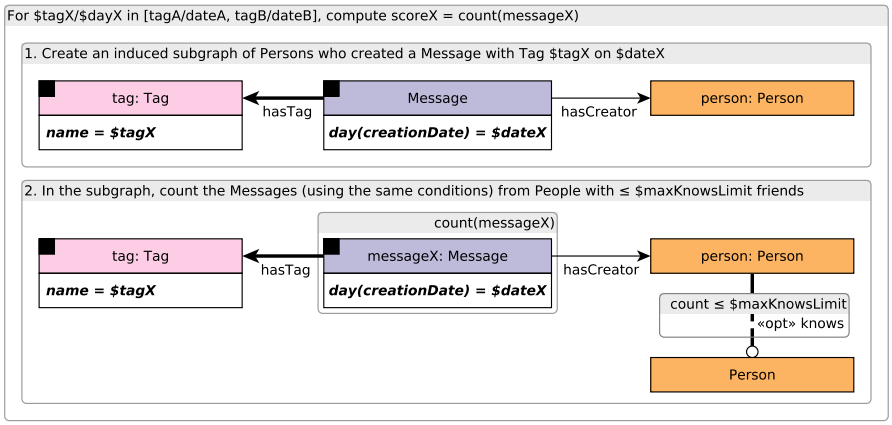
BI / read / 16

BI 1
BI 2
BI 3
BI 4
BI 5
BI 6
BI 7
BI 8
BI 9
BI 10
BI 11
BI 12
BI 13
BI 14
BI 15
BI 16
BI 17
BI 18
BI 19
BI 20

query	BI / read / 16
title	Fake news detection
pattern	
desc.	Given two Tag/date pairs (`tagA`/`dateA` and `tagB`/`dateB`), for each pair `tagX`/`dateX`: • Create an induced subgraph between Persons where for each pair of Persons `person1`/`person2`, both have created a Message on the day of `dateX` with Tag `tagX`. • In the induced subgraph, only keep pairs of Persons who have at most `maxKnowsLimit` friends (in the induced subgraph). • For these Persons, count the number of Messages created on `dateX` with Tag `tagX`. Return Persons who had at least one Messages for both `tagA`/`dateA` and `tagB`/`dateB` ranked by their total number of Messages (descending).

params

1	`tagA`	Long String	
2	`dateA`	Date	
3	`tagB`	Long String	
4	`dateB`	Date	
5	`maxKnowsLimit`	32-bit Integer	Selected between 3 and 6

result

1	`person.id`	ID	R	
2	`messageCountA`	32-bit Integer	A	Message count for `tagA`/`dateA`
3	`messageCountB`	32-bit Integer	A	Message count for `tagB`/`dateB`

sort

1	`messageCountA` + `messageCountB`	↓
2	`person.id`	↑

limit	20
CPs	5.3, 8.4, 8.5
relevance	There are two major ways to compute this query: (1) create the induced subgraph as suggested by the specification (either as a view or in materialized form), or (2) skip creating the induced subgraph and perform on-the-fly check for the number of friends (who also posted at least one Message with the given Tag on the given date). The latter approach is easier to express in systems which do not provide graph views but might result in redundant computations (the query engine will might repeatedly check whether a Person has at least one Message that satifies the conditions).