ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.11, November 2009, pp.2950−2964
doi: 10.3724/SP.J.1001.2009.03678

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

语义Web数据管理研究进展*

杜小勇[1,2+], 王 珊[1,2], 吕 彬[1,2]

[1](教育部数据工程与知识工程重点实验室,北京 100872)

[2](中国人民大学 信息学院,北京 100872)

Research and Development on Semantic Web Data Management

DU Xiao-Yong[1,2+], WANG Yan[1,2], LÜ Bin[1,2]

[1](Key Laboratory of Data Engineering and Knowledge Engineering, Ministry of Education, Beijing 100872, China)

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

+ Corresponding author: E-mail: duyong@ruc.edu.cn

Du XY, Wang Y, Lü B. Research and development on semantic Web data management. *Journal of Software*, 2009,20(11):2950−2964. http://www.jos.org.cn/1000-9825/3678.htm

Abstract: This paper describes the current state of the art of RDF (resource description framework) data management from 4 aspects, storage organization, query processing and optimization, prototypes as well as benchmarks. It also points out some future research topics.

Key words: semantic Web; RDF data; data management; index; storage organization; query optimization

摘 要: 从数据存储组织、查询优化和处理、原型系统和基准测试等方面介绍当前RDF数据管理的研究进展和比较分析,讨论存在的问题并给出未来的研究方向.

关键词: 语义Web;RDF数据;数据管理;索引;存储组织;查询优化

中图法分类号: TP311 **文献标识码**: A

语义Web是Tim Berners-Lee于 2000 年提出的关于下一代互联网的设想[1].它为人们描述了一个美好的远景,通过为Web上所有种类的数据引入清晰的语义和结构化描述,使得计算机可以理解Web上的资源,实现计算机之间基于语义的信息交换.由于这符合人们对互联网的期望,语义Web的建设开始成为研究的热点.目前,自然语言处理的研究尚未达到可大规模实际使用的程度,为了使机器能够正确理解语义信息,最可行的办法就是通过制订标准,对资源的含义进行统一、形式化的描述.

在Tim Berners-Lee关于语义Web的体系结构框架中[2],RDF(resource description framework)是W3C(World Wide Web Consortium,详见http://www.w3c.org/)提出的描述Web信息的通用语言.它借鉴了语义网络[3]的知识表示方法,采用三元组(*s*,*p*,*o*)来描述Web信息,其中,*s*代表主语(subject),*p*代表谓语(predicate),*o*代表宾语(object).由于三元组为主语和宾语建立了语义关联,数据因此而增加了语义信息.例如,http://example.org/foaf/person1 所对应的人的名字是“Alex”这样一条信息,可以用三元组(〈http://example.org/foaf/person1〉,〈http://example.org/

* Supported by the National Natural Science Foundation of China under Grant Nos.60873017, 60573092, 60496325 (国家自然科学基金)

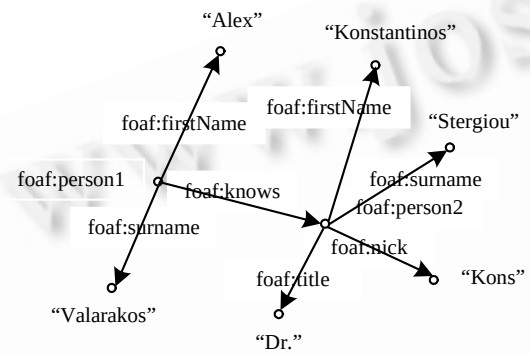
Received 2009-01-19; Revised 2009-05-21; Accepted 2009-07-09

foaf/firstName〉,“Alex”)来表示.其中〈http://example.org/foaf/firstName〉是谓语,表示“名字”,使用URI作为标识符.“Alex”是宾语,是〈http://example.org/ foaf/person1〉在〈http://example.org/foaf/firstName〉这个属性上的取值.如果将三元组中的主语和宾语映射为图上的节点,谓语即连接两节点的边的标记.这样,语义Web数据就被抽象为有向标记图,也称为RDF图.图 1(a)是RDF图的一个例子,描述的是两个人和他们之间的关系.

W3C还为RDF数据设计了查询语言,先后提出了OWL-QL[4],RQL[5],RDQL[6],SPARQL[7]等语言.其中,SPARQL是目前最为流行的查询语言.SPARQL有 4 种查询方式,分别为SELECT,CONSTRUCT,DESCRIBE和ASK.其中,SELECT是最为常见的查询方式,类似于SQL中的SELECT,它返回满足条件的变量值.CONSTRUCT是另一种查询方式,它可以根据查询结果在满足条件的变量值之间建立关联构造新的RDF三元组.DESCRIBE返回与满足条件的变量相关的所有三元组.ASK用来测试查询结果是否为空集.与RDF数据一样,SPARQL查询也可以用图形方式表示.另外,SPARQL提供了OPTIONAL操作符,该操作符类似于关系代数中的左外连接,其结合顺序是左优先的,不满足结合律.

由于 SELECT 查询方式是最为典型的 SPARQL 查询方式,我们举例说明如下:

例 1:在图 1(a)表示的一组 RDF 数据中(此处删去了某些关于命名空间的定义),要求查找名字为“Konstantinos”的人的姓.图 1(b)是该查询的 SPARQL 表达式.图 1(c)是该查询的结果,



(a) An example of RDF data set

(a) RDF 数据集的例子

```
SELECT ?surname
WHERE {?x foaf:firstName “Konstantinos”.
       ?x foaf:surname ?surname
      }
```

(b) An example of SPARQL query

(b) SPARQL 查询的例子

Surname
“Stergiou”

(c) Query result

(c) 查询结果

Fig.1 An example of RDF data set and SPARQL query

图 1 一个 RDF 数据集和 SPARQL 查询的例子

在传统关系数据库中存储语义Web数据,存在一个难题,即语义Web比传统关系数据库更为灵活.传统的数据库建模采用的是自顶向下的方法,先分析应用中的数据需求,了解数据之间的关系,而后定义数据的结构模式,然后再根据已构建好的模式创建数据库表,往表中插入数据.建表操作和数据修改操作分别由数据定义语言(DDL)和数据操作语言(DML)来完成.在传统的应用(如商业数据处理)中,应用逻辑相对稳定,模式变化较少,采用模式和数据分开处理的方法能够有效提高数据处理的速度.而在语义 Web 这种开放环境下,不同用户有不同的需求,当加入一个新用户和新数据源时,模式信息就可能发生改变.系统管理员无法及时将新加入数据的模式信息融合到原有模式中,需要系统抽取模式信息,并与原有模式相融合,最终形成一个满足需求的新模式.这种模式不稳定的特点使得再采用自顶向下的预先建模的方式变得不可行,因此基于数据模式相对稳定的假设而设计的传统数据库管理系统已不能适应 RDF 建模的应用,需要开发一个高效的 RDF 数据管理系统,作为语义Web 的数据管理平台.

本文对现有的语义 Web 数据管理工作进行总结和分析,为读者提供进一步研究的方向和基础.除了事务系统和安全控制外,存储组织和查询处理与查询优化是数据管理系统的最主要的任务,而目前绝大多数语义 Web 数据管理采用关系数据库作为底层的存储器,这样可以最大程度地享受关系数据库具有强大事务管理能力的

优点.因此,本文从存储结构、查询处理与优化这两方面对 RDF 数据管理问题进行综述,并介绍原型系统和基准测试集这两方面的研究进展,分别对应第 1 节~第 4 节,第 5 节讨论目前语义 Web 数据管理研究存在的问题以及未来的发展方向.

1 存储组织

数据存储组织是数据管理的基本问题,是指数据逻辑上和物理上如何在存储设备上进行组织.由于目前的主流是将关系数据库作为底层存储,因此,本文重点放在数据的逻辑组织方面.现有的方案可分为三元组表存储方案、垂直存储方案、水平存储方案和模式生成方案.首先分别介绍这几种方案,然后讨论实现上的考虑.

1.1 三元组表

三元组表(triple)是最直接的存储组织方案,很多语义Web数据管理系统都采用该方案.其思想很简单,由于RDF数据就是由三元组组成的,可以直接存储在一张由 3 列构成的表中,分别对应三元组的主语、谓语、宾语,如图 2 所示.这种方案简洁、明了且易于在关系数据库上实现,但存在查询处理效率低的问题.这是因为所有的数据都存储在一张表中,那么一个SPARQL查询将会被转换成一个存在大量自连接的SQL查询,在表很大的情况下,随着自连接次数的增加,查询时间也会随之显著增长.由于这种三元组表存储组织方式比较简单,初期的很多语义Web系统均采用该方案,如Sesame[8],Jena[9],CODE[10],YARS[11]等,但是,随着系统中数据量的增加,这些系统都存在查询效率低的问题.有些系统对存储作了一些优化,把一些描述模式信息的三元组抽取出来单独存放,但本质上并没有改变三元组表过于庞大的局面.为了提高查询效率,人们不得不在三元组表下增加大量的索引.

Triple table

Subject (URI)	Predicate (property)	Object (value)

Fig.2 Triple table

图 2 三元组表

文献[12]提出一个基于 P2P 的语义 Web 存储方案,通过在主语、谓语、宾语上分别建立哈希索引的方式提高查询效率.在分布式环境中,建立复杂索引的代价较高,因此仅考虑建立这 3 个简单索引.文献[13]提出了另一个分散存储的系统 YARS2.为了提高连接查询的效率,它将索引从一维扩展到多维.另外,YARS2 为每条三元组增加了上下文信息(context),表示该三元组的出处.这样,三元组(*s*,*p*,*o*)被扩展成四元组(*s*,*p*,*o*,*c*).YARS2 于是增加了 6 个索引,分别是{*s*,*p*,*o*,*c*,*spoc*,*pocs*}.这里的 *s*,*p*,*o*,*c* 分别代表以主语、谓语、宾语、上下文信息为索引列而建立的一维索引;*spoc* 代表以主语 *s* 为第 1 顺序列,以谓语 *p* 为第 2 顺序列,以宾语 *o* 为第 3 顺序列,以上下文信息 *c* 为第 4 顺序列而建立的四维索引;*pocs* 与 *spoc* 的不同在于列顺序的调换.虽然,实验证明增加索引确实对查询效率有提高,但这两篇文章更侧重于解决方案的研究,并没有深入权衡增加索引对查询效率的提高和对存储开销的增加这两方面的影响.文献[14]在三元组表存储的基础上增加了{*spo*,*sop*,*pos*,*pso*,*osp*,*ops*}这 6 个索引,通过实验表明在增加了索引的条件下,三元组表存储方案的查询效率不低于将在第 1.2 节中提到的垂直存储方案.但是,对于增加索引而带来的存储开销,该文也没有作进一步讨论.文献[15]使用与文献[14]相同的 6 个索引,它引入压缩机制将索引的存储开销减少到未压缩情形的 1/3.换句话说,当增加 6 个索引后,在未压缩情形下,存储开销应扩展为原始存储开销的 6 倍,通过引入压缩机制,存储开销仅增长了 1 倍.通过在 3 个数据集上的实验,该文证明了查询效率要比无索引状态和垂直存储状态下提高多个数量级.

1.2 垂直存储方案

垂直存储方案是在三元组表存储基础上的一种优化.它根据谓语对三元组表进行划分,将拥有相同谓语的三元组存储到同一张表中.由于该表的谓语都相同,因此可以把谓语列去掉,仅保留两列,分别对应主语和宾语,如图 3 所示.例如CODERS[16]就是采用垂直存储方案.

文献[17]讨论了采用列存储数据库来实现垂直存储方案.列存储数据库以列为单位存储数据,对于一个多属性的模式,在行存储数据库中体现为一张由多列组成的表,而在列存储数据库中体现为多张一列的表.在列存储数据库中,为了将相同实例的信息(行存储数据库中称为元组)串起来,需要在每张表中为每个实例留下位置,即使某个实例在某个属性上的取值为空值,在对应表中也需为其保留一行位置,因为列存储数据库是通过行位置信息将实例串起来的.在用列存储数据库实现垂直存储方案时,需要作一些变换.由于垂直方案中每张表仅有两列,且第 1 列信息是主语,每个主语即一个资源 id,因此,可以将所有表的第 1 列抽取出来形成一张资源 id 表,保留第 2 列信息.这样,就实现了将垂直存储转换为列存储的过程,图 4 由图 3 的垂直存储方案转化而来.实验结果表明,基于列存储的RDF数据存储方案能够明显改善查询效率.由于列存储中通过表间的行位置信息作连接,每个资源在每张表上都必须有对应的行,因此,会存在大量的空值.文献[18]提出了 3 个空值压缩方案,根据空值比例的不同,采用相应的压缩方案解决空值过多的问题.压缩方案的实现,使得列存储数据库上RDF数据的查询效率有多个数量级的提升.

	FirstName
1	Alexandros
2	Konstantinos
3	Vaggelis
4	John
5	Konstantinos

	Surname
1	Valarakos
2	Pazalos
3	Kourakos
4	Partsakoulakis
5	Stergiou

	Gender
1	Male
2	Male
3	Male

	Title
1	Mr.
2	Mr.
5	Dr.

1	5

Fig.3 Vertical partitioning

图 3 垂直分区

ID	FirstName	Surname	Gender	Title	Knows
1	Alexandros	Valarakos	Male	Mr.	5
2	Konstantinos	Pazalos	Male	Mr.	null
3	Vaggelis	Kourakos	Male	null	null
4	John	Partsakoulakis	null	null	null
5	Konstantinos	Stergiou	null	Dr.	null

Fig.4 Column-Based database storage

图 4 列数据库存储

文献[19]重现了文献[17]的实验,确认了从文献[17]的列数据库存储实验中观察到的趋势,并且发现,对于指定的测试数据集,列数据库中垂直存储方案确实优于三元组表方案.但是存在以下问题:在经典的行存储关系数据库中无法证实垂直存储的方案优于三元组表方案,并且随着语义 Web 数据属性的增加,垂直存储方案存在可扩展性问题.

1.3 水平存储方案

从图4可以看出,如果将所有的列合并起来,就形成了一张表,它包含了所有属性的信息.因此,RDF数据用这张表存储就够了,这个方案称为水平存储方案.但是这种方案存在一个问题,即水平表中列的数量过多,且空值太多会造成表的稀疏,导致存储空间的浪费,这个问题在第 1.2 节已经提到.另外一个问题是,随着新属性的插入或删除,表中的列可能需要修改.在现有的关系数据库中,列的增删就是模式的改变,这是一操作的代价是昂贵的.目前在语义Web数据管理研究中尚未见到采用这种存储方案,而在Web数据存储的研究中,已有研究者采用这种方案存储Web数据(如WideTable[20]).由于语义Web数据是Web数据的一种扩展,因此从长远来看,这种方案也会成为一种选择.

文献[20]提出采用解释型存储格式解决水平存储存在的问题.解释型存储格式是将模式信息与数据信息混合存储的一种解决方案.在现有的数据库存储方案中,模式信息的存储与数据信息的存储截然分离.数据页中不存储模式信息,因此元组默认在所有属性上都会有值.如果元组的某个属性值为空,则需要在数据页中增加附加信息来表示该元组在第几个属性上的值为空.随着属性数量的增长,这类附加信息也就越来越多.由于在水平存储方案中,所有属性都存储在一张表中,因此这类附加信息会占用大量空间.极端的情况可能导致附加信息的存储开销大于实际数据的存储开销.在 Web 应用和语义 Web 应用中,由于数据的稀疏性,出现这种情况的可能性极大,所以,需要对存储进行必要的优化,而解释型存储是一种很好的方式.文献[20]提出的元组解释型存储方案在

数据页中存储模式信息,每条元组的头部存储该元组的 id、记录总长度.对于非空值,元组需要存储对应属性的标识符、属性值的长度和具体的属性值,如图 5 所示.该文还引入了稀疏索引的支持,即不将索引属性为空值的元组加入到索引中.通过这两项技术的使用,可以在水平存储方案中提高稀疏数据的存储和查询效率.

Schema information

Name	id	Type	Length
id	1	INT	4
firstName	10	VARCHAR(16)	16
surname	12	VARCHAR(16)	16
Gender	23	VARCHAR(16)	16
Title	25	VARCHAR(16)	16
Knows	34	INT	4

Traditional tuple

1	“Konstantinos”	“Stergiou”	null	“Dr.”	null

Interpreted tuple

1	23	10	12	“Konstantinos”	12	8	“Stergiou”	25	3	“Dr.”

Tuple id Tuple length Property id Property length Value

Fig.5 An example of interpreted tuple

图 5 元组解释型存储示例

1.4 模式生成方案

解释型存储虽然减少了水平存储方案的存储开销,但是仍然存在一些缺点.首先,为每条元组都增加解释信息势必会增加存储的开销.其次,由于所有数据都在一张表中,查询连接的开销较大.因此,研究人员提出应该根据 RDF 数据中空值的分布情况,由系统对水平表作混合切分(包括垂直切分和水平切分),形成多个表,使得切分后各表内的空值较少.通过这种切分,一方面可以降低空值的存储开销.另一方面,数据存储在多张表中,每张表中数据的减少又降低了查询连接的执行开销.这种方案称为模式生成方案(即自动生成一些表的模式),其中的每一张表称为属性表.

文献[21]描述了模式生成方案在 Jena 系统中的设计和实现,它将三元组信息转换成一些属性表.但是,模式生成是一件很繁琐的事情,如果完全交给 DBA 来执行,这种方案则是没有生命力的,应由系统自动生成.

1.5 分析与总结

上述 RDF 数据存储组织方案可以依据考虑的对象的不同来进行区分.三元组表存储方案,以三元组为考虑对象,故可称为基于三元组的存储,它直接将逻辑表示方式与物理存储方式等同起来,很容易在行存储数据库中实现,但是查询效率较低,需要构造大量索引.垂直存储方案根据谓语的值的不同对三元组作垂直划分,形成多个二元组表,故可称为基于属性值的存储.该方案可以节省存储空间,并很容易在传统的行存储数据库中实现,但是,实验结果表明,在行存储数据库中该方案并不一定比三元组表方案优越.将二元组表作进一步的垂直划分,将主语抽取出来,作为一个单独的列,这样就把二元组表转换为单列表,可以自然地用列存储数据库实现.在列存储数据库中进一步采用压缩、延迟物化等优化技术,对于某些测试集,能够极大地提高查询性能.然而,这种优化能否推广到其他测试集,仍有待探讨.将所有列合并到一张表就形成了水平存储方案,称为基于实例的存储.而模式生成方式是在水平存储方案的基础上,将属性相类似的实例合并在一起存储在物理单元中,称为基于实例集的存储.

根据存储对象的复杂性对 RDF 数据存储方案进行了分类,从基于属性值的存储、基于三元组的存储、基于实例的存储到基于实例集的存储.随着存储对象趋于复杂,数据内部的关联趋于紧密,查询性能随之得到提

高.为了在存储对象简单的存储方案中提高查询效率,就必须使用索引或压缩等优化技术,而过多地使用这些技术后又导致数据修改的效率降低,不适用于数据变化的场景.根据存储对象复杂程度和数据动态变化对性能影响的高低,本文对以上所列的解决方案进行总结(见表 1).

Table 1 Comparisons of storage solution

表 1 存储方案的比较

Effects of data changes	Storage unit			
	Value (vertical partitioning)	Triple (triple table)	Instance (horizontal table)	Instance set (schema designing)
Big	Refs.[17,18]	Refs.[12−15]	Ref.[20]	Ref.[21]
Small	Ref.[16]	Refs.[8−10]		

从表 1 来看,目前大量的工作还集中于存储对象较为简单的垂直存储方案和三元组表方案,而且在这两种方案中,更多的文章讨论如何通过增加索引和压缩提高查询效率,忽略了这些优化工作对修改的影响.从三元组表到垂直存储再到水平存储,可以看出,RDF 数据的模式设计工作陷入了一个困境.它们并没有直面模式融合这个难题,而是通过对描述对象的不断细分来回避这个问题,给人以隔靴搔痒的感觉.模式自动生成方案希望根据实例的特点自动生成模式,它直面模式融合带来的挑战,通过将解释型存储引入数据页面中以降低模式融合带来的开销,并且其对于数据变化有着较高的适应性.可以设想,当基于拆分的思路走到极致时,采用属性合并的方法自动生成模式,提高存储对象中数据的内聚性,更有助于提高查询的效率,适应人们对数据的理解和使用.因此,这种方案将是未来语义 Web 数据存储的发展方向,值得进一步深入研究.

2 查询处理和优化

SPARQL[7]是W3C提出的针对RDF数据的查询语言标准,并得到了广泛的认同.以下将分别介绍在查询语言、查询处理和查询优化方面的研究成果.

2.1 查询语言的扩展

在 SPARQL 提出后,关于语言本身的研究,除了从理论上弄清该语言的表达能力以外,主要工作还集中在如何实现和扩充 SPARQL 查询语言,使其能够更好地满足用户的需求.

(1) 语言的表达能力

为了能够利用关系数据库的存储和查询功能,需要将SPARQL映射为关系数据库所能识别的语法.由于关系代数与SPARQL的语法不同,研究者需要对SPARQL的表述能力和复杂性进行分析,研究二者之间的可转换性.文献[22]研究如何将SPARQL翻译成关系代数.由于SPARQL和SQL的语义不同,一些算子无法直接翻译.例如,由于OPTIONAL算子不满足结合律,在多重嵌套的OPTIONAL操作中,OPTIONAL无法直接翻译成关系代数中与之最相近的左外连接.文献[22]给出了将SPARQL主要部分翻译成关系代数的方法.文献[23]将SPARQL的核心操作抽取出来,称为$SPARQL_C$,并分析了它的计算复杂度和形式语义基础.而后,讨论了在哪种SPARQL查询中,多重嵌套的OPTIONAL算子与左外连接可相互转换,并将这样的查询称为精心设计的SPARQL查询.文献[23]从理论上确定了数据库支持的SPARQL查询的范围.文献[24]在文献[23]的基础上对$SPARQL_C$作了扩展,增加了对包操作语义的支持.由于演绎数据库界在逻辑表示和推理方面已经做了很长时间的工作,为了利用演绎数据库的成果,研究人员尝试将语义Web数据映射成演绎数据库的数据.Datalog是演绎数据库的查询语言,文献[25]从复杂性分析的角度出发提出了一个基于Datalog的SPARQL形式语义.文献[26]证明了$SPARQL_C$的表述能力等同于SPARQL,并证明了SPARQL的表述能力等同于带否定词的非递归安全Datalog.

(2) 语言的扩展

研究者们还试图对 SPARQL 的功能进行扩展.主要有两种扩展思路:

第一,从语义Web应用的需求出发扩展功能,称为面向语义Web应用的扩展.例如,导航功能是图数据库查询的基本特征,语义Web数据可抽象为图数据,因此也需要添加导航功能.文献[27]在SPARQL中增加支持导航功能

的嵌入式正则表达式,使得扩展后的SPARQL(称为nSPARQL)可以支持对RDFS**数据的查询.文献[28]是文献[27]工作的延伸,它对nSPARQL语言和内嵌正则表达式进行形式化分析,并证明了执行一个内嵌正则表达式的时间复杂度是$O(|G|\cdot|E|)$(其中,$|G|$和$|E|$分别表示RDF数据图上节点和边的数量),从而说明nSPARQL导航功能的可行性.nSPARQL能够支持对演绎规则的处理,这是SPARQL所做不到的.另外,文献[29]对SPARQL作了扩展,通过增加函数的方法,使得它能够查询满足OWL-DL规范的本体数据.

第二,从支持一般的数据库应用的需求出发进行的功能扩展,称为面向数据库应用的扩展.例如,关系数据库中的数据带有一些简单的语义,如果将关系数据库中存储的数据映射成为语义 Web 数据则可以减少构建语义 Web 的代价.为了利用原先存储在关系数据库中的信息,需要扩展 RDF 词汇表示主键、外键之类约束信息.文献[30]在这方面做了一些工作,将关系数据库信息无损转换为语义 Web 数据,并证明了 SPARQL 具有抽取约束信息和检查数据是否满足约束的功能.文献[31]对 SPARQL 作了扩展,允许用户设定查询结果的优先度,这样,系统将用户优先级较高的查询结果排在前面,适用于 Top-k 查询和 skyline 查询.文献[32]在 SPARQL 上增加了路径变量和路径表达式形成 SPARQ2L 语言,允许用户使用变量来指代未知路径,这样,用户可以使用该语言查询两实例间存在何种直接联系或间接联系.该文给出了 SPARQ2L 的实现框架、预处理方法和具体实现算法.

2.2 查询处理

为了利用关系数据库成熟的查询处理技术,流行的 SPARQL 查询处理方式采用将 SPARQL 直接映射到 SQL 的方式来实现,这样仅需编写 SPARQL 编译器,简化了实现 SPARQL 查询执行引擎的工作量.由于 SPARQL 的语法与 SQL 的语法存在差异,需要研究如何将 SPARQL 完整地翻译为 SQL.文献[33]提出了模式无关的 SPARQL-to-SQL 转换算法,将 SPARQL 中的查询语法转换为对应的 SQL 语句,并对转换结果进行优化,例如利用实例的类型信息和表统计信息来选择查询路径.文献[34]提出了一个 SPARQL 查询图模型,基于此模型定义了一些指导查询重写的转换规则,并运用启发式规则找到高效的查询计划.文献[35,36]提出了 3 类转换算法,解决了基本查询、嵌套查询和并行查询 3 类查询的转换算法,并提出了关于 SPARQL 基本查询和 OPTIONAL 算子映射的一套完整解决方案.文献[37]提出一种实现 OPTIONAL 算子到 SQL 的查询匹配的处理算法,并且从理论上分析该算法的时间复杂度.该算法不仅支持简单 OPTIONAL 匹配,而且支持多重 OPTIONAL 匹配和嵌套 OPTIONAL 匹配,以及复杂的多重嵌套 OPTIONAL 匹配.文献[38]主要提出了“Filter”表达式转换规则.该转换规则支持大部分 SPARQL 特征.

另一种方法是在原有的 SQL 编译器上扩充函数,由人们使用扩充的 SQL 语法书写语义 Web 数据查询请求.文献[39]是这方面工作的代表.该文在 Oracle 的查询引擎上增加 SQL 表函数 RDF_MATCH,结合 SQL 语句对语义 Web 数据作查询,这样就避免了编写 SPARQL 编译器的工作.用户直接利用 SQL 查询,可以避免 SPARQL 与 SQL 转换过程中可能出现的转换后的 SQL 语句代价过高的问题,并且可以利用 SQL 现有的功能,如聚集函数等.

2.3 SPARQL查询优化

在传统的 DBMS 中,查询优化可以分为基于启发式规则和基于代价两类.在 SPARQL 查询的执行过程中,由于要先将 SPARQL 翻译成 SQL,于是,大量基于启发式规则进行优化的工作可交给 SQL 优化器执行.所以,在 SPARQL 查询优化的研究中,研究者关注的焦点在基于代价优化方面.基于代价的查询优化由索引构建和统计信息收集两部分组成.

2.3.1 索引构建

RDF 数据是基于图的数据,常见的查询有以下几类:一是查找两节点间存在何种关联,二是查询哪些节点与给定节点间存在指定边,三是查找与给定节点最邻近的节点.前两类查询由于需要知道节点间的路径信息,因此

** RDFS是由W3C提出的一个用来描述RDF模式信息的语言标准.它提供了一组机制,用户可以通过该机制将相关资源分组,并在这些组间建立关联.

称为基于路径的查询.为提高这类查询的性能而建立的索引则称为路径索引.第 3 类查询需要知道节点间的距离,可称为基于距离的查询.在具体实现时,为了减少搜索空间,需要根据节点间的关联,将图切分成若干块,构建索引.因此,这类索引又称为分块索引.

(1) 路径索引

根据 RDF 图中两个被索引节点的距离,可将路径索引分为单步路径索引、多步路径索引等.单步路径索引存储直接关联节点的信息,类似于第 1.2 节中提到的索引,它们仅存储长度为 1 的路径.例如,*spo* 存储以 *s* 为顶点、以 *p* 为关联边、以直接关联的 *o* 为终点的单步路径.

多步路径索引存储从某顶点出发,经过多步可达的节点及边的信息,如$sp_1o_1p_2o_2$是一条双步路径,索引需要存储始点s、终点o_2、中间结点o_1和连接边p_1p_2的信息.由于具体要存储几步路径不容易确定,所以多步路径索引的使用较少,最多用到双步路径索引.比较而言,全路径索引更有用.

全路径索引是单步路径索引和多步路径索引之和.它将所有可能的路径存储在索引结构中,以便提高路径查询的效率.文献[40]提出一种贪婪算法,将 RDF 数据图上所有可能的路径都存储起来.这样,可以方便地根据用户需求在预存的路径上进行查询,避免在每次查询时进行计算.虽然该方法提高了查询效率,但是问题在于可扩展性差,且不利于数据的修改.路径可达索引是全路径索引的一个变种,它只存储两节点间是否存在路径的信息,不存储具体的路径信息.文献[41]针对 RDF 数据上常见的传递闭包查询,提出了有向图的素数编码标记机制(prime number labeling scheme for directed graph,简称 PLSD).为了快速地判断两节点间是否存在路径,该文利用素数互质的特点为每个资源赋予一个编码.如果两节点间不存在路径,则节点对应的编码互质,否则二者之间存在非 1 的公因子.通过这种方法,用户能够根据节点编码迅速判断二者是否存在路径.然而,随着资源数量的增加,编码的时间代价会迅速增长,因此这种方法的可扩展性存在问题.

(2) 分块索引

在 RDF 数据的应用中,需要查找给定节点的邻近节点.为了提高该类查询的效率,人们引入了分块索引机制,将 RDF 图作聚类操作,将其合并为若干块的集合.通过与块中心节点作比较,系统过滤掉不需要的区域,减少了 I/O 的次数.

文献[42]引入 GRIN 索引,它基于如下假设:在 RDF 数据图中,距离越近的资源越有可能成为同一查询所需要的结果.GRIN 根据用户指定的中心,将资源分为若干块,根据块中心节点的关系将数据组织成二叉树索引.索引节点对应资源块的中心,并存储该块的半径.文献[43]将 GRIN 索引引入到时态 RDF 数据的查询中,将 GRIN 索引扩展为 tGRIN 索引.由于时态数据增加了时间信息,因此与 GRIN 索引相比,tGRIN 在计算资源距离时不仅要考虑平面距离,还要增加对时态距离的考虑.tGRIN 的另一贡献是将索引结构由二叉树扩展到 *n* 叉树.GRIN 类索引的问题在于,这类根据距离将资源分块存储的索引方法适用范围较小,可推广性较差.

针对RDF数据中常见的类别查询[***],文献[44]提出一种优化算法,它合并数据集中属于相同类的资源,形成一个新数据集,称为摘要数据集.经证明,在摘要数据集上作类别查询的效果等同于在原始数据集上作类别查询.通过在数量大为精简的摘要数据集上作查询可以有效地提高查询效率.文献[45]在文献[44]的基础上进行扩展,将这种优化方法推广到关系查询[****]和合取查询[*****]上.该文提出一种方法,将关系查询转换成类别查询,这样就可以在摘要数据集上执行关系查询,同样提高了查询效率.从本质上而言,这种方法也是一种构建分块索引的方式.

*** 类别查询要求返回用户指定类别的所有实例.例如,$man(x)$是一个类别查询,查询结果返回所有男人.

**** 关系查询要求返回满足用户指定关系的所有实例对.例如,$hasFather(x,y)$是一个关系查询,查询结果返回所有存在父子关系的人.

***** 合取查询是类别查询和关系查询的合取组合,查找同时满足各个子查询的实例.例如,$man(x)\wedge hasFather(x,y)$,查询结果返回所有男人和他们的父亲.

2.3.2 统计信息收集

在构建索引以后,为了有效利用索引信息进行查询优化,数据库需要收集和估计统计信息.不同的索引方法需要收集和估计不同的统计信息.在目前的研究中,由于索引的提出还刚刚起步,因此统计信息收集和估计方面的研究相对而言比较少.

文献[46]主要解决 SPARQL 查询中基本查询(类似 SQL 的 Select-From 块)的优化问题.它定义和分析了用于优化的启发式规则,分别根据主语、谓语、宾语统计对应的平均三元组数,而后根据得到的结果估计单步路径查询的选择度和双步路径查询的选择度.为了解决计算量大的问题,它引入了抽样方法和人工干预,以减少计算代价.

文献[47]引入依赖属性的概念来计算三元组连接后的结果集大小.它将 SPARQL 的基数估值分解为两部分,一部分是基本查询,另一部分是在基本查询上增加过滤条件的查询(类似 SQL 的 Select-From-Where 块).对于基本查询,需要基于属性依赖关系分析其结果集的大小.对于后者,需要在基本查询结果集基数估值的基础上考虑过滤因素的影响,使用直方图计算需要过滤的比率.但是在计算属性关系时,需要考虑到具体值的连接,计算量比较大.与文献[46]相比,它考虑了属性间的依赖关系,在估计双步路径或多步路径查询时,其估计值较前者会更准确些.

文献[48]将 SPARQL 统计信息与魔集(magic set)技术结合起来考虑,以提高查询效率.该文用演绎数据库建模 RDF 数据,用混合代价模型估计候选集,结合了基于代价的优化和魔集技术的双重优化策略来选择最优的执行计划.基于代价的优化和魔集技术分别对应显式信息和隐式信息查询的优化.该方案使用双重优化策略提高了隐式信息的查询效率.

2.4 分析与总结

目前对语义 Web 查询语言的研究还处于初始阶段.研究者们逐渐认识到语义 Web 应用将成为未来应用的热点,因此开始尝试在各种领域上采用语义 Web 方法描述信息,也发现了 SPARQL 查询语言无法满足用户对语义 Web 数据操纵的需要,需要对 SPARQL 进行扩展.SPARQL 的扩展分为两类:面向语义 Web 应用的扩展和面向数据库应用的扩展.前者指为了支持对语义 Web 体系结构的高层应用,例如本体、RDFS 等,对 SPARQL 所作的扩展.后者指为了将 SPARQL 移植到传统的数据库应用而对 SPARQL 进行扩展,如支持用户优先级查询和 Skyline 查询等.

SPARQL 查询优化主要是基于代价的优化,为了计算查询代价,需要构建索引作为统计基础并搜集统计信息.RDF 数据索引是针对 RDF 图而提出的,分路径索引和分块索引两种.路径索引包括单步路径索引、多步路径索引和全路径索引,分块索引包括摘要信息索引和 GRIN 类索引.目前主要的研究热点在路径索引,因为主要的应用还是精确查询而非范围查询.由于索引的方法多种多样,语义 Web 应用也刚刚起步,因此,统计信息的收集研究仍停留在单步路径索引的估计和收集.

根据第 1 节的分析,语义 Web 数据的存储要与模式的融合结合考虑,而模式融合又与实例的聚类相关.因此,语义 Web 数据的存储更需要构建分块索引,以便于用户确定原有的分块是否科学,是否需要与附近的分块进行合并来提高查询效率.目前,分块索引的研究还比较少,主要的关注点在于路径索引的构建方面,所以,未来 SPARQL 的发展方向是构建分块索引并统计分块信息.

3 原型系统

现有的语义Web数据管理系统大多是基于三元组表方案进行存储的,主要有以下几个:Sesame,Jena,YARS+ActiveRDF[49].另外,值得一提的还有KAON2[50],它采用模式生成方式将数据存储在演绎数据库中,下面将分别加以介绍.

3.1 Sesame

Sesame(详见 http://www.openrdf.org/)是由德国 Aduna 公司开发和维护的一个开源的基于 Java 的语义 Web

数据存储系统.该系统起源于“On-To-Knowledge”项目.其特点是在数据存储和查询语言之间增加了一个中间层——SAIL.通过引入 SAIL,用户将查询语言对数据存储的调用转变为两个接口:查询语言与 SAIL 的接口和 SAIL 与数据存储之间的接口.这样,当系统需要扩展支持其他存储模式及其他存储格式的语义 Web 数据时,仅需修改 SAIL 与数据存储间的接口即可满足系统扩展的需求,提高了 Sesame 系统的移植能力.在 SAIL 层之上,系统提供了管理、查询和导出模块,在这 3 个模块之上,客户端可以通过 HTTP 协议和 SOAP 协议来访问 Sesame 服务器.

3.2 Jena

Jena(详见http://jena.sourceforge.net/)是惠普公司开发的一个开源的基于Java的语义Web数据存储系统,在语义Web领域应用广泛.与Sesame相比,Jena更侧重于对语义Web上层数据的支持,如RDFS和OWL.因此,Jena的推理功能更加全面.Jena包含一个内置的支持DIG[******]接口的推理器.当然,Jena也支持用户使用外置的推理器,例如Pellet.Jena提供了一套面向三元组格式的API,支持对语义Web数据进行存储和查询.其缺点在于,API直接面向存储底层,缺乏对面向对象的支持,需要开发人员详细定义操作步骤,因此不容易扩展到其他存储模式中.

3.3 YARS+ActiveRDF

YARS(详见 http://sw.deri.org/2004/06/yars/)和 ActiveRDF(详见 http://www.activerdf.org/)的组合构成一个完整的语义 Web 数据管理系统,二者都是 Ireland 大学 DERI 的研发成果.它们的功能不同,正如第 1 节所描述的,YARS 侧重于对语义 Web 数据的底层存储,而 ActiveRDF 是一套基于 Ruby/Rails 的支持语义 Web 数据查询的面向对象 API,侧重于对上层请求的支持.ActiveRDF 是 Ruby/Rails 的数据层,取代了原先的 ActiveRecord,通过它可以灵活地在语义 Web 数据与面向对象数据之间进行映射,极大地方便了用户使用 Ruby/Rails 开发语义 Web 应用.ActiveRDF 抽象了面向三元组的 API,对外提供面向对象的方法来存储和查询语义 Web 数据,而底层的数据存储仍然使用三元组表的存储方式.ActiveRDF 是一套灵活的框架,可以架在 YARS 之上,也可以架在 Sesame,Jena 等其他存储系统之上.

3.4 KAON2

KAON2(详见 http://kaon2.semanticweb.org/)是德国 Karlsruhe 大学开发的开源语义 Web 工具包,它是 KAON 的第 2 版.KAON2 基于演绎数据库存储数据,采用模式生成方式,将语义 Web 数据中的资源映射成表,如同关系数据库的模式设计,并将数据存储于表中.KAON2 支持客户端使用 DIG 接口访问服务器数据,侧重于语义 Web 体系结构中本体层数据的存储,因此它既是一个语义 Web 数据存储系统,又是一个推理系统.KAON2 的核心模块是推理引擎.推理引擎由 3 部分组成,本体公式化模块负责把本体数据翻译成一阶逻辑的公式集合,定理证明模块用于规约算法实现逻辑推理,而析取 Datalog 引擎用于回答结果.由于使用了演绎数据库的推理能力,KAON2 可以使用魔集等优化手段,提高了推理性能.

3.5 CODE

CODE 是中国人民大学信息学院数据库与智能信息检索实验室开发的语义 Web 数据管理工具.与其他侧重于数据推理的工具相比,它更侧重于通过语义 Web 数据存储方案和查询优化的研究提高查询和修改性能.CODE1.0 与 Sesame 的存储方案类似,都是基于三元组表的存储,为了提高查询性能,将模式信息分解出来存储.CODE2.0 又称 CODERS,采用垂直存储模式存储数据.目前正在开发的 CODE3.0 将是基于实例集合存储的一个方案.

[******] DIG是DL Implementation Group的缩写,是一套描述逻辑(DL)推理机的接口规范.通过该规范,用户可以方便地调用第三方开发的描述逻辑推理机.参见http://dl.kr.org/dig

3.6 分析与总结

从这几个原型系统的分析中可以看出,现有的语义 Web 管理系统更多地侧重于功能的实现,包括用户接口、推理算法等方面.对性能优化的考虑最多的是利用关系数据库的优化功能.如前所述,语义 Web 数据与关系数据是两种特点截然不同的应用,前者侧重于开放环境,而后者侧重于封闭环境,这导致原有基于关系数据库的存储和优化方式都需要进行修改.但是,目前的语义 Web 管理系统在这方面的工作较少.根据表 1 的分类方式,可以对语义 Web 管理系统进行类似的分类,见表 2.从中可以发现,目前的语义 Web 管理工具的发展比较缓慢,与语义 Web 存储方案相比,某些存储方案并不存在相对应的语义 Web 管理系统.从未来发展的角度来看,基于实例集且对数据变化适应力较好的语义 Web 管理系统是进一步研究的方向.

Table 2 Comparison of semantic Web management systems

表 2 语义 Web 管理系统的比较

Effects of data changes	Storage unit			
	Value (vertical partitioning)	Triple (triple table)	Instance (horizontal table)	Instance set (schema designing)
Big		Sesame, YARS+ActiveRDF		KAON2
Small	CODE2.0	Jena, CODE1.0		

4 测试集

在数据管理研究中,基准的测试给研究人员提供了一个公平的比较平台,诸如关系数据库领域中的 TPCC,TPCH 等基准测试.基准测试的有无和是否科学从另一个角度反映了目前该领域研究是否成熟.语义 Web 数据管理领域中还没有严格意义的基准测试,但已经出现了一些简单的测试集.

4.1 LUBM和UOBM

LUBM[51]是Lehigh大学提出的语义Web数据测试集.它基于大学这个领域,采用机器自动生成的数据作为测试数据,提供 14 个测试查询和一套性能指标.它可以根据用户指定的参数产生不同规模的数据, 由此测试在不同规模的环境下系统的实例查询性能.LUBM测试集是目前最流行的语义Web测试集.它生成的数据满足本体层的规范,因此,也可以作为推理系统的测试数据集.但是,也存在一个问题,即生成的数据中属性的个数是固定的,仅有 64 个.随着数据量的增加,数据会失去语义Web的一大特点——稀疏性,导致测试的结果不能反映实际应用的效果.

UOBM[52]是IBM中国研究院提出的一个测试集.它在LUBM的基础上进行扩展,增加了推理和可扩展性方面的测试.主要的工作有两个,一是生成了两个本体,分别是OWL-DL和OWL-Lite格式,完善了对不同格式本体的推理方面的测试;二是完善了不同学校的学生间的联系,使得数据看起来更像是一个整体而不是一个个孤岛.

4.2 SP2Bench

SP2Bench[53]是德国Freiburg大学提出的一套针对SPARQL语言设计的语义Web数据测试集.与LUBM相比,它更侧重于测试系统对SPARQL语法的支持程度,具体体现在SP2Bench完善了测试查询,在测试查询中增加了SPARQL语法的UNION和OPTIONAL等算子.它以DBLP目录库作为数据集,因为DBLP数据反映了社会网络分布的情况,因此,SP2Bench的测试结果可以反映语义Web数据的社会网络特征.

4.3 Yago

Yago[54]是德国Max-Planck-Institute提出的测试集,其特点在于综合了从WordNet和Wikipedia两个领域抽取出来的语义Web数据.目前的测试集大多是面向单个领域的测试集,而语义Web数据管理系统应该是一个面向多领域的应用,需要同时存储多个不同领域的数据.因此,目前的测试集在这一点上有所欠缺,Yago增加了这方面的支持,有助于测试系统的可扩展性、可重用性和应用独立性.Yago采用一种方法将WordNet和Wikipedia的数据联系起来,避免造成两个孤岛的情况.Yago的缺点在于其数据是实际数据,导致数据集的大小固定,无法针对

不同规模的应用系统作扩展.

4.4 Barton

Barton[55]测试集是美国麻省理工学院提出的一个测试集,它基于MIT图书馆的Barton在线目录而构建.测试查询用SQL表示,由于SPARQL缺乏对聚集函数的支持,因此很多查询无法翻译成SPARQL.另外,测试查询缺乏左外连接(类似于SPARQL的OPTIONAL算子)和修饰符.因此,从整体上说,该测试集不适用于测试SPARQL查询引擎.然而,由于数据来自不同的数据源,与语义Web数据的特点相同,Barton数据集表现出无结构的特征.如果对查询进行修改,则它可以作为一个很好的测试集.

4.5 分析与比较

现有的测试集都存在各自的缺点:要么测试查询与 SPARQL 语义不匹配,要么数据集存在缺陷,无法扩展或者仅仅针对实例进行扩展,扩展后丧失语义 Web 数据的稀疏性特征.进一步完善测试集或者构建更好的基准测试将是一个很重要的课题.

5 未来的挑战与发展趋势

语义 Web 数据管理带来多方面的挑战,也给数据库及相关领域的研究者带来了新的机会.

首先,尽管用 RDF 来描述语义 Web 数据已经成为标准,但是 RDF 数据的有效存储仍然是个难点.由于人们对于事物的认识是一个不断渐进的过程,因此,语义 Web 数据的存储应有足够的灵活性.传统的关系数据库由于面向封闭环境下的应用,约束过多,无法满足这方面的需求,需要进行改造,放宽对关系的约束.

其次,与 RDF 存储组织方式相适应的 SPAQRL 查询处理与优化技术的研究还刚刚起步,还有很大的空间.关系数据库查询优化的研究长期以来主导了数据库的研究潮流,我们有理由相信,这个领域也将主导 SPAQRQL 的研究.例如,利用存在于 RDF 数据本身中的一些语义信息来优化查询语句从而改善查询处理的性能,就是一个值得深入研究的问题.

再次,语义 Web 应用需要有强大的推理功能,能够根据显式信息推导出相关的隐式信息.目前采用的表推演算法效率较低,实现一个高效的推理器也将是语义 Web 能否获得推广的关键因素之一.

最后,XML 是 WWW 上表示结构化信息的一种标准文本格式.为了便于用户共享已开发的 RDF 数据,W3C 提出了一个基于 XML 语法的 RDF 描述语言,称为 RDF/XML.通过该语言,人们可以使用满足 XML 规范的文档表示 RDF 数据,并将这些文档存储到 XML 数据库中.据我们了解,目前尚未有使用 XML 数据库存储 RDF 数据的研究,不过有一些研究者考虑为 RDF 数据设计专门的存储模式,例如文献[42]的做法.这种做法与使用 XML 数据库存储 RDF 数据有相似之处.随着 XML 数据库的成熟,将其作为语义 Web 的底层数据管理平台也是完全可能的.我们认为,使用 XML 数据库存储语义 Web 也会成为未来的一个发展方向.

总而言之,语义 Web 数据管理是个交叉领域,包含了人工智能和数据库两方面的研究.人工智能研究者侧重于考虑隐含信息的推理和用户查询接口,而不关心数据存储方式和索引构建方式等.与人工智能的研究不同,数据库研究者更侧重于 RDF 数据的组织方式和查询优化策略.RDF 数据是通过从网络中自动抽取或从已有的关系数据中自动转换而来,数据量大,性能问题会越来越严重.两者之间的研究角度、方法和策略都不相同.未来的研究方向应是二者的融合,形成一个功能强大、性能优越的语义 Web 数据管理系统.该系统要能权衡数据存储与推理算法对性能的影响,以达到更好的效果.

语义 Web 概念的出现不过短短几年,已有相当数量的关于语义 Web 数据管理方面的论文面世.这从另一个侧面说明,语义 Web 数据管理将是数据管理研究领域的下一个热点.文献[17]被评为 VLDB2007 的优秀论文已经预示了这个领域的前景.这个领域的突破将有可能成为划时代的革新.

致谢 中国人民大学信息学院胡鹤老师为本文第 3 节提供了部分素材,在此表示感谢.

References:

[1] Berners-Lee T, Hendler J, Lassila O. The semantic Web. Scientific American, 2001. http://www.scientificamerican.com/article.cfm?id=the-semantic-Web

[2] Berners-Lee T. Artificial intelligence and the semantic Web. In: Proc. of the AAAI 2006 Keynote. 2006. http://www.w3.org/2006/Talks/0718-aaai-tbl/Overview.html

[3] Simmons RF. Storage and retrieval of aspects of meaning in directed graph structures. Communications of the ACM, 1966,9(3): 211−215.

[4] Fikes R, Hayes P, Horrocks I. OWL-QL: A language for deductive query answering on the semantic Web. Journal of Web Semantics, 2004,2(1):19−29.

[5] Karvounarakis G, Alexaki S, Christophides V. RQL: A declarative query language for RDF. In: Lassner D, Roure DD, Iyengar A, eds. Proc. of the WWW 2002. New York: ACM Press, 2002. 592−603.

[6] Seaborne A. RDQL—A query language for RDF. W3C, 20040109, 2004. http://www.w3.org/Submission/RDQL/

[7] Prud'hommeaux E, Seaborne A. SPARQL query language for RDF. W3C, 20080115, 2008. http://www.w3.org/TR/rdf-sparql-query/

[8] Broekstra J, Kampman A, Harmelen F. Sesame: A generic architecture for storing and querying RDF and RDF schema. In: Horrocks I, Hendler JA, eds. Proc. of the ISWC 2002. Berlin, Heidelberg: Springer-Verlag, 2002. 54−68.

[9] Carroll JJ, Dickinson I, Dollin C, Reynolds D, Seaborne A, Wilkinson K. Jena: Implementing the semantic Web recommendations. In: Feldman SI, Uretsky M, Najork M, Wills CE, eds. Proc. of the WWW Alt 2004. New York: ACM Press, 2004. 74−83.

[10] Li M. Study on ontology repository management system [Ph.D. Thesis]. Beijing: Renmin University of China, 2006 (in Chinese with English abstract).

[11] Harth A, Decker S. Optimized index structures for querying RDF from the Web. In: Proc. of the LA-WEB 2005. Boston: IEEE Computer Society, 2005. 71−80. http://sw.deri.org/2005/02/dexa/yars.pdf

[12] Cai M, Frank MR, Yan B, MacGregor RM. A subscribable peer-to-peer RDF repository for distributed metadata management. Journal of Web Semantics, 2004,2(2):109−130.

[13] Harth A, Umbrich J, Hogan A, Decker S. YARS2: A federated repository for querying graph structured data from the Web. In: Aberer K, Choi KS, Noy NF, Allemang D, Lee KI, Nixon LJB, Golbeck J, Mika P, Maynard D, Mizoguchi R, Schreiber G, Cudre-Mauroux P, eds. Proc. of the ISWC 2007. Berlin, Heidelberg: Springer-Verlag, 2007. 211−224.

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

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

[16] Qin G. Some key issues of ontology repository management system [MS. Thesis]. Beijing: Renmin University of China, 2006 (in Chinese with English abstract).

[17] Abadi DJ, Marcus A, Madden SR, Hollenbach K. Scalable semantic Web data management using vertical partitioning. In: Koch C, Gehrke J, Garofalakis MN, Srivastava D, Aberer K, Deshpande A, Florescu D, Chan CY, Ganti V, Kanne CC, Klas W, Neuhold EJ, eds. Proc. of the VLDB 2007. New York: ACM Press, 2007. 411−422.

[18] Abadi DJ. Column Stores for wide and sparse data. In: Proc. of the CIDR 2007. 2007. http://db.csail.mit.edu/projects/cstore/abadicidr07.pdf

[19] 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 Endowment, 2008,1(2):1553−1563.

[20] 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, Gehrke J, Garofalakis MN, Srivastava D, Aberer K, Deshpande A, Florescu D, Chan CY, Ganti V, Kanne CC, Klas W, Neuhold EJ, eds. Proc. of the VLDB 2007. New York: ACM Press, 2007. 1045−1056.

[21] Wilkinson K. Jena property table implementation. HPL-2006-140, HP Laboratories Palo Alto, 2006. http://www.hpl.hp.com/techreports/2006/HPL-2006-140.pdf

[22] Cyganiak R. A relational algebra for SPARQL. HPL-2005-170, HP Laboratories Bristol, 2005. http://www.hpl.hp.com/techreports/2005/HPL-2005-170.pdf

[23] Perez J, Arenas M, Gutierrez C. Semantics and complexity of SPARQL. In: Cruz IF, Decker S, eds. Proc. of the ISWC 2006. Berlin, Heidelberg: Springer-Verlag, 2006. 30−43.

[24] Perez J, Arenas M, Gutierrez C. Semantics of SPARQL. Technical Report, TR/DCC-2006-17, Department of Computer Science, Universidad de Chile, 2006. http://www.dcc.uchile.cl/TR/2006/TR_DCC-2006-009.pdf

[25] Schenk S. A SPARQL semantics based on datalog. In: Hertzberg J, Beetz M, Englert R, eds. Proc. of the 30th Annual German Conf. on Advances in Artificial Intelligence (KI). Berlin, Heidelberg: Springer-Verlag, 2007. 160−174.

[26] Angles R, Gutierrez C. The expressive power of SPARQL. In: Sheth AP, Staab S, Dean M, Paolucci M, Maynard D, Finin TW, Thirunarayan K, eds. Proc. of the ISWC 2008. Berlin, Heidelberg: Springer-Verlag, 2008. 114−129.

[27] Arenas M, Gutierrez C, Perez J. An extension of SPARQL for RDFS. In: Christophides V, Collard M, Gutierrez C, eds. Proc. of the SWDB-ODBIS 2007. Berlin, Heidelberg: Springer-Verlag, 2007. 1−20.

[28] Perez J, Arenas M, Gutierrez C. nSPARQL: A navigational language for RDF. In: Sheth AP, Staab S, Dean M, Paolucci M, Maynard D, Finin TW, Thirunarayan K, eds. Proc. of the ISWC 2008. Berlin, Heidelberg: Springer-Verlag, 2008. 66−81.

[29] Sirin E, Parsia B. SPARQL-DL: SPARQL 1uery for OWL-DL. In: Golbreich C, Kalyanpur A, Parsia B, eds. Proc. of the OWLED 2007. 2007. http://ceur-ws.org/Vol-258/paper14.pdf

[30] Lausen G, Meier M, Schmidt M. SPARQLing constraints for RDF. In: Kemper A, Valduriez P, Mouaddib N, Teubner J, Bouzeghoub M, Markl V, Amsaleg L, Manolescu I, eds. Proc. of the EDBT 2008. New York: ACM Press, 2008. 499−509.

[31] Siberski W, Pan JZ, Thaden U. Querying the semantic Web with preferences. In: Cruz IF, Decker S, Allemang D, Preist C, Schwabe D, Mika P, Uschold M, Aroyo L, eds. Proc. of the ISWC 2006. Berlin, Heidelberg: Springer-Verlag, 2006. 612−624.

[32] Anyanwu K, Maduko A., Sheth A. SPARQ2L: Towards support for subgraph extraction queries in RDF databases. In: Williamson CL, Zurko ME, Patel-Schneider PF, Shenoy PJ, eds. Proc. of the WWW 2007. New York: ACM Press, 2007. 797−806.

[33] Chebotko A, Fei X, Lu S, Fotouhi F. Scientific workflow provenance metadata management using an RDBMS-based RDF store. Technical Report, TR-DB-092007-CFLF, Wayne State University, 2007. http://www.cs.wayne.edu/~artem/main/research/TR-DB-092007-CFLF.pdf

[34] Hartig O, Heese R. The SPARQL query graph model for query optimization. In: Franconi E, Kifer M, May W, eds. Proc. of the ESWC 2007. Berlin, Heidelberg: Springer-Verlag, 2007. 564−578.

[35] Chebotko A, Lu S, Jamil HM, Fotouhi F. Semantics preserving SPARQL-to-SQL query translation for optional graph patterns. Technical Report, TR-DB-052006-CLJF, Wayne State University, 2006. http://www.cs.wayne.edu/~artem/main/research/TR-DB-052006-CLJF.pdf

[36] Chebotko A, Atay M, Lu S, Fotouhi F. Relational nested optional join for efficient semantic Web query processing. In: Dong GZ, Lin XM, Wang W, Yang Y, Yu JX, eds. Proc. of the APWeb/WAIM 2007. Berlin, Heidelberg: Springer-Verlag, 2007. 428−439.

[37] Liu J. Research on the optional matching issue in RDF queries [MS. Thesis]. Nanjing: Hehai University, 2007 (in Chinese with English abstract).

[38] Lu R, Cao F, Ma L, Yu Y, Pan Y. An effective SPARQL support over relational databases. In: Christophides V, Collard M, Gutierrez C, eds. Proc. of the SWDB-ODBIS 2007. Berlin, Heidelberg: Springer-Verlag, 2007. 57−76.

[39] Chong EI, Das S, Eadon G, Srinivasan J. An efficient SQL-based RDF querying scheme. In: Böhm K, Jensen CS, Haas LM, Kersten ML, Larson P, Ooi BC, eds. Proc. of the VLDB 2005. New York: ACM Press, 2005. 1216−1227.

[40] Udrea O, Pugliese A, Subrahmanian VS. GRIN: A graph based RDF index. In: Proc. of the AAAI, 2007. Menlo Park: AAAI Press, 2007. 1465−1470.

[41] Pugliese A, Udrea O, Subrahmanian VS. Scaling RDF with time. In: Huai JP, Chen R, Hon HW, Liu YH, Ma WY, Tomkins A, Zhang XD, eds. Proc. of the WWW 2008. New York: ACM Press, 2008. 605−614.

[42] Wu G. Research on key technologies of RDF graph data management [Ph.D. Thesis]. Beijing: Tsinghua University, 2008 (in Chinese with English abstract).

[43] Akiyoshi M, Toshiyuki A, Masatoshi Y, Shunsuke U. An indexing scheme for RDF and RDF schema based on suffix array. In: Cruz IF, Kashyap V, Decker S, Eckstein R, eds. Proc. of the SWDB 2003. 2003. http://www.cs.uic.edu/~ifc/SWDB/papers/Matono_etal.pdf

[44] Fokoue A, Kershenbaum A, Ma L, Schonberg E, Srinivas K. The summary Abox: Cutting ontologies down to size. In: Cruz IF, Decker S, Allemang D, Preist C, Schwabe D, Mika P, Uschold M, Aroyo L, eds. Proc. of the ISWC 2006. Berlin, Heidelberg: Springer-Verlag, 2006. 343−356.

[45] Dolby J, Fokoue A, Kalyanpur A, Ma L, Schonberg E, Srinivas K, Sun XZ. Scalable conjunctive query evaluation over large and expressive knowledge bases. In: Sheth AP, Staab S, Dean M, Paolucci M, Maynard D, Finin TW, Thirunarayan K, eds. Proc. of the ISWC 2008. Berlin, Heidelberg: Springer-Verlag, 2008. 403−418.

[46] Stocker M, Seaborne A, Bernstein A, Kiefer C, Reynolds D. SPARQL basic graph pattern optimization using selectivity estimation. In: Huai JP, Chen R, Hon HW, Liu YH, Ma WY, Tomkins A, Zhang XD, eds. Proc. of the WWW 2008. New York: ACM Press, 2008. 595−604.

[47] Shironoshita EP, Ryan MT, Kabuka MR. Cardinality estimation for the optimization of queries on ontologies. ACM SIGMOD Record, 2007,36(2):13−18.

[48] Ruckhaus E, Ruiz E, Vidal ME. OnEQL: An ontology efficient query language engine for the semantic Web. In: Polleres A, Pearce D, Heymans S, Ruckhaus E, eds. Proc. of ALPSWS 2007. 2007. http://ceur-ws.org/Vol-287/paper_7.pdf

[49] Oren E, Delbru R, Gerke S, Haller A, Decker S. ActiveRDF: Object-Oriented semantic Web pProgramming. In: Williamson CL, Zurko ME, Patel-Schneider PF, Shenoy PJ, eds. Proc. of the WWW 2007. New York: ACM Press, 2007. 817−824.

[50] Motik B. Reasoning in description logics using resolution and deductive databases [Ph.D. Thesis]. Karlsruhe: Univesität Karlsruhe, 2006.

[51] Guo YB, Pan ZX, Heflin J. LUBM: A benchmark for OWL knowledge base systems. Journal of Web Semantics, 2005,3(2-3): 158−182.

[52] Ma L, Yang Y, Qiu ZM, Xie GT, Pan Y, Liu SP. Towards a complete OWL ontology benchmark. In: Sure Y, Domingue J, eds. Proc. of the ESWC 2006. Berlin, Heidelberg: Springer-Verlag, 2006. 125−139.

[53] Schmidt M, Hornung T, Lausen G, Pinkel C. SP^{2}Bench: A SPARQL performance benchmark. Technical Report, arXiv:0806.4627v1 cs.DB, arXiv.org, 2008. http://arxiv.org/pdf/0806.4627v1

[54] Theoharis Y, Christophides V, Karvounarakis G. Benchmarking database representations of RDF/S stores. In: Gil Y, Motta E, Benjamins VR, Musen MA, eds. Proc. of the ISWC 2005. Berlin, Heidelberg: Springer-Verlag, 2005. 685−701.

[55] Abadi DJ, Marcus A, Madden SR, Hollenbach K. Using the Barton libraries dataset as an RDF benchmark. Technical Report, MIT-CSAIL-TR-2007-036, MIT, 2007. http://cs-www.cs.yale.edu/homes/dna/papers/bench.pdf

附中文参考文献:

[10] 李曼.本体库管理系统研究[博士学位论文].北京:中国人民大学,2006.

[16] 秦国.本体库管理系统中若干关键技术的研究[硕士学位论文].北京:中国人民大学,2006.

[37] 刘静.RDF 查询中非强制匹配问题研究[硕士学位论文].南京:河海大学,2007.

[42] 吴刚.RDF 图数据管理的关键技术研究[博士学位论文].北京:清华大学,2008.

杜小勇(1963－),男,浙江开化人,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.

吕彬(1981－),女,硕士,CCF 学生会员,主要研究领域为语义 Web,高性能数据库.

王琰(1977－),男,博士生,CCF 学生会员,主要研究领域为语义 Web,高性能数据库.