

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 company: Company node with a variable name = \$company. A person1: Person node is connected to the company via a workAt relationship. A person2: Person node is connected to person1 via a shortest path on knows.weight relationship, with a constraint id = \$person2Id. A University node is connected to both person1 and person2 via studyAt relationships. The weight of the knows relationship is defined as knows.weight: abs(saA.classYear - saB.classYear) + 1.</p>			
BI 10	desc.	<p>Given a Company <i>company</i> and a Person <i>person2</i> (who is known to be working at another Company), find a different Person (<i>person1</i>) who works in <i>company</i> and is reachable by from <i>person2</i> 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 (<i>studyAt.classYear</i> + 1). We return the 20 shortest paths.</p> <p>If there are multiple Person <i>person1</i> 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.			