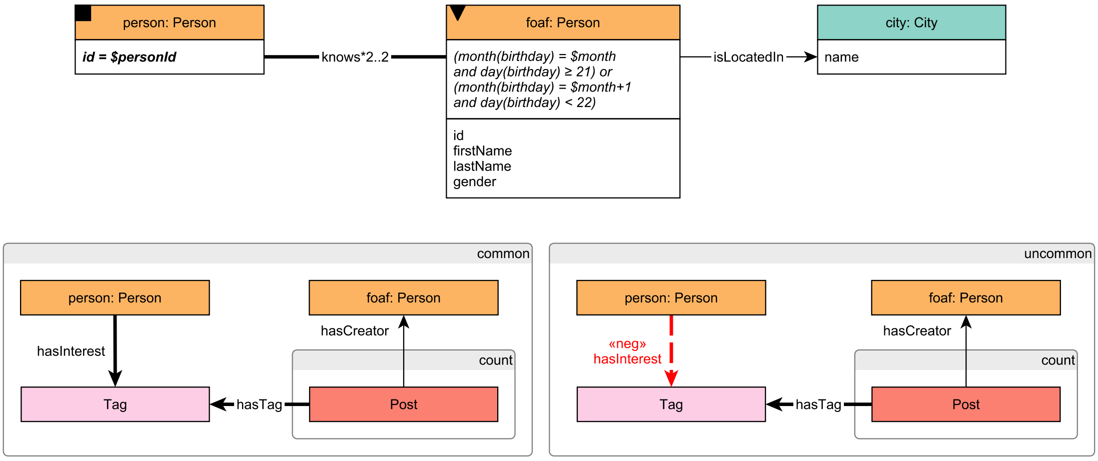
Interactive / complex / 10

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 14v1
IC 14v2

<table>
<tr><td>query</td><td>Interactive / complex / 10</td></tr>
<tr><td>title</td><td>Friend recommendation</td></tr>
<tr><td>pattern</td><td>
</td></tr>
<tr><td>description</td><td>Given a start Person with ID $personId, find that Person's friends of friends (foaf) – excluding the start Person and his/her immediate friends –, who were born on or after the 21st of a given $month (in any year) and before the 22nd of the following month. Calculate the similarity between each friend and the start person, where commonInterestScore is defined as follows:
• common = number of Posts created by friend, such that the Post has a Tag that the start person is interested in
• uncommon = number of Posts created by friend, such that the Post has no Tag that the start person is interested in
• commonInterestScore = common - uncommon</td></tr>
<tr><td>params</td><td><table>
<tr><td>1</td><td>$personId</td><td>ID</td><td></td></tr>
<tr><td>2</td><td>$month</td><td>32-bit Integer</td><td>Between 1 and 12. Implementations may also pass the next month as an additional $nextMonth parameter</td></tr>
</table></td></tr>
<tr><td>result</td><td><table>
<tr><td>1</td><td>foaf.id</td><td>ID</td><td>R</td><td></td></tr>
<tr><td>2</td><td>foaf.firstName</td><td>String</td><td>R</td><td></td></tr>
<tr><td>3</td><td>foaf.lastName</td><td>String</td><td>R</td><td></td></tr>
<tr><td>4</td><td>commonInterestScore</td><td>32-bit Integer</td><td>A</td><td></td></tr>
<tr><td>5</td><td>foaf.gender</td><td>String</td><td>R</td><td></td></tr>
<tr><td>6</td><td>city.name</td><td>String</td><td>R</td><td></td></tr>
</table></td></tr>
<tr><td>sort</td><td><table>
<tr><td>1</td><td>commonInterestScore</td><td>↓</td><td></td></tr>
<tr><td>2</td><td>foaf.id</td><td>↑</td><td></td></tr>
</table></td></tr>
<tr><td>limit</td><td>10</td></tr>
<tr><td>CPs</td><td>2.3, 3.3, 4.1, 4.2, 5.1, 5.2, 6.1, 7.1, 8.6</td></tr>
<tr><td>relevance</td><td>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.</td></tr>
</table>