HDVM:基于关系矩阵的关联数据压缩查询模型

符海东[1,2],彭 燊[1,2,3],黄 莉[1,2],顾进广[1,2,3,4]
(1. 武汉科技大学,计算机科学与技术学院,湖北武汉 430065;
2. 智能信息处理与实时工业系统湖北省重点实验室,湖北武汉 430065;
3. 国家新闻广电出版总局富媒体数字出版内容组织与知识服务重点实验室,北京 100038;
4. 湖北语言与智能信息处理研究基地(武汉大学),湖北武汉 430072)

摘 要: 随着大数据时代的到来,大量的RDF数据充斥着整个数据网络. RDF(Resource Description Framework)后台引擎管理巨大的数据集时,数据集索引不能全部加载到内存中,导致系统需要执行缓慢的磁盘访问来解决SPARQL查询. 本文提出了一种HDVM(Header Dictionary Vector Matrix)压缩查询模型,通过在关联数据集中提取潜在的三元组关系矩阵,以主语向量、谓语向量和宾语矩阵的模型序列化存储来减少关联数据重复出现的次数,允许SPARQL查询在压缩状态下全内存执行. 实验结果表明,本文提出的模型比常用的HDT(Header-Dictionary Triples)压缩方式提高了3% ~20%的压缩率,同时在三元组个数达到十亿级别的数据集上平均查询时间在400ms左右.

关键词: 关系矩阵;关联数据;查询;压缩
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2018) 03-0721-09
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2018.03.030

HDVM: Compression & Query Model of Linked-Data Based on Relational Matrix

FU Hai-dong[1,2], PENG Shen[1,2,3], HUANG Li[1,2], GU Jin-guang[1,2,3,4]
(1. *College of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan, Hubei* 430065, *China*;
2. *Hubei Province Key Laboratory of Intelligent Information Processing and Real Time Industrial System, Wuhan, Hubei* 430065, *China*;
3. *Key Laboratory of Rich-media Knowledge Organization and Service of Digital Publishing Content, SAPPRFT, Beijing* 100038, *China*;
4. *Language and Intelligent Information Processing Research Base (Wuhan University), Wuhan, Hubei* 430072, *China*)

Abstract: With the arrival of big data era, a large number of RDF(Resource Description Framework) data is flooding the entire Web of Data. Since the indexes of these datasets cannot be fully loaded in main memory when the RDF engines manage these huge datasets, these systems need to perform slow disk accesses to solve SPARQL queries. In this paper, a method named HDVM is proposed to reduce the number of linked data repeated times by extracting the latent triplet relation matrix from the linked dataset, and storing them in the form of subject vector, predicate vector and object matrix, which allows SPARQL queries to be full-in-memory performed without decompression. The experimental results show that the HDVM(Header Dictionary Vector Matrix) model proposed in this paper can improve the compression rate by 3% ~20% compared with HDT(Header-Dictionary Triples), and the query time on billion-level-size dataset reaches average 400 milliseconds.

Key words: relation matrix; linked-data; query; compression

收稿日期:2016-11-07; 修回日期:2017-05-18; 责任编辑:梅志强
基金项目: 国家自然科学基金(No. 61673304, No. 61272110);国家社会科学基金重大计划(No. 11&ZD189);软件工程国家重点实验室(武汉大学)开放基金(No. SKLSE2012-09-07)

1 引言

W3C 最先提出将 RDF(Resource Description Framework)[1]作为处理元数据的基础,其目的在于定义一种广泛认可的资源描述机制. RDF 的提出很显然是受到 Web 信息交换过程中以文档数据为中心观点的影响. 随着 RDF 不断演变和进化,人们致力使 RDF 实现信息处理自动化,就像万维网的超链接允许数据能在自身被创建的环境外被访问和处理一样. 因此,RDF 成为信息自动化处理以及关联数据研究重点.

关联数据采用 RDF 和 HTTP 两种方式发布结构化数据,并将它们从不同的数据源中关联起来,这让网络从最初以文档为中心逐渐向以数据为中心转变. RDF 提供了一种基于图形的数据模型来描述真实世界的结构化关联数据,为 Web 数据研究奠定了基础[2]. RDF 基于主谓宾原子三元组(S,P,O)进行建模,其中主语"S"是被描述的资源,谓语"P"是资源属性,宾语"O"是资源属性值.

网络数据包含不同领域的、海量的 RDF 数据,当需要管理和查询这些数据时,查询性能和数据可扩展性便成了焦点. 越来越庞大的 RDF 数据可以使用足够多的存储介质来存储,但是庞大的数据集不仅会导致查询效率降低,还会加剧其它常见过程的性能问题,如 RDF 发布和交换. 随着远程执行 SPARQL 方式越来越受欢迎,RDF 的发布和交换在关联数据的查询中越来越频繁. 因此减小关联数据集体积的同时提高查询效率具有重大意义,而如何构建一个合适的关联数据压缩查询模型成了关键性问题. 本文提出一种内存压缩查询模型,在关联数据的压缩和查询中找到一个平衡点.

2 关联数据压缩研究现状

Wu 等人[3]根据对关联数据冗余类型的分析和对现有压缩方案的总结,将现有的压缩技术分成语法压缩和语义压缩两类. 语义压缩指利用规则推理减少关联数据中三元组的数量以达到压缩的目的. 但在数据压缩过程中,语义压缩的效果并没有语法压缩好.

语法压缩指通过改变关联数据的文件结构并对数据集进行序列化,用更简洁的数据结构[4]表示关联数据中的信息,从而达到压缩的目的. Fernández 等人[5]提出了压缩关联数据的基础方案,利用一般的文件压缩技术(如 LZMA[6]和 bizp2[7])对关联数据的文件结构进行了修改,能够明显减小文件体积. Atre 等人[8]提出了 BitMat 方案,BitMat 用一种紧凑的比特矩阵结构来存储关联数据集,并且支持在压缩数据集上直接进行查询操作,但是这种方案的扩展性不高. 与 BitMat 方法不同,K^2-triple[9,10]采用了基于谓词的方法将数据集中的三元组分为了不相交的子集对〈主语,宾语〉,并对这些子集对进行了高度压缩. 实验结果表明 K^2-triple 方案具有较好的压缩效果,但 K^2-triple 并没有提出一种序列化模型.

文献[11,12]介绍了一种新的关联数据压缩方案 HDT,将关联数据中的信息分解成三个部分:头部(Header)、字典(Dictionary)、三元组(Triples). 头部保存了描述整个数据集的逻辑和物理元数据,例如数据集的来源、编辑信息以及数据集的一些统计信息等;字典部分主要将数据集中的资源映射为唯一标识;三元组部分将数据集中的三元组字典化后,再根据三元组的主语信息将三元组分组,最后对这些三元组进行压缩. HDT 保存了数据集的底层结构,同时避免了数据集中较长及重复的资源描述. 它是一种比较有效的压缩方案,经过 HDT 压缩的文件还可通过 bzip2 等文件压缩技术进行二次压缩. 受 HDT 方案的启发,一些基于 HDT 方案的压缩技术相继被提出,如 HDT FoQ[13]、WaterFowl[14]、HDT++[15].

大型关联数据的压缩一般比较耗时,于是 Urbani 等人[16]将字典编码技术与 MapReduce[17]相结合,采用分布式方案快速的压缩大型数据集. 文献[18]也同样将 HDT 与 MapReduce 结合在一起,加快了 HDT 的压缩速度.

上述的压缩方法,在不同的适用场景下都具有良好的压缩率. 但是除 K^2-triple、BitMat 外的压缩方法并不支持在压缩状态下进行查询[19]. 本文提出的基于分块向量矩阵的关联数据压缩查询模型不仅具有良好的压缩率,还可在压缩状态下直接进行查询操作.

3 构建压缩查询模型

3.1 关联数据冗余分析

目前关联数据语法压缩从原理上可以分为以下两类:

(1)基于关系三维矩阵的压缩方法. 这种压缩方案在结构上形象呈现了主语、谓语和宾语存在的关联关系. 但为了维护矩阵的结构,该类方法存储了部分不存在的关联关系,特别是当关联数据集大到一定规模的时候,关联数据的三维矩阵就是一个超级稀疏矩阵(如图 1 所示,本文 4.1 节实验数据集中 DBLP 的三维空间矩阵),直接存储三维矩阵将会产生很多冗余信息. 虽然这种存储结构对查询操作很友好,但在压缩率不够显著.

(2)通过提取关联数据潜在的三元组关联规则,建立线性序列化和线性比特预测位作为存储结构. 该类方法去掉了很多重复的数据项,因此压缩率比较高. 这类方法虽然在语法结构上达到了压缩的极致,但是在

压缩状态下,主语、谓语和宾语的关联关系并不明显,因此对查询操作的支持并不友好.

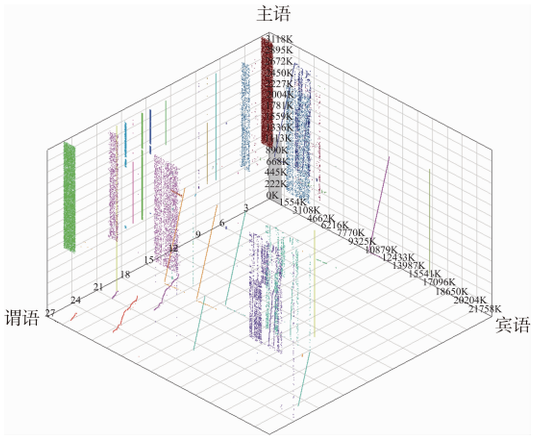


图1　关联数据的三维矩阵空间

基于关系三维矩阵的压缩方法的冗余信息主要来自于三维空间矩阵中不存在的关联关系. 如图 1 所示,三维坐标轴分别表示数据集的主语、谓语和宾语. 在三维坐标系中,离散点表示存在关联关系的数据,其它的空白空间是冗余信息. 只要将不存在的关联关系从矩阵中移除,那么压缩率也就随之提高.

基于上文描述的两类压缩方法,本文提出了一种基于主语向量、谓语向量和宾语矩阵的压缩查询模型,简称 HDVM(Header Dictionary Vector Matrix). 该方法通过挖掘关联数据潜在的三元组关系,减少重复存储的数据项,重新构建了三维矩阵的模型,减小了存储占用空间,允许查询在未解压缩状态下全内存执行. 这种压缩查询模型综合了第一种压缩方法对查询的良好支持和第二种方法的高压缩率特性.

3.2　HDVM 关系矩阵推导

HDVM 模型的核心思想是基于主语和谓语建立两层索引,从三维矩阵空间中去除零块,只记录非零块. 建立索引需要挖掘数据集中潜在的笛卡尔积关系矩阵,本文通过主语向量、谓语向量和宾语矩阵描述笛卡尔积关系. 图 2 介绍了如何按照推导规则挖掘数据集中潜在的关系矩阵.

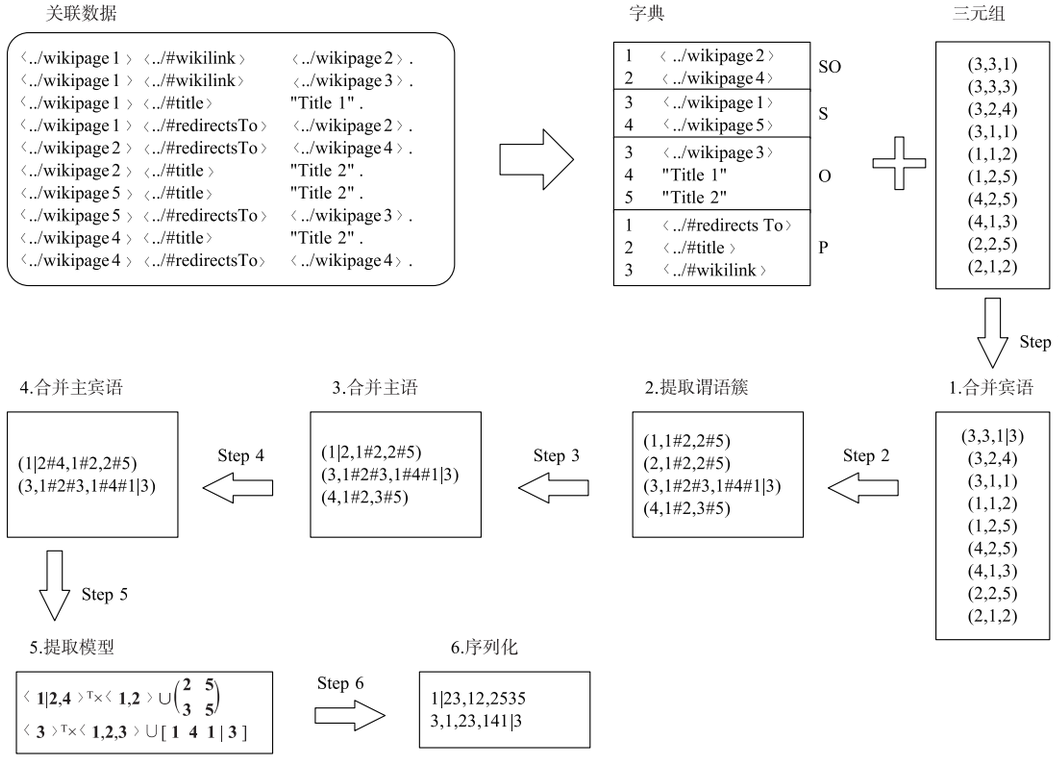


图2　HDVM关系矩阵推导

定义　$(s_?,p_?,o_?)$为任意三元组. 对任意的三元组$(s_?,p_?,o_?)$有如下推导规则.

推导规则 1　对同主语、谓语的三元组合并宾语:

$$(s_1,p_1,o_1)+(s_1,p_1,o_2)\rightarrow(s_1,p_1,o_1|o_2) \quad (1)$$

对应图 2 中 Step 1,

$$(3,3,1)+(3,3,3)\rightarrow(3,3,1|3)$$

推导规则 2　对每个主语分类,提取谓语簇(将每个主语所有谓语构成的集合称为一个谓语簇)并按谓语簇排序:

$$(s_1,p_1,o_1)+(s_1,p_2,o_2)\rightarrow(s_1,p_1\#p_2,o_1\#o_2) \tag{2}$$

对应图 2 中 Step 2,

$$(1,1,2)+(1,2,5)\rightarrow(1,1\#2,2\#5)$$
$$(2,1,2)+(2,2,5)\rightarrow(2,1\#2,2\#5)$$
$$(3,1,1)+(3,2,4)+(3,3,1|3)\rightarrow(3,1\#2\#3,1\#4\#1|3)$$
$$(4,1,3)+(4,2,5)\rightarrow(4,1\#2,3\#5)$$

推导规则 3 对相同谓语(谓语簇)、宾语(宾语向量)的三元组合并主语:

$$(s_1,p_1,o_1)+(s_2,p_1,o_1)\rightarrow(s_1|s_2,p_1,o_1) \tag{3}$$

对应图 2 中 Step 3,

$$(1,1\#2,2\#5)+(2,1\#2,2\#5)\rightarrow(1|2,1\#2,2\#5)$$

推导规则 4 对每个谓语簇分类,合并主语和宾语向量:

$$(s_1,p_1\#p_2,o_1\#o_2)+(s_2,p_1\#p_2,o_3\#o_4)\rightarrow(s_1|s_2,p_1\#p_2,o_1\#o_2/o_3\#o_4) \tag{4}$$

对应图 2 中 Step 4,

$$(1|2,1\#2,2\#5)+(4,1\#2,3\#5)\rightarrow(1|2\#4,1\#2,2\#5/3\#5)$$

经过上述的推导规则可以得到初步的关联矩阵信息,接着从得到的矩阵关系中提取主语向量、谓语向量和宾语矩阵. 对矩阵关系$(s_1|s_2,p_1\#p_2,o_1\#o_2/o_3\#o_4)$的提取结果如下,主语向量$\langle s_1,s_2\rangle^{\mathrm{T}}$,谓语向量$\langle p_1,p_2\rangle$,宾语矩阵$\begin{pmatrix}o_1 & o_2\\ o_3 & o_4\end{pmatrix}$,其中主语向量和谓语向量不会存在重复元素.

定义 ×为向量的笛卡尔积乘法运算符,其运算规则如下:

$$\langle s_1,s_2\rangle^{\mathrm{T}}\times\langle p_1,p_2\rangle=\begin{pmatrix}s_1p_1 & s_1p_2\\ s_2p_1 & s_2p_2\end{pmatrix}$$

定义 ∪为矩阵的一种加法运算符,其运算规则如下:

$$\begin{pmatrix}x_{11} & x_{12}\\ x_{21} & x_{22}\end{pmatrix}\cup\begin{pmatrix}o_{11} & o_{12}\\ o_{21} & o_{22}\end{pmatrix}=\begin{pmatrix}x_{11}o_{11} & x_{12}o_{12}\\ x_{21}o_{21} & x_{22}o_{22}\end{pmatrix}$$

那么有

$$\langle s_1,s_2\rangle^{\mathrm{T}}\times\langle p_1,p_2\rangle\cup\begin{pmatrix}o_1 & o_2\\ o_3 & o_4\end{pmatrix}=\begin{pmatrix}s_1p_1o_1 & s_1p_2o_2\\ s_2p_1o_3 & s_2p_2o_4\end{pmatrix}$$

根据上述运算法则,可以使用主语向量$\langle s_1,s_2\rangle^{\mathrm{T}}$、谓语向量$\langle p_1,p_2\rangle$和宾语矩阵$\begin{pmatrix}o_1 & o_2\\ o_3 & o_4\end{pmatrix}$来表示四个三元组$(s_1,p_1,o_1)$、$(s_1,p_2,o_2)$、$(s_2,p_1,o_3)$、$(s_2,p_2,o_4)$.

如图 2 中 Step 5:

$$(1|2\#4,1\#2,2\#5/3\#5)\leftrightarrow\langle\mathbf{1|2,4}\rangle^{\mathrm{T}}\times\langle\mathbf{1,2}\rangle\cup\begin{pmatrix}\mathbf{2} & \mathbf{0}\\ \mathbf{3} & \mathbf{1}\end{pmatrix}$$

$$(3,1\#2\#3,1\#4\#1|3)\leftrightarrow\langle\mathbf{3}\rangle^{\mathrm{T}}\times\langle\mathbf{1,2,3}\rangle\cup\begin{pmatrix}\mathbf{2} & \mathbf{0}\\ \mathbf{3} & \mathbf{1}\end{pmatrix}$$

将这样的一个主语向量、一个谓语向量和一个宾语矩阵组成的结构称为一个数据块,一个数据集可以由多个数据块组成.

3.3 HDVM 模型

经过上述推导规则,可以构建出 HDVM 的内存和序列化模型,如图 3 所示,其中每一行的主语向量、谓语向量和宾语矩阵构成一个数据块.

头部信息		
字典		
主语向量	谓语向量	宾语矩阵
主语向量	谓语向量	宾语矩阵
主语向量	谓语向量	宾语矩阵
...		

图3 HDVM内存和序列化模型

HDVM 模型序列化写入文件需要三个特殊字符来标识具有特殊意义的符号,或标识符"|",主语向量、谓语向量和宾语矩阵分隔符",",和数据块分隔符,如图 2 中 Step 6 所示.

本文为这三个特殊符号建立了字典映射并为其分配 ID,在写入文件时设定每个 ID 占用的比特长度相同. 以二进制格式的形式将 HDVM 模型序列化到文件,可以形成一个紧凑的文件结构,减少了 HDVM 存储占用的磁盘空间.

3.4 HDVM 并发查询方案

经过 HDVM 方法压缩的数据保存着数据间直观的关联关系. HDVM 将数据存储成向量和矩阵形式,主语向量和谓语向量内部均已排序,因此在做主谓语查询时可以使用折半查找方式. 同时由于数据集被拆分为多个数据块,数据块之间没有重复数据项,因此在数据集上做查询时,可以为每个数据块开启一个线程进行查询操作,最后将查询结果进行聚合操作,这样的方法极大地提高了查询效率,如图 4 所示.

定义 (S,P,O)为三元组查询模式,使用"?"标识不确定的元素. HDVM 压缩查询模型能实现七种基本的查询模式.

① 基本匹配模式(S,P,O)用于查询给出的三元组信息是否存在. 并发遍历数据集中的数据块,如果在某个数据块中,主语向量中存在 S,谓语向量中存在 P,

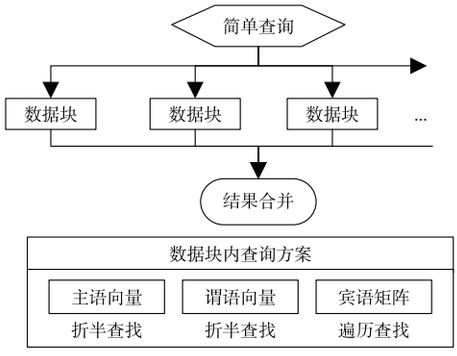


图4　HDVM模型并发查询方案

且宾语矩阵中对应的宾语是 O,则查询的三元组存在.

② 匹配模式(S,? P,O)用于查询符合指定主语和宾语的三元组. 并发遍历数据集中的数据块,如果在每个数据块中,主语向量中存在 S 且宾语矩阵中存在 O,取出所有对应的三元组即为匹配结果.

③ 匹配模式(S,P,? O)用于查询符合指定主语和谓语的三元组. 并发遍历数据集中的数据块,如果在每个数据块中,主语向量中存在 S 且谓语向量中存在 P,取出所有数据块中符合条件的三元组即为匹配结果.

④ 匹配模式(? S,P,O)用于查询符合指定谓语和宾语的三元组. 并发遍历数据集中的数据块,如果每个数据块中,谓语向量中存在 P 且宾语矩阵中存在 O,取出所有数据块中符合条件的三元组即为匹配结果.

⑤ 匹配模式(? S,P,? O)用于查询指定谓语的三元组. 并发遍历数据集中的数据块,如果数据块谓语向量存在 P,取出所有谓语为 P 的三元组即为匹配结果.

⑥ 匹配模式(S,? P,? O)用于查询指定主语的三元组. 并发遍历数据集中的数据块,如果数据块主语向量存在 S,取出所有主语为 S 的三元组即为匹配结果.

⑦ 匹配模式(? S,? P,O)用于查询指定宾语的三元组. 并发遍历数据集中的数据块,如果数据块宾语矩阵存在 O,取出所有宾语为 O 的三元组即为匹配结果.

根据查询匹配模式可知,HDVM 方法进行主语和谓语查询时具有并发和折半查找的优势,而对于宾语查询仅具有并发查询的优势. 鉴于每个主语在所有的主语向量中仅出现一次,本文提出一种数据块拆分的方法. 如下所示,将一个较大的数据块

$$\langle \boldsymbol{s}_1,\boldsymbol{s}_2,\boldsymbol{s}_3,\boldsymbol{s}_4\rangle^{\mathrm{T}}\times\langle \boldsymbol{p}_1,\boldsymbol{p}_2\rangle\cup\begin{pmatrix}\boldsymbol{o}_{11} & \cdots & \boldsymbol{o}_{14}\\ \vdots & \ddots & \vdots\\ \boldsymbol{o}_{21} & \cdots & \boldsymbol{o}_{24}\end{pmatrix}$$

分解为

$$\langle \boldsymbol{s}_1,\boldsymbol{s}_2\rangle^{\mathrm{T}}\times\langle \boldsymbol{p}_1,\boldsymbol{p}_2\rangle\cup\begin{pmatrix}\boldsymbol{o}_{11} & \boldsymbol{o}_{12}\\ \boldsymbol{o}_{21} & \boldsymbol{o}_{22}\end{pmatrix}$$

和

$$\langle \boldsymbol{s}_3,\boldsymbol{s}_4\rangle^{\mathrm{T}}\times\langle \boldsymbol{p}_1,\boldsymbol{p}_2\rangle\cup\begin{pmatrix}\boldsymbol{o}_{13} & \boldsymbol{o}_{14}\\ \boldsymbol{o}_{23} & \boldsymbol{o}_{24}\end{pmatrix}$$

两个较小的数据块. 在保证 HDVM 模型所有特性的基础上,减小了宾语矩阵的大小,增加了数据块的数量,提高了查询操作中可启用并发线程的数量,弥补了大数据块上宾语匹配查询的劣势.

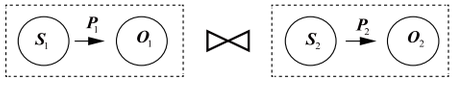


图5　连接查询

将上述七种基本匹配模式进行连接操作可用做更复杂的查询[20]. 如图 5 所示,连接查询分为主语连接主语(Subject-Subject)、主语连接宾语(Subject-Object)、宾语连接宾语(Object-Object)和谓语连接谓语(Predicate-Predicate)四类连接操作. 匹配模式(S_1,P_1,O_1)和匹配模式(S_2,P_2,O_2)进行连接查询时,匹配模式(S_1,P_1,O_1)和匹配模式(S_2,P_2,O_2)中必定有一个已知值,先将查询拆分为两个基本查询,再将两个查询结果做聚合操作. 聚合操作包含集合的差集、并集、交集和补集. 例如,将(S_a,? P,? O)和(? S,? P,O_b)进行主语连接宾语的查询,其中 S_a 和 O_b 是已知的主语和宾语,则可以拆分为两个简单查询(S_a,? P,? O)和(? S,? P,O_b),再将两个查询结果按照主语和宾语连接关系进行交集操作.

4　实验结果分析

4.1　实验数据集

为了验证 HDVM 模型的有效性,本文对表 1 中的数据集进行了构建压缩查询模型的实验. 实验数据集按从小到大排序,其中谓语数量表示关联数据集中出现的谓语总个数,谓语簇数量是指关联数据集中所有不相同的谓语簇的数量. 从后续实验结果中得知,内存压缩模型最终的压缩率和查询效率与实验数据集的谓语簇数量具有一定关系,谓语簇的数量是衡量 HDVM 在一个关联数据集上实验效果优劣的重要指标.

4.2　HDVM 模型构建结果

在构建 HDVM 模型的过程中,本文统计了不同阶段元数据(本文将任意一个主语、谓语或者宾语称为元数据)的总数量,如表 2 所示. 从表 2 可以看出,从语义的角度上看,HDVM 模型减少了一半以上的元数据数量,压缩率能达到 62% 以上. 但是在将 HDVM 模型序列化到磁盘的时候,需要特殊辅助符号来维持模型的结

构,因此 HDVM 实际的压缩率相较语义压缩率来说有所下降,具体降低的幅度取决于数据集的初始分布.

表 1 实验数据集

数据集	文件大小	三元组数量	谓语数量	谓语簇数量
CN2012	17.7M	137,441	26	4
DogFood	38.7M	242,256	132	1001
Archive Hub	71.7M	431,088	52	611
Jamendo	143.9M	1,047,950	336	32
LinkedMdb	850.0M	6,147,996	694	8460
DBpedia rdftypes	1.20G	9,237,320	1	1
DBLP	8.84G	55,586,971	27	250

下载链接:

CN2012 (https://datahub.io/dataset/combined-nomenclature-2012)

DogFood (https://datahub.io/dataset/scholarlydata)

Archive Hub (https://datahub.io/dataset/archiveshub-linkeddata)

Jamendo (https://datahub.io/dataset/jamendo-dbtune)

LinkedMdb (https://datahub.io/dataset/linkedmdb)

DBpedia rdftypes (http://downloads.dbpedia.org/3.6/en/instance_types_en.nt.bz2)

DBLP (https://datahub.io/dataset/rkb-explorer-dblp)

假设一个数据集中有 N 个数据块,三元组总数为 total,每块数据块的主语向量长度和谓语向量长度分别为 $ls_1 \sim ls_N$ 和 $lp_1 \sim lp_N$. 虽然"|"关系能极大地减少元数据的数量,提高压缩率,但由于该种关系不利于实验分析,因此在分析的过程中假设 HDVM 模型中不存在"|"关系. 则有如下关系,

$$\text{total} = \sum_{i=1}^{N} ls_i * lp_i$$

有效减少的元数据数量为:

$$\begin{aligned}\text{count} &= \sum_{i=1}^{N} (2 * ls_i * lp_i - ls_i - lp_i) \\ &= 2 * \text{total} - \sum_{i=1}^{N} ls_i - \sum_{i=1}^{N} lp_i\end{aligned}$$

其中 $\sum_{i=1}^{N} ls_i$ 是数据集中主语的个数, $\sum_{i=1}^{N} lp_i$ 是所有谓语簇长度之和. 在三元组数量不变的情况下,数据集中主语数量与所有谓语簇长度之和越小(从三维矩阵空间的角度来看,如图 1 所示,点在三维空间上分布越均匀),减少的元数据数越多,压缩效果越好.

表 2 构建 HDVM 模型各阶段元数据的数量

数据集	初始数量	规则 1	规则 2	规则 3	规则 4	语义压缩率(%)
CN2012	412,323	332,941	249,743	249,743	152,025	63.2
DogFood	726,768	578,794	433,835	431,507	275,219	62.2
Archive Hub	1,293,264	1,110,108	822,009	690,127	421,769	67.4
Jamendo	3,143,850	2,779,624	2,249,712	1,673,538	1,095,940	65.2
LinkedMdb	18,444,363	17,140,959	12,338,940	12,337,466	6,977,111	62.2
DBpedia rdftypes	27,750,234	12,912,516	12,912,516	1,832,787	1,832,528	93.4
DBLP	166,760,913	130,381,443	96,325,342	96,324,957	58,931,325	64.7

4.3 压缩实验结果

HDVM 模型序列化需要三个特殊字符来标识三个具有特殊意义的符号,因此实际的压缩率相较表 2 中的语义压缩率要低一些,低出的这部分是三个特殊字符占用所导致. 表 3 统计了 HDVM、HDT 和 HDT++压缩文件大小(本文只统计三元组,暂不考虑 Header 和 Dictionary)和压缩率,PLAIN 表示二进制编码下的三元组数据大小,Size 表示压缩后的大小,Rate 表示压缩率.

从表 3 中可以看出,在大部分数据集上,HDVM 压缩方式比 HDT 更高效,但是低于 HDT++. 究其原因,HDT++是一种文件存储结构,它采用了二级字典的结构设计,去除了很多冗余信息,有利于数据压缩. 如果以它原始的结构直接载入内存,并不利于查询,如果将其解压后载入内存,通常会因为解压后的数据太大而无法完全载入内存. 但 HDVM 不仅是一种文件存储结构,而且是一种内存模型,它可以按照文件存储的原始结构,以向量和矩阵模型的形式在未解压的状态下完全载入内存.

利用 HDVM 模型的数据分块特性实现了并发查询,并且在数据块内部使用二分查找方式进行主语和谓语匹配,相较于大部分查询方法,具有显著的查询优势.

表 3　压缩实验结果(三元组)

数据集	PLAIN(MB)	HDT		HDVM		HDT++	
		Size(MB)	Rate(%)	Size(MB)	Rate(%)	Size(MB)	Rate(%)
CN2012	0.54	0.34	37.1	0.30	44.5	0.20	63.0
DogFood	1.04	0.64	38.5	0.61	41.4	0.40	61.5
Archive Hub	1.85	1.19	35.7	1.05	43.3	0.69	62.7
Jamendo	4.87	3.20	34.3	3.15	35.3	1.75	64.1
LinkedMdb	34.00	22.21	34.7	14.82	56.4	12.02	64.7
DBpedia rdftypes	49.62	21.14	57.4	49.62	0	17.64	64.5
DBLP	337.95	199.52	41.0	193.15	42.9	119.29	64.7

4.4　查询实验结果

设定关联数据集 S,在构建 HDVM 模型后有 N 个数据块,查询每个数据块花费的时间分别为 $t_1 \sim t_N$,查询结果合并时间为 t_s,那么在并行查询下耗费的最大时间为 $\max\{t_1 \sim t_N\} + t_s$. 对于数据集 S 来说,并行查询的时间取决于所有数据块的最大查询耗时,最大查询耗时又与数据块的主语向量长度、谓语向量长度和宾语矩阵大小相关. 假设一个数据集的最大主谓宾语长度为(l_s, l_p, l_o). 由于主语向量和谓语向量是有序的,因此可采用二分查找,那么最大比较次数是 $\log_2(l_s)$和$\log_2(l_p)$.

按照大数据块拆分方法,将宾语矩阵的大小阈值设置为5000,那么所有的数据集宾语查询的最大比较次数为5000. 所有的查询模式都可以由基本查询模式$(S,?\ P,?\ O)$、$(?\ S,P,?\ O)$和$(?\ S,?\ P,O)$组合而成. 表4统计了实验数据集在不同查询模式下的最大比较次数.

表 4　简单查询模式的最大比较次数(次)

数据集	l_s	l_p	l_o	$(S,?\ P,?\ O)$	$(?\ S,P,?\ O)$	$(?\ S,?\ P,O)$
CN2012	9650	11	67550	14	4	5000
DogFood	2207	24	15237	12	5	5000
Archive Hub	7022	36	55467	13	6	5000
Jamendo	125948	9	273804	17	4	5000
LinkedMdb	179161	31	1612449	18	5	5000
DBpedia rdftypes	1831219	1	1308	21	1	1308
DBLP	1178718	18	18165292	21	5	5000

将工作线程(并发数量)设置为20,实验统计了每个数据集的分块数量和每种查询匹配模式在不同数据集上的耗时. 由于实验机器是多核 CPU,受其操作系统进程调度和查询语句命中的影响,每次实验结果存在一定的误差,因此本文以多次实验的平均值作为最终实验结果,如表5所示.

表 5　并发查询实验结果(单位:毫秒)

数据集	分块数量	$(S,?\ P,?\ O)$	$(?\ S,P,?\ O)$	$(?\ S,?\ P,O)$	$(S,P,?\ O)$	$(?\ S,P,O)$	$(S,?\ P,O)$
CN2012	22	1.9	5.6	2.8	1.8	1.8	1.5
DogFood	1007	23.5	27.6	29.1	29.3	32.2	31.7
Archive Hub	646	19.7	17.8	15.6	21.6	22.7	22.3
Jamendo	198	7.9	10.1	13.8	9.7	8.9	11.4
LinkedMdb	9418	324.4	235.7	238.8	334.6	365.3	341.6
DBpedia rdftypes	367	13.4	2137.3	67.3	18.5	1135.5	13.7
DBLP	7670	367.7	2116.2	369.4	379.1	1249.7	401.1

从表5可以看出,HDVM 模型下的并发查询耗时和表4的最大比较次数并不成正比,这是因为表4是在查询系统并发能力充足的理想环境下的推导结果,而表5是在查询系统最大并发能力限制为20的实验结

果. 同时,查询匹配模式(? S,P,? O)和(? S,P,O)相较其它查询匹配模式耗时较高,对实验结果进行分析,发现这两种匹配模式拥有较大的查询结果集,至少有几十万条数据.

大部分体验交互式系统取到部分结果就返回,以达到及时响应请求和改善用户体验的目的. 而本文的查询系统是取得所有的查询结果后再返回,因此对于相同的查询匹配模式,返回结果集越大,用于构建返回对象的耗时就越长,总体查询时间也会相应增加. 本文统计了在 DBLP 数据集上查询结果数量和查询耗时关系曲线图,如图 6 所示.

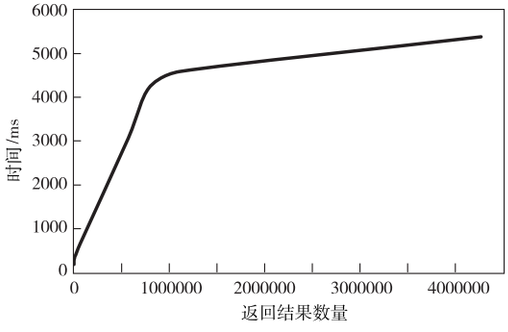


图6 查询结果数量与耗时曲线图

从图 6 可以看出,查询结果数量和耗时存在一定的关系. 当返回结果数量小于 8E +6 时,查询时间和查询结果数量大致成正比,斜率为 k_1;当返回结果数量大于 8E +6 时,查询时间和查询结果数量以 k_2 为斜率大致成正比,且 $k_2 < k_1$. 分析结果可知,当返回结果小于 8E +6 时,用于查找匹配的时间比重较高,用于结果聚合的时间比重较低. 当返回结果的数量大于 8E +6 时,在查询匹配时间稳定的情况下,查询系统做数据块的结果聚合操作增加幅度较小,所以 $k_2 < k_1$.

为了验证 HDVM 模型查询的高效性,本文在数据集 DBLP 上使用了 K^2-triple 方式(工作线程同样限制为 20)进行查询,统计了不同匹配模式下的查询耗时,如图 7 所示. 实验结果表明除了查询匹配模式(S,P,O),在其它的查询匹配模式上 HDVM 相对于 K^2-triple 方式都具有较大的查询优势.

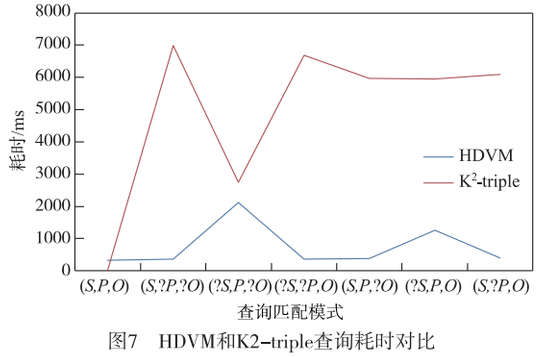


图7 HDVM和K2-triple查询耗时对比

5 结论和展望

本文从语法压缩和语义压缩出发,总结了两者的优缺点,综合两者优势而提出本文的压缩方法:基于主语向量、谓语向量和宾语矩阵的关联数据压缩. 该方法在体积压缩和查询效率之间取得了一个较好的平衡点,兼顾查询的前提下具有良好的压缩率,缓解了关联数据在网络上存储传输和查询之间的矛盾.

但在关联数据增加或减少数据项时,HDVM 必须要重新进行建模操作,因此本文后续将把 HDVM 的动态建模作为研究重点. 同时从实验结果中可以看出,宾语矩阵中存在一些相同的行数据或者列数据,因此块内的行(列)压缩也是本文后续工作之一.

参考文献

[1] PAN J Z. Resource Description Framework[M] Berlin Heidelberg:Springer,2009. 71 – 90.

[2] BIZER C,TOM H,TIM B L,et al. Linked data:the story so far[J]. International Journal on Semantic Web & Information Systems,2009,5(3):1 – 22.

[3] WU H,VILLAZON-TERRAZAS B,PAN J Z,et al. How redundant is it? -an empirical analysis on linked datasets [A]. Proceedings of the 5th International Conference on Consuming Linked Data-Volume 1264 [C]. Aachen:CEUR-WS. org,2014. 97 – 108.

[4] NAVARRO G,MÄKINEN V. Compressed full-text indexes[J]. ACM Computing Surveys (CSUR),2007,39(1):2.

[5] FERNáNDEZ J D,GUTIERREZ C,MARTíNEZ-PRIETO M A. RDF compression:basic approaches[A]. International Conference on World Wide Web[C]. New York:ACM,2010. 1091 – 1092.

[6] PAVLOV I. Lzma sdk (software development kit)[OL]. http://www. 7-zip. org,2013-06-16.

[7] SEWARD J. bzip2 and libbzip2[OL]. http://www. bzip. org,2007-12-10.

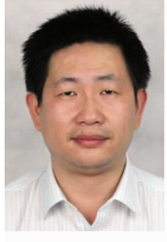
[8] ATRE M,CHAOJI V,ZAKI M J,et al. Matrix Bit loaded:a scalable lightweight join query processor for RDF data [A]. Proceedings of the 19th International Conference on World Wide Web[C]. New York:ACM,2010. 41 – 50.

[9] ÁLVAREZGARCÍA S,BRISABOA N R,FERNÁNDEZ J D,et al. Compressed k2-triples for full-in-memory RDF engines[A]. Americas Conference on Information Systems [C]. Detroit:MACIS,2011.

[10] ÁLVAREZ-GARCÍA S,BRISABOA N,FERNÁNDEZ J D,et al. Compressed vertical partitioning for efficient RDF management[J]. Knowledge and Information Sys-

tems,2015,44(2):439 – 474.
[11] FERNÁNDEZ J D,MARTÍNEZ-PRIETO M A,GUTIERREZ C. Compact Representation of Large RDF Data Sets for Publishing and Exchange[M]. Heidelberg, Berlin: Springer,2010. 193 – 208.
[12] FERNÁNDEZ J D,MARTÍNEZ-PRIETO M A,GUTIÉRREZ C,et al. Binary RDF representation for publication and exchange (HDT)[J]. Web Semantics Science Services & Agents on the World Wide Web,2013,19(1):22 – 41.
[13] MARTÍNEZ-PRIETO M A,GALLEGO M A,FERNÁNDEZ J D. Exchange and consumption of Huge RDF Data[M]. Heidelberg,Berlin:Springer,2012. 437 – 452.
[14] CURÉ O,BLIN G,REVUZ D,et al. Waterfowl:A Compact,Self-Indexed and Inference-Enabled Immutable RDF Store[M]. Heidelberg,Berlin:Springer International Publishing,2014. 302 – 316.
[15] HERNÁNDEZ-ILLERA A,MARTINEZ-PRIETO M A,FERNÁNDEZ J D. Serializing RDF in compressed space[A]. Data Compression Conference (DCC)[C]. New York:IEEE Press,2015. 363 – 372.
[16] URBANI J,MAASSEN J,DROST N,et al. Scalable RDF data compression with MapReduce[J]. Concurrency and Computation: Practice and Experience, 2013, 25(1): 24-39.
[17] 李建江,崔健,王聃,等. MapReduce 并行编程模型研究综述[J]. 电子学报,2011,39(11):2635 – 2642.
LI J J,CUI J,WANG D,et al. Survey of mapreduce parallel programming model[J]. Acta Electronica Sinica, 2011,39(11):2635 – 2642. (in Chinese)
[18] GIMÉNEZ-GARCÍA J M,FERNÁNDEZ J D,MARTÍNEZ-PRIETO M A. HDT-MR:A Scalable Solution for RDF Compression with HDT and MapReduce[M] Heidelberg,Berlin: Springer International Publishing,2015:253-268.
[19] LIU J,WEI L I,LUO L,et al. Linked open data query based on Natural Language[J]. Chinese Journal of Electronics,2017,26(2):230 – 235.
[20] 章登义,吴文李,欧阳黜霏. 基于语义度量的 RDF 图近似查询[J]. 电子学报,2015,43(7):1320 – 1328.
ZHANG D Y,WU W L,OUYANG C F. Approximating query with semantic-based measure on RDF graphs[J]. Acta Electronica Sinica,2015,43(7):1320 – 1328. (in Chinese)

作者简介



符海东　男,1971 年出生湖北武汉. 2003 年毕业于华中科技大学,获工学博士学位. 现任武汉科技大学计算机科学与技术学院副院长、湖北高等教育学会高校计算机教育专业委员会常务理事、湖北省人工智能学会理事. 主要从事人工智能、软件工程和分布式计算方面研究.
E-mail:drfu@163.com

彭　燊　男,1992 年出生于湖北随州. 2014 年毕业于武汉科技大学软件工程系. 现为武汉科技大学计算机应用技术专业研究生,主要从事分布式计算和语义网的研究工作.
E-mail:sean_ps@163.com

黄　莉　女,1982 年出生于湖北. 2011 年于华中科技大学计算机科学与技术学院获得博士学位. 现为武汉科技大学计算机科学与技术学院教师. 主要从事语义网,数据管理和知识发现方面的研究工作.
E-mail:huangli82@wust.edu.cn

顾进广　男,1974 年出生于湖北仙桃. 2005 年毕业于武汉大学计算机学院,获工学博士学位. 现为武汉科技大学计算机科学与技术学院教授,博士生导师. 主要从事语义网与分布式计算方面的研究.
E-mail:simon@wust.edu.cn