

Interactive / complex / 10

IC 1
IC 2
IC 3
IC 4
IC 5
IC 6
IC 7
IC 8
IC 9
IC 10
IC 11
IC 12
IC 13
IC 14

query	Interactive / complex / 10			
title	Friend recommendation			
pattern	<p>The diagram illustrates the query pattern. It starts with a person: Person node (orange) with a variable <code>id = \$personId</code>. This node is connected via a <code>knows*2..2</code> relationship to a foaf: Person node (orange). The foaf: Person node has properties: <code>id</code>, <code>firstName</code>, <code>lastName</code>, and <code>gender</code>. It is connected via an <code>isLocatedIn</code> relationship to a city: City node (teal) with property <code>name</code>. Below this, two sub-patterns are shown: common: A person: Person node has a <code>hasInterest</code> relationship to a Tag node (pink). A foaf: Person node has a <code>hasCreator</code> relationship to a Post node (red), which in turn has a <code>hasTag</code> relationship to the same Tag node. A <code>count</code> operation is indicated. uncommon: Similar to the common pattern, but the <code>hasInterest</code> relationship is negated (indicated by a red arrow and <code>«neg»</code>), meaning the start person is not interested in the Tag.</p>			
desc.	<p>Given a start Person with id <code>personId</code>, find that Person's friends of friends (<code>foaf</code>) – excluding the start Person and his/her immediate friends –, who were born on or after the 21st of a given month (in any year) and before the 22nd of the following month. Calculate the similarity between each friend and the start person, where <code>commonInterestScore</code> is defined as follows:</p> <ul style="list-style-type: none"> • <code>common</code> = number of Posts created by friend, such that the Post has a Tag that the start person is interested in • <code>uncommon</code> = number of Posts created by friend, such that the Post has no Tag that the start person is interested in • <code>commonInterestScore</code> = <code>common</code> - <code>uncommon</code> 			
params	1	<code>personId</code>	ID	
	2	<code>month</code>	32-bit Integer	Between 1 and 12. Implementations may also pass the next month as an additional <code>nextMonth</code> parameter
result	1	<code>foaf.id</code>	ID	R
	2	<code>foaf.firstName</code>	String	R
	3	<code>foaf.lastName</code>	String	R
	4	<code>commonInterestScore</code>	32-bit Integer	A
	5	<code>foaf.gender</code>	String	R
	6	<code>city.name</code>	String	R
sort	1	<code>commonInterestScore</code>	↓	
	2	<code>foaf.id</code>	↑	
limit	10			
CPs	2.3, 3.3, 4.1, 4.2, 5.1, 5.2, 6.1, 7.1, 8.6			
relevance	<p>This query looks for paths of length two, starting from a Person and ending at the friends of their friends. It does widely scattered graph traversal, and one expects no locality of in friends of friends, as these have been acquired over a long time and have widely scattered identifiers. The join order is simple but one must see that the anti-join for “not in my friends” is better with hash. Also the last pattern in the scalar sub-queries joining or anti-joining the Tags of the candidate's Posts to interests of self should be by hash.</p>			