计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200600084

基于双时态 RDF 模型的索引方法

王引娣　章哲庆　严　丽
南京航空航天大学计算机科学与技术学院　南京 210000
(2515551281@qq.com)

摘　要　RDF(Resource Description Framework)已被广泛用于大数据的语义表示与处理。传统的 RDF 只能表示静态语义,无法满足时间敏感场景下随时间动态处理语义的需求。为此,几种时态 RDF 模型已被提出,包括支持事务时间或有效时间的时态 RDF 模型,以及同时支持事务时间和有效时间的双时态 RDF 模型。为有效支持大规模时态 RDF 的高效处理,文中提出了一种基于双时态模型的时态 RDF 三层索引结构。第一层根据最大更新次数将双时态 RDF 数据划分为不同的数据子集;第二层在每一个数据子集上分别建立一棵四叉树来索引时间信息;第三层构建了包含 3 种组合键的复合位图来索引 RDF 三元组的主体、谓词和客体信息。实验从索引构建时间、索引占用空间,以及查询所需时间 3 个方面对所提时态 RDF 索引结构进行验证,结果表明,所提索引方案能有效缩短查询时间并提高查询效率。

关键词:RDF;时态信息;三层索引;四叉树;位图索引
中图法分类号　TP399

Indexing Bi-temporal RDF Model

WANG Yin-di,ZHANG Zhe-qing and YAN Li
College of Computer Science and Technology,Nanjing University of Aeronautics and Astronautics,Nanjing 210000,China

Abstract　RDF(Resource Description Framework) has been widely used for semantic representation and processing of big data. Traditional RDF can only represent static semantics and can not meet the needs of processing semantics dynamically over time in time-sensitive scenarios. Therefore, many temporal RDF models are proposed, including RDF model for transaction time, RDF model for valid time, and bi-temporal RDF model that supports both transaction time and valid time. To support efficient processing of large-scale temporal RDF data, this paper proposes a three-level index structure based on bi-temporal RDF model. Specifically, in the first level of this index structure, the dataset is divided into different subsets according to the update times of the temporal RDF data. In the second level, a quadtree is built for indexing time information in each subset, and in the third level, the bitmap with three composite keys is used to index the subject, predicate, object of RDF triples. Experiments are conducted from three aspects: the time of building index, the index size, and the required query time. Experimental results show that the proposed indexing scheme can reduce the query time effectively and improve the query performance.

Keywords　RDF, Temporal data, Three-level index, Quadtree, Bitmap index

1　引言

RDF 是 W3C 推荐的标准,用于表达和共享 Web 上的元数据,并且可以在应用程序之间交换而不丢失语义信息,因此,越来越多的组织和项目采用 RDF 作为数据表示模型。例如,DBPedia[1]是早期的语义网项目,采用了一个较为严格的本体,包含人、地点、音乐、电影、组织机构、物种、疾病等类定义。DBPedia 采用 RDF 语义数据模型,共包含 30 亿个 RDF 三元组。IBM 智慧地球[2]的研究中广泛采用 RDF 进行数据描述和集成语义。YAGO[3]是一个集成了 Wikipedia,WordNet 和 GeoNames 的链接数据库。YAGO 不仅具有丰富的实体分类体系,还考虑了时间和空间知识,为很多知识条目增加了时间和空间维度的属性描述。目前,YAGO 包含 1.2 亿条三元组知识。随着 RDF 数据的不断增长,对其进行有效的管理和查询显得愈发重要,其中大规模 RDF 数据索引发挥着重要作用[4]。

在现实生活中,许多应用场景中的信息会随时间的推移不断变化,如新闻、天气、股票、软件的更新等。为有效管理和处理时态信息,研究者们提出了时态数据库的概念,开发了时态数据库系统 TO2[5]、时态数据库管理系统 TDBMS[6]和双

到稿日期:2020-06-15　返修日期:2020-08-11　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:江苏省自然科学基金(BK20191274);国家自然科学基金(61772269)
This work was supported by the Natural Science Foundation of Jiangsu Province(BK20191274) and National Natural Science Foundation of China (61772269).
通信作者:严丽(yanli@nuaa.edu.cn)

时态关系数据库系统 TimeDB[7]等。特别是时态关系数据库在 SQL:2011 中得以标准化[8]，充分表明了实现时态信息管理的重要性。正如文献[9]所述，对 Web 上半结构化数据的处理需要完成对时间信息的建模。以 Web 上广泛用于数据表示和交换的可扩展标记语言 XML 为例，XML 文档不仅可能包含由时间限定的有效信息，还经常会被更新。为了记录 XML 文档中时间限定的有效信息和历史版本、描述 XML 文档的动态特性，“时态表达”的概念被引入 XML 文档中，出现了时态 XML 模型。文献[10]在 XML 中引入了“版本结点”，并为每条边增加一个“时态元素标记”来表示有效时间，将时态 XML 模型转化为有向无环图。针对事务时间，文献[11]为 XML 模型的每个结点增加一个“事务时间戳”，并扩展了 XPath，增加“事务时间轴”“事务时间结点测试”等来支持时态查询。文献[12]提出了一个双时态 XML 数据模型，该模型将一个双时态 XML 文档建模成一个带有“双时态元素”边标记的有向树形图，给出了该模型映射到双时态 XML 文档的两种映射方法。

随着 RDF 模型的广泛使用和大量 RDF 数据的出现，基于 RDF 的时态信息管理受到研究者的重视。文献[13]提出了一个 RDF 版本管理系统，用于处理事务时间。文献[14]针对有效时间，将时态三元组扩展为四元组(s,p,o,t)，形成 stRDF，以表示与查询随时间变化的地理空间数据。事务时间侧重于更新操作，有效时间侧重于有效性，为同时表示事务时间和有效时间，文献[15]提出了一种双时态 RDF 模型，该模型通过在谓词上添加时间信息和更新次数信息，对传统的三元组进行了扩展。

随着含时态信息的 RDF 数据规模的不断增大，为时态 RDF 构建索引结构以对其实现有效管理和高效查询处理变得越来越重要。虽然目前已有一些研究工作致力于时态 RDF 的研究，但有关时态 RDF 索引的研究还少有成果发表。基于单一时态 RDF 模型的索引结构研究已经取得了一定进展，关于有效时间的研究工作可参考文献[16-17]。文献[16]提出了 tGRIN 索引结构，文献[17]提出基于时态 RDF 图的路径索引方法。为了支持事务时间，文献[18]提出了用于处理 RDF 三元组的两级索引结构。目前还没有基于双时态 RDF 模型的索引研究成果，而大规模时态 RDF 数据的有效查询依赖于索引结构，合理的索引设计可以避免全局搜索，对不相关的分支进行裁剪，快速定位正确的三元组信息，有效缩短查询时间。因此，本文针对双时态 RDF 模型，提出了一种三层索引结构，用以提高双时态 RDF 数据的查询效率。本文的主要贡献如下：

(1)基于双时态 RDF 模型，提出一种三层索引结构。第一层先获取 RDF 三元组的更新次数信息，将数据集划分为不同的数据子集；第二层使用四叉树来索引时间信息；第三层采用复合位图结构来索引三元组的主体、谓词和客体信息。

(2)提出双时态 RDF 索引结构的实现算法。

(3)对所提索引方法进行实验分析，并将其与其他索引方法进行实验对比。结果表明，本文提出的索引方法能有效缩短查询时间，提高查询性能。

本文第 2 节简要介绍了基于经典 RDF 的索引结构、时态 RDF 模型及其索引结构；第 3 节提出了基于双时态 RDF 模型设计的三层索引结构的具体实现方案和算法；第 4 节通过对比实验对所提方案进行分析；最后总结全文并展望未来。

2 相关工作

2.1 基于经典 RDF 的索引结构

基于经典 RDF 数据的索引方法主要是利用 B+树、哈希等索引结构，以主体 S、谓词 P、客体 O 的不同顺序设计索引结构，实现对 RDF 数据的存储和查询。Hexastore[19]，RDF-3X[20]，RDF cube[21]都使用这种方法来组织和存储数据。Hexastore[19]是一种六元索引方案，该方案基于同等处理三元组的主体、谓词和客体的思想，设计了 6 种不同的复合索引 SPO，SOP，PSO，POS，OPS 和 OSP，其键值分别为 SP，SO，PS，PO，OP 和 OS。该索引以空间为代价来减少检索时间。RDF-3X[20]利用上述方法，进一步创建了 SP，SO 等 4 种二维索引和 S，P，O 3 种一维索引，并且在这些聚集的索引中利用剩余的比特记录该索引方式下的三元组组合出现的次数，当查询需要进行去重操作以及选择率估计时，可利用这些聚集索引来进一步加速查询。RDF cube[21]基于三元组结构设计了一种三维哈希索引。其利用哈希函数，将三元组的 S，P，O 分别映射到 cube 的三维空间。cube 中的每个单元表示一个 RDF 三元组，单元中的一个位表示在每个三维坐标组合下三元组存在与否。

2.2 时态 RDF 模型及索引

一般来说，时态数据主要分为两种类型：1)事务时间[22]，表明用户访问数据库时执行数据操作(插入、删除和修改等)的时间；2)有效时间[23]，指数据在数据模型中有效的时间。

为了表示这种时态数据，研究者提出了许多时态 RDF 模型，这些模型主要分为两种类型：版本控制类型和添加时间信息扩展 RDF 类型。版本控制模型用来捕获事务时间，维护图的每个状态的快照，并跟踪文件(如程序源代码)及其内容的发展。例如，文献[24]提出了一个正式的模型，用于跟踪基于图的数据模型中的变化，并将存储库中的更新历史定义为状态序列和更新序列来记录每个时间段的状态信息，但是无法直接获得与此项相关的记录总数。第二种类型是向 RDF 数据添加时间信息和注释。文献[25]使用时态标签 t 来扩展 RDF 三元组，表示为$(s,p,o)[t1;t2]$，它可以表示时间信息，但不能表示更新计数信息。文献[26]提出了带注释的 RDF，它通过一个带有底部元素的部分有序集对 RDF 三元组进行注释，并且支持表示 RDF 三元组中的不确定性和时间信息。文献[27]使用时间信息扩展了 YAGO 中的事实，并使用关系 on 来描述在某个时间点有效的事实，使用两个关系来表示时间间隔：使用 since 关系表示开始时间点，使用 until 关系表示结束时间点。文献[14]进一步将时态三元组扩展到使用四元组(s,p,o,t)表示的 stRDF，以表示查询随时间变化的地理空间数据。文献[14,26-27]提出的时态模型都只能表示在特定时间段内与三元组相关的状态信息，无法统计相关历史记录。基于以往的工作不足以在实际应用中同时对事务时间、有效时间和更新计数信息进行表示和管理，文献[15]提出了一种双时态 RDF 模型 RDFt，通过在谓词部分添加时间信息和更

新计数信息来扩展传统三元组,既能表示有效时间和事务时间,又能处理更新计数信息。

基于时态 RDF 模型的索引工作主要集中在文献[16-17]中。文献[16]提出了对有效时间进行处理的 tGRIN 索引结构。该索引为物理存储在 RDBMS 中的时态 RDF 数据构造一个特殊的索引,tGRIN 索引中的每个结点 m 有一个中心顶点 Cm 和一个半径 Rm,表示与图中顶点 Cm 的距离不超过 Rm 的所有顶点。文献[18]提出了用于处理事务时间模型的两级索引结构,首先使用 k-d 树索引作为全局索引来索引 RDF 数据的时间信息,然后使用组合位图索引作为局部索引来索引 RDF 三元组的信息。文献[17]提出了一种用于处理有效时间的基于时态 RDF 图的索引结构。首先,将时态 RDF 图存储为邻接表,并通过宽度优先遍历搜索图的某些路径;然后分别为图中的前缀路径和后缀路径建立前缀路径索引和后缀路径索引;最后,为了加快元素的搜索速度,构造了相应的 B 树索引。

以往的工作都是基于单一时态模型设计的索引结构,而本文提出了一种基于双时态模型的三层索引结构,既能处理事务时间,又能处理有效时间,很好地提高了管理和查询时态 RDF 数据的性能。

3 基于双时态 RDF 模型的索引

本文基于双时态 RDF 模型设计了一种三层索引结构。本节首先介绍了提出的双时态 RDF 模型,然后阐述了索引的结构设计,最后给出了建立索引的具体步骤和算法。

3.1 双时态 RDF 模型

单一的时态 RDF 模型在实际应用中无法表示和查询所有类型的时间信息(见表 1),同时用户可能会不断提出新的查询需求。为了满足这些查询需求,表示更多种类的时态信息,本文提出了一种双时态 RDF 模型,在同时表示事务时间和有效时间的情况下,还能记录更新次数信息,便于我们了解事物的波动情况和变化趋势。

表 1 查询和表示需求

Table 1 Representation and query requirments

	Representation and query requirments
R_1	Represent and query the information of employee's job transfer during a period of time
R_2	Represent and query the times of stock gains of more than 10% recently
R_3	Represents and queries the update records of a software

传统的 RDF 三元组包含主体(s)、谓词(p)和客体(o),而双时态 RDF 模型通过在谓词部分添加时间信息 t 和更新次数信息 n 来扩展 RDF 三元组,记为$(s,p[ts,te]-n,o)$。时间信息 ts 和 te 分别表示有效时间的开始时间和结束时间,当 $ts=te$ 时,表示为一个时间点,当 $ts\neq te$ 时,表示为一个时间段。更新次数 n 表示这条记录更新了 n 次,更新次数的变化即表示发生了更新操作,变化的时间点即为事务时间。根据更新次数,我们可以快速查看三元组的更新历史。

为更好地理解双时态模型,本文以图 1 中用双时态模型表示的数据集为例进行解释。从图 1 可以看到,在[1,2]时间段,Jack 第一次工作于 New Time,在时间节点 2,Jack 发生了工作调动,入职 Creative Force,工作期限为[3,6],在随后的[7,8]时间段,又工作于 Big Data。因此,使用双时态模型,我们既能看到 Jack 的最新工作状况,还能获悉 Jack 的工作变动情况以及具体任职经历,弥补了单一时态模型只能表示当前有效状态或历史信息的不足。

<Jack> workIn[1,2]-1 <New Time>
<Jack> workIn[3,6]-2 <Creative Force>
<Jack> workIn[7,8]-3 <Big Data>
<Mary> like[4,8]-1 <music>
<Sandy> position[8,10]-1 <staff>
<Sandy> position[11,13]-2 <assistant>
<Sandy> like[4,8]-1 <music>
<Alex> position[12,17]-1 <staff>
<Alex> position[18,20]-2 <assistant>
<Alex> position[21,23]-3 <manager>

图 1 用双时态 RDF 模型表示的数据集

Fig. 1 Dataset represented by bi-temporal RDF

3.2 索引结构

由于双时态 RDF 模型既包含事务时间和有效时间,也包含更新次数信息,因此本文提出的索引结构分为三层。第一层根据更新次数信息将数据集划分为不同的数据子集,第二层分别在这些数据子集上建立四叉树,用来索引时间信息。第三层为包含了 3 种组合键值的复合位图索引结构,用来索引三元组的主体、谓词和客体信息。

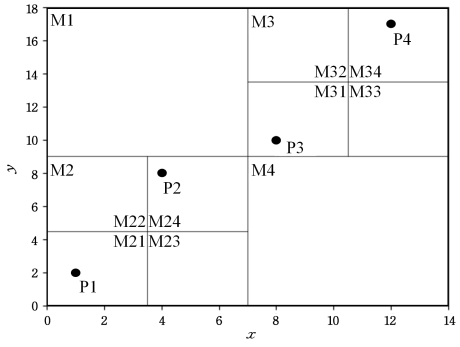
为了更好地介绍索引结构,我们引入 data subset,time set,triple collection 这 3 个概念。根据不同更新次数划分初始数据集得到 data subset;在 data subset 中,所有不重复的时间信息构成了 time set;time set 中的一个时间信息可能会对应一个或者多个三元组,这些具有相同时间信息的三元组被放入 triple collection。

以图 1 的数据为例,在我们的索引结构中,第一层用来处理更新次数,可以看到,最大的更新次数是 3,根据更新次数 1,2,3 分别将数据划分为 3 个不同的子集 data subset1,data subset2,data subset3。在 data subset1 中,将不重复的时间信息 [1,2],[4,8],[8,10],[12,17]放入 time set 中,对于时间间隔[1,2],triple collection 就是{(〈Jack〉,workIn[1,2]-1,〈New Time〉)},对于[4,8],triple collection 就是{(〈Sandy〉,like[4,8]-1,〈music〉),(〈Mary〉,like[4,8]-1,〈music〉)}。对于其他更新次数和时间信息的处理,以此类推。

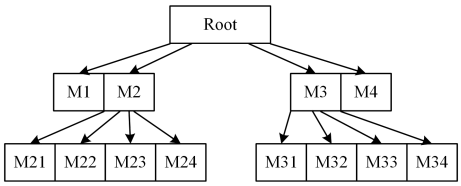
时间信息可以表示成二维空间的一个二维点,在第二层中我们采用四叉树结构来索引时间信息。四叉树也称为 Q 树,其运用了空间分割的理念,将二维空间递归划分为不同层次的树结构。它的每个结点下至多可以有 4 个子结点,通常把一部分二维空间细分为 4 个区域并把该区域里的相关信息存入四叉树节点中,如此递归下去,直至树的层次达到一定深度或者满足某种要求后停止分割。空间对象都存储在叶子结点上,中间节点以及根节点不存储地理空间对象。由于四叉树在划分空间时不会重叠,因此在查找点时不用回溯,缩短了查找时间。

图 1 中更新次数为 1 的时间信息在空间中的表示如图 2(a)所示。可以看到,由于初始空间内的点多于 1 个,因此将其等分为 4 个部分 M1,M2,M3,M4,而 M2 和 M3 中点的数量又超过了一个,继续递归划分成 M21,M22,M23,M24,以

及 M31,M32,M33,M34,划分后的每个空间的点数均不超过1个,至此,划分结束。最终得到的四叉树如图2(b)所示。



(a)Data space



(b)Quadtree index

图2 数据空间和四叉树

Fig. 2 Data space and quadtree index

最后,同一个时间间隔会对应一个三元组集合 triple collection,在第三层结构中,采用复合位图结构来索引 triple collection 中三元组的主体、谓词和客体信息。复合位图结构包含了3个分别以主体-谓词(s-p)、谓词-客体(p-o)和客体-主体(o-s)为组合键的位图索引,并用二维数组存储,因此对于同一个时间间隔存在三元组的3个不同视图。以 s-p 位图索引为例,将它存入 bmn 数组中,其中 b[0][0]为时间间隔,不同行对应不同的主体,不同列对应不同的谓词,行与列的交点就是对应这个主体和谓词的所有客体。对于时间间隔[4,8],它的3个位图索引如图3所示,由此可见,位图索引在数据部分重合时能够很好地减少冗余信息。

[4,8]	like
Sandy	{music}
Mary	{music}

(a)s-p bitmap index

[4,8]	music
like	{Sandy,Mary}

(b)p-o bitmap index

[4,8]	Sandy	Mary
music	{like}	{like}

(c)o-s bitmap index

图3 时间信息[4,8]对应的位图索引结构

Fig. 3 Bitmap index structure for time interval [4,8]

图4就是图1中更新次数为1的数据子集的树形索引结构。可以看到,树形结构的上部分是四叉树结构,用来索引时间信息,其中叶子结点还指向对应这个时间信息的复合位图索引。位图索引包含三元组集合中的主体、谓词和客体信息。在查询时,先利用时间信息在四叉树中找到相应的叶子结点,然后根据结点指向的位图索引来查找具体的三元组信息。

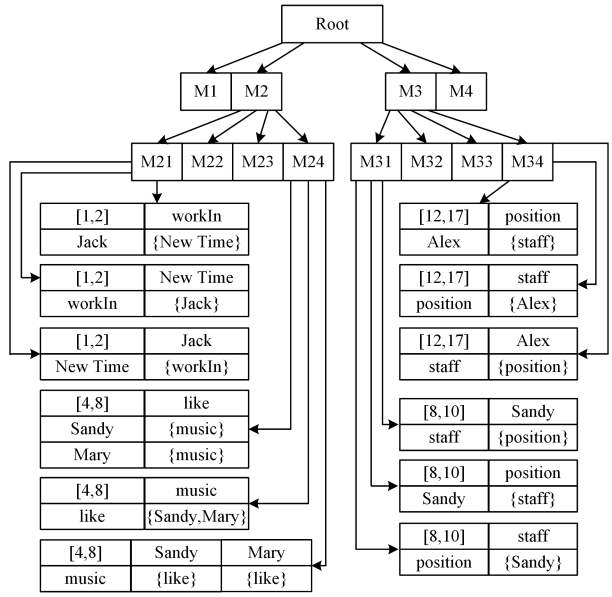


图4 更新次数为1的树形索引

Fig. 4 Temporal RDF tree index with update time of 1

3.3 构建索引

本文提出的三层索引结构中,第一层根据不同的更新次数划分得到若干个 data subset,第二层利用 data subset 中的 time set 信息构建四叉树索引,第三层用复合位图结构来索引 time set 中每个时间间隔信息对应的 triple collection。

步骤1(Divide) 如算法1所示,先遍历初始时态 RDF 数据集 d,获取最大的更新次数 maxUpdateCount(第1行)。然后再次遍历 d,根据不同的更新次数将其划分到 maxUpdateCount 个 data subset 中(第2—第7行)。最后调用 BuildQtree 来完成剩下的索引构建工作(第8行)。

算法1 数据划分算法

```
输入:时态 RDF 数据集 d
输出:时态 RDF 索引
1.  maxUpdateCount←getMaxCount(d)
2.  Set i=0
3. For(;i<maxUpdateCount;i++)
4.  Set ds /*存放同一更新次数的数据集*/
5.  For(all data in d)
6.    if(data.UpdateCount==i)
7.      ds.add(data)
8.  callBuildQtree(ds) /*执行第二步构建四叉树索引*/
```

步骤2(BuildQtree) 如算法2所示,先遍历数据集得到时间信息集合 interval(第1行),再新建四叉树(第2行),然后遍历集合 interval(第3行)。对于每一个时间信息,需要从四叉树根结点递归找到插入的叶子结点,选择的原则是分别与当前结点的竖直方向的中点和水平方向的中点进行比较,从而找到下一步要插入的孩子结点。递归找到合适的插入位置后,还需要判断当前叶节点是否还有空间,有空间就可以直接插入,否则需要在当前节点下再分裂出4个孩子结点,并将当前结点的数据划分到孩子结点中(第6—15行)。最后,需要调用 BuildBitmap 方法来索引与这个时间间隔相关的三元组信息(第5行)。

算法 2 四叉树构建算法

输入：时态 RDF 数据子集 ds

输出：四叉树索引

```
1. interval←getInterval(ds)
2. quadTree qtree=new quadTree()
3. For(all i in interval)
4.    qtree.insert(i,root)
5.    call BuildBitmap(i,ds) /*执行第三步构建复合位图*/
6. insert(i,root) /*插入结点*/
7.    nodeCapacity←qtree.getNodeCapacity()
8.    if(root.isLeaf)
9.       if(root.usedSpace < nodeCapacity)
10.         root.add(i)
11.      else
12.         splitNode(root) /*空间不足，需要分裂结点*/
13.   else
14.      qnode child=findChildNode(root,node)
15.      insert(i,child)
```

步骤 3(BuildBitmap) 每一个时间间隔指向 3 个不同组合键的位图，这里着重介绍组合键为主体-谓词的位图结构，其他两个类似。位图索引可以用二维数组存放，算法 3 先将时间信息 i 存放至 b_{00} 处(第 1 行)，再遍历数据集 ds 获得与时间信息 i 相关的三元组集合 C(第 2 行)，继续遍历 C 获得主体集合 subjects 和谓词集合 predicates(第 3、第 4 行)，将主体分别放入数组的第一列(第 5、第 6 行)，将谓词分别放入数组的第一行(第 7、第 8 行)，对应行与列的交点处就存放三元组的客体(第 9－第 11 行)。

算法 3 复合位图构建算法

输入：时间信息 i，时态 RDF 数据子集 d_s

输出：位图索引 b

```
1. b00←i /*位图索引的 b00 存放时间间隔*/
2. C←getTripleCollection(ds,i)
3. subjects← getSubject(C)
4. predicates← getPredicate(C)
5. For(all subject in subjects)
6.    bk0←subject
7. For(all predicate in predicates)
8.    b0j← predicate
9. For all t in C
10.  if(t.subject=bi0 && t.predicate=b0j)
11.  bij=t.object
```

4 实验评估

本评估实验是基于 JDK1.8 版本的 eclipse 平台上完成的，使用的数据库是 MySql。本实验在具有 Intel i5-4210u 处理器和 windows8.1 操作系统的系统上完成。

4.1 数据集

本文数据集来源于 Lehigh University benchmark data set(LUBM)。LUBM 是一个用于研究的综合基准数据集[28]，可以用来生成若干个三元组，然后随机生成更新次数信息和时间间隔信息，并将这些信息添加到三元组上。

如表 2 所列，我们通过选择不同数量的时间间隔、三元组以及最大更新次数来生成 8 个不同的数据集，后续对索引进行评估的实验分别在这些数据集上进行。

表 2 实验数据集

Table 2 Experimental datasets

Datasets	Max update count	Quantity of time intervals	Number of triples
ds1	3	5000	16256
ds2	3	8000	16256
ds3	5	8000	16256
ds4	5	8000	50472
ds5	5	16000	50472
ds6	7	16000	50472
ds7	7	16000	140592
ds8	7	50000	140592

4.2 实验分析

目前并没有研究者基于双时态 RDF 三元组提出相关的索引结构，而文献[18]基于单时态 RDF 三元组提出了一种二级索引结构，使用 KD 树索引时间信息，使用位图索引具体的三元组信息。本文使用的双时态 RDF 三元组模型相比文献[18]中的单时态 RDF 三元组额外添加了更新次数信息。为了测试所提索引结构的查询性能，本文设计出一种对比索引结构 LUBM-KD，其索引更新次数的方式和本文相同，处理时间信息和具体的三元组信息采用文献[18]提出的索引结构。实验中，将本文提出的索引方案 LUBM-Q 与 LUBM-KD 进行了对比。实验方案包含索引性能分析和查询性能分析，其中索引性能又分为索引的构建时间和占用空间，我们将从这几个方面完成索引评估。

4.2.1 索引性能分析

我们在不同数据集上比较了索引结构的构建时间和占用空间，结果分别如图 5、图 6 所示。

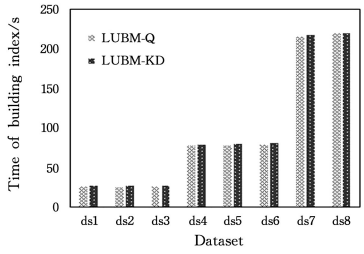


图 5 索引构建时间

Fig. 5 Time of building index

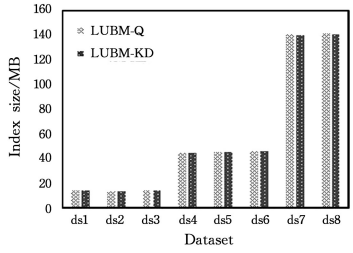


图 6 索引占用空间

Fig. 6 Index size

从图 5 可以看到，索引方案 LUBM-Q 与 LUBM-KD 的构建时间几乎相同，这是因为索引的构建步骤包含根据更新次

数将数据集划分为数据子表、构建二级索引(四叉树或 KD 树)和构建位图索引并将数据最终存储至数据库表,其中创建数据子表、对数据表进行更新等数据库操作占据绝大部分的构建时间,本文与文献[18]中索引的差异集中在二级索引上,因此这两种方案对索引构建时间的影响几乎可以忽略不计。

索引的占用空间分为两部分:二级索引(四叉树或 KD 树)和复合位图索引。在二级索引中,四叉树中只有叶子结点保存时间信息,而 KD 树中所有的结点都用来保存时间数据,因此相比 KD 树,四叉树需要占用额外的空间来保存非叶子结点信息。从图 6 来看,二级索引所导致的占用空间差异并不明显,这是因为在第三层索引结构(复合位图)中包含 3 个位图存储最终的 RDF 数据信息,相当于原始 RDF 数据量的三倍,所以位图索引占用最主要的空间。总体来说,索引方案 LUBM-Q 的占用空间比 LUBM-KD 略大。

整体来看,本文提出的索引方案 LUBM-Q 的构建时间和占用空间都与数据集的规模呈线性相关,这表明这种索引方案是可扩展的。同时,索引方案 LUBM-Q 与 LUBM-KD 相比,构建时间和占用空间均没有明显的额外损耗,两种索引方案在索引性能方面的表现几乎没有差异。

4.2.2 查询性能分析

为了测试索引对查询性能的提高,本文使用了两种类型的查询,一种是单个三元组查询,用来查询单个时态 RDF 三元组内的某个元素;另一种是路径查询,用来查询几个时态 RDF 三元组构成的一条路径上的某个元素。

在每种查询中我们做了两组对比实验,第一组使用的是本文提出的索引方案 LUBM-Q,第二组使用索引方案 LUBM-KD,这两种查询的结果分别如图 7、图 8 所示。

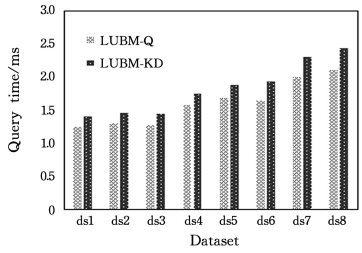


图 7 单个三元组的查询时间

Fig. 7 Query time of single triple pattern

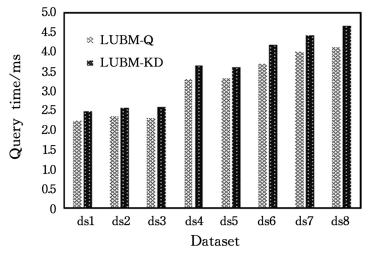


图 8 路径查询时间

Fig. 8 Path query time

从图中结果可以看出,无论是第一种查询还是第二种查询,使用索引结构 LUBM-Q 的查询时间都比 LUBM-KD 短,并且随着三元组中时间间隔数量的增加,查询时间的差异越来越明显。这是因为 KD 树相当于平衡二叉树,并且在寻找最相近的时间结点时,还会进行回溯操作,而四叉树结点有 4 个分支,降低了树的高度,同时,四叉树在查找时不需要进行回溯,从而有效地减少了数据查找的次数和时间。

结束语 随着语义 Web 的不断发展,RDF 作为数据表示模型被广泛应用于各种场景。然而有许多场景是对时间敏感的,仅使用 RDF 无法描述这种随着时间变化的动态特性。于是,研究者们指出需要对时间建模并提出了一些支持事务时间或有效时间的单一时态 RDF 模型和同时支持事务时间和有效时间的双时态 RDF 模型。由于含时态信息的 RDF 数据规模不断增大,亟需为时态 RDF 构建索引以实现高效管理和查询。基于单一时态 RDF 模型的索引结构的研究已经取得了一定进展,而基于双时态 RDF 模型的索引结构还未被研究。本文基于这种双时态 RDF 模型提出了一种三层索引结构,在数据划分的基础上,采用四叉树和复合位图结构来索引双时态 RDF 三元组。实验表明,相较于对比方案,本文提出的索引方案在索引构建时间和占用空间方面几乎没有额外损耗,并且有效缩短了查询时间,提高了查询效率。在接下来的工作中,我们将考虑使用频繁算法对数据集和实际查询中频繁出现的数据构建索引,以降低索引的构建时间并减少索引的存储空间。

参 考 文 献

[1] AUER S. DBpedia: A Nucleus for a Web of Open Data. [C]// Semantic Web, International Semantic Web Conference, Asian Semantic Web Conference. Iswc + Aswc, Busan, Korea, DBLP, 2007.

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

[3] HOFFART J, SUCHANEK F M, BERBERICH K, et al. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia [J]. Artificial Intelligence, 2013, 194: 28-61.

[4] MA Z M, CAPRETZ M A, YAN L, et al. Storing massive Resource Description Framework(RDF) data: a survey[J]. Knowledge Engineering Review, 2016, 31(4): 391-413.

[5] ZHANG F. Research and Implementation of an Object-Oriented Temporal Database System[D]. Beijing: Chinese Academy of Sciences, 2000.

[6] EDELWEISS N, HUBLER P N, MORO M M, et al. A temporal database management system implemented on top of a conventional database[C]// Proceedings 20th International Conference of the Chilean Computer Science Society. IEEE, 2002.

[7] TimeConsult. TimeDB[EB/OL]. http://www.timeconsult.com/.

[8] KULKARNI K, MICHELS J E. Temporal features in SQL: 2011[J]. SIGMOD record, 2012, 41(3): 34-43.

[9] ABITEBOU S. Querying Semi-Structured Data[C]// International Conference on Database Theory. Berlin, Heidelberg: Springer, 1997.

[10] VAISMAN A. Temporal XML: Data Model, Query Language and Implementation[J]. VLDB Journal, 2008, 17(5): 1179-1212.

[11] DYRESON C E. Observing transaction-time semantics with

TTXPath[C]// International Conference on Web Information Systems Engineering. IEEE,2001.

[12] TANG N,TANG Y,CAI M M. Bitemporal Extension of XPath Data Model[J]. Journal of Computer Research and Development,2006,43(z3):504-509.

[13] GRANDI F. Multi-temporal RDF ontology versioning[J]. CEUR Workshop Proceedings,2009,519:1-10

[14] BERETA K , SMEROS P , KOUBARAKIS M. Representation and Querying of Valid Time of Triples in Linked Geospatial Data[C]// Extended Semantic Web Conference. Berlin, Heidelberg:Springer,2013.

[15] ZHANG F , WANG K , LI Z,et al. Temporal Data Representation and Querying Based on RDF[J]. IEEE Access,2019(99):1-1.

[16] PUGLIESE A , UDREA O , SUBRAHMANIAN V S. Scaling RDF with time[C]// Proceedings of the 17th International Conference on World Wide Web (WWW 2008). Beijing, China: ACM,2008:21-25.

[17] YAN L, ZHAO P, MA Z. Indexing temporal RDF graph[J]. Computing,2019,101(10):1457-1488.

[18] ZHAO P, YAN L I. A Methodology for Indexing Temporal RDF Data[J]. Journal of Information ence and Engineering, 2019,35(4):923-934.

[19] WEISS C, KARRAS P, BERNSTEIN A. Hexastore: sextuple indexing for semantic web data management[J]. Proceedings of the VLDB Endowment,2008,1(1):1008-1019.

[20] NEUMANN T,WEIKUM G. RDF-3X:a RISC-style engine fNeumann T,Weikum G. RDF-3X:a RISC-style Engine for RDF [J]. Proceedings of the VLDB Endowment,2008,1(1).

[21] MATONO A, PAHLEVI S M, KOJIMA I. RDFCube: A P2P-Based Three-Dimensional Index for Structural Joins on Distributed Triple Stores [J]. Lecture Notes in Computer Science,2006, 4125:323-330.

[22] MCBRIDE B, BUTLER M. Representing and Querying Historical Information in RDF with Application to E-Discovery[R]. Hewlit Packard Laboratories Technical Report,2009.

[23] MOTIK B. Representing and querying validity time in RDF and OWL:A logic-based approach[J]. Journal of Web Semantics, 2012,12-13(2):3-21.

[24] OGNYANOV D , KIRYAKOV A. Tracking Changes in RDF (S) Repositories[C]// Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference(EKAW 2002). Siguenza, Spain:Springer-Verlag,2002:1-4.

[25] GUTIERREZ C, HURTADO C A, VAISMAN A A,et al. Introducing Time into RDF[J]. IEEE Transactions on Knowledge and Data Engineering,2007,19(2):207-218.

[26] UDREA O , RECUPERO D R , SUBRAHMANIAN V S. Annotated RDF[C]// Proceedings of the 3rd European conference on The Semantic Web:Research and Applications. ACM,2006.

[27] WANG Y , ZHU M , QU L,et al. Timely YAGO:Harvesting, Querying,and Visualizing Temporal Knowledge from Wikipedia [C]// 13th International Conference on Extending Database Technology(EDBT 2010). Lausanne, Switzerland: ACM, 2010: 22-26.

[28] GUO Y, PAN Z, HEFLIN J. An evaluation of knowledge base systems for large OWL datasets[J]. Lecture Notes in Computer Science,2004,3298:274-288.

WANG Yin-di, born in 1996, postgraduate. Her main research interests include RDF data and semantic web.

YAN Li, born in 1964, professor. Her main research interests include big data, knowledge graph, spatiotemporal information processing and NoSQL database.