Template-Based SPARQL Query and Visualization on Knowledge Graphs

Xin Wang[1,2](✉), Yueqi Xin[1], and Qiang Xu[1]

[1] School of Computer Science and Technology, Tianjin University, Tianjin, China
{wangx,xinyueqi,xuqiang3}@tju.edu.cn
[2] Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin, China

Abstract. With the popularity of Linked Open Data, a large amount of RDF data have been published and developed in the form of knowledge graphs, which can be publicly accessible via SPARQL endpoints. The efficiency of SPARQL querying on large-scale knowledge graphs has attracted increasing research efforts. In this paper, we propose a template-based query approach, which involves temporal, spatial, and domain-specific constraints to focus on certain resources of interest. Furthermore, query results which include a set of RDF triples are visualized in graph format to display entities and relationships in a user-friendly manner. We also analyze the visualized graph with ranking, partitioning, filtering, and statistics. Various template-based queries are designed and evaluated on the knowledge graph of DBpedia. It can be observed that template-based queries with temporal-spatial and domain-specific constraints can effectively facilitate users to obtain target answers by filtering out irrelevant information.

Keywords: Template query · Temporal-spatial · Visualization SPARQL

1 Introduction

With the popularity of *Linked Data*, a series of systematic methods to organize and publish RDF graphs have been developed [6], which aim to build large-scale *knowledge graphs*. As a flexible graph-like data model, an RDF graph is a set of triples, where each triple, consisting of a subject, predicate, and object, can be viewed as an edge in a directed graph from a subject to an object with the predicate as the edge label. SPARQL is the standard query language, endorsed by W3C, to retrieve data from RDF graphs. Since RDF has been gradually recognized as a major representation format by the knowledge graph community, in recent years, increasing importance has been attached to SPARQL for querying large-scale knowledge graphs more effectively.

To provide effective SPARQL query experience, there have been some research efforts on large-scale knowledge graphs. With the interface of TriniT [15], users need to input a complete SPARQL query, which is difficult for end users. On the other hand, in RelFinder [8,13], users need to provide two resources in the interface. Then, from a starting node, RelFinder explores its neighboring nodes and properties, which form the edges in the paths between the two user-specified resources. Meanwhile, Fusion [1] is also designed to offer discovery of relationships between two given resources. As a graph-based query method, NAGA [11] provides a user interface and focuses on a novel scoring model to rank results. Shekarpour et al. [14] propose basic graph pattern templates to generate SPARQL queries, where users merely need to provide some words of interest. However, the above methods are designed for professional programmers in most cases and the query results are returned in form of triples or paths, which are not visible and usable for users.

In this paper, we propose an approach to evaluating template-based SPARQL queries, which are defined as general SPARQL queries with Basic Graph Patterns (BGPs) and FILTER clauses. End users just need to provide several specific keywords or values to replace the *placeholders* in BGPs or FILTER clauses without knowing the syntax and semantics of SPARQL. Since the attributes about time and space are common and essential for the resources in knowledge graphs, we define the *basic template query* by adding temporal-spatial constraints that are provided by users. However, in the real world, users' requests not only focus on the temporal and spatial attributes of resources but also other various domain-specific properties. If users need to obtain precise relationships, they can choose the *refined template query*, which replaces the variables with certain values or adds triple patterns with extra constraints based on the basic template query.

Despite the proliferation of knowledge graphs, there still exist a number of obstacles, which hinder the large-scale deployment of knowledge graphs [2]. In general, for end users, the query result on knowledge graphs is a set of triples that are not yet sufficiently visible and usable. The *visualization* for knowledge graphs is commonly displayed in form of a labeled graph, such as the Paged Graph Visualization [4]. It turns out that the efficient innate human capabilities can be inspired to perceive and process data when knowledge graphs are presented visually [10]. Therefore, we convert the result of the template-based query into a series of nodes and edges and display them in graph format using the following three steps: (1) the result is loaded into the R tool; (2) *igraph* package is applied to construct an adjacency graph of the result in R; and (3) the adjacency graph can be visualized as a labeled graph in Gephi. We designed and evaluated 8 template-based queries on the knowledge graph of DBpedia. Moreover, We demonstrate these queries in several typical case studies via a SPARQL endpoint on a single machine.

Our main contributions include: (1) We propose the basic template SPARQL query with temporal-spatial constraints, where the keywords or values are provided by users to extract entities and relationships of interest. (2) Further we design the refined template query, which is defined by adding constraints with certain domain-specific values based on the basic one. (3) Various queries are designed and conducted as different case studies on knowledge graphs, such as the DBpedia dataset. Moreover, we realize a user-friendly visualization of each query result in graph format.

The rest of the paper is organized as follows. We discuss the related work in the areas of SPARQL query and visualization on knowledge graphs in Sect. 2. Section 3 provides the fundamental definitions of background knowledge. In Sect. 4, we describe in detail the template-based query. We also present the universal procedure of visualization in Sect. 5. Queries based on the basic and refined templates are evaluated in case studies in Sect. 6, and we conclude in Sect. 7.

2 Related Work

The existing query and visualization methods on knowledge graphs can be classified into the following four categories:

Language-Based Query. TriniT search engine [15] provides a user interface for querying, where users need to input a complete SPARQL query with knowing the syntax of SPARQL. Elbassuoni et al. [5] present a structured query mechanism on RDF graphs, which shows the inter-relationships between entities based on a language model. Despite with rich expressiveness, the above language-based query methods are designed for professional programmers in most cases. Thus, it is difficult for end users to use it effectively.

Keyword-Based Query. Heim et al. [8] propose an approach, called RelFinder, to searching the relationships between two user-specified nodes and displaying all the edges between the two nodes as a graph. Users can choose a starting node and incrementally explore a knowledge graph. The found resources are visualized as nodes connected by the edges labeled with the relationships. In addition, Lohmann et al. [13] present an approach with relationships filtering in four dimensions based on RelFinder. Fusion [1] implements a path discovery algorithm, which can find a path represented in form of a triple between two known resources given by users in a Web interface. However, the above methods only focus on the paths satisfying between the two given nodes labeled with keywords, which result in ignoring the global information that is vital for data statistics and analysis.

Graph-Based Query. NAGA [11] is a semantic search engine, which is built based on a knowledge base consisting of millions of entities and relationships. It presents a graph-based query language that allows the formulation of queries with semantic information, which can be more expressive than those standard keyword-based search approaches. The query result of NAGA is ranked using a scoring model, while it is not visualized in form of a graph and not intuitive for end users.

Template-Based Query. Shekarpour et al. [14] propose a set of predefined basic graph pattern templates to generate SPARQL queries with the user-supplied keywords. Users just need to provide several keywords, then the corresponding SPARQL query can be generated and the query result can be returned. LESS [2] presents a language, called LESS Template Language (LeTL), which can define arbitrary text-based output representations and support the integration of information. It provides a Web interface for users to edit the template with user-defined parameters, then returns the query result for users, whereas it cannot display the result in form of a graph.

Actually, most of the above methods simply automatically display the results in the Web applications without graphics visualization. Unlike the above methods, we combine keyword-based and template-based queries with a balance between flexibility and expressiveness to design the basic and refined template queries. In our method, the query result is displayed as a graph with interactive features and filter options considering the global information in different granularity and dimensions.

3 Preliminaries

In this section, we introduce the definitions of relevant background knowledge.

RDF data is a collection of triples denoted as (s, p, o), which states that the resource s has a relationship p to the resource o, where s is called the subject, p the predicate (or property), and o the object, which can be formally defined as follows:

Definition 1 (RDF graph)**.** *Let U and L be the disjoint infinite sets of URIs and literals, respectively. A tuple $(s, p, o) \in U \times U \times (U \cup L)$ is called an* RDF triple*. A finite set of RDF triples is denoted as $G = (V, E, \Sigma)$, called an RDF graph, where V is a set of vertices that correspond to all subjects and objects; $E \subseteq V \times V$ is a set of directed edges that correspond to all triples; and Σ is a set of edge labels.*

SPARQL is the standard RDF query language, in which Basic Graph Pattern (BGP) queries are fundamental building blocks [7]. The BGP queries can be easily extended to general SPARQL queries with FILTER, UNION, and OPTIONAL.

Definition 2 (Basic graph pattern (BGP))**.** *Assume there exists an infinite set Var of variables disjoint from U and L, and every element in Var starts with the character ? conventionally, e.g., $?v \in Var$. A triple $(s, p, o) \in (Var \cup U) \times (Var \cup U) \times (Var \cup U \cup L)$ is called a* triple pattern. Basic graph pattern (BGP) *is denoted as a finite set of* triple patterns*. For a triple pattern t, let $vars(t)$ be the set of variables occurring in t.*

In order to help users query on knowledge graphs without knowing the syntax and semantics of SPARQL, we design the *basic template query*. First, we predefine a series of *placeholders*, denoted by $K = \{K_1, K_2, \ldots, K_n\}$, where $K_i \in \mathcal{P}(U \cup L)$ and K_i denotes the resources that belong to some specific domains. Then, as a placeholder, each element $k_i \in K_j$ can denote a subject, a predicates, or an object. For example, given a certain triple pattern (s, k_i, o), users can specify $k_i \in K_j$ to denote a predicate in the triple pattern. In particular, we define K_{pt} as a set of predicates for restricting the temporal attributes, such as $K_{pt} = \{$`birthYear,birthDay,...`$\}$, K_{ot} as a set of objects for representing certain temporal values, such as $K_{ot} = \{$`1990,1990-01-01,...`$\}$, K_{ps} as a set of predicates for restricting the spatial attributes, such as $K_{ps} = \{$`nationality,birthPlace,...`$\}$, and K_{ot} as a set of objects for representing certain spatial values, such as $K_{os} = \{$`United_Kingdom, London,...`$\}$. Therefore, we define the *basic template query* as follows:

Definition 3 (Basic template query)**.** *Assume that a SPARQL query, denoted as $Q_t = \{t_1, \ldots, t_n\}$, includes a set of triple patterns and additional FILTER statements. There exist several triple pattern statements, such as, $t_p = (s, k_{pt}, k_{ot})$ satisying $k_{pt} \in K_{pt} \wedge k_{ot} \in K_{ot}$, $t_f = (s, k_{pt}, o)$ satisying $k_{pt} \in K_{pt}$ and o is restricted by FILTER in a range, and $t_s = (s, k_{ps}, k_{os})$ satisying $k_{ps} \in K_{ps} \wedge k_{os} \in K_{os}$. If $(t_p \in Q_t \vee t_f \in Q_t) \wedge t_s \in Q_t$, then Q_t is a* basic template query*. The placeholders in Q_t can be replaced by proper values specified by users.*

The properties related to temporal-spatial attributes of the resources in knowledge graphs are general in most cases. For Q_t, users just need to provide the values to replace the placeholders. However, in the real world, users'requests not only focus on the temporal-spatial attributes, but also other attributes in different domains. If users would like to obtain more specific or detailed information, they can choose the *refined template query*, which is defined as follows.

Definition 4 (Refinement relation)**.** *Let S_q and S_r denote two sets of basic template queries, a binary relation from S_q to S_r, denoted as $R \subseteq S_q \times S_r$, is called a* refinement relation *if and only if $\forall(Q_t, Q_r) \in R$ the following conditions hold, for a triple pattern $t \in Q_t$: (1) $t \in Q_r$ or $t \notin Q_r \wedge \exists\, t_r \in Q_r \wedge vars(t_r) \subset vars(t)$; (2) $vars(Q_r) \subset vars(Q_t)$.*

Definition 5 (Refined template query)**.** *Given a basic template query $Q_t = \{t_1, \ldots, t_n\}$, where t_i is a triple pattern or a FILTER statement, we design a query $Q_r = \{t_1, \ldots, t_n, t_{n+1}, \ldots, t_{n+m}\}$, where t_{n+i} $(i \geq 1)$ denotes a triple pattern in general* SPARQL*. If Q_t and Q_r satisfy a refinement relation, i.e., $(Q_t, Q_r) \in R$, then Q_r is called a* refined template query *w.r.t. Q_t.*

Obviously, for a *refinement* relation R, $\forall(Q_t, Q_r) \in R$, the result set of Q_r is a subset of that of Q_t [12]. For instance, in Fig. 1, the SPARQL expressions highlighted in orange represent temporal-spatial constraints, in gray represent extra refined triple patterns. The query result is a set of triples, i.e., a series of subjects, predicates, and objects, which need to be visualized explicitly. To this end, we visualize these triples in graph format.

SELECT *	**SELECT** *
WHERE {	**WHERE**{
`?x ?z ?m .`	`?x type MusicalArtist .`
`?x ?u ?y .`	`?x associatedMusicalArtist ?y .`
`?x` k_{pt} k_{ot} `.`	`?x birthYear 1989 .`
`?x` k_{ps} k_{os} `.`	`?x birthPlace United_Kingdom .`
`?y ?v ?n .` }	`?y type Person .` }

Fig. 1. Examples of the basic and refined template queries

Definition 6 (Visualization of RDF triples)**.** *We define the visualization of a set of RDF triples as an labeled undirected graph $G = (V, E)$, called the* visualized graph*. Then, for $\forall$ a triple (s, p, o), the subject s (or the object o) denotes a vertex $v \in V$ (or $v' \in V$) labeled with s (or o), and the predicate p denotes an undirected edge between v and v'. The nodes are sorted in different colors and sizes and the graph is shown in proper layouts.*

4 Template-Based Query

In this section, we describe two template-based queries in detail. Moreover, we present how to evaluate the queries to obtain the meaningful target relationships and information of rich semantics.

4.1 Basic Template Query

There exist several predicates that are restricted to the sets K_{pt} and K_{ps} in the basic template query. The basic template query applies temporal and spatial constraints to the resources, which contributes to an important influence on the visualized graph.

Given a basic template query Q_t, users need to provide k_{ot} and k_{os} to complete the query and execute it against a SPARQL endpoint. Since the predicates in K_{pt} and K_{ps} are determined by the knowledge graph, we can evaluate the basic template query on a knowledge graph in the real world, such as DBpedia. For example, when asking the query "to search the one that belongs to `dbo:Preson` and return the person and his related attributes", two triple patterns $t_1 =$ (`?x rdf:type dbo:Person`) and $t_2 =$ (`?x ?p ?y`) can return the target answers. In basic template query, we add temporal and spatial constraints based on t_1, which is shown in template Q_t. It aims to find all the resources that have the property `rdf:type` with `dbo:Person`, `dbo:nationality` with k_{os}, and `dbo:birthYear` with the value that is larger than k_{ot}. When we increase (or reduce) the range of k_{os} or k_{ot}, the number of the results can change dramatically, which can be clearly observed in the visualized graph.

```
Template $Q_t$:
PREFIX rdf: <http://www.w3.org/1992/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX db: <http://dbpedia.org/resource/>
SELECT ?x ?y
WHERE {
   ?x rdf:type dbo:Person.
   ?x ?p ?y.
   ?x $k_{ps}$ (e.g., dbo:nationality) $k_{os}$ (e.g., db:United_Kingdom).
   ?x $k_{pt}$ (e.g., dbo:birthYear) ?birthYear .
   FILTER(?birthYear >= $k_{ot}$ (e.g., 1990)) .
}
```

In this paper, we mainly focus on the resources and their relationships involving persons, i.e., social networks. We also consider the resources in other domains to reveal the meaningful relationships in the real world. When the domain varies, the corresponding temporal and spatial attributes also change.

4.2 Refined Template Query

In the basic template query, we only add temporal and spatial constraints. However, the resources in knowledge graphs have covered various domains and the query result is too large to be analyzed directly due to the massive knowledge graphs. We propose the refined template query to specify and restrict the resources further to search more meaningful relationships. For example, we specify `?p` in t_2 = (`?x ?p ?y`) of Q_t as `dbo:parent` and add new triple patterns (`?x ?z ?m`), (`?y ?v ?n`), ..., to restrict `?x` and `?y`, where `?z`, `?m`, `?v`, `?n`, etc. are all specified by users.

```
Template $Q_r$:
SELECT ?x ?y
WHERE {
   ?x rdf:type dbo:Person.
   ?x dbo:parent ?y.
   ?x $k_{ps}$ (e.g., dbo:nationality) $k_{os}$ (e.g., db:United_Kingdom).
   ?x $k_{pt}$ (e.g., dbo:birthYear) ?birthYear .
   FILTER(?birthYear >= $k_{ot}$ (e.g., 1990)) .
   ?x ?z ?m .
   ?y ?v ?n .
   ...
}
```

Obviously, there exists a refinement relation R from Q_t to Q_r, where the result set of Q_r is a subset of that of Q_t.

5 Template-Based Visualization

In this paper, the result of the template-based query is a set of RDF triples. Therefore, we can transform these RDF triples into an undirected labeled graph.

As a statistical analysis software, R combines statistical analysis and graph visualization well. The *igraph* package in R can easily convert the result into an adjacent graph in *GraphML* format. Gephi is a very powerful software for processing graphs with many functions, such as sorting, partitioning, statistics, filtering, layout, and so on. To this end, we process the query results with R and realize visualization of RDF triples in Gephi.

5.1 General Data Transform Algorithm

Given a template-based query Q, we execute the query in R against a specified SPARQL endpoint. Once receiving the result, we transform it to an adjacency graph in Algorithm 1, which includes the general steps of visualization in R.

Algorithm 1. `visualizeInR`

```
    Input  : The template-based query $Q$.
    Output: A visualized graph.
 1  $R_e \leftarrow$ the result of $Q$ against a specified SPARQL endpoint;
 2  Sort $R_e$ as a table $T_r$;
 3  Select two columns $\overrightarrow{x}$, $\overrightarrow{y}$ in $T_r$;
 4  $m, n \leftarrow$ the number of values in $\overrightarrow{x}$, $\overrightarrow{y}$, respectively;
 5  $M_{xy} \leftarrow$ construct a matrix with $m$ rows and $n$ columns;
 6  if i-th value in $\overrightarrow{x}$ relates to j-th value in $\overrightarrow{y}$ then
 7      $M_{xy}[i][j] = 1$;
 8  else $M_{xy}[i][j] = 0$;
 9  if visualizing a direct relationship then
10      $xy \leftarrow graph.incidence(M_{xy})$;
11  else if visualizing an indirect relationship then
12      $M_{xy} \leftarrow M_{xy} * M_{xy}$;
13      $diag(M_{xy}) \leftarrow 0$;
14      $xy \leftarrow graph.incidence(M_{xy})$;
15  Attach the labels to the corresponding nodes in $xy$;
16  return xy in GraphML format;
```

The query result in R includes several columns of different entities with certain attributes. We select two columns of entities to construct an adjacency matrix (lines 2–8), which can be transformed into an adjacency graph. There are two cases: (1) when visualizing a direct relationship, we transform the adjacency matrix into an adjacency graph directly (lines 9–10); (2) when visualizing an indirect relationship, we conduct multiplication with the adjacency matrix itself to build a new matrix, then we transform the new one into an adjacency graph (lines 11–14). For clarity of the visualized graph, we attach the labels to the nodes in the graph. Finally, the output is a labeled graph in GraphML format, which can be further demonstrated in Gephi.

5.2 Visualization and Analysis

Gephi is a common visualization tool, which is mainly used for exploratory data analysis, link analysis, social network analysis, and biological network analysis. As a powerful software for processing graphs, it provides systematic analysis with ranking, partitioning, filtering, and statistics for graph analysis [3]. In this paper, the input of Gephi is an adjacency graph from R, shown in a random layout, which can be sorted and displayed in a proper layout as output to intuitively provide useful information for end users.

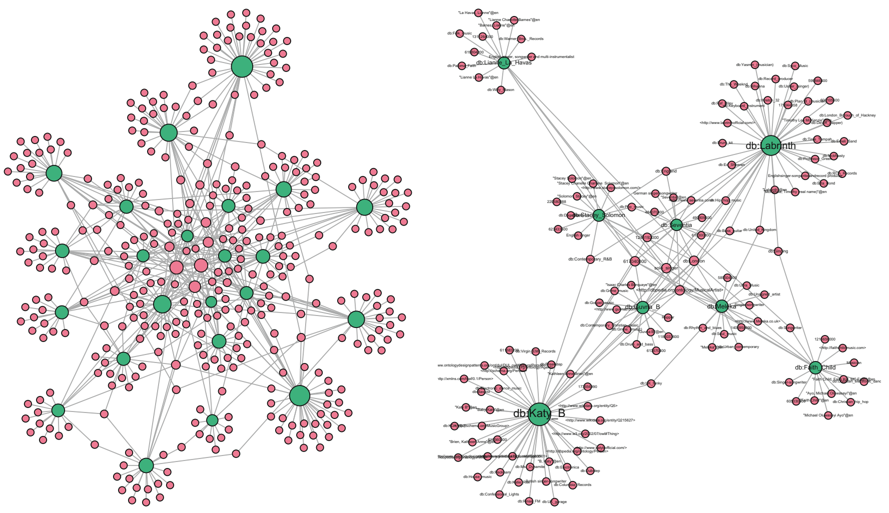
The process of visualization in Gephi can be realized in the following steps. First, we can allocate different colors to the nodes in accordance with the parameters of *betweenness centrality* or *PageRank* and sort the size of each node by its degree. Then we can choose an appropriate layout strategy to display the nodes. There are a variety of layout strategies which consider the gravitational and repulsive forces between each node. Furthermore, to obtain more refined information, we can select nodes or edges with thresholds, ranges, and other properties to filter out irrelevant information.

6 Case Study

The SPARQL queries were executed against a SPARQL endpoint provided by *Virtuoso* on a PC machine. Eight template-based queries were designed and evaluated on the knowledge graph of DBpedia, shown in four case studies. The datasets we employed are four subsets in DBpedia, called `instance_types_en`, `labels_en`, `mappingbased_literals_en`, and `mappingbased_objects_en`, respectively. We also displayed the visualized graph with sorted nodes in terms of colors and sizes. In most cases, a strategy, called *ForceAtlas2*, was chosen to layout the nodes in the graph, which aims to generate a readable shape of the graph [9].

Case Study 1. Without temporal and spatial constraints, the result includes a large number of triples which cannot be analyzed intuitively when visualized. We carried out 4 template-based queries as follows, for selecting the entities and relationships in the real world, in three granularities: (i) template query with temporal and spatial constraints, (ii) refinement by replacing variables with constants, and (iii) refinement by adding extra triple patterns.

When asking the query *"which musical artists have a genre?"*, in template query users just need to provide `MusicalArtist`, `genre`, and temporal and spatial constraints instead of a complete SPARQL query. For example, we select musical artists and corresponding resources who were born in `United_Kingdom` and between 1987 and 1997, shown as Q_1. The result of Q_1 is shown in Fig. 2(a). Although the nodes that represent `dbo:MusicalArtist` are highlighted in green



(a) The result of Q_1 **(b) The result of Q_2**

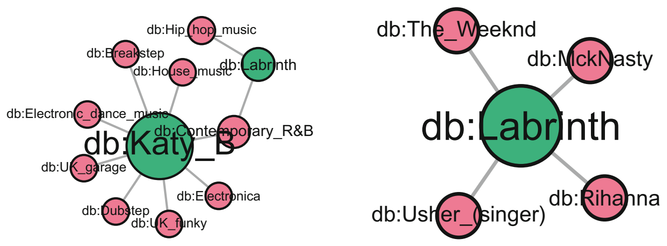
Fig. 2. The visualization of the query results of Q_1 and Q_2 (Color figure online)

and its related nodes are marked in red, the whole results are still difficult to be analyzed intuitively.

```
Q1: SELECT ?x ?y
WHERE {
?x rdf:type dbo:MusicalArtist .    ?x dbo:genre ?z .
?x ?p ?y .       ?x dbo:birthPlace db:United_Kingdom .
?x dbo:birthYear ?birthYear.    FILTER(?birthYear >= 1987) .
FILTER(?birthYear <= 1997) .
}
```

Furthermore, we can narrow the range of spatial and temporal values, such as `London` and 1989, shown as Q_2. As shown in Fig. 2(b), the size of the result decreases in comparison with the result of Q_1. We can observe that the node labeled with `db:Katy_B` and `db:Labrinth` have the larger outdegree than the other nodes. However, temporal and spatial constraints are inadequate, the above results include a certain number of resources that users are not interested in.

```
Q2: SELECT ?x ?y
WHERE {
?x rdf:type dbo:MusicalArtist .    ?x dbo:genre ?z .
?x ?p ?y .             ?x dbo:birthPlace db:London .
?x dbo:birthYear 1989.
}
```



(a) The result of Q_3 **(b) The result of Q_4**

Fig. 3. The visualization of the query results of Q_3 and Q_4

To obtain more specific information, more constraints need to be added based on the basic template query. For example, if users are interested in the musical artists who can play `Bass guitar`, or the people who are associated with the musical artists, then variables can be replaced with constants or extra new triple patterns can be added. Suppose Q_3 be a query that satisfies the refinement relation $R(Q_2, Q_3)$. Based on Q_2, Q_3 is formed by replacing the triple pattern `(?x ?p ?y)` with `(?x dbo:instrument db:Bass_guitar)`, whose result is shown in Fig. 3(a). As we can see, the size of the result of Q_3 decreases significantly compared with that of Q_2. We use Q_4 to select the persons who have the relationship called `associatedMusicalArtist` with musical artists, shown as follows.

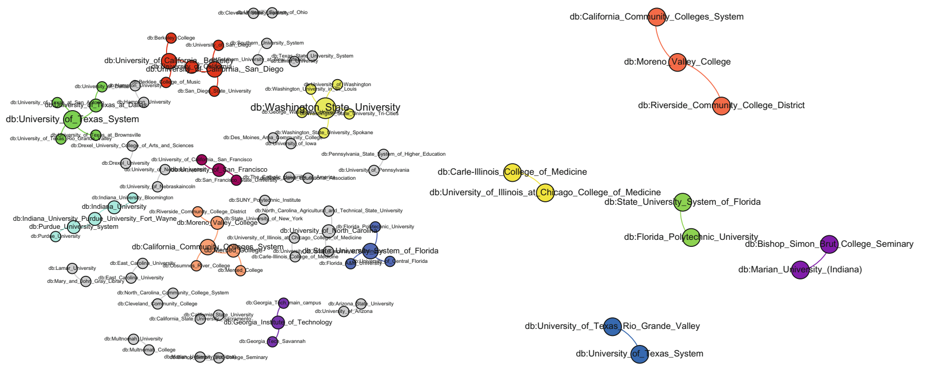
```
Q4: SELECT ?x ?y
WHERE {
?x rdf:type dbo:MusicalArtist .    ?x dbo:genre ?z .
?x dbo:birthPlace db:London .       ?x dbo:birthYear 1989.
?x dbo:associatedBand ?y .          ?y rdf:type dbo:Person .
}
```

The query Q_4 also satisfies $R(Q_2, Q_4)$, whose result is shown in Fig. 3(b). It can be observed that the answers to Q_3 and Q_4 are both subsets of the answers to Q_2.

Case Study 2. When querying the universities that have direct relationships, we add corresponding temporal and spatial constraints, i.e., `dbo:foundingDate` and `dbo:country`. We design Q_5 and vary the value of k_{ot} in temporal dimension, i.e., 1800-01-01 and 2000-01-01 to analyze the query results, as shwon in Fig. 4.

```
Q5: SELECT ?x ?y
WHERE {
?x rdf:type dbo:University .    ?y rdf:type dbo:University .
?x ?p ?y .                      ?x dbo:country db:United_States .
?x dbo:foundingDate ?date .
FILTER(?date >= 1800-01-01 (or 2000-01-01)) .
}
```

Obviously, the number of nodes in the result of Q_5 declines as the range of temporal constraint being narrowed, as shown in Fig. 4(a) and (b). The main relationships between two universities are that one has `dbo:affiliation` with the another. We allocate the color of each node using the *community detection* algorithm and sort the size of each node with the value of the degree. Thus, different communities consisting of several universities are highlighted in different colors. In particular, the less important nodes are marked in gray. It can be observed that all the universities are divided into several groups, which can be analyzed further.


db:University_of_California,_Berkeley
db:University_of_California,_San_Diego
db:Washington_State_University
db:University_of_Texas_System
db:University_of_California,_San_Francisco
db:Indiana_University_Purdue_University_Fort_Wayne
db:Purdue_University_system
db:Moreno_Valley_College
db:California_Community_Colleges_System
db:University_of_North_Carolina
db:State_University_System_of_Florida
db:Georgia_Institute_of_Technology
db:California_Community_Colleges_System
db:Moreno_Valley_College
db:Riverside_Community_College_District
db:Carle-Illinois_College_of_Medicine
db:University_of_Illinois_at_Chicago_College_of_Medicine
db:State_University_System_of_Florida
db:Florida_Polytechnic_University
db:Bishop_Simon_Bruté_College_Seminary
db:Marian_University_(Indiana)
db:University_of_Texas_Rio_Grande_Valley
db:University_of_Texas_System

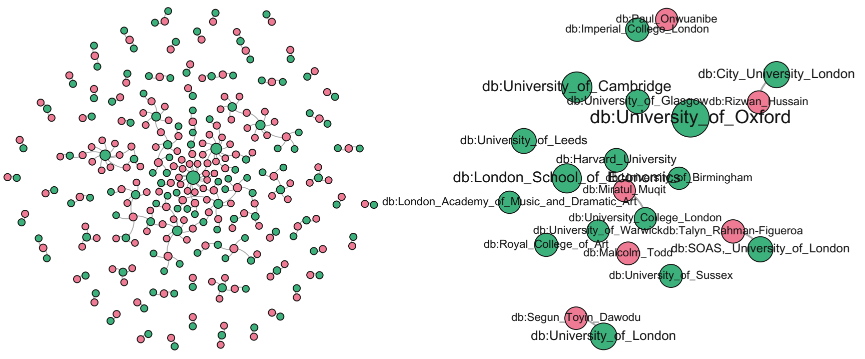
(a) $\geq$ **1800-01-01** **(b)** $\geq$ **2000-01-01**

Fig. 4. The visualization of the query result of Q_5 (Color figure online)

Case Study 3. In Q_5, it searches the universities that have direct relationships, where each node in Fig. 4 represents a university. Now we look for a person and his/her related university. Temporal-spatial constraints are added to the person rather than the university, as shown in Q_6.

```
Q6: SELECT ?x ?y
WHERE{
?x ?p ?y .    ?x dbo:nationality db:United_Kingdom .
?x rdf:type dbo:Person .    ?y rdf:type dbo:University .
?x dbo:birthYear ?birthYear .    FILTER(?birthYear >= 1900) .
}
```

The result of Q_6 contains the persons and the related universities, including 296 nodes and 236 edges in total, among which 169 persons marked in red and 127 universities in green in Fig. 5(a). We allocate color of each node by its kind; rank the size of each node by *PageRank* algorithm in Gephi, where the rank of a node is higher, the size is larger. Based on the result of Q_6, we just display the nodes whose rank are greater than or equal to 0.59, then we obtain several



(a) Persons and universities **(b) The nodes with rank ≥ 0.59**

Fig. 5. The visualization of the query result of Q_6 (Color figure online)

significant nodes as shown in Fig. 5(b). Further, the modularity class is applied to allocate the color of each node in order to build 9 main communities consisting of the persons and the related universities in Fig. 6.

In particular, we take advantage of the above direct relationships between the persons and the universities to mine the indirect connections, i.e., the schoolfellow relationship. After using matrix multiplication operation in Algorithm 1 to process the result of Q_6, we can obtain the social relationship between the persons who are studying or working in the same university, as visualized in Fig. 7, which is a complex social network and different from the other visualized graphs. If a person has complex relationships with other persons, then the color of the node that labeled with this person is closer to red. We use the *K-core* algorithm to refine the social network further and employ *between_centrality* to filter out the nodes, as shown in Fig. 8(a) and (b).

Case Study 4. In this subsection, we focus on the relationships that belong to some specific domains. For example, based on Q_1, we change the value of object correspond with to `rdf:type` in the first triple pattern, which can limit the subject to a certain class of people, such as swimmer, writer, soccerplayer, or tennisplayer. Then we design Q_7 to specify the relationship called `dbo:influencedBy`, which is a unique attribute in `dbo:Writer`. Similarly, we desgin Q_8 to search the relationship called `dbo:soccerPlayer` between a soccerplayer and a team. For obtaining specific results, we search the soccerplayers and their teams, who are born after 1800 s and have British citizenship.

The result of Q_7 includes 29 nodes and 18 edges in Fig. 9(a), which means that there are 18 relationships satisfy the conditions. We sort color of each node by the modularity class and rank the size of each node by PageRank. In Fig. 9(a),

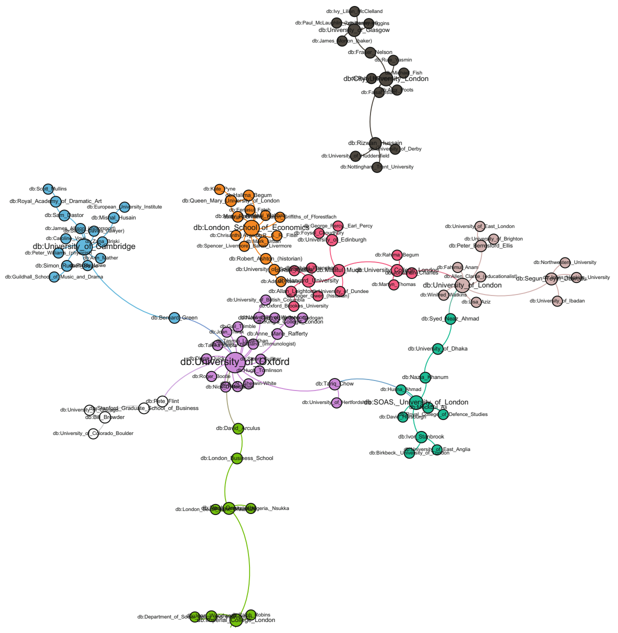


Fig. 6. The visualization of the main communities in Q_6 (Color figure online)

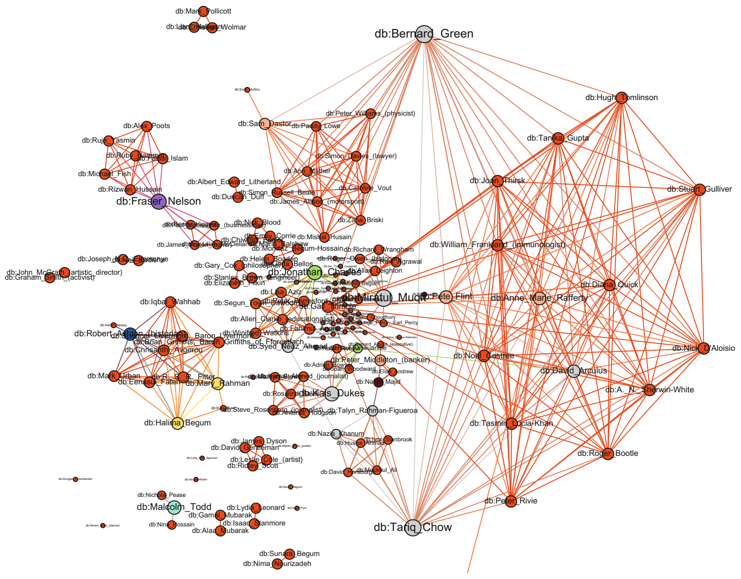


Fig. 7. The visualization of the schoolfellow relationship based on Q_6 (Color figure online)

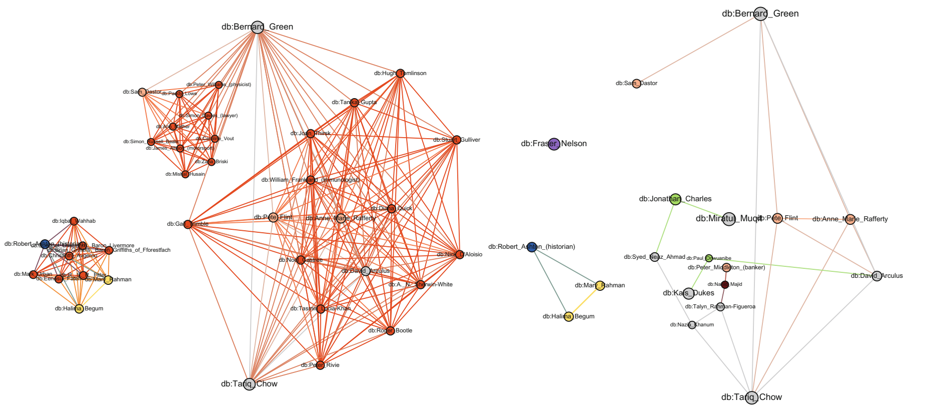

(a) K-core (k=9) (b) $1.08 \leq$ between_centrality ≤ 1080

Fig. 8. The visualization of the schoolfellow relationship with filtering (Color figure online)

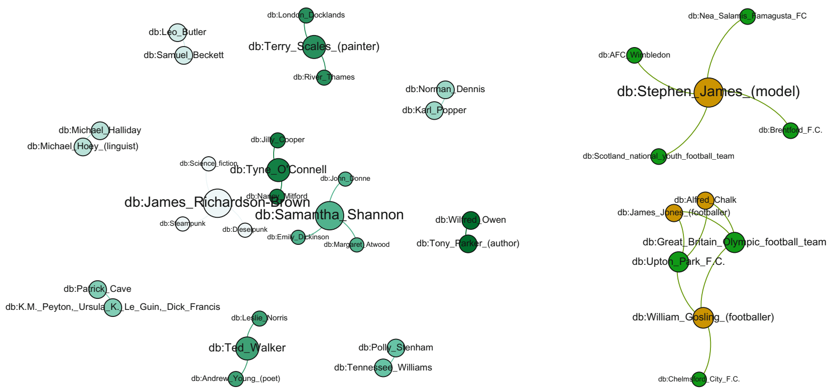

(a) The influencedBy relationship (b) The soccerPlayer relationship

Fig. 9. The relationships of some specific domains in Q_7 and Q_8 (Color figure online)

the size of a node is proportional to the rank of the node. The relationship `soccerPlayer` in Q_8 involves soccer players and teams, which are marked in yellow and green, respectively, as shown in Fig. 9(b). Since the above two relationships belong to the specific domains, the number of result is relatively fewer and it can be easily visualization.

7 Conclusion

We present template-based queries on knowledge graphs, which is a query method based on temporal, spatial, and domain-specific constraints. We mainly propose two template queries, i.e., the basic and refined template queries, to extract valuable information on knowledge graphs, whose results are visualized in undirected labeled graph. Our experimental results are well displayed in graph format for data analysis. With different constraints, knowledge graphs are visualized in different granularity. Our future work includes the visualization of more knowledge graphs and implementation of a more user-friendly interface.

Acknowledgments. This work is supported by the National Natural Science Foundation of China (61572353) and the Natural Science Foundation of Tianjin (17JCYBJC15400).

References

1. Araujo, S., Houben, G.-J., Schwabe, D., Hidders, J.: Fusion – visually exploring and eliciting relationships in linked data. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010. LNCS, vol. 6496, pp. 1–15. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17746-0_1
2. Auer, S., Doehring, R., Dietzold, S.: LESS - template-based syndication and presentation of linked data. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6089, pp. 211–224. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13489-0_15
3. Bastian, M., Heymann, S., Jacomy, M., et al.: Gephi: an open source software for exploring and manipulating networks. In: ICWSM, vol. 8, pp. 361–362 (2009)
4. Deligiannidis, L., Kochut, K.J., Sheth, A.P.: RDF data exploration and visualization. In: Proceedings of the ACM First Workshop on CyberInfrastructure: Information Management in eScience, pp. 39–46. ACM (2007)
5. Elbassuoni, S., Ramanath, M., Schenkel, R., Sydow, M., Weikum, G.: Language-model-based ranking for queries on RDF-graphs. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management, pp. 977–986. ACM (2009)
6. Fernández, J.D., Martínez-Prieto, M.A., Gutierrez, C.: Compact representation of large RDF data sets for publishing and exchange. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010. LNCS, vol. 6496, pp. 193–208. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17746-0_13
7. Harris, S., Seaborne, A., Prudhommeaux, E.: SPARQL 1.1 query language. W3C Recommendation **21**(10) (2013)
8. Heim, P., Hellmann, S., Lehmann, J., Lohmann, S., Stegemann, T.: RelFinder: revealing relationships in RDF knowledge bases. In: Chua, T.-S., Kompatsiaris, Y., Mérialdo, B., Haas, W., Thallinger, G., Bailer, W. (eds.) SAMT 2009. LNCS, vol. 5887, pp. 182–187. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10543-2_21

9. Jacomy, M., Venturini, T., Heymann, S., Bastian, M.: Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. PloS One **9**(6), e98679 (2014)
10. Kapler, T., Wright, W.: Geotime information visualization. Inf. Visual. **4**(2), 136–146 (2005)
11. Kasneci, G., Suchanek, F.M., Ifrim, G., Ramanath, M., Weikum, G.: NAGA: Searching and ranking knowledge. In: 2008 IEEE 24th International Conference on Data Engineering, ICDE 2008, pp. 953–962. IEEE (2008)
12. Kolaitis, P.G., Vardi, M.Y.: Conjunctive-query containment and constraint satisfaction. J. Comput. Syst. Sci. **61**(2), 302–332 (2000)
13. Lohmann, S., Heim, P., Stegemann, T., Ziegler, J.: The relfinder user interface: interactive exploration of relationships between objects of interest. In: Proceedings of the 15th International Conference on Intelligent User Interfaces, pp. 421–422. ACM (2010)
14. Shekarpour, S., Auer, S., Ngonga Ngomo, A.C., Gerber, D., Hellmann, S., Stadler, C.: Generating SPARQL queries using templates. Web Intell. Agent Syst. Int. J. **11**(3), 283–295 (2013)
15. Yahya, M., Barbosa, D., Berberich, K., Wang, Q., Weikum, G.: Relationship queries on extended knowledge graphs. In: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, pp. 605–614. ACM (2016)