计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2019.20190169
56(8): 1758-1771, 2019

支持范围查询的低冗余知识图谱管理

王 飞 钱铁云 刘 斌 彭智勇
(武汉大学计算机学院 武汉 430072)
(feiw@whu.edu.cn)

Low-Redundancy Knowledge Graph Management with Range Query Support

Wang Fei, Qian Tieyun, Liu Bin, and Peng Zhiyong
(*School of Computer Science*, *Wuhan University*, *Wuhan* 430072)

Abstract As more and more data is published in the form of knowledge graph, the management of which attracts a lot of attention. Existing approaches for knowledge graph management have two drawbacks: 1) logical storage modeling generates lots of redundancy and ineffectively supports range queries on continuous attributes; 2) semantic storage modeling costs much and inefficiently adapts to the dynamic evolution of knowledge graph. In this paper, we propose a novel method called cluster object deputy model (CODM) to manage knowledge and metadata. The model has two key properties, namely logical storage modeling of schema and semantic storage modeling of lightweight. To this end, we design a schema cluster algorithm based on the set editing distance to convert knowledge graph into schema data, which realizes schema storage of data and supports index specification of attribute type. Besides, CODM constructs a class hierarchical system to model different associations among entities. It adopts object pointers to achieve the lightweight materialization of generalized semantic association. Experimental results show that CODM can tremendously reduce the data redundancy and outperforms the state-of-the-art methods in terms of range queries. And those results also indicate that CODM can accelerate the processing of complex queries.

Key words knowledge graph; metadata modeling; range query; schema storage; cluster object deputy model (CODM)

摘 要 随着越来越多的数据以知识图谱的形式进行组织和发布,知识图谱的管理引起了大量的关注.现有知识图谱管理方法存在2个明显的缺陷:1)逻辑存储建模产生了大量的数据冗余,无法有效地支持连续属性的范围查询;2)语义存储建模代价昂贵,不能有效地适应查询的动态演化.提出了聚簇对象代理模型(cluster object deputy model, CODM)进行知识和元知识的建模管理.该模型具有2个特点,分别是模式化的逻辑存储建模和轻量级的语义存储建模.CODM设计了基于集合编辑距离的模式聚簇算法将知识图谱转化为模式数据,实现了数据的模式化存储,支持了面向属性数据类型的索引特化.此外,CODM构建类的层次系统建模实体之间的各种语义关联,采用对象指针实现了轻量级的泛化语义关联

收稿日期:2019-03-20;**修回日期**:2019-05-21
基金项目:国家重点研发计划项目(2018YFB1003400);国家自然科学基金项目(61572376);中央高校基本科研业务费专项资金项目(2042019k10278);国家自然科学基金联合基金重点项目(U1811263)
This work was supported by the National Key Research and Development Program of China (2018YFB1003400), the National Natural Science Foundation of China (61572376), the Fundamental Research Funds for the Central Universities (2042019k10278), and the Joint Funds of National Natural Science Foundation of China (U1811263).
通信作者:彭智勇(peng@whu.edu.cn),钱铁云(qty@whu.edu.cn)

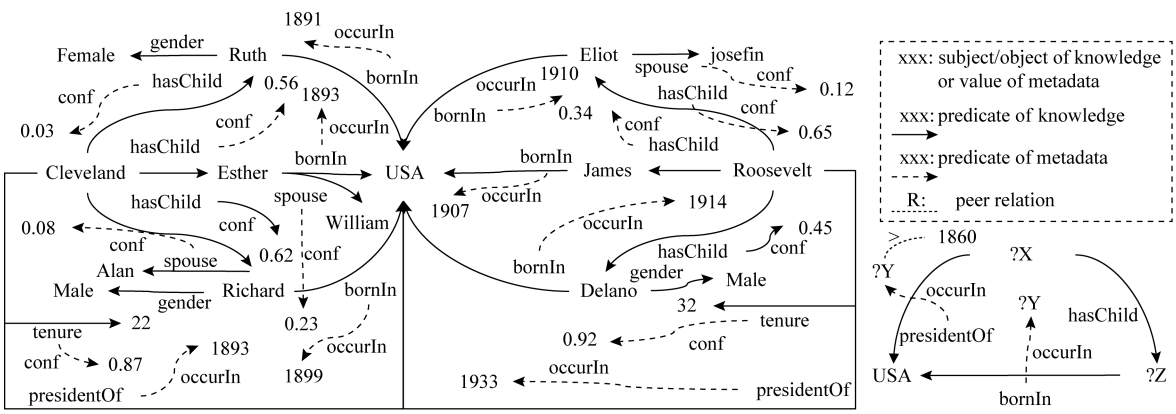
物化.实验结果证明:CODM 不仅能够极大地减少数据冗余和有效地支持范围查询,而且加速了复杂查询的处理效率.

关键词　知识图谱;元数据建模;范围查询;模式化存储;聚簇对象代理模型

中图法分类号　TP391

知识图谱是一种重要的知识表示形式,能够支持智能推荐[1]、智能问答[2]和知识发现[3]等大量的人工智能应用.随着越来越多的知识图谱数据陆续发布出来,知识图谱的管理吸引了研究者的大量关注[4-8].尽管许多研究成果展示了其在知识图谱管理方面的巨大成功,但是,带有连续属性的知识图谱元知识建模管理并没有引起太多的关注.知识图谱自然地涵盖知识和元知识 2 种陈述.一条知识就是一条陈述,而一条元知识则是一条关于知识的起源、时间、空间和置信度等信息的陈述[9].

知识通常呈现为〈主语,谓语,宾语〉的形式,而元知识则表现成〈知识陈述,谓语,宾语〉的形式.这样,由知识和元知识组成的知识图谱就可以表示为一个带有标签的有向图.在这种有向标签图中,一个节点代表一个主语或宾语,一条边则代表主语和宾语之间的一种谓语关系.图 1 呈现了采用有向标签图表示的一个微型知识图谱和查询.例如,一条知识表示为〈Cleveland, presidentOf, USA〉,一条元知识则可以用于解释这个事实发生在 1893 年.如果一个应用想要知道“who became president of USA in the year when his/her child was born in USA since 1860?”,那么,这个查询就可以表示成图 1(b)的形式.这样,知识图谱回答查询请求的流程则转变为从一个完整的有向标签图中寻找匹配查询的子图的过程.



(a) A toy knowledge graph　　(b) A query graph

Fig. 1　Knowledge Graph and Query Graph

图 1　知识图谱和查询图

由于知识图谱包含知识和元知识 2 种陈述类型,因此,知识图谱管理的一个重要任务就是进行知识和元知识的逻辑存储建模.现有的逻辑存储建模主要分为三元组框架和类三元组框架 2 种建模方式.三元组框架建模方式[9-11]将知识陈述抽象成句柄,通过知识句柄关联知识陈述和元知识陈述,并统一表示成三元组形式.类三元组框架建模方式[12]突破了传统三元组知识表示的约束,通过扩展知识维度进行公共元知识属性的模式化存储.由于知识图谱是一种无模式的数据类型,因此,三元组框架和类三元组框架对知识陈述都采用泛化形式的语义表达.在进行知识和关联元知识表示时,泛化建模方式需要同时进行数据值声明和模式声明,这导致了知识图谱建模的冗余度非常高.这种类型的知识图谱冗余不仅浪费存储空间和 I/O 代价,还需要大量的连接操作来重构知识陈述和元知识陈述,制约着知识图谱的查询响应效率,直接影响着知识图谱的可用性.此外,知识和元知识可能涉及时间、地理位置以及置信度等连续型属性,因此,面向连续型属性的查询已经成为知识图谱应用迫切需要解决的问题[13-14].现有的泛化知识图谱建模方法无法有效地支持带有连续属性的范围查询.

不同于泛化建模方式,模式化建模在一定程度上固化了数据的存储模式,在进行知识和元知识

建模时只需要进行数据值声明,而不需要进行数据模式声明,这样就显著地降低了数据冗余.同时,模式化存储也有助于实现对知识属性和元知识属性的数据类型特化,支持连续属性的范围查询.为了实现知识图谱的模式化存储,就需要将无模式的知识图谱数据转化为模式化数据.

虽然知识图谱数据的查询需求动态变化,但是,数据之间的固有关联以及相关的查询模式却又具有一定的稳定性[15].因此,知识图谱数据的语义存储建模成为知识图谱管理提升查询效率的一条有效途径.一些研究者提出对知识图谱进行谓语分组存储来适应绑定谓语的查询需求[16].另一些研究者采用实体形式组织存储,根据数据组织和处理的差异性分为表上的实体存储[17]和图上的实体存储[6].这种建模方式实现了一定程度的模式存储,有助于减少星型查询的关联操作.除此之外,利用查询日志理解数据之间的查询关联,动态地调整数据的存储组织方式也是一种建模数据语义关联的方法[18].基于谓语分组和实体的关联建模通常只能适应固有的数据关联查询,无法满足各种应用的动态查询需求.基于查询日志动态调整数据存储组织方式涉及到数据的迁移与合并,通常需要较大的性能开销.因此,我们需要设计一种轻量级的语义建模机制.这种语义建模机制不仅能够建模数据之间的各种语义关联,还能够以非常低的性能代价适应查询需求的动态变化.

本文提出了聚簇对象代理模型(cluster object deputy model, CODM)用于知识图谱的建模管理.CODM 模型将元知识陈述关联到相应的知识陈述,提出了基于集合编辑距离的模式聚簇算法实现无模式的知识图谱到模式化数据的转变.为了进行统一的建模设计,CODM 模型将模式化存储建模和关联语义建模融合到一个统一的建模框架,引入了类-对象概念,构建了一个由聚簇实体类和代理关系类组成的类层次系统.基于聚簇模式构建聚簇实体类实现模式化存储,能够更好地支持知识图谱的星型查询.CODM 模型定义了一系列语义操作算子用于构建代理关系类,实现对各种语义关联的建模.代理关系类采用指针机制来表达数据关联,实现对知识图谱聚簇对象泛化语义关联的轻量级物化.这种轻量级的泛化语义关联建模机制可以将连接操作转换为选择操作,以此来加快复杂查询的处理效率.

1 相关工作

以知识图谱为代表的知识工程是进一步提升人工智能智能化水平的基石日益成为一种行业共识.越来越多的信息以知识图谱的形式进行发布[19-20],知识图谱的管理引起了学术界和工业界的高度关注.知识陈述和元知识陈述组成知识图谱.早期的知识图谱管理主要聚焦于知识陈述的管理.随着知识图谱应用的逐步扩展和深化,知识图谱管理研究开始关注元知识陈述建模和连续属性的查询支持[21].知识图谱管理主要分为关联知识陈述和元知识陈述的逻辑存储建模和优化查询性能的语义存储建模.

逻辑存储建模主要用于构建知识陈述和元知识陈述的逻辑关联,依据建模的元组形式分为三元组框架和类三元组框架.三元组框架通过知识陈述的唯一意涵关联元知识陈述,其特点是将知识和元知识整合到统一的三元组框架.标准增强方式(standard reified statement)[10]是资源描述框架官方推荐的带有元知识的知识图谱建模方式.这种建模方式将知识陈述和关联的元知识陈述归并为一个泛化陈述,通过泛化陈述的意涵句柄分别声明主语、谓语、宾语以及关联的元知识.属性唯一方式(singleton property approach)[9]是另一种三元组框架的元知识建模方式,该方式认为知识陈述的唯一性源于主语和宾语之间谓语所代表关系的唯一性.属性唯一方式构建了谓语属性类,唯一性属性成为谓语属性类的一个属性情景实例,将谓语属性情景实例作为意涵句柄进行元知识陈述的声明.由于三元组框架进行元知识陈述建模时固有的冗余特性,类三元组框架[22]扩展了经典的 SPO 三元组为 SPO+X 的元组框架.SPO 代表经典的主语、谓语和宾语,而 X 则代表扩展的元知识公共谓语集.YAGO2 增强方式(YAGO2 reified statement)[12]是一种典型的类三元组框架,保留了 SPO 的经典模式,同时扩展了时间、位置以及情景等公共元知识谓语集.表 1～3 呈现了 3 种逻辑存储建模方式.

Table 1 Standard Reified Statement

表 1 标准增强方式

Subject	Predicate	Object
stmt#1	rdf:type	statement
stmt#1	rdf:subject	Cleveland
stmt#1	rdf:predicate	presidentOf
stmt#1	rdf:object	USA
stmt#1	occurIn	1893
stmt#1	deriveFrom	wiki
stmt#1	conf	0.86

Table 2　Singleton Property Approach

表 2　属性唯一方式

Subject	Predicate	Object
p#1	rdf:singletonPropertyOf	presidentOf
Cleveland	p#1	USA
p#1	occurIn	1893
p#1	deriveFrom	wiki
p#1	conf	0.86

Table 3　YAGO2 Reified Statement

表 3　YAGO2 增强方式

ID	Subject	Predicate	Object	OccurIn (Time)	Place (Location)
#1	Cleveland	presidentOf	USA	1893	null
#2	#1	deriveFrom	wiki	null	null
#3	#1	conf	0.86	null	null

语义存储建模主要是指通过知识图谱聚簇实施存储来提升知识图谱数据的查询响应性能,依据聚簇语义的差异性分为基于谓语的聚簇、基于实体的聚簇以及基于查询语义的聚簇等 3 种类型的语义存储建模方式.基于谓语的聚簇建模[16],也称为列式存储,是指依据谓语信息对知识图谱数据进行分组,并为每一个知识图谱集合构建一个存储模式,这样有助于提升 I/O 操作的有效性.由于基于谓语的聚簇建模方式实施分散存储带来了频繁的连接操作,因此,研究者提出了基于实体的聚簇建模.这种聚簇建模方式将具有相同主语的知识图谱数据进行关联,并将这些实体映射到一个统一的聚簇文件.基于实体的聚簇建模有 2 种实现方式:1)采用散列或者图着色的方式将实体的谓语信息映射到一个大的关系表,构建一种具有弱模式的映射存储[17];2)将实体的谓语和宾语进行绑定,生成实体的指纹编码,再采用类似散列树的形式构建实体指纹编码的层次索引结构[20],加速离散型知识图谱数据的查询处理.谓语聚簇和实体聚簇通常只能反映静态的数据关联,无法满足不断演化的知识图谱查询需要.基于查询语义的聚簇建模依据查询流和数据响应流进行数据的关联聚簇,动态地调整数据的存储关联,适应当前的数据查询需要[18].

2　聚簇对象代理模型 CODM

CODM 构建了由聚簇实体类和代理关系类组成的类层次系统.一个模式聚簇算法能够实现知识图谱从无模式数据到强类型模式数据的转变.基于聚簇模式生成聚簇实体类,实现知识图谱的模式化逻辑存储和属性数据类型的特化,而代理关系类构建各种泛化的实体语义关联用于加快复杂语义的特化查询.表 4 是本文的符号表:

Table 4　Symbol Description Table

表 4　符号描述表

Symbol	Description
E,e	entity set E, $e\in E$
Φ,φ	entity schema set $\Phi,\varphi\in\Phi$
Ψ,ψ	cluster schema set Ψ, $\psi\in\Psi$
O,o	object set $O,o\in O$
C,c	class set $C,c\in C$
K,k	knowledge predicate set $K,k\in K$
M,m	metadata predicate set $M,m\in M$
P,p	(knowledge/metadata)predicate set $P,p\in P$
A,a	action set $A,a\in A$
ce	implication of cluster entity
dr	implication of deputy relation
s	implication of source object(class)
id	implication of unique identifier
h,i,j	serial number

2.1　类的层次系统

现实生活中的一个对象可能拥有多个身份,在不同的情景中扮演不同的角色.在对象代理模型(object deputy model, ODM)[23]中,基于源对象生成的代理对象主要用于表示一组对象在不同情景中的角色或者对象的特定侧面.对象代理模型在源类上定义了 SELECTION,JION,UNION,PROJECTION 和 EXTENSION 等操作语义生成具有不同角色的代理类.源类和代理类分别解释源对象和代理对象的模式信息(schema).对于一个代理类,定义在源类上的一系列操作成为生成该代理类的代理规则.同时,代理类可以迭代地作为源类生成新的代理类.双向指针能够实现代理对象和源对象之间的双向通信.关联代理对象的查询通过双向指针可以获取相应的数据,从而避免代价昂贵的连接操作.不同于物化视图,对象代理模型采用双向指针物化对象关联能够显著减少维护代价.因此,本文引入对象代理模型管理知识图谱数据以及数据之间的关联性.

知识图谱是无模式数据,对象代理模型通常只能管理具有固定模式的数据,因此,本文扩展对象代理模型 ODM 为聚簇对象代理模型 CODM 来解决这种模式冲突.现在给出聚簇实体类(聚簇实体

对象)、代理关系类(代理关系对象)组成的类层次系统的定义.

定义 1. 聚簇实体对象(cluster entity object).一个聚簇实体对象定义为 $o^{ce}=\langle id^o, id^c, P^o \mid P^o \rightarrow \{(k_i^{ce}, M_i^{ce})\}, A^o \mid A^o \rightarrow \{A_i^{ce}\}\rangle$,其中,$id^o$ 代表聚簇实体对象的唯一标识符,id^c 代表 o^{ce} 所属聚簇实体类的唯一标识符,一个聚簇实体对象的谓语对(k_i^{ce}, M_i^{ce})由一个知识谓语及其关联元知识谓语集合组成,P^o 代表 o^{ce} 实例化的谓语对集合,A^o 代表定义在对象 o^{ce} 谓语集合属性上的所有读写操作.

定义 2. 聚簇实体类(cluster entity class).一个聚簇实体类定义为 $c^{ce}=\langle\{o_j^{ce}\}, id^c, P^c \mid P^c \rightarrow \{P_j^o\}, A^c \mid A^c \rightarrow \{A_j^o\}\rangle$,其中,$\{o_j^{ce}\}$代表一个聚簇实体类实例组成的对象集合,$id^c$ 代表 c^{ce} 的唯一标识符,P^c 代表一个由所有 c^{ce} 实例对象属性的谓语对(包括知识谓语和元知识谓语)组成的潜在谓语集合,A^c 代表一个由所有定义在 c^{ce} 实例对象上的读写操作组成的潜在读写操作集合.

定义 3. 代理关系对象(deputy relation object).一个代理关系对象定义为 $o^{dr}=\langle id^o, id^c, P^o \mid P^o \rightarrow \{(k_i^{dr}, M_i^{dr})\}, A^o \mid A^o \rightarrow \{A_i^{dr}\}\rangle$,其中,$id^o$ 代表代理关系对象的唯一标识符,id^c 代表 o^{dr} 所属代理关系类的唯一标识符,一个代理关系对象的谓语对(k_i^{dr}, M_i^{dr})由一个知识谓语及其关联元知识谓语集合组成,P^o 代表 o^{dr} 实例化的谓语对集合,A^o 代表定义在对象 o^{dr} 谓语集合属性上的所有从源对象继承衍生而成的读写操作.

定义 4. 代理关系类(deputy relation class).一个代理关系类定义为 $c^{dr}=\langle\{\{o_j^{dr} \mid o_j^{dr} \rightarrow \{o_i^s\}, sp(\{o_i^s\})=\text{true}\}, id^c, P^c \mid P^c \rightarrow \{P_j^o\}, A^c \mid A^c \rightarrow \{A_j^o\}\rangle$,其中,$o_j^{dr} \rightarrow \{o_i^s\}$表示代理关系对象 o_j^{dr} 由一个源对象集合$\{o_i^s\}$衍生而来,sp 代表源对象集合需要满足的选择条件,id^c 代表 c^{dr} 的唯一标识符,P^c 代表一个由所有 c^{dr} 实例对象属性的谓语对(包括知识谓语和元知识谓语)组成的潜在谓语集合,A^c 代表一个由所有定义在 c^{dr} 实例对象上的读写操作组成的潜在读写操作集合.

图 2 展示了聚簇对象代理模型的类层次系统.聚簇实体类位于类层次系统的底层,是构建代理关系类的基础.聚簇实体类和下层的代理关系类可以迭代地作为源类构建上层的代理关系类.从功能上来看,聚簇实体类主要负责知识图谱的模式化逻辑存储建模,而代理关系类则用于构建加快复杂查询的语义存储建模.

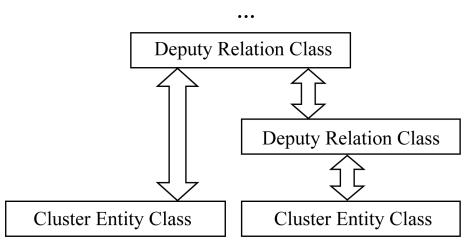


Fig. 2 CODM hierarchical system

图 2 CODM 类层次系统

2.2 聚类和聚簇实体类

在知识图谱的查询处理中,星型查询具有非常特殊意义[21].一方面,星型简单查询在实际应用中占据绝对的数量优势;另一方面,许多复杂查询可以分解为一些块状的星型查询.拥有固定存储模式具有天然的优势支持知识图谱上的星型查询.本文提出了一种新颖的模式聚簇算法将无模式的知识图谱数据转换为模式数据.算法的核心是定义了集合编辑距离(set editing distance),用于衡量不同模式间的相似性,作为模式合并的依据.集合编辑距离定义为

$$D(K_1,K_2)=\begin{cases}\dfrac{|K_2-K_1|}{|K_2|}, & \text{if } K_1 \subset K_2;\\ D_1+D_2, & \text{otherwise;}\end{cases} \tag{1}$$

$$D_1(K_1 \mid K_1K_2)=\frac{|K_1 \cup K_2-K_2|}{|K_1 \cup K_2|}; \tag{2}$$

$$D_2(K_2 \mid K_1K_2)=\frac{|K_1 \cup K_2-K_1|}{|K_1 \cup K_2|}; \tag{3}$$

其中,$D(K_1,K_2)$表示谓语集合 K_1 和潜在聚簇谓语集合 K_2 之间的集合编辑距离.如果 K_1 是 K_2 谓语集合的子集,则 D 表示从 K_1 到 K_2 的转换代价.否则,$K_1 \cup K_2$ 取代 K_2 成为新的潜在聚簇谓语中心.这样,谓语集合 K_1 和 K_2 都需要归并到新的聚簇谓语中心(式(2)和式(3)),转换代价为 2 个谓语集合分别转换为聚簇谓语中心的集合编辑距离之和.

算法 1. 模式聚簇算法(Schema Cluster Algorithm).

输入:三元组集合 T、聚簇模式数目 k;

输出:聚簇模式集合 Ψ.

① 基于相同主语归并三元组,生成实体集合 E.

② 基于实体(集合)$e \in E$ 的谓语,生成实体模式(集合)$\varphi \in \Phi$.

③ 实体模式集合的最大谓语数目,记为 L.

④ 分割 L 为$\dfrac{k}{2}$个聚簇初始化区域,依据谓语数目将实体模式分配到聚簇初始化区域.

⑤ 依据聚簇初始化区域的实体模式数目为权重分配聚簇模式数目,并进行相应的初始化.

⑥ Repeat

⑦　基于式(1)计算实体模式到每一个当前聚簇中心的集合编辑距离,将其分配给集合编辑距离最小的聚簇中心.

⑧　更新聚簇中心的谓语集合,使得每一个新聚簇中心能够包含其所涵盖实体模式的所有谓语.

⑨ Until no change

⑩ 返回聚簇中心(聚簇模式 Ψ).

算法 1 展示了基于集合编辑距离的知识图谱模式聚簇过程.整个算法分为 3 个部分:三元组生成实体模式集合(步骤①～②);分区初始化模式聚簇中心(步骤③～⑤);迭代进行模式聚簇(步骤⑥～⑨).拥有相同主语的三元组归并为一个实体,这样,所有的三元组就转换成一个实体集合(步骤①).每一个实体的谓语组成一个实体模式,所有实体的实体模式组成一个实体模式集合(步骤②).一个直观的现象是不同谓语数目的实体模式必然存在模式的差异性,而且谓语数目差值越大,其模式差异性也相应地增大.因此,模式聚簇算法将实体模式的谓语数目作为重要考虑因素进行聚簇中心的初始化.统计所有实体模式的谓语数目(步骤③),最大谓语数目 L 被分解为$\frac{k}{2}$个聚簇初始化区域,根据谓语数目将实体模式分配到聚簇初始化区域(步骤④),依据聚簇初始化区域内实体模式数目为权重来分配聚簇模式数目进行随机初始化(步骤⑤).步骤⑥～⑨是 k-means 聚类过程.依据集合编辑距离计算每一个实体模式到当前聚簇中心的距离,并将其分配至拥有最小集合编辑距离的聚簇中心,接着,依据当前聚簇中心的实体模式集合调整聚簇中心位置.当所有的聚簇中心不再发生实体模式变动时,结束模式聚簇,并将最后的聚簇中心作为聚簇模式进行返回.

我们将在 2.3 节结合物化聚簇实体类产生的空值现象讨论模式聚簇算法的合理性,并给出设置超参数应当遵循的规则.

2.3　物化聚簇实体类

本节将介绍知识图谱的模式存储和属性数据类型的特化.知识和元知识组成知识图谱.知识图谱的知识三元组经过 2.2 节的模式聚簇生成一个聚簇模式集合.基于聚簇模式构建聚簇实体类,聚簇模式涵盖的实体模式关联的所有实体称为聚簇实体类的实例对象,即聚簇实体对象.为了进行统一建模,元知识作为附属陈述与知识进行关联,实现元知识的模式化存储.聚簇实体类的每一个知识谓语连同关联的元知识谓语集合形成一个谓语对,一个聚簇实体类的所有谓语对构成聚簇实体类的模式信息.这样,通过模式聚簇算法就将无模式的知识图谱转换成模式化数据,实现对象代理模型对知识图谱的模式化建模.

聚簇实体类主要负责知识图谱的模式化存储建模.在聚簇实体类表中,聚簇实体类的每一个实例对象物化为对象指针(OID)进行类域范围的唯一性指代.在聚簇代理模型的物化系统中,对象指针有 2 个突出用途:1)构建聚簇模式主表和多值附表之间的关联;2)支持代理关系类的物化.在聚簇实体类物化时,元知识谓语取得了与关联知识谓语相同的地位,分别占据聚簇实体类表的一个表列.元知识谓语列的一个值构成了对知识谓语列相应知识事实的补充陈述.图 3 呈现图 1 中知识图谱的模式化物化形式.CEC1 和 CEC2 是聚簇实体类的 2 个物化模式表.在 CEC1 表中,元知识谓语列 a2_occurIn 中的 1891

OID	subj	gender	bornIn	a2_occurIn	spouse	a3_conf
1	Ruth	Female	USA	1891		
2	Esther		USA	1893	William	0.23
3	Richard	Male	USA	1899	Alan	0.08
4	James		USA	1907		
5	Eliot		USA	1910	Josefin	0.12
6	Delano	Male	USA	1914		

(a) CEC1

OID	X	xxx
…	…	

(b) CEC1_X

OID	subj	tenure	a1_conf	hasChild	presidentOf	a3_occurIn
1	Cleveland	22	0.87		USA	1893
2	Roosevelt	32	0.92		USA	1933

(c) CEC2

OID	hasChild	a1_conf
1	Ruth	0.03
1	Esther	0.56
1	Richard	0.62
2	James	0.34
2	Eliot	0.65
2	Delano	0.45

(d) CEC2_hasChild

Fig. 3　Materialization of cluster entity class

图 3　物化聚簇实体类

代表了一个关于知识事实〈Ruth, bornIn, USA〉的补充陈述.

多值属性广泛地存在于知识图谱.例如,一个成人可能拥有多个孩子,一个城市可能拥有多个简称等.聚簇实体类的每一个多值谓语关联一个附属表.这个附属表的模式信息由主表对象指针、多值谓语以及相关元知识谓语集合组成.聚簇实体类的主表和多值附属表通过对象指针实现了有效的关联.这样,通过 SELECTION 操作而非 JOIN 操作就可以获得需要的聚簇实体对象,显著地提升星型查询的处理效率.在图 3 中,CEC2_hasChild 表是主表 CEC2 中多值属性 hasChild 谓语的附属表.

面向知识和元知识设计强模式的物化策略能够极大地减少数据冗余,加速星型查询的处理效率.与此同时,这种模式化的存储设计实际上为每一种谓语属性(包括知识谓语和元知识谓语)分配了一个特定的表列,这客观上创造了条件为每一个表列增加面向不同数据类型的索引来支持高效的范围查询.

一个理想的模式聚簇方法应该满足这样一条准则:在聚簇数目可接受的情况下,物化聚簇实体类应尽可能地减少空值现象.模式聚簇算法是符合这个准则的.极端情况下,将实体模式作为聚簇实体模式,无须进行实际的聚簇操作.一个实体模式设计一个存储模式,聚簇实体类的物化不会产生任何空值现象.但是,这种情境下却需要最大数目的聚簇模式.在聚簇模式数目(超参数 k)减少时,实体模式就需要进行模式合并,这就造成了空值现象.随着聚簇模式数目的逐步减少,将会有更多的实体模式合并成聚簇实体模式或者归并进当前的聚簇实体模式,聚簇实体类的物化将会产生更多的空值.最后,一种极端情况是将所有的实体模式合并成一个聚簇实体模式,将造成最严重的空值现象.但是,这种情况下的聚簇模式数目又是最少的.因此,在我们提出的模式聚簇算法中,聚簇模式数目和空值数目整体上将呈现近似反比的函数关系.这就给我们提供了一个可操作的设置聚簇模式数目的规则:在聚簇模式数目可接受的情况下,适当地增大聚簇模式数目就可以减少物化时的空值现象.

2.4 物化代理关系类

知识图谱查询处理的瓶颈主要集中于连接操作.知识图谱查询通常可以分为星型查询(star-shaped)、链式查询(chain-shaped)和雪花状查询(snowflake-shaped)[15]等 3 种查询类型.聚簇对象代理模型的强模式存储特性固有地支持星型查询.为了加速链式和雪花状等复杂查询的处理效率,我们提出了采用对象代理机制将复杂对象关系变换为具有特定规则的对象序列,通过物化具有泛化语义表达能力的对象关系来加速复杂语义关系的特化查询.

一个代理关系类表达了一种代理规则,实际上解释了一组源对象以何种方式关联起来展现一种复杂的语义关系.在聚簇对象代理模型中,对象封装了所有的属性和动作,由对象指针进行唯一性标识.对象之间的复杂关系可以通过对象指针进行表达,物化代理关系类的过程就是构建起一组源类对象指针和一个代理类对象指针之间的映射关系,这样就实现了轻量级的语义存储建模.聚簇对象代理模型提供了 SELECTION,JOIN 和 UNION 三种原子操作来构建代理关系类,这些操作的组合使用能够表达各种复杂的实体关系.3 种原子操作的定义为

1) SELECTION

$$Select(C^{dr})=\{Select(C^{s},sp),$$

$$P^{dr}=\bigcup_{i=1}^{n}(k_i^{s}\vee M_i^{s})\},$$

其中,C^{s} 表示执行选择操作的源类,sp 代表选择规则,P^{dr} 指代从源类集合继承的知识谓语和元知识谓语集合组成的谓语对集合.图 4(a)中的对象关系图呈现了构建在以聚簇实体类 CEC2 为源类的 SELECTION 代理类的代理规则,而图 4(a)中的物化表图则呈现了基于 CEC2 对象指针的代理关系类的物化形式.

2) JOIN

$$Join(C^{dr})=\{Join(\{C_1^{s},C_2^{s}\},(p_1^{s},p_2^{s})),$$

$$P^{dr}=(\bigcup_{i=1}^{n1}(k_i^{s}\vee M_i^{s})\vee\bigcup_{j=1}^{n2}(k_j^{s}\vee M_j^{s}))\},$$

其中,$\{C_1^{s},C_2^{s}\}$ 表示 2 个存在 JOIN 关系的源类,p_1^{s},p_2^{s} 分别指代源类中存在 JOIN 关系的 2 个谓语对,而 P^{dr} 则表示从 2 个源类继承演化得到的谓语对集合,是 2 个源类谓语对集合的合集.图 4(b)中的对象关系图呈现了构建在以聚簇实体类 CEC1 和 CEC2 为源类的 JOIN 代理类的代理规则,而图 4(b)中的物化表图则呈现了基于 CEC1 和 CEC2 对象指针的代理关系类的物化形式.

3) UNION

$$Union(C^{dr})=\left\{Union\left\{\bigcup_{i=1}^{n1}C_i^{s}\right\},\right.$$

$$\left.P^{dr}=\bigcap_{i=1}^{n1}(\bigcup_{j=1}^{n2}(k_{j\_i}^{s}\vee M_{j\_i}^{s}))\right\},$$

其中,$\bigcup_{i=1}^{n1}C_i^{s}$ 表示一组存在 UNION 关系的源类集合,P^{dr} 表示从源类继承演化来的谓语对集合,是所有源

类谓语对集合的公有合集.图 4(c)中的对象关系图呈现了构建以聚簇实体类 CEC2 为源类的 UNION 代理类的代理规则,而图 4(c)中的物化表图则呈现了基于 CEC2 对象指针的代理关系类的物化形式.

需要指出的是,聚簇实体类和代理关系类都可以作为源类构建上层的代理关系类.SELECTION 代理类表达一种面向源对象的选择规则,而 JOIN 代理类则表达 2 个源类之间存在一种关联关系.UNION 代理类通过融合多种选择和连接规则实现了复杂代理关系的表达.这些原子操作的组合使用能够支持各种复杂语义关系的建模表示.

一个查询称为一个代理关系类的特化查询意味着这个查询结果可以由代理关系类的物化结果特化生成.与此同时,这个代理关系类也称为这个查询的泛化类.直观来看,一个代理关系类是一个查询的泛化类需要至少满足其中一个条件:1)查询是代理关系类中知识谓语或元知识谓语的属性实例化;2)查询是代理关系类的知识谓语或元知识谓语的关系附加.图 5 呈现了一个泛化类和 2 个特化查询,其中,图 5(a)表示了由 SELECTION 和 JOIN 两个原子操作组合表达的链式泛化类,图 5(b)和图 5(c)分别表示了属性实例化和关系附加 2 种特化查询.

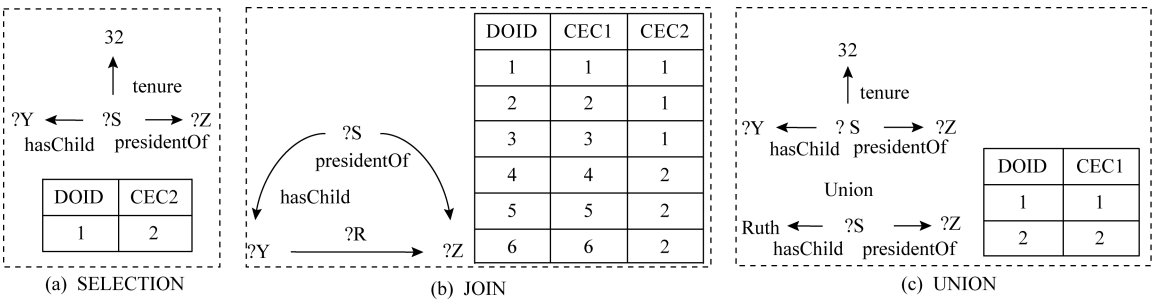


Fig. 4　Materialization of deputy relation class

图 4　物化代理关系类

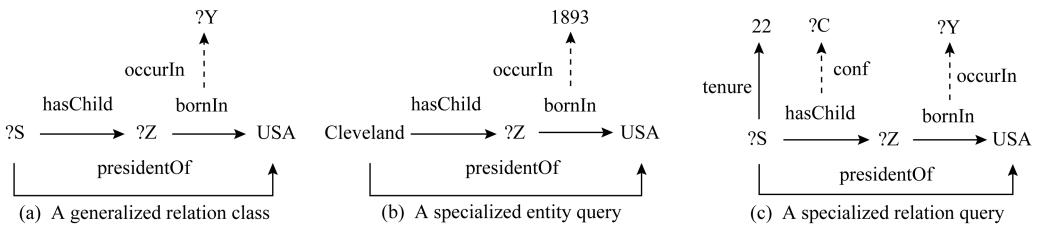


Fig. 5　A generalized class and two specialized queries

图 5　泛化类和特化查询

3　实验与结果

在本节中,我们介绍数据集和评价指标等实验设置,采用本文提出的技术进行知识图谱的管理,并报告聚簇对象代理模型在减少数据冗余和支持范围查询方面的有效性.

3.1　实验设置

3.1.1　数据集

我们选择 YAGO2s[12] 和 LUBM[24] 2 个数据集作为实验评估的测试数据.YAGO2s 是一个大规模事实型知识图谱,主要涵盖从 Wikipedia 抽取的各类事实,并整合了 WordNet 和 GeoNames 等相关的实体信息.由于 YAGO2s 是一个真实的知识图谱数据,因而能够如实地反应生产场景中知识图谱的各种存储需求.LUBM 是一个大学领域的人工合成型本体库,广泛地应用于评估知识型知识图谱管理系统[5-7,25].LUBM 的一个显著特征是能够根据评估需要生成任意大小的知识图谱,可验证知识图谱管理方法的伸缩性.

YAGO2s 是由知识陈述和元知识陈述组成的知识图谱,包含时间、高度以及地理经纬度等大量连续属性.由于 LUBM 是一个知识型知识图谱,因此,我们针对特定的知识陈述进行连续型元知识陈述的扩展,使其满足带有元知识知识图谱建模的各种评估需要.这些扩展的连续元知识属性基于给定的取值范围随机生成.表 5 给出了 LUBM 连续元知识属性的取值范围.

Table 5 Range of Metadata Expansion

表 5 元知识扩展范围

Type	Predicate	Year	Confidence
work	ub:worksFor	[1998,2003]	[0.2,0.98]
	ub:memberOf	[2003,2009]	[0.2,0.98]
	ub:headOf	[2009,2018]	[0.2,0.98]
degree	ub:undergraduate DegreeFrom	[2000,2005]	[0.45,0.98]
	ub:masterDegreeFrom	[2005,2008]	[0.45,0.98]
	ub:doctoralDegreeFrom	[2008,2012]	[0.45,0.98]
author	ub:publicationAuthor	[1995,2018]	[0.6,0.98]

由于 YAGO2s 没有提供任何的测试查询用例，因此，本文采用 LUBM 提供的查询用例进行检索性能的有效性验证.文献[24,26]分别给出了设计在 LUBM 测试集上的评估查询用例.从这些查询中选择至少包含表 5 中一个谓语的测试用例，移除重复和相似查询项，剩余查询组成本文实验的评估查询用例.根据查询的复杂程度，将这些测试用例分为星型查询(sq)和复杂查询(cq)2 个类别.最后，对这些评估查询语句进行元知识属性的扩展，例如，查询添加“Year＞2005 and Confidence＜0.8”的过滤条件.表 6 呈现了评估查询语句的分类情况，并给出了新查询用例和原查询用例的编号对应关系，其中，lq * 和 mq * 分别表示查询来自文献[24]和文献[26]，而 * 则表示当前查询在原查询用例集的编号.

Table 6 Type of Evaluated Query

表 6 评估查询的类别

Type	New ID of query	Old ID of query
Star-shaped query(sq)	sq1	lq3
	sq2	lq5
	sq3	lq13
	sq4	mq4
Complex-shaped query(cq)	cq1	lq8
	cq2	mq1
	cq3	mq3
	cq4	mq6

3.1.2 评估设置

在逻辑存储建模方面，我们选择 Standard Reified Statement(SRS)[10]，Singleton Property Approach (SPA)[9]和 YAGO2 Reified Statement(YRS)[12]等 3 种经典的建模方法用于知识图谱元知识建模的实验评估.最近一些研究工作通过理论和实验反复验证了 SRS，SPA 以及 YRS 等方法在知识图谱建模方面的有效性[27-29].SRS 和 SPA 将知识图谱建模成三元组，而 YRS 和 CODM 则分别采用元组和对象形式进行知识图谱建模.在语义存储建模方面，SW-Store[16]和 gStore[6]分别是基于谓语建模和基于实体建模的典型代表，而基于查询的语义存储建模还没有发布支持带有连续属性的知识图谱建模管理方法.由于 gStore 已经展示了对于 SW-Store 的压倒性优势，因此，我们主要以 gStore 作为语义存储建模的对比评估方法.

为了评估知识图谱建模方法对查询性能的影响，我们需要为不同的建模方法选择相应的管理系统.在逻辑存储建模方面，现有的 RDF3X[4]，gStore[6]和 Virtuoso[30]等都能够支持三元组框架的知识图谱建模管理.但是，由于 RDF3X 不支持连续属性的查询，因此，选择 gStore 和 Virtuoso 用于 SRS 和 SPA 建模的知识图谱管理.YAGO2[12]推荐了一种基于对象关系型数据库 Postgresql 的知识图谱管理方法.由于对象建模的需要，我们选择 Postgresql 作为 CODM 建模的知识图谱管理系统.在语义存储建模方面，gStore 提供了一种在有向图上进行实体建模的原型系统.

在本文实验中，我们选择 gStore，Virtuoso Open Source 7.2.5 以及 Postgresql 10.4 等版本的知识图谱管理系统.这些管理系统运行在 16G 内存的 Ubuntu Desktop 14.04.3 操作系统上进行实验性能的评估.

3.2 空间有效性

基于 LUBM 数据生成器获得了 10 M 条的知识陈述，并结合 3.1 节策略补充了相应的元知识陈述，形成了 LUBM10M 数据集.我们分别按照 SRS，SPA，YRS 以及 CODM 将 LUBM10M 和 YAGO2s 两个评估测试集转换成相应的知识图谱逻辑存储建模表示形式.由于这 4 种方法采用三元组、元组以及对象等不同形式进行建模，因此，我们以元素数目取代三元组数目进行数据冗余的评估.一条三元组含有 3 个元素，而一个对象(元组)的元素数目则对应于属性(属性列)的数目.类似于文献[9]，我们将不同知识图谱建模方法的元素划分为知识元素、控制元素以及元知识元素 3 种类别.这种划分方法有助于评估不同知识图谱建模方法的哪些具体因素影响知识图谱建模的数据冗余.

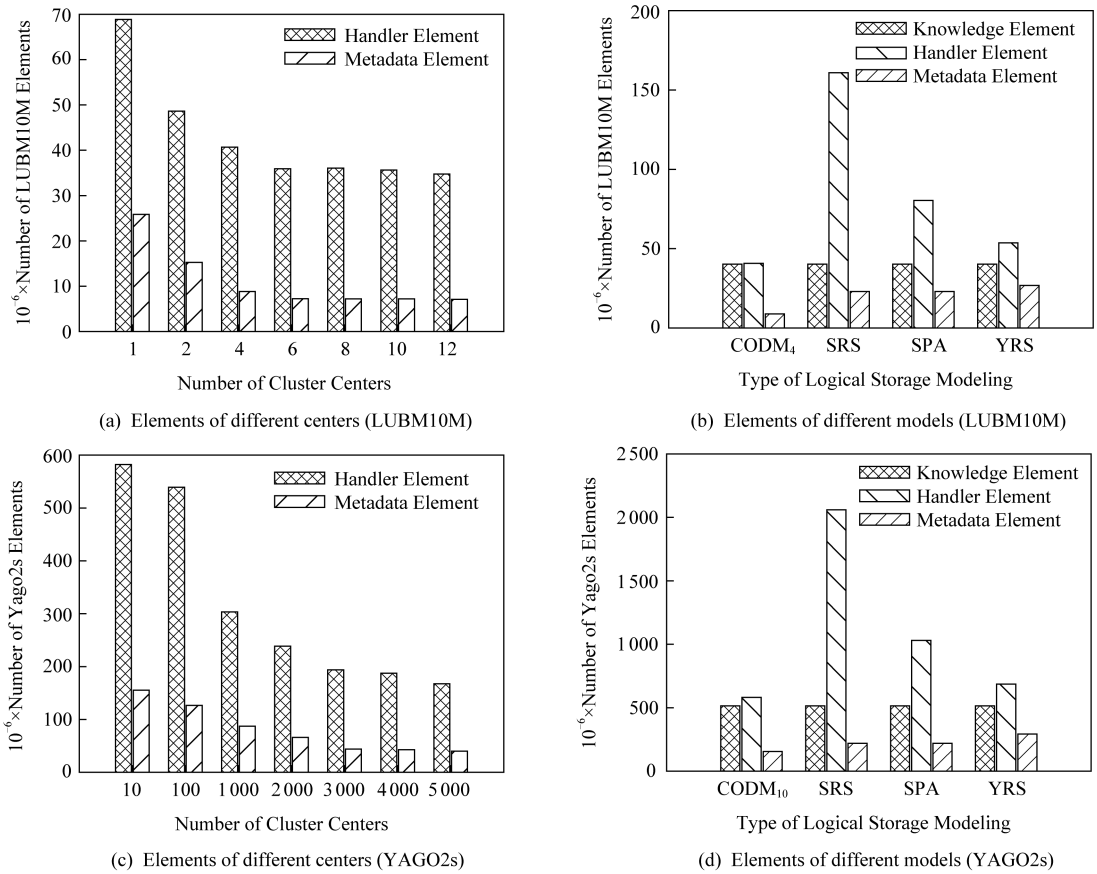
1) 知识元素(knowledge element).知识陈述和元知识陈述组成知识图谱，而知识元素是由知识陈述分解而来.传统的资源描述框架是知识陈述的

主流表达形式,因此,知识元素由知识图谱中知识陈述的主语、谓语和宾语组成.这样,一条知识三元组蕴含 3 个知识元素.

2) 控制元素(handler element).在知识图谱建模时,控制元素的主要任务是将一条知识陈述转换为一个独立的知识句柄,用于知识陈述关联元知识的陈述声明.一条知识陈述只有转换成知识句柄才能进行元知识的陈述声明.不同的知识图谱建模方法拥有不同的知识句柄化方式.在三元组框架中,SRS 需要 4 条三元组陈述 12 个控制元素,而 SPA 则需要 2 条三元组陈述 6 个控制元素将一条知识陈述转换成一个知识句柄.在类三元组框架中,YRS 需要一个元组陈述〈ID,Subject,Predicate,Object〉,4 个控制元素句柄化一个知识陈述.由于模式化存储的缘故,CODM 只需要一个对象陈述〈OID,Subject,Object〉,3 个控制元素建模一个知识陈述.在有 n 条知识陈述共享一个主语的情况下,CODM 的表现更加突出,只需要 $n+2$ 个控制元素就可以实现这些知识陈述的句柄化.

3) 元知识元素(metadata element).在逻辑存储建模时,元知识通常基于知识句柄进行陈述声明.在三元组框架内,SRS 和 SPA 分别以〈知识增强句柄,元知识谓语,元知识宾语〉和〈属性唯一句柄,元知识谓语,元知识宾语〉的形式关联一条元知识陈述.YRS 有 2 种元知识陈述声明情景.如果存储模式含有元知识谓语(例如时间、位置和情景),YRS 只需要一个元知识元素声明一条元知识陈述.否则,YRS 则至少需要 4 个元知识元素来声明一条元知识陈述.CODM 将元知识陈述作为知识陈述的附属,实现了元知识陈述的模式化存储.因此,CODM 只需要一个元知识元素就可以声明一条元知识陈述.

我们以 LUBM10M 和 YAGO2s 作为评估知识图谱逻辑存储建模的数据集.在评估空间有效性时,将空值以及标识(YRS 的 ID 和 CODM 的 OID)都作为元素进行统计.图 6 展示了 4 种知识图谱建模方法的空间存储情况.在 CODM 模型中,超参数代表最终聚簇实体模式的数目,影响哪些实体模式的实体对象存储到同一张聚簇模式表.LUBM 是一个模拟的知识图谱数据集,拥有 12 个实体模式,而 YAGO2s 则是一个真实的知识图谱数据集,实体



(a) Elements of different centers (LUBM10M)　(b) Elements of different models (LUBM10M)

(c) Elements of different centers (YAGO2s)　(d) Elements of different models (YAGO2s)

Fig. 6　Space effectiveness of four modelings for logical storage

图 6　4 种逻辑存储建模方法的空间有效性

关系非常复杂,拥有超过2万多个实体模式.LUBM和YAGO2s都很好地验证了2.3节的结论,即随着聚簇实体模式的逐步增大,控制元素和元知识元素的空值数量逐渐减少.从图6(a)(c)可以看出,随着聚簇模式数目的逐渐增加,超参数对于减少空值的边际效益是逐步降低的.这种现象表明,只需要设置一个少量的聚簇中心数目就可以取得令人满意的聚簇结果,例如YAGO2s和LUBM10M的超参数可以设置为1 000和4.

图6(b)(d)分别呈现了4种逻辑存储建模方法在2个数据集上的建模结果,其中,CODM在LUBM10M和YAGO2s上的超参数分别设置为4和10.从横向比较来看,SRS,SPA和YRS这3种建模方法的控制元素数量都明显高于知识元素,而CODM建模的控制元素数量和知识元素数目基本保持一致.在元知识建模方面,CODM建模的元知识元素数量大致保持在SRS和SPA建模的40%~70%之间.YRS和CODM都采用了模式化方法进行元知识建模.但是,由于YRS的存储模式只涵盖公有谓语,这样就造成了大量的空值情况.因此,YRS建模的元知识元素数量达到了CODM建模的2倍(YAGO2s)到4倍(LUBM10M).总体来看,CODM进行知识图谱建模能够极大地减少数据冗余,取得了最好的空间存储优势.

3.3 时间有效性

在3.3节,我们将从查询响应时间、数据的可伸缩性以及代理机制加速复杂查询等方面验证CODM支持范围查询的有效性.

知识图谱建模主要分为逻辑存储建模和语义存储建模.逻辑存储建模构建知识陈述和元知识陈述的逻辑关系,其存储方式则依据建模方式的不同而有所差异.三元组框架的直接存储方式就是构建三列表分别用于主语、谓语和宾语的存储,因此,我们在Virtuoso系统上构建三列表来存储标准增强方式(SRS_Virtuoso)和属性唯一方式(SPA_Virtuoso)生成的知识图谱数据.类三元组框架采用YAGO2推荐的存储方法(YRS_PG),而聚簇对象代理模型则构建模式化存储(CODM_PG).在语义存储建模方面,gStore是基于实体聚簇存储在图结构上的一种实现方式.由于gStore并不支持属性唯一方式的元知识逻辑表示,因此,语义存储建模主要采用标准增强方式进行知识陈述和元知识陈述的逻辑建模表示(SRS_gStore)

聚簇对象代理模型有2个显著特性.一个是模式化存储减少数据冗余和JOIN操作.以低冗余方式进行知识图谱建模能够减少I/O代价,而更少的JOIN操作则能够缩小查询的响应时间.另一个特性则是数据类型的特化支持,可以构建面向特定数据类型的索引加速,提升连续属性查询的响应效率.图7呈现了知识图谱建模方法的时间有效性.由于系统设计的完备性问题,gStore无法支持BIND语法操作,因此,SRS_gStore没有sq3的响应时间.

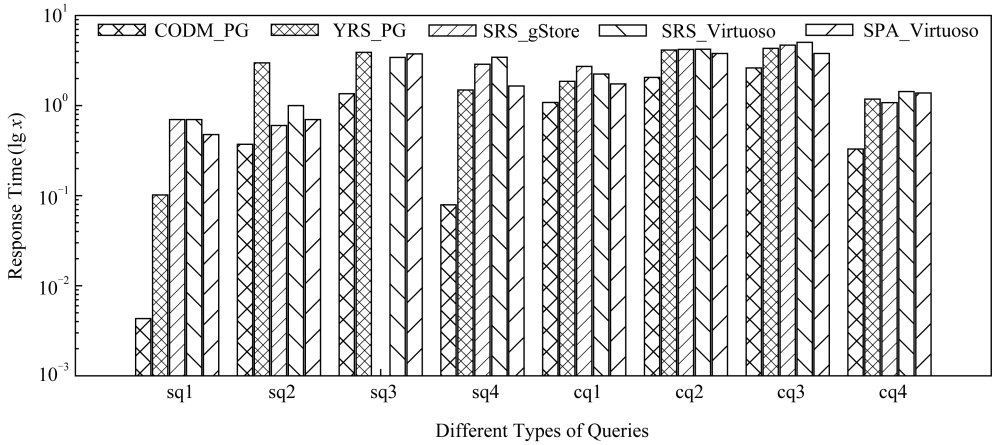


Fig. 7 Time effectiveness of knowledge graph modelings (LUBM10M)

图7 知识图谱建模方法的时间有效性(LUBM10M)

从逻辑存储建模角度来看,CODM建模方法取得了最好的查询响应时间,在所有查询上均大幅度地领先于其他类型查询.在三元组框架内,标准增强方式(SRS_Virtuoso)在进行知识陈述句柄化过程中产生的控制元素远远高于属性唯一方式(SPA_Virtuoso).因此,不管是星型查询,还是复杂查询,

标准增强方式需要更多的 I/O 和 JOIN 操作,其查询响应时间也明显高于属性唯一方式.三元组框架知识陈述句柄化产生的控制元素数目多于类三元组框架,但是,三元组框架建模的元知识元素数量通常少于类三元组框架,因此,这 2 种建模方法的实际查询响应时间依赖于具体的连接条件.

从语义存储建模角度来看,YAGO2(YRS_PG)实现了公共元知识的模式化存储,而聚簇对象代理模型(CODM_PG)则实现了知识陈述和元知识陈述的完全模式化存储.因此,这 2 种存储建模方式实现了不同程度的实体聚簇,属于基于实体聚簇的存储建模方式.从图 7 可以看出,CODM_PG 和 YRS_PG 的查询性能明显优于 SRS_gStore 的查询性能.这主要是因为 gStore 构建的类哈希树索引结构在离散字符型数据上拥有非常优越的查询性能,但是,当数据含有大量连续性属性,特别是涉及范围查询时,这种索引结构的性能就表现的不是非常突出.与 YRS 的部分属性模式化相比,CODM 实现了完全的模式化存储,不仅避免了大量的连接操作,更重要的是能够以非常低的代价构建面向数据类型支持的各种索引结构,自然具有更加优秀的查询性能.

在评估图 7 的时间有效性时,聚簇实体类采用了超参数为 4 的 LUBM10M 数据集的模式聚簇结果.超参数的调整影响聚簇模式,决定哪些实体存储到同一个磁盘文件,进而造成了不同的数据冗余.由于聚簇对象代理模型实现了面向数据类型的索引支持,所有的查询字面值都是先走索引再读取磁盘数据,因此,超参数的设置对于查询响应的影响主要体现在不同数量的磁盘 I/O.由于超参数的设置对于数据冗余和磁盘 I/O 的影响是类似的,这样,超参数对于查询性能的影响可以参照图 6 中超参数对于数据冗余的影响分析得到.因此,本节不再给出模式聚簇算法的超参数对于查询性能的影响.

为了验证 CODM 模型的可伸缩性,我们利用 LUBM 数据生成器分别生成 10 M,30 M,50 M,70 M 以及 100 M 等 5 个不同三元组数量规模的评估测试集.图 8 呈现了 8 个评估查询在 5 个测试评估集上的运行结果.得益于 CODM 的模式化逻辑存储建模和数据类型特化,绝大多数查询处理的响应时间并没有随着数据规模的增大而明显的扩大,始终维持在 500 ms 以下,这就很好地验证了 CODM 模型能够有效地支持大规模知识图谱的查询应用.需要指出的是,模式化存储无法避免非星型查询之间的连接操作.在数据规模增大时,需要执行连接操作的中间结果也会随之增大,导致了数据规模对复杂查询 cq2 和 cq3 的响应时间影响比较明显,这就给验证代理机制的有效性提供了机会.

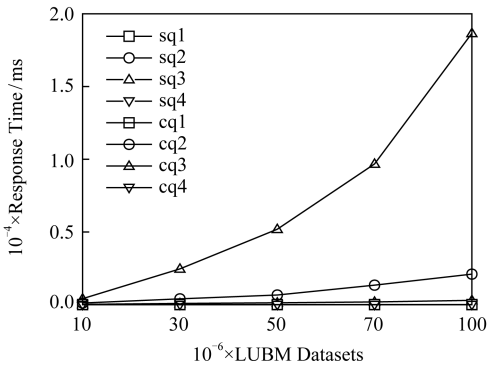


Fig. 8　Scalability of CODM

图 8　CODM 的可伸缩性

许多实体之间可能存在一些固有的关联,例如学术会议和研究主题.由于知识图谱主要面向互联网用户,一些特定的历史事件可能导致相关查询的急剧增长.对象代理机制可用于建模实体之间的复杂语义关联,通过物化固有的泛化对象关系来缩短复杂查询的响应时间.由于复杂查询 cq2 和 cq3 分别呈现了实体不规则形状的固有关系,因此,我们可以分别为这 2 个查询创建泛化的代理关系类来物化这 2 类固有实体关联.图 9 呈现了构建在代理机制上的查询性能,响应时间从原来的几十秒迅速回落到几毫秒.因此,通过代理关系类的物化结果来特化查询结果,代理机制能够极大地降低复杂查询的响应时间.聚簇实体类构建了实体的内部关联,而代理关系类则构建了实体的外部关联.实验结果证明了两者的配合使用,能够极大地提升复杂查询的处理能力.

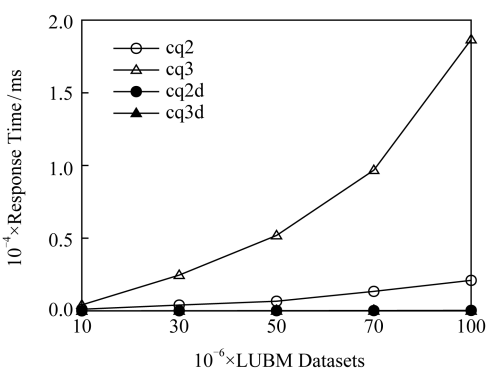


Fig. 9　Effectiveness of deputy mechanism

图 9　代理机制的有效性

4 总 结

本文提出了一种支持范围查询的低冗余知识图谱建模方法-聚簇对象代理模型(CODM).我们设计了集合编辑距离用于知识图谱聚簇,将无模式的知识图谱数据转换为模式化数据.在逻辑存储建模方面,CODM 将元知识陈述作为知识陈述的附属,基于聚簇模式实现知识陈述和元知识陈述的模式化存储和数据类型特化,减少了数据冗余,支持了连续属性的范围查询.在语义存储建模方面,CODM 通过代理机制实现知识图谱固有关联的轻量级语义建模,能够有效地适应不断演化的知识图谱查询,提升复杂查询的处理能力.实验结果验证了 CODM 模型在减少数据冗余和支持范围查询方面的有效性.

模式聚簇算法的超参数在一定程度上影响 CODM 模型的有效性,超参数的设定与知识图谱数据本身的特性存在很强的关联性.此外,CODM 模型在实际应用场景中的性能表现依然存在一些不确定因素.因此,CODM 模型在现实场景中应用和基于知识图谱数据特征来优化超参数设计将作为下一阶段的研究工作.

参 考 文 献

[1] Yang Deqing, He Jingrui, Qin Huazheng, et al. A graph-based recommendation across heterogeneous domains [C] //Proc of the 24th ACM Int on Conf on Information and Knowledge Management. New York: ACM, 2015: 463-472

[2] Cui Wanyun, Xiao Yanghua, Wang Haixun, et al. KBQA: Learning question answering over QA corpora and knowledge bases [J]. Proceedings of the VLDB Endowment, 2017, 10 (5): 565-576

[3] Sahoo S S, Bodenreider O, Hitzler P, et al. Provenance context entity (PaCE): Scalable provenance tracking for scientific RDF primer data [C] //Proc of the Int Conf on Scientific and Statistical Database Management. Berlin: Springer, 2010: 461-470

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

[5] Bornea M A, Dolby J, Kementsietsidis A, et al. Building an efficient RDF store over a relational database [C] //Proc of the 2013 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2013: 121-132

[6] Zou Lei, Özsu M T, Chen Lei, et al. gStore: A graph-based SPARQL query engine [J]. The VLDB Journal, 2014, 23 (4): 565-590

[7] Mohamed Y, Klaus B, Maya R, et al. Exploratory querying of extended knowledge graphs [J]. Proceedings of the VLDB Endowment, 2016, 9(13): 1521-1524

[8] Peng Peng, Zou Lei, Özsu M T, et al. Multi-query optimization in federated RDF systems [C] //Proc of the 23rd Int Conf on Database Systems for Advanced Applications. Berlin: Springer, 2018: 745-765

[9] Nguyen V, Bodenreider O, Sheth A. Don't like RDF reification?: Making statements about statements using singleton property [C] //Proc of the 23rd Int Conf on World Wide Web. New York: ACM, 2014: 759-770

[10] Manola F, Miller E, McBride B. RDF primer [EB/OL]. (2004-02-10) [2014-02-25]. https://www.w3.org/TR/2004/REC-rdf-primer-20040210

[11] Luis G, Kim A, Katja H, et al. Answering provenance-aware queries on RDF data cubes under memory budgets [C] //Proc of the 17th Int Semantic Web Conf. Berlin: Springer, 2018: 547-565

[12] 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

[13] Leeka J, Bedathur S, Bera D, et al. Quark-X: An efficient top-k processing framework for RDF quad stores [C] //Proc of the 25th ACM Int on Conf on Information and Knowledge Management. New York: ACM, 2016: 831-840

[14] Frey J, Müller K, Hellmann S, et al. Evaluation of metadata representations in RDF stores [J]. Semantic Web, 2019, 10 (2): 205-229

[15] Aluç G, Özsu M T, Daudjee K. Workload matters: Why RDF databases need a new design [J]. Proceedings of the VLDB Endowment, 2014, 7(10): 837-840

[16] Abadi D J, Marcus A, Madden S R. SW-Store: A vertically partitioned DBMS for semantic Web data management [J]. The VLDB Journal, 2009, 18(2): 385-406

[17] Sun Wen, Fokoue A, Srinivas K, et al. Sqlgraph: An efficient relational-based property graph store [C] //Proc of the 2015 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2015: 1887-1901

[18] Aluç G, Özsu M T, Daudjee K. Building self-clustering RDF databases using tunable-LSH [J]. The VLDB Journal, 2018 (1): 1-23

[19] Liu Qiao, Li Yang, Duan Hong, et al. Knowledge graph construction techniques [J]. Journal of Computer Research and Development, 2016, 53(3): 582-600 (in Chinese)
(刘峤, 李杨, 段宏, 等. 知识图谱构建技术综述[J]. 计算机研究与发展, 2016, 53(3): 582-600)

[20] Hou Mengwei, Wei Rong, Lu Liang, et al. Research review of knowledge graph and its application in medical domain [J]. Journal of Computer Research and Development, 2018, 55 (12): 2587-2599 (in Chinese)
(侯梦薇, 卫荣, 陆亮, 等. 知识图谱研究综述及其在医疗领域的应用[J]. 计算机研究与发展, 2018, 55(12): 2587-2599)

[21] Hernández D, Aidan H, Markus K. Reifying RDF: What works well with wikidata? [C] //Proc of the 11th Int Workshop on Scalable Semantic Web Knowledge Base Systems co-located with the 14th Int Semantic Web Conf. Berlin: Springer, 2015: 32-47

[22] Färber M, Frederic B, Carsten M, et al. Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago [J]. Semantic Web, 2018, 9(1): 77-129

[23] Peng Zhiyong, Kambayashi Y. Deputy mechanisms for object-oriented databases [C] //Proc of the Eleventh Int Conf on Data Engineering. Piscataway, NJ: IEEE, 1995: 333-340

[24] Guo Yuanbo, Pan Zhengxiang, Heflin J. LUBM: A benchmark for OWL knowledge base systems [J]. Web Semantics: Science, Services and Agents on the World Wide Web, 2005, 3(2-3): 158-182

[25] Peng Peng, Zou Lei, Özsu M T, et al. Processing SPARQL queries over distributed RDF graphs [J]. The VLDB Journal, 2016, 25(2): 243-268

[26] Atre M, Chaoji V, Zaki M J, et al. Matrix bit loaded: A scalable lightweight join query processor for RDF data [C] // Proc of the 19th Int Conf on World Wide Web. New York: ACM, 2010: 41-50

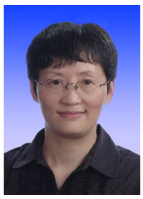
[27] Fu Gang, Bolton E, Rosinach N Q, et al. Exposing provenance metadata using different RDF models [J]. arXiv preprint arXiv:1509.02822, 2015

[28] Frey J, Müller K, Hellmann S, et al. Evaluation of metadata representations in RDF stores [J]. Semantic Web, 2019, 10 (2): 205-229

[29] Nguyen V, Sheth A. Logical inferences with contexts of RDF triples [J]. arXiv preprint arXiv:1701.05724, 2017

[30] Erling O. Virtuoso, a hybrid RDBMS/graph column store [J]. Data Engineering, 2012, 35(1): 3-8

Wang Fei, born in 1989. PhD candidate. His main research interests include database, complex data management, patent mining, information retrieval and natural language processing.



Qian Tieyun, born in 1970. PhD, professor. Member of CCF. Her main research interests include text mining, Web mining, and natural language processing.

Liu Bin, born in 1975. PhD, lecturer. Member of CCF. His main research interests include complex data management, patent data mining.

Peng Zhiyong, born in 1963. PhD, professor and PhD supervisor. Senior member of CCF. His main research interests include complex data, trusted data and Web data management.