海量RDF数据管理*

邹 磊[1] 陈跃国[2]
[1]北京大学
[2]中国人民大学

关键词：RDF数据管理 关键词检索

语义网和RDF数据

语义网是万维网之父蒂姆·伯纳斯-李（Tim Berners-Lee）在1998年提出的，它提供了一种在不同的应用和个体之间共享和重用数据的整体框架[1]，其核心是构建以数据为中心的网络，即web of data。我们将万维网称为web of pages。万维网是利用超链接技术将不同的文档链接起来，从而方便用户浏览和共享文档。例如HTML（hypertext markup language，超文本标记语言）文档的语法是告诉浏览器按照何种格式来显示文档，而并不是告诉计算机文档中的数据分别表示什么语义信息。语义网的核心是让计算机能够理解文档中的数据以及数据与数据间的语义关联关系，从而使机器可以更加智能化地处理这些信息。因此可以把语义网想象成一个全球性的数据库系统。由于语义网技术涉及面较广，本文仅讨论语义网框架中的一项核心概念——资源描述框架（resource description framework，RDF）。

RDF[6]是一种数据模型，是由万维网联盟（World Wide Web Consortium，W3C）组织的资源描述框架工作组于1999年提出的一个解决方案，并于2004年2月正式成为万维网联盟推荐标准。其目标是构建一个综合性的框架来整合不同领域的元数据，实现在万维网（Web）上交换元数据，促进网络资源的自动化处理。随着互联网的发展，对元数据的研究逐步深入，出现了多种元数据标准，如DC（Dublin core）[3]和PICS（platform of Internet content selection）[5]等等。这些元数据描述、组织并重新整理了网络信息，使用户可以更方便地利用网络数据。RDF的基本数据模型包括资源（resource）、属性（property）及陈述（statements）。

资源 所有能够使用RDF表示的对象都称为资源，包括网络上的所有信息、虚拟概念和现实事物等等。资源以唯一的统一资源标识（uniform resource identifiers，URI）来表示，通常使用的URL是它的一个子集。不同的资源拥有不同的URI。

属性 用来描述资源的特征或资源间的关系。每一个属性都有其意义，用于定义资源的属性值（property value）、描述属性所属的资源形态、与其他属性或资源的关系。

陈述 一条陈述包含三个部分，通常称为RDF三元组<主体，属性，客体>。其中主体一定是被描述的资源，由URI表示。客体表示主体在属性上的取值，可以是另外一个资源（由URI表示）或者是文本。

要实现从万维网到语义网的转变，构建海量

* 本文作者邹磊的研究获国家自然科学基金青年基金：基于图数据库理论的海量RDF数据存储和查询方法研究（61003009）的资助。本文作者陈跃国的研究获国家核高基重大科技计划项目：非结构化数据管理系统之人大部分（2010ZX01042-002-002-03）资助。

和分布式的RDF数据集是一项重要且不可或缺的步骤，为此万维网联盟提出了LOD（Linked Open Data）项目[7]将各个零散的RDF数据集链接起来构成未来语义网的基础。RDF数据的获取和构建目前有人工编辑、基于信息抽取方法构建和基于Web2.0的协同编辑三种方法。传统的人工编辑只限于单个领域的小规模RDF数据的构建。基于信息抽取技术，可以自动地从大规模非结构化数据中抽取和构建开放领域的RDF数据。例如Barton[8]抽取自美国麻省理工学院的图书馆数据，YAGO[9]和DBpeida[10]都是从维基百科上通过信息抽取的方法来构建RDF数据集合。利用类似于维基百科的协同编辑方法，由一个网络社区的用户共同构建RDF数据集也是构建高质量RDF数据的一种可行方法，例如Freebase[11]等。

RDF数据管理研究现状

目前，海量RDF数据存储和查询的方案分为两种：一种是将三元组数据映射成关系数据库中的表结构，利用现有的关系型数据库管理系统（relational database management system，RDBMS）来完成面向RDF查询检索；另一种是基于图结构的存储方式。因为RDF数据本身就是基于图结构的，所以可以利用图数据库中的操作来完成对RDF数据的查询。

SPARQL查询语言

SPARQL查询语言是由万维网联盟的“RDF Data Access”工作组（DAWG）开发的一种面向RDF数据的查询语言，目前已经成为万维网联盟的RDF查询语言的推荐标准。SPARQL语言与关系数据库中的SQL语言类似，这方便了用户对于SPARQL语言的使用。例如在SPARQL语法中，也是在SELECT部分指定查询变量，在WHERE部分指定查询条件，而查询条件通常由三元组来构成，其中三元组的某一项或者某几项可以由变量来表示[20]。

图1给出了一个RDF数据集的例子。假设我们需要从图中的RDF数据中查询“在1809年2月12日出生，并且在1865年4月15日逝世的人的姓名”这个自然语言的问题，可以表示成图2所示的SPARQL语句。

Prefix: y= http://en.wikipedia.org/wiki/

主体	属性	客体
y:Abraham_Lincoln	hasName	“Abraham Lincoln”
y:Abraham_Lincoln	BornOnDate	“1809-02-12”
y:Abraham_Lincoln	DiedOnDate	1865-04-15
y:Abraham_Lincoln	DiedIn	y:Washington_D.C
y:Washington_D.C	hasName	“Washington D.C.”
y:Washington_D.C	FoundYear	1790
y:Washington_D.C	rdf:type	y:city
y:United_States	hasName	“United States”
y:United_States	hasCapital	y:Washington_D.C
y:United_States	rdf:type	Country
y:Reese_Witherspoon	rdf:type	y:Actor
y:Reese_Witherspoon	BornOnDate	“1976-03-22”
y:Reese_Witherspoon	BornIn	y:New_Orleans,_Louisiana
y:Reese_Witherspoon	hasName	“ReeseWitherspoon”
y:New_Orleans,_Louisiana	FoundYear	1718
y:New_Orleans,_Louisiana	rdf:type	y:city
y:New_Orleans,_Louisiana	locatedIn	y:United_States

图1 RDF例子

```
SELECT ?name                          //查询返回的变量值
WHERE
{ ?m  <hasName>   ?name.              //查询条件
  ?m <BornOnDate>  “1809-02-12” .
  ?m <DiedOnDate>  “1865-04-15” .
}
```

图2 SPARQL查询的例子

我们也可以将RDF和SPARQL分别表示成图的形式。例如在RDF中，主体和客体可以分别表示成RDF图中的节点，一条RDF三元组可以表示成一条边，其中属性是边的标签。图3显示了图2对应的RDF图和SPARQL查询图结构。回答SPARQL查询本质上就是在RDF图中找到SPARQL查询图的子图匹配的位置，这就是基于图数据库的回答SPARQL查询的理论基础。

基于关系数据模型的方法

由于关系型数据库管理系统在数据管理方面的成功及其成熟的商业软件产品，而且RDF数据的三元组模型可以很容易映射成关系模型，因此大量研究者尝试了使用关系数据模型来设计RDF存储和检索的方案[21~23]。根据所设计的表结构的不同，相应的存储和查询方法也各异。

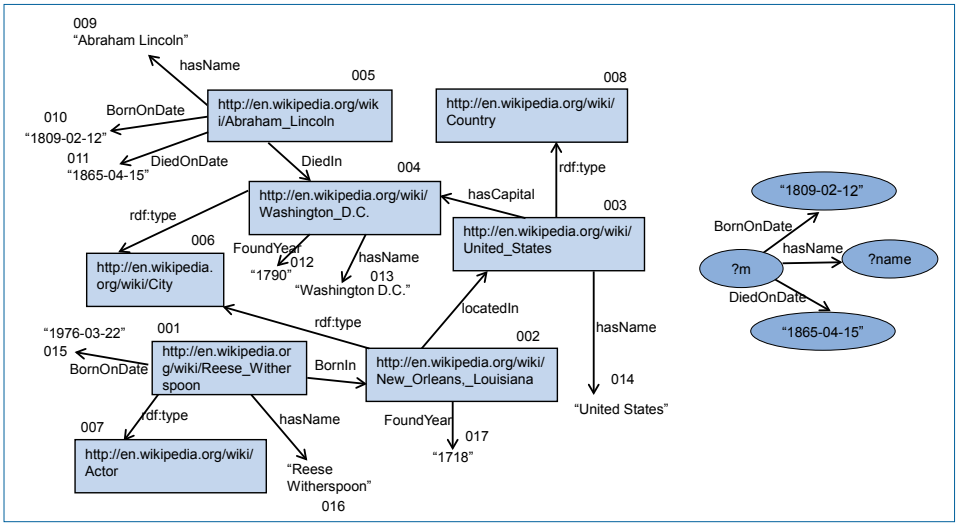

图3 RDF图和SPARQL查询图

```
SELECT  T3.Subject
FROM  T as T1, T as T2, T as T3
WHERE T1. Property = "BornOnDate"
and T1.Object= "1809–02–12"
and T2.Property= "DiedOnDate"
and T2.Object= "1865–04–15"
and T3. Property = "hasName"
and T1.Subject = T2.Subject
and T2. Subject= T3.subject
```

图4 转换以后的SQL查询

简单三列表

一种将RDF数据映射到关系数据库表的最简单的方法是构建一张三列表（Subject，Property，Object），将所有的RDF三元组都放在这个表中。给定一个SPARQL查询，设计查询重写机制将SPARQL转化为对应的SQL语句，由关系数据库来回答此SQL语句。例如将图2中的SPARQL查询转换为图4中的SQL语句。

虽然这种方法具有很好的通用性，但其最大的问题是查询性能差。如图4所示的SQL语句中有多个表的自连接操作，三列表的规模可能非常庞大，将严重影响其查询性能。

水平存储

文献[24]中提到的水平方法（horizontal schema）是将一个RDF主体表示为数据库表中的一行。表中的列包括该RDF数据集合中所有的属性。这种策略的好处在于设计简单，同时很容易回答面向某单个主体的属性值的查询，如图5所示。

根据图5中表的结构，为了回答图2中的SPARQL查询，可以转换为如图6所示的SQL语句。与图4比较，图6中的SQL语句没有耗时的连接操作，因此其查询效率要远高于图4中的SQL语句。

然而这种水平存储方法也有明显的缺点[24,25]：第一，表中存在大量的列。一般来讲，属性数目会比主体和属性值的个数少很多，但还是有可能超过当前数据库能够承受的范围；第二，表的稀疏性问题。通常一个主体并不是在所有的属性上都有值，而仅仅在极少数的属性上有值。然而由于一个主体存成一行，那么表中将存在大量空值。空值不仅增加了存储负载，还会带来其他问题，比如增加索引大小、影响查询效率。文献[24~26]详述了空值带来的问题；第三，水平存储存在多值性问题。一个表中列的数量是固定的，这就使主体在一个属性上只

Subject	rdf:type	hasName	BornOnDate	DiedOnDate	DiedIn	FoundYear	hasCapital	locatedIn	bornIn
y:Abraham_Lincoln		Abraham Lincoln	1809–02–12	1865–04–15	y:Washington_D.C				
y:Washington_D.C	y:city	Washington D.C.				1790			
y:United_States	y:country	United States					y:Washington_D.C		
y:Reese_Witherspoon	y:actor	Reese Witherspoon	1976–03–22						y:New_Orleans,_Louisiana
y:New_Orleans,_Louisiana	y:city					1718		y:United_States	

图5 水平存储

```
SELECT hasName from T WHERE
BornOnDate = “1809-02-12” and
DiedOnDate = “1865-04-15”.
```

图6 水平存储上的SQL查询

能有一个值。而真实数据往往并不符合这个限制条件；第四，数据变化可能带来很大的更新成本。在实际应用中，数据的更新可能带来增加属性或删除属性等改变，这就涉及到整个表结构的变化，水平结构很难处理类似的问题。

属性表

属性表是将三元组根据属性分类，每一类采用水平存储策略的数据库表。属性表在继承了水平储策略优势的基础上，通过对相关属性的分类避免了表中列数过多等问题。Jena2[27,28]中使用属性表以提高对RDF三元组的查询效率。研究者提出了两种不同的属性表，一类称为聚类属性表（clustered property table），另一类称为属性类别表（property-class table）。

聚类属性表将概念相关的属性聚成一类，每一类定义一个单独的数据库表，使用水平方式存储这些三元组。如果有三元组不属于任何一个类别，就被放在一张剩余表（left-over table）中。在图5中，根据属性的相关性，将所有的属性聚类成三个类，每个类用一张水平表来存储。根据图7给出的属性表结构，我们同样可以将图3中的SPARQL查询转换成类似于图6的SQL语句。属性类别表将所有的实体按照rdf:type来分类，每一类用一个张水平表来表示。这种组织方式要求每个实体都必须有一个rdf:type属性。

属性表最主要的优点在于可以减少查询时主体-主体间的自连接代价，从而极大地提高查询效率。属性表的另外一个优点是属性相关的属性值存储在一列中，可以针对该列的数据类型设计一些存储策略来减少存储空间。这样就避免了三元组存储策略中，由于数据类型不同的属性值存储在一列中造成存储上的不便。Jena2等研究工作证明了属性表的有效性，但属性表的缺陷使其除了某些特殊应用以外，没有更广泛的应用。第一，文献[29]指出，虽然属性表对于某些查询能够提高查询性能，但是大部分查询都会涉及多个表的连接或合并操作。对聚类属性表而言，如果查询中属性作为变量出现，则会涉及多个属性表；对属性分类表而言，如果查询并未确定属性类别，则查询也会涉及多个属性表。在这种情况下，属性表的优点就不明显了；第二，RDF数据由于来源庞杂，其结构性较差，属性和主体间的关联性不强，类似的主体可能并不包含相同的属性。空值的问题随之出现。数据的结构性越差，空值的问题就越明显；第三，在现实中，一个主体在一个属性上可能存在多值。用关系型数据库管理系统管理这些数据时就带来麻烦。其中，前两个问题是相互影响的。当一个表的宽度减小时，对结构性的要求降低，空值问题就会得到缓解，但查

Prefix: y= http://en.wikipedia.org/wiki/

People

Subject	hasName	BornOnDate	DiedOnDate	DiedIn	BornIn	rdf:type
y:Abraham_Lincoln	“Abraham Lincoln”	1809–02–12	1865–04–15	y:Washington_D.C		
y:Reese_Witherspoon	“ReeseWitherspoon”	1976–03–22		y:Washington_D.C	y:New_Orleans,_Louisiana	y:Actor

City

Subject	FoundYear	rdf:type	locatedIn	hasName
y:New_Orleans,_Louisiana	1718	y:city	y:United_States	
y:Washington_D.C	1790	y:city	y:United_States	“Washington D.C.”

Country

Subject	hasName	hasCapital	rdf:type
y:United_States	“United States”	y:Washington_D.C	Country

图7 聚类属性表

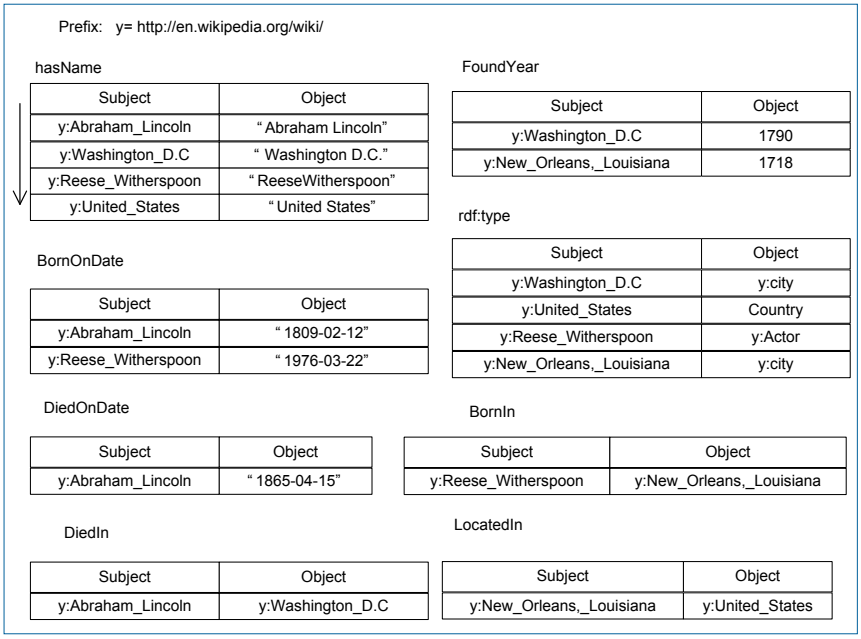

图8 二元垂直分割表

询会涉及更多的表；而当表的宽度增加时，如果数据结构性不强，就会出现更多的空值问题[29]。

二元存储

阿巴迪（Abadi）等人[29]以一种完全的分解存储模型（decomposed storage model，DSM[30]）为基础，将DSM引入了语义网数据的存储，提出了垂直分割技术。在垂直分割的结构下，三元组表被重写为N张包含两列的表，N等于RDF数据中属性的个数。每一张表都以对应的属性为表名，第一列是所有在这个属性上有属性值的主体，第二列是该主体在这个属性上的值。每一张表中的数据按照主体进行排序，从而能够迅速定位特定主体，而且将所有涉及主体-主体的表连接转换为可以迅速完成的排序合并连接（merge join）。在对存储空间限制较少时，也可以对属性值这一列建立索引或对每个表建立一个按照属性值排序的副本，以提高对特定属性值的访问和属性值-主体、属性值-属性值连接的性能。如图8显示了将图1中的RDF数据集分解成8个二元表，每个二元表按照主体进行排序。

与三元组存储相比，二元存储方式有如下优点：由于属性名不再重复出现，因此有效地减少了存储空间；在查询时，只需要处理涉及查询条件的表，从而有效地减少了I/O代价。与属性表方式相比，垂直分割的优点有：垂直分割适应于多值数据。与三元组存储方式类似，当一个主体在一个属性上有多个属性值时，只需要将其存储为多行；垂直存储适用于结构化较差的数据，如果一个主体未定义某个属性，那么这条记录就不会在这种存储方式中出现，避免了空值的产生。二元存储技术不需要对属性进行聚类，就不需要寻找好的聚类算法。在查询时，如果属性名被限定，那么查询的内容就不会出现在多个表中，减少了合并操作。SW-Store[29]利用了垂直分割技术，进一步减少了主体存储的冗余。但垂直分割技术同样存在缺点。首先，这种存储方式增加了表连接的运算数。即使这些连接都是时间代价较低的合并连接，总的运算代价也是不可忽略的；其次，表的增多增加了数据更新的难度。对一个主体的更新涉及多个表，可能因为外存存储方式的影响增加I/O代价。当表上存在索引时，更新代价更高。另外，文献[31,32]认为在多个表中存储结构化不强的数据（如某些RDF数据集）会存在一些问题，将多个表返回的结果重构成一张视图所进行的运算代价较高。研究者建议将较稀疏、结构化较差的数据存储于一张表中，并对存储结构加以描述。

全索引策略

为了提高简单三列表存储的查询效率，目前一种普遍认可的方法是“全索引”（exhaustive indexing）策略，例如RDF3x[33]和Hexastore[34]。列举三列表的所有排列组合的可能性（6种），并且按照每一种排列组合建立聚集B+-树。建立全索引的好处有两点：其一，对于SPARQL查询中的每个查询

三元组模式，都可以转换成对于某个排列组合上的范围查询。例如?m<BornOnDate> “1809-02-12”查询三元组模式，可以转换为对于（P，O，S）排列上的范围查询。因为在（P，O，S）排列中，所有P，O为<BornOnDate>和“1809-02-12”的三元组都连在一起；其二，全索引的好处在于可以利用合并连接降低连接的复杂度。我们知道嵌套循环连接（nested loop join）的复杂度是O（$|L_1|*|L_2|$），这里$|L_1|$和$|L_2|$分别表示两个待连接列表的长度。然而合并连接的复杂度是O（MAX（$|L_1|, |L_2|$））。例如从（P，O，S）排列中可以得到满足?m <BornOnDate>“1809-02-12”查询条件的?m的取值，并且这些取值按顺序排列。同样的，从（P，O，S）排列中也可以得到满足?m <DiedOnDate> “1865-04-15”查询条件的?m的取值，这些取值也是按顺序排列。通过合并排序，我们可以很容易找到同时满足这两个查询条件的?m的取值。

虽然用全索引策略可以弥补一些简单垂直存储的缺点，但三元存储方式难以解决的问题还有很多。第一，不同的三元组其主体/属性/属性值可能重复，浪费了存储空间；第二，复杂的查询需要进行大量表连接操作，即使精心设计的索引可以将连接操作都转化为合并连接，但当SPARQL查询复杂时，其连接操作的查询代价依然不可忽略；第三，随着数据量增长，表的规模会不断膨胀，系统的性能下降严重。而且目前此类系统都无法支持分布式的存储和查询，这限制了系统的可扩展性；第四，由于数据类型多种多样，无法根据特定数据类型进行存储优化，可能会造成存储空间的浪费（例如，客体的值可能多种多样，如URI、一般字符串或数值。客体一栏的存储空间必须满足所有的取值，而无法进行存储优化）。为了解决这个问题，目前的全索引方法是利用字典方式将所有的字符串和数值映射成一个独立的整数ID。但是这种方法很难支持带有数值范围约束和字符串中的子串约束的SPARQL查询。

基于图数据模型的方法

通过将RDF三元组看作带标签的边，RDF数据很自然地符合图模型结构。因此，有的研究者从RDF图模型结构的角度来看待RDF数据。他们将RDF数据视为一张图，并通过对RDF图结构的存储来解决RDF数据存储问题。图模型符合RDF模型的语义层次，可以最大限度地保持RDF数据的语义信息，也有利于对语义信息的查询。此外，以图的方式来存储RDF数据，可以借鉴成熟的图算法、图数据库来设计RDF数据的存储方案与查询算法。然而，利用图模型来设计RDF存储与查询也存在难以解决的问题：第一，相对于普通的图模型，RDF图上的边具有标签，并可能成为查询目标；第二，典型的图算法时间复杂度较高，需要精心的设计以降低实时查询的时间复杂度。

本斯特伦（V. Bönström）等人[35]提出，相比于将RDF数据视为XML格式数据或三元组的集合，RDF的图模型包含了RDF数据中涵盖的语义信息。他们认为，用图结构存储RDF数据的优点在于：（1）图结构能够直接映射RDF模型，避免了为适应存储结构对RDF数据进行转换；（2）查询RDF数据的语义信息需要重构RDF图，以图结构存储RDF数据避免了重构。在存储方案的具体实现上，他们采用了面向对象数据库，并论证了其有效性。

安格鲁（R. Angles）和古铁雷斯（C. Gutierrez）对用图模型数据库存储RDF数据的一些细节问题进行了讨论[36]。他们比较了各种抽象存储模型（如关系数据模型、语义模型、面向对象数据模型等）与RDF数据模型之间的关系，并重点关注了图数据库模型。另外，他们还讨论了现有的RDF查询语言对图数据查询的适应能力和图数据库查询语言对RDF数据的适用性。

乌德雷亚（Udrea）等人提出用GRIN算法[39]来回答SPARQL查询。GRIN的核心是构建一个类似M-tree结构[40]的GRIN索引。所有的RDF图上的节点表示成GRIN索引上的叶子节点。GRIN索引上的非叶子节点包括两个元素（center，radius），其中center是一个中心点，radius是半径长度。在RDF图上到center的最短距离小于等于radius的节点在GRIN上是该非叶子节点的子孙节点。利用距离约束，

GRIN可以迅速判断并过滤掉RDF图上不满足查询条件的部分，从而提高查询性能。

邹（Zou）等人提出用gStore系统[41]来存储RDF和回答SPARQL查询语句。通过编码的方法，将RDF图G中的每个实体节点、邻居属性和属性值编码成一个带有Bitstring的节点，从而得到一张标签图G*。也可以将查询图Q表示成一张查询的标签图Q*。可以证明Q*在G*上的匹配是Q在G上匹配的超集。为了有效地支持在G*上查找Q*的匹配位置，gStore系统提出VS-tree索引。具体地说，在所有G*的叶子节点上建立S-tree索引[42]，S-tree每个叶子节点对应G*上的一个点。为了构建VS-tree，当且仅当S-tree上的两个非叶子节点的子孙之间在RDF图上存在一条边时，引入一条边连接这两个非叶子节点。也就是说，VS-tree每个非叶子层是底层G*的不同粒度的摘要图（summary graph）。利用每层的摘要图，gStore系统提出利用VS-query算法来削减查询空间，加快查询速度。

海量分布式的RDF存储查询引擎

RDF数据集的规模在不断扩大，而且几乎以每年翻一番的速度在快速地扩充。另外RDF数据集本身就是分布式的，各个站点提供各自的RDF数据集，各个数据集之间通过OWL:sameAS等互相链接起来从而构成一张巨大的RDF图。海量RDF的数据规模和分布式特点，使得传统的集中式管理方法面临挑战。目前有不少工作在讨论海量分布式RDF存储查询引擎的构建问题。例如文献[64]讨论了在AWS（Amazon Web Services）上利用SimpleDB（AWS提供的一种Key-Value Store）来回答SPARQL查询。该文献提出了一系列索引方式，用来确定给定一个查询以后，利用其索引结构能够很快地判定哪个RDF数据集可能包含查询结果。文献[65]给出了利用Hadoop来存储和检索RDF数据集的方案，并讨论了此平台下的查询计划生成算法。H2RDF[66]利用MapReduce的框架设计了基于云平台的RDF存储和查询算法。关于这方面的研究，虽然尚处于起步阶段，但是已经引起了学术界和产业界的高度关注。

面向RDF的关键词检索

结构化的SPARQL查询语言有着较为复杂的语法定义，需要用户熟悉它的语法规则和了解RDF数据的模式信息（属性和前缀等），才能使用该语言检索RDF数据。对于普通用户而言，SPARQL语言实用性极差。而与之对应的关键词查询，则因灵活度大、实用性好，而成为RDF数据查询的常见形式。这方面的研究可以分为两大类：（1）通过关键词匹配直接检索RDF数据图；（2）由关键词构造出结构化的查询语句，再利用转换后的查询语句检索结果。

直接检索RDF数据图

由关键词直接检索结果的方法采用的是图上的关键词查询方法，即在RDF数据图上定位包含检索词的斯坦纳树（Steiner tree）[50~53]。斯坦纳树是满足根节点到每一个关键词节点都存在一条路径的子树。此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，通过索引可以快速地定位图中包含关键词信息的节点（关键词节点）。最初的BANKS[50]采用的基本搜索策略是，对包含关键词的节点在图上进行扩展（对扩展路径的长度有限制），直到找到满足斯坦纳树的根节点。为了提高查询性能，文献[51]提出被扩展的节点有一个权重值，对每一个关键词节点进行权重计算，优先扩展权重值比较高的，在扩展过程中先反向扩展n步，再正向扩展，若找到子树结构则停止，否则再反向扩展。

BLINKS[52]则建立了一些额外的索引结构来存储关键词可达性信息：第一种索引是关键词至节点的最短距离，给定一个关键词，得到能到达关键词的所有节点及最短路径；第二种索引是节点至关键词的最短距离，给定节点和关键词，在O(1)时间内得到最短距离。EASE[53]提出了一个扩展的倒排索引结构，是两个关键词对包含这两个关键词的所有

子图的一个倒排索引。在对多个关键词进行查询时，只需要对两两关键词的配对在倒排索引中搜索出子图，并对子图求交集。

结构化查询转换

结构化查询将关键词查询转换成结构化的查询语句（如SPARQL）。首先，匹配查询关键词到RDF图的边或者顶点上；然后，在模式信息的辅助下，找到查询关键词之间的关联，确定用户的查询对象；最后，构造符合语法规则的结构化查询语句并将其排序返回。该类方法一般需要借助用户的反馈，按所选查询语句向RDF数据库发起结构化查询并获得最终的查询结果。

特兰（Tran）等人[54]提出的方法首先在模式图上找到包含用户查询关键词的最小子图，得到top-k个子图，然后结合数据图将这些子图映射成合取查询语句，这些查询会翻译成SPARQL查询语句，进行数据库的查询操作并返回结果。文献[55]将关键词查询转换成结构化查询语句返回，也可以在此基础上直接构造查询结果返回。该方法从RDF数据中抽取结构信息，构造查询搜索图，搜索符合要求的子图生成结构化查询，再结合实体索引和关系索引直接得到查询结果。此外，还有方法借助人机交互的手段获得关键词查询的结构化语句转换[56]。

以上两类方法各有优缺点。直接检索RDF数据图的方法需要建立大量的索引结构，并且一般忽略RDF的属性信息，无法处理用户将属性或关系名作为关键词查询的情况，但它直接在RDF图中构造查询结果返回，查询粒度细，不需要背景知识和RDF模式信息的支持；基于模式图的结构化查询转换方法不需要建立大规模的索引，但依赖RDF模式信息，确定查询关键词之间的关联，出于查询语句转换精度的考虑，有时还需要人工干预。

还有很多针对RDF搜索结果排序的相关工作[57~59]。帕塔萨拉蒂（Parthasarathy）等人[57]提出了在搜索过程中没有距离限制的算法。在以往的图搜索过程中，为了降低搜索和计算代价设定一个固定的距离以圈定一个较小的关键词节点的扩展范围。艾巴索尼（Elbassuoni）等人[58]提出了一个基于语言模型的排序机制，分别计算查询语言模型的上概率分布和查询结果语言模型的概率分布，将这两者用KL距离进行计算，以此对结果集进行排序。

一些RDF数据集的介绍

随着RDF被广泛应用到表达web of data中，互联网上出现了越来越多的RDF数据集。这些数据集来自化学、生物、地理、环境、娱乐和体育等众多领域。LOD项目正努力将越来越多的RDF数据集关联起来。截至2012年3月，LOD所收集的325个数据集已经包含了超过250亿条RDF三元组。在这样的背景下，出现了诸如Swoogle[60]和Sindice[61]等专用于检索以RDF数据为代表的语义网数据搜索引擎。RDF数据管理中常见的数据集如下：

Freebase[11]是语义网数据库技术公司Metaweb创建的。该公司在2010年被谷歌收购。Freebase包含了维基百科、IMDB、Flickr等众多网站或数据集中的结构化数据，目前包含了2000多万条实体信息。Freebase以一个叫做MQL（metaweb query language）的查询语言向用户提供了RDF形式的数据访问接口。整个数据可以下载，其压缩格式的数据集容量目前为5GB左右。

DBpedia[10]是一个被广泛使用的RDF数据集，是由德国的研究人员从多种语言的维基百科网页中抽取出的结构化信息，包含了众多领域的实体信息，三元组的数量超过了10亿条。它被认为是LOD中最为重要的一个数据集，也因此被广泛用作RDF数据查询的一个标准数据集。

YAGO[9]是一个由德国马克斯-普朗克研究所（the Max Planck Institute，MPI）的工作人员从维基百科网页中抽取出来的RDF数据集。为了提高精度，它还借助了WordNet中的本体信息。除了像DBpedia充分利用维基百科的Infobox来抽取结构化信息外，YAGO还直接从文本信息中抽取了大量的RDF数据，其信息抽取的精度高达95%以上。目前

CCF首个城市分部在深圳成立

杜子德（左）向CCF深圳分部主席纪震授旗

中国计算机学会首个城市会员活动中心——CCF深圳会员活动中心（CCF深圳分部）2012年10月12日成立。

CCF深圳分部成立大会在国家超级计算深圳中心举行。CCF理事长郑纬民、CCF秘书长杜子德亲临会场，深圳市副市长陈彪、科技创新委员会主任陆健、深圳市科协主席周路明、中科院深圳先进技术研究院副院长许建国应邀出席了成立大会。CCF在深圳的常务理事、CCF YOCSEF深圳成员以及深圳的会员约100人参加了会议。

杜子德在成立大会上指出，CCF城市分部是CCF为广大会员提供的又一服务平台，借助此平台，CCF总部将整合更多学术资源为会员服务，会员间也将建立更为密切的联系。陈彪副市长也对CCF深圳分部寄予厚望，“CCF深圳分部将团结并组织深圳以及珠三角广大的科技工作者，广泛地开展国内外的学术交流活动，为提高华南地区的计算机技术水平，大力推广计算机技术，并尽快促进产业化产生积极的促进作用。”

在成立大会上，杜子德向CCF深圳分部首任主席纪震教授授予了CCF深圳会员活动中心的旗帜，并向CCF深圳分部执行委员会成员颁发了聘书。CCF深圳分部的办公地点设在国家超级计算深圳中心。

YAGO2包含了1000多万实体的上亿条三元组信息。

BTC（billion triple challenge）[62]是一个从网上爬取的包含了多个数据集（如Freebase和DBpedia等）的RDF数据集。它在近几年的国际语义万维网会议（the international semantic web conference，ISWC）会议上被用作语义网挑战的基础测试数据集。该数据集每年发布一次，最新的BTC2012数据集包含了14亿条三元组，总数据量在压缩形式下为17GB左右。除了上面提到的数据集外，在万维网联盟网站上还列出了一些其它的RDF数据集，如Uni-Prot和Bio2RDF[63]。

结语

本文介绍了在海量语义网数据管理中的一个核心问题“RDF数据的存储和检索方法”，阐述了目前海量RDF存储的两种基本策略——基于关系数据库和基于图数据库以及面向RDF的查询语言SPARQL和检索方法等，讨论了面向RDF的智能检索方法。■

邹　磊

CCF会员。北京大学副教授。主要研究方向为海量图数据和RDF数据的管理，RDF知识库构建等。
zoulei@pku.edu.cn

陈跃国

CCF会员。中国人民大学副教授。主要研究方向为知识库的创建、管理和利用。chenyueguo@ruc.edu.cn

参考文献

[1] Berners-Lee, Tim; James Hendler and Ora Lassila (May 17, 2001). The Semantic Web. Scientific American Magazine. Retrieved March 26, 2008

[2] http://www.foaf-project.org/

[3] http://dublincore.org/

[4] http://www.w3.org/TR/xhtml-rdfa-primer/

[5] http://www.w3.org/PICS/

[6] RDF Current Status. http://www.w3.org/standards/techs/rdf#w3c_all

[7] http://linkeddata.org/

[8] MIT Libraries Barton Catalog Data. http://simile.mit.edu/rdf-test-data/barton/

[9] Fabian M. Suchanek, Gjergji Kasneci, Gerhard Weikum. Yago: a core of semantic knowledge. In Proceedings of WWW'2007. 697~706

[10] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Soren Auer, Christian Becker, Richard Cyganiak, Sebastian Hellmann. DBpedia - A crystallization point for the Web of Data. J. Web Sem., 2009: 154~165

[11] Kurt D. Bollacker, Robert P. Cook, Patrick Tufts. Freebase: A Shared Database of Structured General Human Knowledge. In Proceedings of AAAI'2007. 1962~1963

[12] Charlotte Jenkins, Mike Jackson, Peter Burden, Jon Wallis. Automatic RDF Metadata Generation for Resource Discovery. Computer Networks, 1999: 1305~1320

[13] Thanh Tho Quan, Siu Cheung Hui, Alvis Cheuk M. Fong, Tru Hoang Cao. Automatic Generation of Ontology for Scholarly Semantic Web. In Proceedings of International Semantic Web Conference'2004. 726~740

[14] Charlotte Jenkins, Dave Inman. Server-Side Automatic Metadata Generation using Qualified Dublin Core and RDF. In Proceedings of Kyoto International Conference on Digital Libraries'2000. 245~253

[15] Eugene Agichtein, Luis Gravano. Snowball: extracting relations from large plain-text collections. In Proceedings of ACM DL'2000. 85~94

[16] Michael J. Cafarella, Doug Downey, Stephen Soderland, Oren Etzioni. KnowItNow: Fast, Scalable Information Extraction from the Web. In Proceedings of HLT/EMNLP'2005

[17] Fabian M. Suchanek, Georgiana Ifrim, Gerhard Weikum. Combining linguistic and statistical analysis to extract relations from web documents. In Proceedings of KDD'2006. 712~717

[18] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan. A framework and graphical development environment for robust NLP tools and applications. In Proceedings of ACL'2002. 168~175

[19] Oren Etzioni, Michael J. Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, Alexander Yates. Web-scale information extraction in knowitall: (preliminary results). In Proceedings of WWW'2004. 100~110

[20] SPARQL Query Language for RDF . http://www.w3.org/TR/rdf-sparql-query/

[21] RDF Access to Relational Databases. http://www.w3.org/2003/01/21-RDF-RDB-access/

[22]W3C Semantic Web Advanced Development for Europe (SWAD-Europe). http://www.w3.org/2001/sw/Europe/reports/scalable_rdbms_mapping_report/

[23] Storing RDF in a relational database. http://infolab.stanford.edu/~melnik/rdf/db.html

[24]Zhengxiang Pan, Jeff Heflin. DLDB: Extending Relational Databases to Support Semantic Web Queries. In Proceedings of PSSS'2003

[25] Daniel J. Abadi. Column Stores for Wide and Sparse Data. In Proceedings of CIDR'2007. 292~297

[26] Jennifer L. Beckmann, Alan Halverson, Rajasekar Krishnamurthy, Jeffrey F. Naughton. Extending RDBMSs To Support Sparse Datasets Using An Interpreted Attribute Storage Format. In Proceedings of ICDE'2006. 58~58

[27] K. Wilkinson. Jena property table implementation. In SSWS. 2006

[28] Kevin Wilkinson, Craig Sayers, Harumi A. Kuno, Dave Reynolds. Efficient RDF Storage and Retrieval in Jena2. In Proceedings of SWDB'2003. 131~150

[29] Daniel J. Abadi, Adam Marcus, Samuel Madden, Kate Hollenbach. SW-Store: a vertically partitioned DBMS for Semantic Web data management. VLDB J., 2009: 385~406

[30] George P. Copeland, Setrag Khoshafian. A Decomposition Storage Model. In Proceedings of SIGMOD Conference'1985. 268~279

[31] Jennifer L. Beckmann, Alan Halverson, Rajasekar Krishnamurthy, Jeffrey F. Naughton. Extending RDBMSs To Support Sparse Datasets Using An Interpreted Attribute Storage Format. In Proceedings of ICDE'2006. 58~58

[32] Eric Chu, Jennifer L. Beckmann, Jeffrey F. Naughton. The case for a wide-table approach to manage sparse relational data sets. In Proceedings of SIGMOD Conference'2007. 821~832

[33] Thomas Neumann, Gerhard Weikum. RDF-3X: a RISC-style engine for RDF. In Proceedings of PVLDB’2008: 647~659

[34] Cathrin Weiss, Panagiotis Karras, Abraham Bernstein. Hexastore: sextuple indexing for semantic web data management. PVLDB, 2008: 1008~1019

[35] Valerie Bonstrom, Annika Hinze, Heinz Schweppe. Storing RDF as a Graph. In Proceedings of LA-WEB'2003. 27~36

[36] Renzo Angles, Claudio Gutierrez. Querying RDF Data from a Graph Database Perspective. In Proceedings of ESWC'2005. 346~360

[37] Jonathan Hayes, Claudio Gutierrez. Bipartite Graphs as Intermediate Model for RDF. In Proceedings of International Semantic Web Conference'2004.47~61

[38] Akiyoshi Matono, Toshiyuki Amagasa, Masatoshi Yoshikawa, Shunsuke Uemura. A Path-based Relational RDF Database. In Proceedings of ADC'2005. 95~103

[39] Octavian Udrea, Andrea Pugliese, V. S. Subrahmanian: GRIN: A Graph Based RDF Index. AAAI 2007: 1465~1470

[40] Paolo Ciaccia, Marco Patella, Pavel Zezula: M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. VLDB 1997: 426~435

[41] Lei Zou, Jinghui Mo, Lei Chen, M. Tamer Özsu, Dongyan Zhao: gStore: Answering SPARQL Queries via Subgraph Matching. PVLDB (2011),4(8): 482~493

[42] Uwe Deppisch: S-Tree: A Dynamic Balanced Signature Index for Office Retrieval. SIGIR 1986: 77~87

[43] Varun Kacholia, Shashank Pandit, Soumen Chakrabarti, S. Sudarshan, Rushi Desai, Hrishikesh Karambelkar. Bidirectional Expansion For Keyword Search on Graph Databases. In Proceedings of VLDB'2005. 505~516

[44] Gaurav Bhalotia, Arvind Hulgeri, Charuta Nakhe, Soumen Chakrabarti, S. Sudarshan. Keyword Searching and Browsing in Databases using BANKS. In Proceedings of ICDE'2002.431~440

[45] Hao He, Haixun Wang, Jun Yang, Philip S. Yu. BLINKS: ranked keyword searches on graphs. In Proceedings of SIGMOD Conference'2007. 305~316

[46] Thanh Tran, Haofen Wang, Sebastian Rudolph, Philipp Cimiano. Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (RDF) Data. In Proceedings of ICDE'2009. 405~416

[47] XQuery 1.0: An XML Query Language. http://www.w3.org/TR/xquery/

[48] Shady Elbassuoni, Maya Ramanath, Ralf Schenkel, Gerhard Weikum: Searching RDF Graphs with SPARQL and Keywords. IEEE Data Eng. Bull. (2010), 33(1): 16~24

[49] Jeffrey Pound, Ihab F. Ilyas, Grant Weddell. Expressive and Flexible Access to Web-Extracted Data: A Keyword-based Structured Query Language. In Proceedings of SIGMOD’2010

[50]Gaurav Bhalotia, Arvind Hulgeri, Charuta Nakhe, Soumen Chakrabarti, S. Sudarshan: Keyword Searching and Browsing in Databases using BANKS. ICDE 2002:431~440

[51]Varun Kacholia, Shashank Pandit, Soumen Chakrabarti, S. Sudarshan, Rushi Desai, Hrishikesh Karambelkar: Bidirectional Expansion For Keyword Search on Graph Databases. VLDB 2005:505~516

[52]Hao He, Haixun Wang, Jun Yang, Philip S. Yu: BLINKS: ranked keyword searches on graphs. SIGMOD 2007:305~316

[53]Guoliang Li, Beng Chin Ooi, Jianhua Feng, Jianyong Wang, Lizhu Zhou: EASE: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data. SIGMOD 2008:903~914

[54]Thanh Tran, Haofen Wang, Sebastian Rudolph, Philipp Cimiano: Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (RDF) Data. ICDE 2009:405~416

[55]Günter Ladwig, Thanh Tran: Combining Query Translation with Query Answering for Efficient Keyword Search. ESWC 2010:288~303

[56]Gideon Zenz, Xuan Zhou, Enrico Minack, Wolf Siberski, Wolfgang Nejdl: From keywords to semantic queries - Incremental query construction on the semantic web. J. Web Sem. (WS) (2009) 7(3):166~176

[57]Parthasarathy K., Sreenivasa P. Kumar, Dominic Damien: Ranked answer graph construction for keyword queries on RDF graphs without distance neighbourhood restriction. WWW (Companion Volume) 2011:361~366

[58]Shady Elbassuoni, Maya Ramanath, Ralf Schenkel, Marcin Sydow, Gerhard Weikum: Language-model-based ranking for queries on RDF-graphs. CIKM 2009:977~986

[59]Shady Elbassuoni, Roi Blanco: Keyword search over RDF graphs. CIKM 2011:237~242

[60] http://swoogle.umbc.edu/

[61] http://sindice.com/

[62] http://challenge.semanticweb.org/

[63] http://www.w3.org/wiki/DataSetRDFDumps

[64] Francesca Bugiotti, François Goasdoué, Zoi Kaoudi, Ioana Manolescu: RDF data management in the Amazon cloud. EDBT/ICDT Workshops 2012: 61~72

[65] Mohammad Farhan Husain, James P. McGlothlin, Mohammad M. Masud, Latifur R. Khan, Bhavani M. Thuraisingham: Heuristics-Based Query Processing for Large RDF Graphs Using Cloud Computing. IEEE Trans. Knowl. Data Eng. (2011)23(9): 1312~1327

[66] Nikolaos Papailiou, Ioannis Konstantinou, Dimitrios Tsoumakos, Nectarios Koziris: H2RDF: adaptive query processing on RDF data in the cloud. WWW (Companion Volume) 2012: 397~400