

BI / read / 19

BI 1	query	BI / read / 19			
BI 2	title	Interaction path between cities			
BI 3	pattern	<p>Find the shortest paths between all pairs of Persons in city1 and city2</p> <p>city1: City id = \$city1id</p> <p>city2: City id = \$city2id</p> <p>person1: Person isLocatedIn</p> <p>person2: Person isLocatedIn</p> <p>compute weighted shortest paths on knows.weight</p> <p>The weight of a knows edge is based on the number of interactions between its Persons: knows.weight = 1 / (count(i1)+count(i2))</p> <p>Case i1: Reply from personA to Person B's Message</p> <p>personA: Person knows personB: Person</p> <p>hasCreator c: Comment replyOf m: Message</p> <p>Case i2: Reply from personB to personA's Message</p> <p>personA: Person knows personB: Person</p> <p>hasCreator m: Message replyOf c: Comment</p>			
BI 13	desc.	<p>Given two Cities city1, city2, find Persons person1, person2 living in these Cities (respectively) with the shortest <i>interaction path</i> between them. If there are multiple pairs of people with shortest paths having the same total weight, return all of them.</p> <p>The shortest path is computed using a weight between two Persons defined as the reciprocal of the number of interactions (direct reply Comments to a Message by the other Person). Therefore, more interactions imply a smaller weight.</p> <p><i>Note:</i> Interactions are counted both ways, i.e. if Alice writes 2 reply Comments to Bob's Messages and Bob writes 3 reply Comments to Alice's Messages, their total number of interactions is 5.</p>			
BI 14	params	1	city1Id	ID	Small Cities within the same Country are selected
BI 15		2	city2Id	ID	
BI 16	result	1	person1.id	ID	R
BI 17		2	person2.id	ID	R
BI 18		3	totalWeight	64-bit Float	C
BI 19	sort	1	totalWeight	↑	
BI 20		2	person1.id	↑	
		3	person2.id	↑	
	limit	20			
	CPs	3.3, 7.6, 7.7, 8.4, 8.6			
	relevance	Finding shortest paths between pairs of Persons in Cities can be implemented in theory with an <i>all-pairs shortest paths</i> algorithm. However, this needs to be executed on the whole Person-knows-Person graph (with edge weights derived from the number of interactions) so it is expected to be prohibitively expensive. A better approach is using multiple <i>single-source shortest path algorithms</i> (e.g. from the City with fewer inhabitants).			