

BI / read / 20

BI 1	query	BI / read / 20			
BI 2	title	Recruitment			
BI 3	pattern	 <p>The diagram illustrates the query pattern. It shows a <code>company: Company</code> node with a property <code>name = \$company</code>. A <code>person1: Person</code> node is connected to the company via a <code>workAt</code> edge. A <code>person2: Person</code> node is connected to <code>person1</code> via a <code>shortest path on knows.weight</code> edge, with a filter <code>id = \$person2Id</code>. A <code>University</code> node is connected to both <code>person1</code> and <code>person2</code> via <code>studyAt</code> edges. The weight of the <code>knows</code> edge between <code>personA</code> and <code>personB</code> is defined as <code>abs(saA.classYear - saB.classYear) + 1</code>.</p>			
BI 10	desc.	<p>Given a Company <code>company</code> and a Person <code>person2</code> (who is known to be working at another Company), find a different Person (<code>person1</code>) who works in <code>company</code> and is reachable by from <code>person2</code> through people who have studied together. On this path, we only consider edges between Persons who know each other and attended the same university and set the weight of the edge to the absolute difference between the year of enrolment plus 1 (<code>studyAt.classYear + 1</code>). We return the 20 shortest paths.</p> <p>If there are multiple Person <code>person1</code> nodes with the same shortest path, return all of them.</p>			
BI 17	params	1	company	Long String	Companies with a similar number of employees (former or current) are selected
BI 18		2	person2Id	ID	
BI 19	result	1	person1.id	ID	R
BI 20		2	totalWeight	64-bit Integer	C
	sort	1	totalWeight	↑	
		2	person1.id	↑	
	limit	20			
	CPs	3.3, 7.6, 7.7, 8.4, 8.6			
	relevance	Implementations can either pre-compute edge weights or compute them on-the-fly. To find a weighted shortest path efficiently, implementations can use e.g. a bidirectional Dijkstra algorithm.			