Recursive Path Query for SPARQL path generation

# Overview

## Description

This example will use the Friend-of-a-Friend (FOAF) model as a simplified data model for demonstration of the pattern for generating paths using RDF data.

## Limitations in SPARQL

SPARQL does not provide a solution within the standard language. To implement the “path query” in the RDF framework using SPARQL, a programmatic solution is needed.

In the Labelled Property Graph (LPG) framework, a path query is provided within the standard query library. This means that there is a solution within the query languages (OpenCypher or Gremlin) to provide this functionality.

There is a “Property Path” functionality in SPARQL, which can be useful in the situation where the length and pattern of the path is known or predefined. However, in our use case of generating a pathway between two nodes, this may be a varying length. There will be a limitation in how many traversals are reasonable considering the data, and the programmatic solution will apply the limitation in the recursive logic. In SPARQL, the “dynamic path query” is used when the path requires an unknown number of traversals or path length. The return of this query is all of the nodes that

SPARQL 1.1 has property path which includes the \* operator for any number of.

It does not tell you what the path is nor the length of the shortest path - only whether there is such a path.

# Basic Example

## Example Data

The following query is used to insert the sample data for the FOAF (Friend-of-a-friend) model.

## Query Example for Dynamic Path

%%sparql

PREFIX : <http://example.org/>

SELECT ?intermediateNode

WHERE {

:Alice (:friendOf|:parentOf)\* ?intermediateNode .

?intermediateNode (:friendOf|:parentOf)\* :Dave .

}

This code snippet demonstrates an example of the query pattern. The code snippet starts with the “magic command” to invoke the SPARQL library functionality within the notebook. In the query, we are returning all “intermediate nodes” that satisfy the pattern