

Interactive / complex / 6

IC 1	query	Interactive / complex / 6			
IC 2	title	Tag co-occurrence			
IC 3	pattern	 <pre> graph TD person[person: Person] -- knows*1..2 --> otherPerson[otherPerson: Person] tag[tag: Tag] -- hasTag --> Post[Post] otherTag[otherTag: Tag] -- hasTag --> Post Post -- hasCreator --> otherPerson subgraph count Post end </pre> <p>The diagram illustrates the query pattern. It features three main entity types: Person (orange), Tag (pink), and Post (red). A Person entity (labeled 'person: Person') is connected to another Person entity (labeled 'otherPerson: Person') via a 'knows' relationship with a cardinality of 1..2. A Tag entity (labeled 'tag: Tag') is connected to a Post entity via a 'hasTag' relationship. Another Tag entity (labeled 'otherTag: Tag') is also connected to the same Post entity via a 'hasTag' relationship. The Post entity is enclosed in a box labeled 'count', indicating that the count of posts is being tracked. The 'otherPerson' entity is connected to the 'Post' entity via a 'hasCreator' relationship. The 'tag' entity has a constraint 'name = \$tagName', and the 'otherTag' entity has a constraint 'name ≠ \$tagName'.</p>			
IC 4					
IC 5					
IC 6					
IC 7					
IC 8					
IC 9					
IC 10					
IC 11					
IC 12					
IC 13	desc.	Given a start Person and some Tag, find the other Tags that occur together with this Tag on Posts that were created by start Person's friends and friends of friends (excluding start Person). Return top 10 Tags, and the count of Posts that were created by these Persons, which contain both this Tag and the given Tag.			
IC 14	params	1	personId	ID	
		2	tagName	Long String	
	result	1	otherTag.name	Long String	R
		2	postCount	32-bit Integer	A
	sort	1	postCount	↓	
		2	otherTag.name	↑	
	limit	10			
	CPs	5.1, 8.2			
	relevance	This query looks for paths of lengths three or four, starting from a given Person, moving to friends or friends of friends, then to Posts and finally ending at a given Tag.			