A CONSTRUCT-based Query for Weighted RDF Graph Analytics

Xin Song, Shizhan Chen, Xiaowang Zhang*, and Zhiyong Feng

College of Intelligence and Computing, Tianjin University, Tianjin, China
Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin, China
* Corresponding Author: xiaowangzhang@tju.edu.cn

Abstract. In this paper, based on SPARQL CONSTRUCT query, we present a new query form called SPARQL Analyze Query, which integrates graph analytics with graph querying. SPARQL Analyze Query extends SPARQL by performing graph algorithms via WITH and PRODUCE operators, where WITH is used for invoking a graph algorithm and PRODUCE for designating results of variables. SPARQL Analyze Query supports two classes of graph algorithms over the weighted RDF graph w.r.t. nodes and shortest paths. Besides, we illustrate the feasibility of the SPARQL Analyze Query in expressing PageRank.

1 Introduction

In graph databases, graph analytics and graph query are often discussed in two separate fields, but we often perform graph analytics and graph query at the same time on one task. Although there are some graph analytical engines that support graph analytics by means of graph query languages, they cannot combine graph analytics with graph querying well. One of them is Neo4j and its query language Cypher [1] which lacks the formalization of graph analytics. In terms of SPARQL, the standard graph query language of RDF, there are some works on extending SPARQL with graph analytics [2–4], but they also lack the formalization and cannot perform graph analytics in a compact way.

As graph analytics should be performed over a graph, we build SPARQL Analyze Query on SPARQL CONSTRUCT query, which is one of the four query forms of SPARQL and also the only form whose result is a graph. We follow the formalization of the CONSTRUCT query introduced in [5] and briefly recall some notions. Given a SPARQL graph pattern P, a CONSTRUCT query $Q_c =$ CONSTRUCT H WHERE P. H is a set of triple patterns from $(I \cup L \cup V) \times (I \cup V) \times (I \cup L \cup V)$, where I, L, V are the sets of $IRIs$, $literals$ and $variables$. Besides, we do not consider blank nodes in our work.

An RDF graph $G = (N, E)$, where N and E denote the sets of nodes and edges of G respectively. We follow the definitions of weighted RDF graph $S =$

(G, ω) where ω denotes the weight function, and shortest path π introduced in [6]. Especially, we only consider the weight function on edges $\omega : E \to \mathbb{R}_{\geq 0}$.

Inspired by Neo4j, SPARQL Analyze Query supports two classes of graph algorithms w.r.t. nodes and shortest paths, which are *PageRank* (PR), *Betweenness Centrality* (BC), *Closeness Centrality* (CC), *Shortest Path* (SP), *Single Source Shortest Path* (SSSP) and *All Pair Shortest Path* (ALSP).

2 Motivating Example

We provide an example of expressing PageRank over a social network to directly illustrate how SPARQL Analyze Query performs. For clarity, the following example is based on our semantics introduced in Section 4 while the SPARQL CONSTRUCT query introduced in [5] cannot output a weighted RDF graph.

The following CONSTRUCT query Q_c generates a social network which consists of all people that $?p1$ knows. The generated weighted RDF graph is shown in Figure 2 where A, B, ..., H are nicknames of each person respectively.

```
CONSTRUCT { ?p1  knows  ?p2.}
WHERE {
  ?m knows ?n.   ?m nickName ?p1.
  ?n nickName ?p2.
}
```

Fig. 1. An example of CONSTRUCT query

Figure 2 shows a weighted RDF graph where each *knows* edge is acquired from WHERE clause along with its original weight. In this example, we assume the weights denote the interpersonal influence.

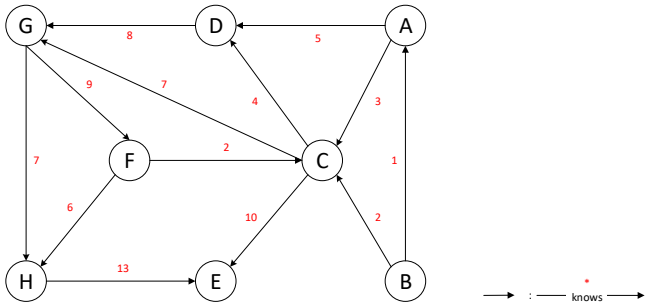

Fig. 2. A social network generated by Q_c in Figure 1

Based on Figure 1, we provide a SPARQL Analyze Query with PageRank algorithm in Figure 3 that reveals the person whose nickname is E has the greatest interpersonal influence in the social network.

```
Qc WITH PR({?p1,?p2},{knows},10,0.85)
PRODUCE ?node ?pr
ORDER BY DESC (?pr)
LIMIT 3
```

?node	?pr
E	0.776535
G	0.604325
H	0.579038

Fig. 3. A SPARQL Analyze Query with PageRank algorithm

3 Syntax of SPARQL Analyze Query

Given a CONSTRUCT query Q_c, a SPARQL Analyze query Q_A is defined as follows:

$$Q_c \text{ WITH } \mathcal{C} \text{ PRODUCE } v\_sq \text{ [ORDER BY] [LIMIT]}$$

where

- Q_c is a SPARQL CONSTRUCT query;
- $\mathcal{C}$ is an analyzing clause of one of the following six forms:
  Given a weighted RDF graph S, let N_H and E_H be sets of nodes and edges occurring in H respectively, where nodes and edges can be variables or constants.
  - PR : PR (N_H, E_H, n, dF), where $n \in \mathbb{N}^+$ is the number of iterations in PageRank and $dF \in (0, 1)$ is the damping factor;
  - BC : BC (N_H, E_H);
  - CC : CC (N_H, E_H);
  - SP : SP (N_H, E_H, u, v) where $u, v \in I \cup L$;
  - SSSP : SSSP (N_H, E_H, u) where $u \in I \cup L$;
  - ALSP : ALSP (N_H, E_H);
- PRODUCE v_sq designates the output of a SPARQL Analyze Query of variables, where v_sq is defined as a sequence of two variables (v_1, v_2), $v_1, v_2 \in V$, and the parentheses () denote an ordered sequence of variables;
- [ORDER BY] and [LIMIT] are standard SPARQL fragments. They are optional to use for controlling the order and the number of solutions.

4 Semantics of SPARQL Analyze Query

To be compatible with weighted RDF graphs, we extend the semantics introduced in [5] by means of a weight $w \in \mathbb{R}_{\geq 0}$. Given a weighted RDF graph S, the evaluation of a SPARQL graph pattern P over S is denoted by $[\![P]\!]_S$.

We introduce some new terminology for SPARQL Analyze Query. Let $V_N \in V$ and $V_E \in V$ be the sets of disjoint node and edge variables respectively. Given a node u, a shortest path π, and let A be the set of real numbers. We have two mappings $\lambda : V_N \to N$, $\tau : V_E \to E$ and two functions $f : u \to a$ and $g : \pi \to a$, where $f \in \mathcal{F}$, $g \in \mathcal{G}$ and $a \in A$. $\mathcal{F} = \{PR, BC, CC\}$ and $\mathcal{G} = \{SP, SSSP, ALSP\}$ are two function sets where each function represents the computation of a graph algorithm. For example, the betweenness centrality of u is obtained by $BC(u)$.

For a SPARQL Analyze Query Q_A, we say that $Q = Q_c$ WITH $\mathcal{C}$ is the analytical part of Q_A. Given a graph pattern P, a weighted RDF graph S, a CONSTRUCT query Q_c and an analyzing clause $\mathcal{C}$. The semantics of an analytical part Q of Q_A returns a set of pairs ϕ, denoted by $[\![Q]\!]_S$, defined as follows:

$$[\![Q]\!]_S = \begin{cases} \{(u, f(u)) \mid \lambda \in [\![P]\!]_S, f(u) \in A\} & \mathcal{C} = \text{PR, BC, CC} \\ \{(\pi, g(\pi)) \mid \lambda, \tau \in [\![P]\!]_S, g(\pi) \in A\} & \mathcal{C} = \text{SP, SSSP, ALSP} \end{cases}$$

On the above formalization and given an ordered sequence of variables v_sq. The semantics of a SPARQL Analyze Query Q_A, denoted by $[\![Q_A]\!]_S$, is defined as follows:

$$[\![Q_A]\!]_S = [\![Q \text{ PRODUCE } v\_sq]\!]_S = \{\phi \mid_{(v_1,v_2)} \mid \phi \in [\![Q]\!]_S\}$$

where $\phi \mid_{v_1,v_2}$ is the restriction of ϕ to (V, V) correspondingly of its domain.

5 Conclusion

In this paper, we present SPARQL Analyze Query which integrates graph querying with graph analytics. Our approach provides a logical foundation for integrating querying with analytics, which makes it possible to study graph querying and analytics in a unified way theoretically. In future work, we will study some fundamental problems of SPARQL Analyze Query such as expressivity and complexity as well as its generalization of more graph algorithms.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (2017YFC0908401) and the National Natural Science Foundation of China (61972455, 61672377). Xiaowang Zhang is supported by the Peiyang Young Scholars in Tianjin University (2019XRX-0032).

References

1. Francis, N., Green, A., Guagliardo, P., Libkin, L., et al.: Cypher: An evolving query language for property graphs. In: *Proc. of SIGMOD*, pp. 1433–1445 (2018).
2. Erétéo, G., Gandon, F., Buffa, M.: Semantic social network analysis. In: *Proc. of WebSci*, pp. 1–5 (2009).
3. Techentin, R.W., Gilbert, B.K., Lugowski, A., Deweese, K., et al.: Implementing iterative algorithms with SPARQL. In: *Proc. of ICDT*, pp. 216–223 (2014).
4. Mizell, D., Maschhoff K.J., Reinhardt, S.P.: Extending SPARQL with graph functions. In: *Proc. of BigData*, pp. 46–53 (2014).
5. Kostylev, E.V., Reutter, J.L., Ugarte, M.: CONSTRUCT queries in SPARQL. In: *Proc. of ICDT*, pp. 212–229 (2015).
6. Tartari, G., Hogan, A.: WiSP: Weighted shortest paths for RDF graphs. In: *Proc. of ISWC*, pp. 37–52 (2018).