采用穷举的方式建立 SPO 完全索引,包括 SPO,SOP,PSO,POS,OSP,OPS 索引.图 5 为 Hexastore 中的 PSO 索引结构.

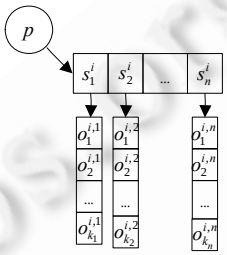


Fig.5 PSO indexes storage structure in Hexastore

图 5 Hexastore 中的 PSO 索引存储结构

数据按照 SPO 索引顺序组织存储在某种索引结构(如 B+树)中,当查询中进行变量绑定时,不需要对整个数据集进行扫描,而是在相应的索引中做范围查找.由于建立了完备的索引机制,在执行连接查询时,这些方法都尽可能地使用归并连接,只有在无法进行归并连接时才转换为哈希连接.

RDF-3X[10]是基于基本三元组的查询技术的典型代表,也是目前效率最高的 RDF 查询引擎.除了 6 种 SPO 索引外,RDF-3X 还创建了 SP,SO 等 4 种二维索引和 *S*,*P*,*O* 等 3 种一维索引,并且在这些聚集的索引中利用剩余的比特(E-flag)记录该索引方式下的三元组组合出现的次数,这样,当查询中包含去重(distinct)操作以及查询优化时的选择率估计中,可利用这些聚集索引来进一步加速查询.执行查询时,RDF-3X 将 SPARQL 语句转换成联合查询,对于组成联合查询的每个三元组模式,只需要根据相应的索引做一次扫描即可完成变量的绑定.当查询中有多个三元组模式时,首先进行查询顺序优化,再将每个单次扫描的结果进行连接,得到最终查询结果.关于查询优化的方法将在第 5 节进一步分析讨论.

Hash 索引存储实现查询:RDF Cube[40]根据 RDF 三元组的结构设计了一个三维的 hash 索引存储结构,用 hash 函数将三元组中的 *s*,*p*,*o* 映射至 cube 的三维,cube 中的每个 cell 代表一个 RDF 三元组,并且用 cell 中的一个比特表示每个三维坐标组合下的三元组是否存在.查询时,查询语句中的常量也按照 hash 值映射,根据映射值在 RDF 数据中找到候选的结果 cell(候选结果三元组).得到的候选三元组可能是一个序列或一个矩阵,若为矩阵,则按行和列抽取出两个序列.当查询中的三元组模式含有相同的连接变量时,需要对候选序列(矩阵)执行 AND 操作.图 6 为 RDF Cube 实现连接查询的例子.

3.2.3 基于基本三元组的查询方法的比较

由于用索引结构直接存储三元组,基于基本三元组的查询实现方法都具有较高的查询性能.表 2 为几种相关方法的比较.

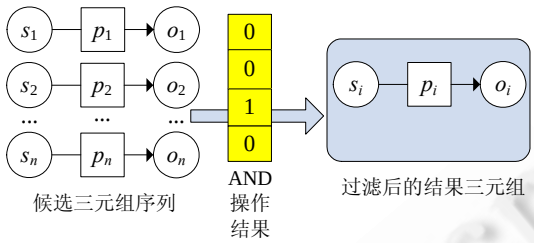


Fig.6 An example of query execution in RDF Cube

图 6 RDF Cube 实现查询的例子

Table 2 Comparison of query processing approach based on basic spo triples

表 2 基于基本三元组的查询方法比较

索引类型	方法	三元组存储(索引)结构				查询实现		
		B+树	向量	倒排	hash 表	归并连接	AND 操作	实现特点
SPO 索引	RDF-3X[10,11]	●				■	■	聚集索引联合查询
	Hexa[38]			●			■	Bitmat 间的交操作
	BitMat[39]		●			■		向量间的归并操作
Hash 索引	RDF Cube[40]				●		■	候选序列的与操作

3.3 基于图的RDF数据查询技术

3.3.1 RDF 的图存储

RDF 数据可以表示成图,因此在一些研究中[41−43]提出了直接存储 RDF 图结构的方法.

文献[41]中,RDF 图被表示为邻接表存储在磁盘上,表中的每一项包括图中顶点的 ID、顶点的标记信息以及该顶点的出边列表.文献[43]将 RDF 数据表示为二分图.文献[42]将 RDF 图存储在面向对象的数据库系统中,图中的每个 *s*,*p*,*o* 都是数据库中的一个对象,出边表示为对象之间的映射属性.

在图存储的方式中,RDF 数据查询可以转换成查询图与 RDF 数据图之间的匹配问题(如文献[41]),也可以借助对象关系数据库中的对象查询语言(OQL)实现对图对象的检索[42].

3.3.2 基于图的查询实现

RDF 的数据模型是图,因此 RDF 上的查询处理可以被转换为大图上的子图匹配问题.

定义. 对于一个给定的图 $G=(V,E)$,V 为边集合,E 为顶点集合.这幅图的一个匹配 M 是图 G 的一个子图 $G'=(V',E')$,其中,$V'\subseteq V$,$E'\subseteq E$.并且其中每两条边都不相邻(没有公共顶点).

为了提高查询效率,基于图的查询方法大都建立有效索引,首先在索引上匹配查询子图,减小搜索空间,再到缩小的子图上找到对应的查询结果.也有一些方法借助查询语言给定的起始和终止结点,在图上按路径搜索查询结果.

- 基于树形索引的方法

图的划分:GRIN[44]首先采用聚类的方法对 RDF 数据图中结点进行划分,按照主语进行聚类,将图中的顶点聚集到 C 个不相交的集合中,然后计算每个聚类的中心点(center)和半径,根据聚类结果建立树状索引.

GRIN 是一棵平衡二叉树,采用自底向上的方法生成树结构.其叶子结点为 RDF 上的聚类结果,生成非叶子结点时,首先任意选取一个叶子结点,并找出与其距离最近的结点,然后计算这两个聚类的中心结点 C 及半径 r,由 C 和 r 构成这两个结点的父亲结点.重复此构造过程,直至最终生成根结点.由于随机选取结点,因此,GRIN 树并不唯一.图 7 为 GRIN 索引的一个例子.

设给定查询中的常量字串为 x,查询时在 GRIN 树中进行搜索,若树中的结点与 x 的距离小于结点中给出的半径,则表示在该(C,r)指定的范围中包含查询结果.之后,根据这些找到的(C,r)结点在 RDF 数据图的相应部分进行搜索找到结果.由于此时定位到的图远小于原始的 RDF 大图,因此可以调用内存算法进行快速匹配返回查询结果.

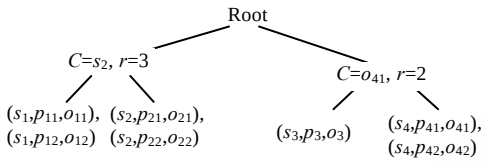


Fig.7 An example of GRIN index

图 7 GRIN 索引的例子

与 GRIN 类似,文献[45]也是在数据划分的基础上建立树形索引——PIG(parameterizable index graph).PIG 中的每个结点都代表 RDF 数据图中的一组元素,这些元素具有结构上相似或相同的邻居结点,即一组无法根据出边和入边来区分的元素.

查询时,首先从 PIG 中检索与查询图中的边同态的边,加入候选边集合,然后在候选边集合中执行 join 操作,找到结果.

S 树索引:GStore[41]在存储 RDF 图时对图中的每个实体进行编码序列化,存储为签名图(signature graph)G^*.查询时,按照 RDF 图的编码转换方法将查询也表示为一个签名图 Q^*,这样,查询被转换为在 G^*上搜索 Q^*的匹配问题.

GStore 在 G^*上建立 VS-tree 来加速查询.按照 G^*中的编码首先生成一棵高度平衡的 S 树(关于 S 树可参考文献[46]),树中的每个叶子结点对应 G^*中的一个顶点.如果两个顶点在 G^*中有边相连,那么在 VS-tree 中也有边连接对应的结点,并且在边上加入原有的标签信息.若 S 树中两个顶点的孩子结点间存在边,则在 VS-tree 中为这两个顶点建立一条超边(super edge),并且对这些相连的孩子结点做“或”操作,作为该超边的标记.这样自底向上的建立 VS-tree,直到生成根结点.

查询时采用自顶向下的搜索方法在 VS-tree 中找 Q^*在 G^*上的匹配.因为 VS-tree 中的非叶子结点中包含其孩子结点间边的信息,则在自顶向下的搜索过程中通过这些信息减小搜索空间,将匹配定位在更小的图结构上.

生成结果树:RDF 上的关键词查询根据关键词和图建立倒排索引和基于图的索引,借助索引在数据图上匹配查询子图,并且将匹配结果生成结果树.

BLINKS[7]首先将图划分成子图/块(block),并在此基础上建立各块的摘要索引及块内的路径索引.对于用户提交的查询关键词,从至少包含一个关键词的某个块开始,采用反向搜索算法对图进行搜索,即搜索时选择最近访问过的结点的入边,沿该边反向访问其源结点,并将源结点和到源结点的最短路径加入结果树,一直找到所有关键词结点的根结点.

Steiner 树:在 RDF 数据图上搜索满足关键词的结果树,并且要找到一棵最小的结果树时,关键词搜索问题就演变成了图论中的 Steiner 树问题.文献[16,17,19]都根据 Steiner 树问题来设计查询方案,即在 RDF 图中找到结点间路径最短且包含关键词信息最多的一棵结果树.为了提高查询效率,这些方法都在图上建立基于关键词的倒排索引和结点间的最短路径索引,采用启发式规则找到 Steiner 树问题的最优解.

文献[16]首先根据倒排索引确定包含每个关键词的句子结点,这样形成多个初始集合,然后从初始集合开始,利用路径索引将关键词结点可以到达的新的结点加入到集合,直到一个结点同时属于每个集合.扩充集合时,根据 Steiner 树原理,每次选择当前势最小且距离当前要扩充的关键词结点最近的结点.这样得到的是备选结果树,算法继续循环,直至找到前 k 个结果树.

文献[17]将 Steiner 树扩展为 Steiner 图,并将问题描述为在数据图上某个阈值范围内找半径(图中任意两结点间的最大距离为该图的直径),同时包含更多关键词的子图.在文献[17]中,图被表示为一个布尔型的邻接矩阵(1 表示两个结点间有边存在,否则为 0),通过对矩阵求 N 次方的多次迭代后找到半径小于阈值的图,则为候选 Steiner 图,然后根据索引找到包含关键词的前 k 个结果.

- 基于路径匹配的方法

RQL,RDQL 等查询语言均支持路径表达式,BRAHMS[47]根据 RQL 表达式中指定的起点和终点,在图上进行深度优先搜索或宽度优先搜索找到匹配的路径.

文献[48]中首先利用 Web 上的 linked data(根据 URI 将 Web 上不同数据源之间的 RDF 数据联系起来)信息,将查询图扩充为上下文图(context graph):扩充时以开始执行查询的结点作为根(root),根据 LD(linked data)图中实体及实体间的联系,依次加入与结点关联的星形结构图.在加入的同时,利用 RDF schema 中的类(class)信息及词汇表信息,对要加入的内容进行筛选,去掉那些与查询不属于相同类(class)和域(domain)的点和边.上下文图存储在缓存中.这样,在生成时可以避免加入重复的信息.执行查询时,在生成的上下文图中进行路径匹配,从根结点开始分层搜索,找到那些边上的标签与查询中标签匹配的路径,得到查询结果.

- 面向图对象的查询处理

文献[42,49]中,RDF 图被存储在对象关系数据库中,借助面向对象的查询语言,实现对 RDF 图数据的查询.文献[50]对查询语言进行了扩充,考虑了图形中邻接点、邻接边、顶点的度、路径等多个特征,对 RDF 图进行查询.

3.3.3 基于图的查询实现方法比较

基于图的方法将 RDF 数据和查询都表示为图,这样既保留 RDF 数据间的关联信息又不丧失语义信息.表 3 对比了几种基于图的查询实现方法.基于图的查询处理算法的时间复杂度及空间复杂度都与结点数目 n 有关,因此面临大数据时,基于图的查询处理方法效率都较低.因为面向对象的图数据查询是借助面向对象的数据库查询技术实现,所以没有对文献[42,49,50]进行比较.

Table 3 Comparison of query processing techniques based on approaches on graph

表 3 基于图的查询处理方法比较

查询策略	方法	索引类型	结果模型		时间复杂度	空间复杂度
			树	图		
基于树的方法	GRIN[44]	基于树的子图索引		√	$O(n!)$	$O(n)$
	PIG[45]	基于树的子图索引		√	$O(kn)$,k 为索引中边的数目的最大值	$O(n)$
	GStore[41]	基于 S 树的子图索引		√	$O(n^2)$	$O(n)$
	BLINKS[7]	块上的路径索引	√		$O(kn)$,k 为块的数目	$O\left(\sum_n n_b^2\right)$, n_b 为块的数目
	Ref.[16]	倒排+路径索引	√		$O(kn)$,k 为与查询相关的结点数	$O(n)$
	EASE[17]	r 半径图索引		√	$O(n^2)$	$O(n^2)$
	Kerag[19]	倒排+路径索引	√		$O(kn)$,k 为查询关键词的个数	$O(n)$
路径匹配的方法	BRAHMS[47]	基于哈希表的中心结点索引		√	$O(n^2)$	$O(n)$
	Ref.[48]	摘要索引		√	$O(kn^2)$,k 为结点的邻接点数平均值	$O(n)$

3.4 静态数据集上RDF查询处理技术的对比与分析

查询的代价主要体现在连接操作上,本节将针对连接操作对静态数据集上的 RDF 查询处理技术进行比较与分析.

- 基于关系的查询处理方法的分析

基于关系技术的 RDF 数据管理将 RDF 数据存储在关系表中,这样可以利用成熟的关系查询技术实现对 RDF 数据的查询处理.但这样的存储方式也给查询处理带来了很多问题.

首先,在三元组表上执行星形连接和链式连接时都需要大量自连接操作,设 RDF 数据集中某主语 s 的数量为 N_s,如果使用嵌套循环连接,那么仅涉及这一个主语的星形查询所需的连接次数将为 N_s^2.如果在 10 亿级的数据集上实现包含多个不同连接变量的查询,那么在某些较为常见的主语或宾语上执行连接时,需要的连接次数将超过 10^{10},加上全表扫描(无索引时)或读索引(有索引时)操作以及多趟连接操作,查询将异常缓慢,甚至无法进行.即使采用归并连接将复杂度降低到 $O(\mathrm{MAX}(N_{s1},N_{s2}))$,当连接主语或宾语的数量巨大时,查询效率仍然无法得到明显改善;在水平划分的三元组表上执行查询时虽然无需再进行自连接,但是由于不同主语下的谓词不同,

会造成表中包含大量的空值,增加索引的负担,影响查询性能;属性表和垂直存储按照谓词将三元组保存在不同的表中,使得查询时可以按照谓词定位相应的表,从而在较大程度上减小搜索空间.但是当星形查询或连接查询中的变量所涉及的主语和宾语不在同一个表时,需要进行表之间的连接,特别是当查询中的有些谓词未知时,会导致大量的表连接运算.设查询中的主语变量出现在 M 个表中,宾语变量出现在 N 个表中,当谓词未知时,最坏情况下需要进行 $M+N-1$ 次表连接运算,最好情况下需要进行 $\mathrm{Min}(M,N)-1$ 次表连接运算.当 M 或 N 的数目较大时,这些存储方案上的连接查询效率也会很低.

同时,水平划分存储的属性列众多,而属性表和垂直存储中同一资源(主语)会涉及到多张表,因此,这两种存储方式上的更新代价都很高,而大数据的处理一直是关系系统的瓶颈,因此,基于关系的查询处理方法在面对动态大规模 RDF 数据时显得力不从心.

- 基于基本三元组的查询处理方法的分析

基于基本三元组的方法大都采用索引结构直接存储三元组,这样,在执行查询时可以尽可能地使用时间复杂度较低的归并连接,在一定程度上提高查询效率.但是当执行复杂查询时,例如包含多个连接变量的链式查询,这种优势将不再明显.如果将索引存储中的一页看做一张表,当查询中包含多个连接变量时,需要在多张表(假设为 M)上进行连接操作.特别是当谓词为某个常见谓词(例如在 Barton Libraries[51]数据集中,共 5×10^7 多个三元组,其中出现频率最高的#type 谓词包含在 24.5%的三元组中,并且出现频率在前 13%的谓词包含在 99%的三元组中.而在 yago,BTC 等很多 RDF 数据集中也为类似的分布情况),当数据量进一步增加时,M 的数目会很大,这样,即使是归并连接也需要很多趟执行,造成不可忽视的查询代价.

基于基本三元组的 RDF 数据存储通常为冗余存储,可扩展性低,这也为数据的更新带来很多问题.因此,基于基本三元组的查询处理方法无法直接应用在动态大规模 RDF 数据集上.

- 基于图的查询处理方法的分析

用图来表示 RDF 数据可以保持其语义信息,更好地实现针对 RDF 语义信息的查询.但典型的图算法时间复杂度高,虽然基于图的 RDF 查询处理方法通过建立子图索引、路径索引等方法提高查询效率,但是从表 3 中可以看出,很多查询方法的时间复杂度都为 $O(n^2)$,n 为图中结点的数目.而 RDF 数据图中结点数目多,图的规模很大,导致这些方法的查询效率很低.即使是那些时间复杂度为 $O(kn)$的方法,当 n 很大时查询效率仍然无法得到较大程度的提高.

基于图的查询方法其空间复杂度也与 n 有关,同样的,当 n 进一步增大时,所需内存空间将无法满足需求,因此,基于图的查询方法不具备良好的可扩展性,无法有效地处理动态大数据集上的 RDF 查询.

表 4 对静态数据集上的查询方法进行了对比.

Table 4 Comparison of query processing techniques on static RDF dataset

表 4 静态 RDF 数据集上的查询处理技术比较

查询处理方法	特点	主要优势	存在的问题
基于关系的查询处理技术	关系存储上的关系查询	1. 成熟的关系查询处理技术 2. 完备的数据管理方案	1. 大量自连接或表连接导致查询效率低下 2. 大量空值影响查询效率 3. 对大数据的扩展性差
基于基本三元组的查询处理技术	全索引存储上的查询处理	1. 全索引机制有效提高查询效率	1. 冗余存储和大量索引导致可扩展性差 2. 对可更新数据的支持不具有良好的可扩展性
基于图的查询处理技术	图匹配查询	1. 更好的支持语义查询 2. 经典的图匹配算法	1. 查询算法的时间复杂度较高 2. 针对大图的可扩展性差 3. 更新数据的代价大

从以上的分析和对比可以看出,静态数据集上的查询方法扩展性不高,无法直接应用在动态数据集上.

4 可用在动态数据集上的查询处理技术

随着语义网和 Web 2.0 的发展,RDF 数据呈现出动态的、海量的增长特征.这就要求 RDF 上的查询技术能够应用在动态的大数据集上.在动态数据集上进行查询处理,要求方法具有扩展性,这样才可能对频繁更新的数

据集及时作出处理.静态数据集上的查询处理技术大都通过大量索引和冗余存储来保证查询效率,显然,这些方法无法直接应用在动态数据集上.目前,一些研究者对已有的静态数据集上的方法进行了改进,其中更多的方法是采用分布式查询处理技术,以适应大数据集动态变化的需求.

4.1 静态查询处理方法的改进

一些静态数据集上的基于基本三元组的方法对原有方案进行了改进,通过多版本或多重索引在单结点上实现对动态可更新数据的查询处理.

x-RDF-3X[52]是对 RDF-3X 的扩展,在原有的 *spo* 索引的基础上采用多版本技术将三元组存储在 B+树中.而对于频繁更新的三元组,若当前索引页面无法存储其所有版本,则创建一棵单独的 B+树存储当前版本.执行查询时,通过扫描索引获取三元组的所有版本,再对这些数据进行版本过滤,保证结果的正确性.

文献[53]首先将三元组分解为 *s-po*,*p-so*,*p-os* 和 *o-ps* 等 4 种形式,并按这些形式在三元组上创建双重索引,分别为 AP(atom position)索引和 BT(binary tuple)索引,其中,AP 索引是针对 *s*,*p*,*o* 的原子索引,采用键值(key-value)对的方法进行存储.例如对于 *s-po* 形式,AP 索引按 *s* 键记录 *po* 二元组的存储位置和相关数据的数量.PT 索引中存储每个键值(*s*,*p*,*o*)对应的二元组数据.PT 索引存储在分块后的本地磁盘上,为了保证读写效率,这些块都被设计成长度固定的小块.

查询时,根据三元组模式变量的位置读取相应的 AP 索引,再根据 AP 索引的内容找到对应的 PT 索引,装载到内存进行变量绑定.这种可以装载到内存的小块和相对的轻量级索引设置,使得该方法可以应用在动态 RDF 数据集上.但仍然只能支持批量的数据插入而非频繁的数据更新.

虽然这些改进后的静态查询处理方法能够用在动态数据集上,但是当面对大数据时,多版本和多重索引所需要的空间开销将非常巨大,这将导致查询时的搜索空间过大;而当数据频繁更新时,索引更新的代价也会成为这些方法的瓶颈.由于集中式单结点的方式不具备良好的可扩展性,因此,这些方法在处理大量动态 RDF 数据时必然带来查询效率低下的问题.

4.2 分布式查询处理技术

4.2.1 分布式存储和云存储

文献[54,55]等对 RDF 数据采用分布式组织存储,在每个结点上用关系数据库来存储数据.RDF Cube[40]实现了 P2P 环境中的 RDF 数据管理,该方法也可以用在分布式环境中.随着云计算技术的发展,目前还有一些研究提出了基于云平台的 RDF 数据存储及管理.云平台下以分布式文件系统(HDFS)对数据进行组织存储,为了将数据分布在各个结点上,通常需要根据某种策略对基本三元组进行划分,然后按照划分结果以键值对的方式存储 RDF 数据.划分的策略主要包括:1) 按照图结构划分,例如,文献[56]根据 RDF 图结构,将图中位置靠近的结点划分在同一机器结点上;2) 按照三元组中的谓词进行划分,文献[57]首先按照谓词将数据划分成不同的文件,在此基础上,再根据宾语的类型将谓词文件划分成更小的文件;也有一些方法不对基本三元组进行预先划分,直接以键值对的方式将数据存储在 HBASE,BigTable 等分布式系统中[58,59].这些方法都借助云平台的分布式并行计算能力实现对海量 RDF 数据的管理.

4.2.2 分布式查询处理

RDF 数据的分布式查询是指在分布式系统上对 RDF 数据进行查询[54,55,57],包括目前正在开展的在云平台上实现 RDF 数据查询的方法[56,58,60,61].由于数据被分布在不同的结点上,执行查询时需要根据数据的划分规则对查询语句进行分解.当查询被分解到单个结点后,可以利用静态数据集上的查询处理方法对局部数据进行处理.因此,与静态数据集上的集中式查询处理技术相比,分布式查询处理的研究还主要包括如何按照数据的分布对查询进行分解和如何在分布式环境中实现连接查询两方面的内容.

- 查询的分解

目前的研究从两个角度对查询进行分解:从数据分布的角度以及从查询语句的构成角度.

根据数据分布的策略对查询进行分解:文献[57]基于图结构将数据划分在不同结点上,并针对划分后的数

据建立数据与划分模式的映射和索引,查询时根据索引将查询分解到不同的机器结点或数据源结点.文献[56]根据数据的分布将查询分解成 PWOC(parallelizable without communication)子查询,每个子查询只在一个机器结点上执行.文献[60]将数据按谓词划分为不同的谓词文件,再根据宾语的类型对谓词文件进行进一步的划分.查询按照谓词文件的类型分解成不同种类,根据查询的种类读取相应的数据文件进行查询.

根据查询语句的结构对查询进行分解:另外一种角度是从查询语句本身出发对查询进行分解.文献[61]针对常用分析型查询进行预先定义,将可能的查询名(查询对应的实体名)存入单独的表,并在表上加载索引.这样,当处理 SPARQL 查询时,可以根据查询名找到对应索引,并装载相应数据进行查询.文献[62]将查询分解成多个星形子查询,在查询处理时使得每个分发回收(map-reduce)的迭代过程中只处理一个星形子查询,最后再将这些子查询的结果进行连接得到最终结果.

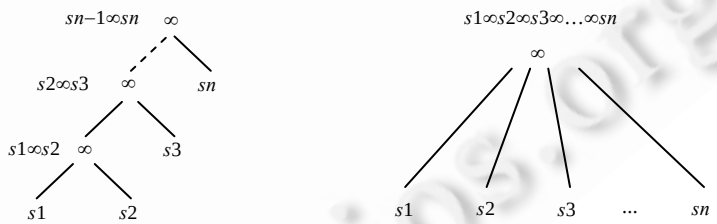
- 分布式查询处理

将查询语句分解后,就可以在不同的数据结点上进行查询得到中间结果,再对这些结果进行处理返回最终结果.

单个数据结点上的查询处理可以采用静态数据集上的方法,根据结点上的数据的分布组织方式,若结点均以关系数据库组织并存储数据,则可采用基于关系的查询处理技术[54];若结点上的数据采用基于基本三元组的组织存储方式,相应地将采用直接利用索引查询处理方法.例如,文献[56]在云平台的每个结点上部署 RDF-3X 引擎实现对数据的组织和查询;在异构的分布式系统中,则需要利用结点上的本地查询引擎进行查询处理[57],再对中间结果进行组合.

在云平台上对 RDF 进行查询处理时,需要选择合适的连接键(join key)来减少分发回收的迭代次数.当查询中只有一个需要绑定的连接变量时,则将该变量作为连接键;当查询中有多个连接变量时,可以根据贪心策略,在每次迭代时根据相关三元组模式的数量选择合适的连接变量作为连接键,或者一次选择多个连接变量进行多路连接操作.

多路连接:由于具有多处理器并行的优势,在分布式环境中,特别是云平台上[63],可以将普通的二元连接查询转换为多路连接查询来处理.图 8 展示了传统的二元连接执行树和转换后的多路连接树.



(a) 二元连接树 (b) 多路连接树

Fig.8 Binary join tree and Multi way join tree

图 8 二元连接树和多路连接树

多路连接是一种基于 hash 的连接方法,每个连接操作可以分成 build 和 probe 两部分,其中,build 用来根据键值创建 hash 表,probe 用来执行 hash 映射和匹配.一般来说,当有 k 个缓存时,首先根据缓存大小对首次输入创建尽可能多的 hash 表,有新的输入到来时,则将数据映射到相应的 hash 表中完成一次连接.接下来再进行下一轮迭代,产生下一个结果.

在云平台上执行多路连接查询时,首先根据相关三元组模式的统计数字用贪心策略选择连接变量,通常选择数目最小的,然后在剩下的三元组模式中将那些具有相同连接变量的三元组模式的相关数据全部装载,在一次分发回收(map-reduce)的迭代中实现一次多路连接.多次迭代后得到最终结果.

4.3 小 结

改进的静态查询处理方法在单结点上依赖复杂庞大的索引或多版本的方式实现对动态数据集的查询,当数据量进一步增加时,过重的空间负载将使得这些方法无法支持动态数据上的高效查询.与之相比,分布式存储和云存储上的 RDF 查询处理方法具有良好的可扩展性,因此可以适用于动态数据集.但这些方法仍处于初步研究阶段,并没有提出有效的支持频繁更新的策略.

5 查询优化

基于关系存储的 RDF 数据查询优化可以借助关系查询优化技术实现,但三元组的资源描述方式使 RDF 上的查询优化具有不同的特点.连接运算一直是查询中代价最高的部分,因此它也是查询优化中需要关注的重点;而 RDF 数据查询中通常含有大量的连接操作,因此 RDF 的查询优化处理研究也主要集中在连接运算的优化上.

5.1 裁剪搜索空间

5.1.1 基于连接变量的剪枝技术

根据查询三元组模式中的常量(已知值)可以对查询空间进行初步裁剪,例如,形如〈?*x* rdf:type: movie〉的查询可以根据 type 和 movie 仅装载查询所需的数据,也可以根据查询所属的域或类信息在遍历查询图的过程中裁减掉一些与查询无关的数据[48].

其次,当三元组模式某个位置(如 *s*)上的变量 *v* 无法与某个值 *C* 绑定时,且该变量为连接变量,则查询中其他在相同位置出现该变量的三元组模式也将不能和 *C* 绑定,这样可以提前在搜索空间中裁剪掉所有在该位置有值 *C* 的相应数据.

当数据上有多种顺序的索引时,可以根据索引预先对连接变量求交集,裁剪掉无关的搜索空间,找到共有的变量绑定集合.例如文献[39]中,当一个查询在 *o* 和 *s* 的位置有连接变量?*x* 分别涉及到 BitMat1 和 BitMat2 索引,则可以对 BitMat1 和 BitMat2 求交集,在此基础上搜索变量绑定值.

5.1.2 旁路信息传递

旁路信息传递(sideways information passing,简称 SIP)是一种用在多线程的哈希连接和稠密查询计划中的优化方法[64],采用 SIP 方法时,首先计算查询计划中的部分结果,再将该结果传送到需要的地方,以此来对查询计划树进行剪枝,实现优化.

RDF-3X[11]中,执行计划被表示为操作符树,在编译和运行时使用 SIP 方法.编译时采用的方法是将某些连接属性值传送到其他操作符,运行时在整个查询块中动态地重写这些属性值.

如图 9 所示的操作符树中,包括两个 *s-s* 连接,假设谓词 *p=born-in-place* 的选择率较高,所以先扫描 *pos* 索引,找到 *p=born-in-place* 所对应的主语,并将这些主语可以传递给左边的连接中的两个 *ops* 扫描操作.借助传递过来的主语,就可以裁剪掉大量无用的索引空间,在小数据上执行 *ops* 扫描.

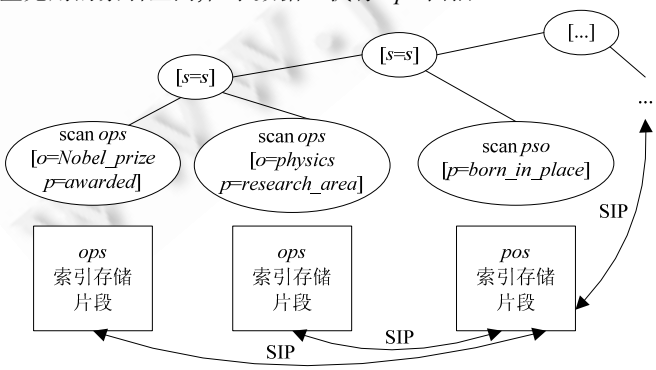


Fig.9 An example of SIP

图 9 一个 SIP 的例子

5.2 连接顺序优化

5.2.1 选择率估计

连接顺序优化需要精确的选择率估计.RDF 数据查询时的选择率估计是利用摘要统计数据,在概率框架(probabilistic framework,简称 PF)的基础上,通过启发式规则计算得到.选择率估计包括两种:三元组模式选择率估计和连接时的三元组模式选择率估计.文献[65]中给出了选择率估计的启发式规则和计算公式.

启发式规则一般包括:

1) 三元组的选择率根据 s,p,o 各部分的种类和没有被绑定的部分计算,且在多数 RDF 数据源中,用 $sel(\alpha)$ 表示 α 的选择率,则有 $sel(s)<sel(o)<sel(p)$;
2) 连接时的三元组选择率根据连接的数目和类型计算,一般来说,绑定的、连接的选择率高于没有绑定的连接.用 β-γ 来表示 β 与 γ 的连接,则有 $sel(s\text{-}s)<sel(s\text{-}o)$;$sel(s\text{-}s)<sel(o\text{-}o)$.

利用统计直方图、概率框架等可以给出选择率估计公式.

- 三元组模式(triple pattern)的选择率

三元组模式的选择率 $sel(t)$ 是一个在[0,1]区间内的实数,其计算公式为

$$sel(t)=sel(s)\times sel(p)\times sel(o) \tag{1}$$

非绑定的三元组模式组成元素(S 或 P 或 O)的选择率总是为 1.因为数据集中的所有三元组都可能匹配非绑定的三元组模式.

对于绑定的三元组中的组成元素,分别由以下公式计算:

$$sel(s)=\frac{1}{N_s} \tag{2}$$

其中,N_s 为 RDF 数据集中的资源总数.

$$sel(p)=\frac{T_p}{T} \tag{3}$$

其中,T_p 为匹配谓词 p 的三元组数目,T 为数据集中的三元组总数.

$$sel(o)=\begin{cases} sel(p,o_c), & \text{如果}p\text{是绑定的} \\ \sum_{p_i\in P} sel(p_i,o_c), & p\text{是非绑定的} \end{cases} \tag{4}$$

其中,(p,o_c) 代表直方图中 o_c 相关的 p 的数目,且 $sel(p,o_c)=\frac{f_c(p,o_c)}{T_p}$,即用匹配 p 的三元组的数目对 (p,o_c) 的频率进行规范化.当 p 是非绑定的,直方图中的每一个谓词都用来对 o 的选择率进行估计,即,此时的选择率等于每一个 p 上的 $sel(p_i,o_c)$ 之和.

- 连接的三元组模式(joined triple pattern)选择率

当给定连接的三元组绑定上界为 U_{tp} 时,连接的三元组 T 的选择率为

$$sel(T)=\frac{U_{tp}}{T^2} \tag{5}$$

其中,T^2 为 RDF 数据集中三元组总数的平方.当具体在 S 或 O 上进行连接时,需要对公式(5)乘一个选择率因子.例如,形如(?x p_1 o)的三元组与(?x p_2 ?y)连接时,其选择率应按如下公式计算:

$$sel(T)=\frac{U_{tp}}{T^2}\times sel(p_1,o_c) \tag{6}$$

文献[10]将连接分为 $s\text{-}o,s\text{-}p,s\text{-}s,p\text{-}o$ 等 9 种类型,根据二维索引,针对这 9 种连接建立统计直方图,利用直方图的统计数据估计连接时的选择率.利用直方图估计选择率的方法假设了谓词之间的独立性,但在实际的查询中,多个谓词之间通常通过主语($s\text{-}s$)或主语宾语($s\text{-}o$)的连接发生关联,因此,一些研究还提出了其他方法:文献[26]利用索引树结构中匹配结点之间的距离来估计连接时的选择率;RDF-3X 同时采用物化常用路径的方法计算连接时的选择率,首先利用图像挖掘技术预先计算 RDF 数据图中一些常用的路径,并保存其中一些路径上结

点的统计数字,这样,在查询中涉及到这些路径时可以直接利用预先保存的统计数字计算选择率.

5.2.2 连接顺序优化

当多个三元组模式连接时,根据估计的选择率可以对连接顺序进行优化,按照中间结果的大小选择最优执行计划.一般采用动态编程的方法枚举查询计划,包括自底向上(从一个简单的子查询开始扩充)和自顶向下(从整个查询开始向下处理)的方法,在每次迭代中,根据选择率估计中间结果代价,选择当前最优的子计划,直至得到最终执行计划[10].

5.3 其他优化方法

文献[66]利用启发式规则对星形查询和链式查询分别进行优化,还有一些文献中提出了物化中间结果[31]的优化方法,将常用的中间结果保存为物化视图,以此来提高查询效率.

6 总结及未来研究方向

作为一种 Web 上的资源描述框架,RDF 越来越多地应用在 Web 2.0、语义网等领域.Web 上出现了大量的 RDF 数据,特别是近年来对非结构化数据和大数据的研究,使得 RDF 数据的管理特别是查询处理成为一个研究热点.本文从数据集的形态(动态、静态)以及 RDF 数据组织存储方法两个维度,对目前已有的 RDF 查询处理技术进行了分析、对比,包括查询表述、查询处理技术和查询优化等方面.但限于篇幅,本文分析讨论的只是目前 RDF 数据查询研究的一部分,还有一些工作(如有些会议的 workshop 等)未能提及,这些工作都为 RDF 数据查询处理技术研究的发展做出了贡献.

结合分析可以得出,目前的研究工作虽然取得了一些成果,但其中仍然有一些问题值得探讨:

1) 已有的方法应对动态增长数据的可扩展性不足.目前的大多数方法都是针对静态的 RDF 数据集,在静态数据集的基础上通过构建大量索引和数据冗余等方法来提高查询效率.这些方法需要消耗大量存储空间,更新代价高,系统缺乏可扩展性.基于云平台的分布式查询处理方法可以用在动态数据集上,但目前的研究较少,技术也相对不够成熟,在本地结点上仍然是采用静态数据集上的查询技术.没有良好的可扩展性;
2) 现有的查询表达方式使普通用户表述查询困难.目前,主流的 RDF 查询处理技术主要围绕 SPARQL 查询语言.然而在 RDF 的很多实际应用中,SPARQL 语言语法复杂,需要用户拥有很强的专业背景,并且对知识库的语义结构(如谓词信息、前缀等)非常熟悉才能使用该查询语言.这极大地限制了 SPARQL 语言的应用范围,进而影响了 RDF 数据集的应用.在另一方面,关键词检索又面临查询语义歧义性大的问题.对大规模 RDF 数据集的查询,还需要探索更为合理的用户交互方式;
3) RDF 数据的应用扩展.尽管很多研究工作集中在 RDF 数据的查询处理上,然而仅仅为用户提供查询接口并不能充分发挥 RDF 数据集,尤其是作为语义网数据集所能起到的作用.不可否认,庞大的 RDF 数据集有很多潜在的应用领域.当前,我们还需要在 RDF 数据集的应用方面扩展思路、做很多工作.这方面还需要数据库、语义网、互联网等领域的研究人员们进行更多的交叉合作.

因此,未来的研究工作将包括:

1) 针对动态 RDF 数据集的查询处理技术的深入研究

在频繁更新的数据集上实现高效查询是亟待解决的问题.分布式查询处理具有良好的可扩展性,针对海量 RDF 数据的管理具有明显优势.但现有的分布式 RDF 查询处理技术的研究尚处于初步阶段,例如在对数据进行分布式存储,特别是按键值对的方式存储在云平台时,需要对数据划分,在此基础上,将查询分解到各个结点进行处理.按谓词分布 RDF 数据会带来极不均衡的问题,按图结构分解又可能导致语义损失,这些都会给查询处理带来影响.同时,如何减少连接查询时分发和回收的迭代次数,也是基于云平台的分布式查询处理技术中需要深入研究的难点问题.几种对静态数据集上查询方法的扩展虽然可以用在可更新数据上,但这些方法的可扩展性不高,因此,在分布式系统的基础上采用多版本等技术,或许是可行的方法.

目前的方法中,无论是静态数据集还是动态数据集上的查询方案,大都需要大量的索引机制和冗余的存储.

如何在大数据上建立轻量级的索引、减少数据冗余,在增强可扩展性的同时提供高效的查询处理,也是一个值得研究的问题.基于并行编程模型框架 BSP 计算模型下的图处理技术(比如 Pregel[67]),以其更高效的对复杂图迭代算法的支持,可能会成为云环境下海量 RDF 查询处理的一个重要研究方向.

2) 基于交互式浏览查询的信息发现技术研究

普通用户无法掌握复杂的 SPARQL 语言语法,而使用关键词查询时,若用户不了解数据的基本情况也无法提出有效的查询.总之,无模式的特点使用户构造查询十分困难.用户对 RDF 数据集信息的分布和结构不清楚,势必会影响其对 RDF 数据的使用.因此,提供交互式的浏览查询方式就显得极其迫切和重要.首先,可以通过 RDF 数据浏览使用户了解数据的内容、类别、域等基本信息;其次,利用交互式的方式帮助用户构建有效的查询,得到满意的查询结果;另外,可以通过交互式的方式对海量数据集进行在线的信息发现,使得在线浏览和 RDF 数据上的知识发现能够高效的结合起来.

目前,在 RDF 数据浏览方面也有一些研究工作[68],提出了 OpenLink[69],Faceted RDF Browser[70]等系统和工具.但这方面的研究工作还处于初步阶段,缺乏通用的模型和系统,没有良好的数据呈现方式,普通用户很难正确使用.因此,需要给出组织良好的浏览方式,在 RDF 数据的交互式查询方面也需要更多研究工作的支持.

3) 针对 RDF 数据的应用扩展

文献[5]作为语义网发展的一个标志性成果(一个庞大的 RDF 数据集),其规模还在迅速不断地扩展.除了为应付 RDF 大数据的挑战而需要深入研究的动态 RDF 数据集查询处理的相关研究外,人们还应关注如何更好地将大规模的 RDF 数据集应用到一些实际应用中,以更好地发挥其价值.在这方面,互联网搜索引擎诸如 Google, Bing 等早已开展了很多在语义网数据基础上的语义搜索研究[71].另外,RDF 数据也可以用来标注非结构化的文本信息,建立非结构化数据与结构化的知识库之间的联系[72],进而可能帮助我们解决诸如非结构化数据管理方面的问题.

References:

[1] RDF model and syntax specification. 1999. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

[2] Carroll JJ, Dickinson I, Dollin C, Reynolds D, Seaborne A, Wilkinson K. Jena: Implementing the semantic Web recommendations. In: Feldman SI, ed. Proc. of the 13th Int'l World Wide Web Conf. on Alternate Track Papers & Posters (WWW 2004). New York: ACM Press, 2004. 74−83. [doi: 10.1145/1013367.1013381]

[3] RDF concepts and abstract syntax. http://www.w3.org/TR/rdf-concepts/

[4] IBM smart planet. http://www.ibm.com/developerworks/cn/web/wa-aj-smartweb/index.html

[5] Linking open data. http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

[6] SPARQL. http://www.w3.org/TR/rdf-sparql-query/

[7] He H, Wang HX, Yang J, Yu PS. Blinks: Ranked keyword searches on graphs. In: Chan CY, ed. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2007). Beijing: ACM Press, 2007. 305−316.

[8] Tran T, Wang HF, Rudolph S, Cimiano P. Top-*k* exploration of query candidates for efficient keyword search on graph-shaped (RDF) data. In: Ioannidis YE, ed. Proc. of the 25th Int'l Conf. on Data Engineering (ICDE 2009). Shanghai: IEEE, 2009. 405−416. [doi: 10.1109/ICDE.2009.119]

[9] Murray C, Alexander N, Das S, Eadon G, Ravada S. Oracle spatial resource description framework (RDF). 10g Release2, 2005.

[10] Neumann T, Weikum G. RDF-3X: A risc-style engine for RDF. Proc. of the VLDB Endowment, 2008,1(1):647−659.

[11] Neumann T, Weikum G. Scalable join processing on very large RDF graphs. In: Çetintemel U, ed. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2009). Providence: ACM Press, 2009. 627−640. [doi: 10.1145/1559845.1559911]

[12] Seaborne A. RDQL—A query language for RDF. 2004. http://www.w3.org/Submission/2004/SUBM-RDQL-20040109

[13] Miller L, Seaborne A, Reggiori A. Three implementations of SquishQL, a simple RDF query language. In: Horrocks I, ed. Proc. of the 1st Int'l Semantic Web Conf. (ISWC 2002). LNCS 2342, Sardinia: Springer-Verlag, 2002. 423−435. [doi: 10.1007/3-540-48005-6_36]

[14] Sintek M, Decker S. TRIPLE—An RDF query, inference and transformation language. In: Yoshie O, ed. Proc. of the 14th Int'l Conf. on Applications of Prolog (INAP 2001). Tokyo: DDLP Press, 2001. 47−56.

[15] Fikes R, Hayes P, Horrocks I. DQL—A query language for the semantic Web. In: Raedt LD, ed. Proc. of the 12th Int'l World Wide Web Conf. (WWW 2003). Budapest: ACM Press, 2003. 324−335.

[16] Li HY, Qu YZ. A Keyword query approach on RDF data. Journal of Southeast University (Natural Science Edition), 2012,40(2):270−274 (in Chinese with English abstract).

[17] Li GL, Ooi BC, Feng JH, Wang JY, Zhou LZ. EASE: An effective 3 in 1 keyword search method for unstructured, Semi-Structured and structured data. In: Tsong J, ed. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2008). Vancouver: ACM Press, 2008. 903−914.

[18] Wu G, Tang J, Li JZ, Wang KH. Fine-Grained semantic web retrieval. Journal of Tsinghua University (Sci & Tech), 2005,45(9): 1865−1872 (in Chinese with English abstract).

[19] Li HY, Qu YZ. KREAG: Keyword query approach over RDF data based on entity-triple association graph. Chinese Journal of Computers, 2011,34(5):825−835 (in Chinese with English abstract).

[20] Elbassuoni S. Effective searching of RDF knowledge bases [Ph.D. Thesis]. Munchen: Max-Planck-Institut für Informatik, 2011.

[21] Tian X, Du XY, Li HH. Computing term-concept association in semantic-based query expansion. Ruan Jian Xue Bao/Journal of Software, 2008,19(8):2043−2053 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/2043.htm [doi: 10.3724/SP.J.1001.2008.02043]

[22] Tran T, Cimiano P, Rudolph S, Studer R. Ontology-Based interpretation of keywords for semantic search. In: Aberer K, ed. Proc. of the Semantic Web, the 6th Int'l Semantic Web Conf., the 2nd Asian Semantic Web Conf. (ISWC 2007). LNCS 4825, Busan: Springer-Verlag, 2007. 523−536. [doi: 10.1007/978-3-540-76298-0_38]

[23] Zhou Q, Wang G, Xiong M, Wang HF, Yu Y. SPARK: Adapting keyword query to semantic search. In: Aberer K, ed. Proc. of the Semantic Web, 6th Int'l Semantic Web Conf., 2nd Asian Semantic Web Conf. (ISWC 2007). LNCS 4825, Busan: Springer-Verlag, 2007. 649−707. [doi: 10.1007/978-3-540-76298-0_50]

[24] Lei YG, Uren V, Motta E. Semsearch: A search engine for the semantic Web. In: Staab S, ed. Proc. of the 15th Int'l Conf. on Managing Knowledge in a World of Networks (EKAW 2006). LNCS 4248, Podebrady: Springer-Verlag, 2006. 238−245. [doi: 10.1007/11891451_22]

[25] Wang HF, Zhang K, Liu QL, Tran T, Yu Y. Q2Semantic: A light weight keyword interface to semantic search. In: Bechhofer S, ed. Proc. of the 5th European Semantic Web Conf. (ESWC 2008). LNCS 5021, Tenerife: Springer-Verlag, 2008. 584−598. [doi: 10.1007/978-3-540-68234-9_43]

[26] Zenz G, Zhou X, Minack E, Siberski W, Nejdl W. From keywords to semantic queries—Incremental query construction on the semantic Web. Journal of Web Semantics, 2009,7(3):166−176. [doi: 10.1016/j.websem.2009.07.005]

[27] Harris S, Gibbins N. 3store: Efficient bulk RDF storage. In: Volz R, ed. Proc. of the 1st Int'l Workshop on Practical and Scalable Semantic Systems. Sanibel Island: CEUR-WS Press, 2003.

[28] Broekstra J, Kampman A, van Harmelen F. Sesame: A generic architecture for storing and querying RDF and RDF schema. In: James I, ed. Proc. of the 1st Int'l Semantic Web Conf. (ISWC 2002). LNCS 2342, Sardinia: Springer-Verlag, 2002. 54−68. [doi: 10.1007/3-540-48005-6_7]

[29] Lu J, Ma L, Zhang L, Brunner JS, Wang C, Pan Y, Yu Y. SOR: A practical system for ontology storage, reasoning and search. In: Koch C, eds. Proc. of the 33rd Int'l Conf. on Very Large Data Bases (VLDB 2007). Austria: ACM Press, 2007. 1402−1405.

[30] Chu E, Baid A, Chen T, Doan AH, Naughton J. A relational approach to incrementally extracting and querying structure in unstructured data. In: Koch C, ed. Proc. of the 33rd Int'l Conf. on Very Large Data Bases (VLDB 2007). Austria: ACM Press, 2007. 1045−1056.

[31] Abadi DJ, Marcus A, Madden SR, Hollenbach K. Scalable semantic Web data management using vertical partitioning. In: Koch C, ed. Proc. of the 33rd Int'l Conf. on Very Large Data Bases (VLDB 2007). Austria: ACM Press, 2007. 411−422.

[32] Abadi DJ, Marcus A, Madden SR, Hollenbach K. Sw-Store: A vertically partitioned dbms for semantic Web data management. The VLDB Journal, 2009,18(2):385−406. [doi: 10.1007/s00778-008-0125-y]

[33] Li M, Du XY, Wang S. Study on architecture of the ontology repository management system in semantic web. Journal of Computer Research and Development, 2006,43(Suppl.):39−45 (in Chinese with English abstract).

[34] Sidirourgos L, Goncalves R, Kersten M, Nes N, Manegold S. Column-Store support for rdf data management: Not all swans are white. Proc. of the VLDB Endowments, 2008,1(2):1553−1563.

[35] Wilkinson K, Sayers C, Kuno H, Reynolds D. Efficient RDF storage and retrieval in jena2. In: Cruz LF, ed. Proc. of the 1st Int'l Workshop on Semantic Web and Databases (SWDB 2003). Humboldt-Universität: Morgan Kaufmann Publishers, 2003. 131−150.

[36] Chong EI, Das S, Eadon G, Srinivasan J. An efficient SQL-based RDF querying scheme. In: Böhm K, ed. Proc. of the 31st Int'l Conf. on Very Large Data Bases (VLDB 2005). Trondheim: ACM Press, 2005. 1216−1227.

[37] Sintek M, Kiesel M. RDFBroker: A signature-based high-performance RDF store. In: Sure Y, ed. Proc. of the 3rd European Semantic Web Conf. (ESWC 2006). LNCS 4011, Budva: Springer-Verlag, 2006. 363−377. [doi: 10.1007/11762256_28]

[38] Weiss C, Karras P, Bernstein A. Hexastore: Sextuple indexing for semantic Web data management. Proc. of the VLDB Endowments, 2008,1(1):1008−1019.

[39] Atre M, Srinivasan J, Hendler J. BitMat: A main-memory bit matrix of RDF triples for conjunctive triple pattern queries. In: Bizer C, ed. Proc. of the Poster and Demonstration Session at the 7th Int'l Semantic Web Conf. (ISWC 2008). Karlsruhe: CEUR-WS.org, 2008.

[40] Matono A, Pahlevi SM, Kojima I. RDFCube: A P2P-based three-dimensional index for structural joins on distributed triple stores. In: Moro G, ed. Proc. of the Int'l Workshops on Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P 2005/2006). LNCS 4125, Trondheim, Seoul: Springer-Verlag, 2007. 323−330. [doi: 10.1007/978-3-540-71661-7_31]

[41] Zou L, Mo JH, Chen L, Özsu MT, Zhao DY. gStore: Answering SPARQL queries via subgraph matching. Proc. of the VLDB Endowment, 2010,4(1):482−493.

[42] Bönström V, Hinze A, Schweppe H. Storing RDF as a graph. In: Teixeira C, ed. Proc. of the 1st Latin American Web Congress (LA-WEB 2003). Sanitago: IEEE Computer Society, 2003. 27−36.

[43] Haye J, Gutiërrez C. Bipartite graphs as intermediate model for RDF. In: McIlraith SA, ed. Proc. of the 3rd Int'l Semantic Web Conf. (ISWC 2004). LNCS 3298, Hiroshima: Springer-Verlag, 2004. 47−61. [doi: 10.1007/978-3-540-30475-3_5]

[44] Udrea O, Pugliese A, Subrahmanian VS. Grin: A graph based RDF index. In: Holte RC, ed. Proc. of the 23rd AAAI Conf. on Artificial Intelligence (AAAI 2007). Vancouver: AAAI Press, 2007. 1465−1470.

[45] Tran T, Ladwig G. Structure index for RDF data. In: Tan KL, ed. Proc. of the Workshop on Semantic Data Management, 36th Int'l Conf. on Very Large Data Bases (SemData@VLDB 2010). Singapore: CEUR-WS Press, 2010. 20−25.

[46] Chen YJ. Signature files and signature trees. Information Processing Letters, 2002,82(1):213−221. [doi: 10.1016/S0020-0190(01)00266-6]

[47] Janik M, Kochut K. BRAHMS: A WorkBench RDF store and high performance memory system for semantic association discovery. In: Gil Y, ed. Proc. of the 4th Int'l Semantic Web Conf. (ISWC 2005). LNCS 3729, Galway: Springer-Verlag, 2005. 431−445. [doi: 10.1007/11574620_32]

[48] Kuldeep BR, Kuma PS. Efficient approximate SPARQL querying of Web of linked data. In: Bobillo F, ed. Proc. of the 6th Int'l Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2010). CEUR Workshop Proceedings 654. Shanghai: CEUR-WS Press, 2010. 37−48.

[49] Alexaki S, Christophides V, Karvounarakis G, Plexousakis D, Tolle K. The ICS-FORTH RDFsuite: Managing voluminous RDF description bases. In: Decker S, ed. Proc. of the 2nd Int'l Workshop on the Semantic Web (SemWeb 2001). Hong Kong: CEUR-WS Press, 2011. 40.

[50] Angles R, Gutierrez C. Querying RDF data from a graph database perspective. In: Asunción GP, ed. Proc. of the Semantic Web: Research and Applications, 2nd European Semantic Web Conf. (ESWC 2005). LNCS 3532, Heraklion, C: Springer-Verlag, 2005. 346−360. [doi: 10.1007/11431053_24]

[51] Newman A, Hunter J, Li YF, Bouton C, Davis M. A scale-out RDF molecule store for distributed processing of biomedical data. In: Huai JP, ed. Proc. of the 17th Int'l Conf. on World Wide Web (WWW 2008). Beijing: ACM Press, 2008.

[52] Neumann T, Weikum G. X-RDF-3X: Fast querying, high update rates, and consistency for RDF databases. Proc. of the VLDB Endowment, 2010,1(1):256−263.

[53] Pu X, Wang JY, Luo P, Wang M. AWETO: Efficient incremental update and querying in rdf storage system. In: Macdonald C, ed. Proc. of the 20th ACM Conf. on Information and Knowledge Management (CIKM 2011). Glasgow: ACM Press, 2011. 2445−2448. [doi: 10.1145/2063576.2063988]

[54] Stuckenschmidt H, Vdovjak R, Houben GJ, Broekstra J. Index structures and algorithms for querying distributed RDF repositories. In: Feldman SI, ed. Proc. of the 13th Int'l Conf. on World Wide Web (WWW 2004). New York: ACM Press, 2004. 631−639. [doi: 10.1145/988672.988758]

[55] Harth A, Umbrich J, Hogan A, Decker S. Yars2: A federated repository for querying graph structured data from the Web. In: Aberer K, ed. Proc. of the 6th Int'l Semantic Web Conf., 2nd Asian Semantic Web Conf. (ISWC 2007+ASWC 2007). LNCS 4825, Busan: Springer-Verlag, 2007. 211−224. [doi: 10.1007/978-3-540-76298-0_16]

[56] Huang JW, Abadi DJ, Ren K. Scalable SPARQL querying of large RDF graphs. Proc. of the VLDB Endowment, 2011,4(11):1123−1134.

[57] Tran T, Wang HF, Haase P. Search Web DB: Data Web search on a pay-as-you-go integration infrastructure. Journal of Web Semantics, 2009,7(1):189−203. [doi: 10.1016/j.websem.2009.07.001]

[58] Guéret C, Kotoulas S, Groth PT. TripleCloud: An infrastructure for exploratory querying over Web-scale RDF data. In: Hübner FJ, ed. Proc. of the 2011 IEEE/WIC/ACM Int'l Joint Conf. on Web Intelligence and Intelligent Agent Technology—Workshops (WI-IAT 2011). Lyon: IEEE Computer Society, 2011. 245−248. [doi: 10.1109/WI-IAT.2011.166]

[59] Ruckhaus E, Ruiz E, Vidal ME. OnEQL: An ontology efficient query language engine for the semantic Web. In: Polleres A, ed. Proc. of the Workshop on Applications of Logic Programming to the Web (ICLP 2007), Semantic Web and Semantic Web Services (ALPSWS 2007). Porto: CEUR-WS Press, 2007.

[60] Husain MF, McGlothlin J, Masud MM, Khan LR, Thuraisingham B. Heuristics-Based query processing for large RDF graphs using cloud computing. IEEE Trans. on Knowledge and Data Engineering, 2011,23(9):1312−1327. [doi: 10.1109/TKDE.2011.103]

[61] Kotoulas S, Urbani J. SPARQL query answering on a sharednothing architecture workshop on semantic data management. In: Tan KL, ed. Proc. of the Workshop on Semantic Data Management, 36th Int'l Conf. on Very Large Data Bases (SemData@VLDB 2010). Singapore: CEUR-WS Press, 2010.

[62] Kim HS, Ravindra P, Anyanwu K. From SPARQL to MapReduce: The journey using a nested TripleGroup algebra. Proc. of the VLDB Endowment, 2011,4(12):1426−1429.

[63] Myung J, Yeon J, Lee SG. SPARQL basic graph pattern processing with iterative MapReduce. In: Rappa M, ed. Proc. of the Int'l Workshop on Massive Data Analytics over the Cloud (MDAC 2010). Raleigh: ACM Press, 2010. [doi: 10.1145/1779599.1779605]

[64] Ives ZG, Taylor NE. Sideways information passing for push-style query processing. In: Alonso G, ed. Proc. of the 24th Int'l Conf. on Data Engineering (ICDE 2008). Cancún: IEEE Press, 2008. 774−783. [doi: 10.1109/ICDE.2008.4497486]

[65] Stocker M, Seaborne A, Bernstein A, Kiefer C, Reynoldsy D. SPARQL basic graph pattern optimization using selectivity estimation. In: Huai JP, ed. Proc. of the 17th Int'l Conf. on World Wide Web (WWW 2008). Beijing: ACM Press, 2008. 595−604. [doi: 10.1145/1367497.1367578]

[66] Lü B, Du XY, Wang Y. SPARQL query optimization based on property correlations. Journal of Computer Research and Development, 2009,46(suppl.):119−125 (in Chinese with English abstract).

[67] Malewicz G, Austern MH, Bik ATC, Dehnert JC, Horn I, Leiser N, Czajkowski G. Pregel: A system for large-scale graph processing. In: Elmagarmid AK, ed. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2010). Indianapolis: ACM Press, 2010. 135−146. [doi: 10.1145/1807167.1807184]

[68] Wu HH, Qu YZ. Browsing RDF data: State of art survey. Computer Science, 2009,36(2):5−10 (in Chinese with English abstract).

[69] Open link. http://demo.openlinksw.corn/DAV/JS/rdfbrowser/index.html

[70] Oren E, Delbru R, Decker S. Extending faceted navigation for RDF data. In: Cruz IF, ed. Proc. of the 5th Int'l Semantic Web Conf. (ISWC 2006). LNCS 4273, Athens: Springer-Verlag, 2006. 559−572. [doi: 10.1007/11926078_40]

[71] Weikum G. Semantic search: From names and phrases to entities and relations. In: Brambilla M, ed. Proc. of the 2nd Int'l Workshop on Searching and Integrating New Web Data Sources. Istanbul: CEUR-WS Press, 2012. 3.

[72] Shen W, Wang JY, Luo P, Wang M. LINDEN: Linking named entities with knowledge base via semantic knowledge. In: Mille A, ed. Proc. of the 21st World Wide Web Conf. (WWW 2012). Lyon: ACM Press, 2012. 449−458. [doi: 10.1145/2187836.2187898]

附中文参考文献:

[16] 李慧颖,瞿裕忠.基于关键词的 RDF 数据查询方法.东南大学学报(自然科学版),2012,40(2):270−274.

[18] 吴刚,唐杰,李涓子,王克宏.细粒度语义网检索.清华大学学报(自然科学版),2005,45(9):1865−1872.

[19] 李慧颖,瞿裕忠.KERAG:基于实体三元组关联图的 RDF 数据关键词查询方法.计算机学报,2011,34(5):825−835.

[21] 田萱,杜小勇,李海华.语义查询扩展中词语概念相关度的计算.软件学报,2008,19(8):2043−2053. http://www.jos.org.cn/1000-9825/19/2043.htm [doi: 10.3724/SP.J.1001.2008.02043]

[33] 李曼,杜小勇,王珊.语义 Web 环境中本体库管理系统体系结构研究.计算机研究与发展,2006,43(增刊):39−45.

[66] 吕彬,杜小勇,王琰.基于属性相关性的 SPARQL 查询优化方法.计算机研究与发展,2009,46(增刊):119−125.

[68] 吴鸿汉,瞿裕忠. RDF 数据浏览的研究综述.计算机科学,2009,36(2):5−10.

杜方(1974－),女,宁夏银川人,博士生,CCF 会员,主要研究领域为智能信息检索,大数据管理.

E-mail: dfang@ruc.edu.cn

杜小勇(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为智能信息检索,高性能数据库,知识工程.

E-mail: duyong@ruc.edu.cn

陈跃国(1978－),男,博士,副教授,CCF 会员,主要研究领域为语义搜索,大数据管理.

E-mail: chenyueguo@ruc.edu.cn