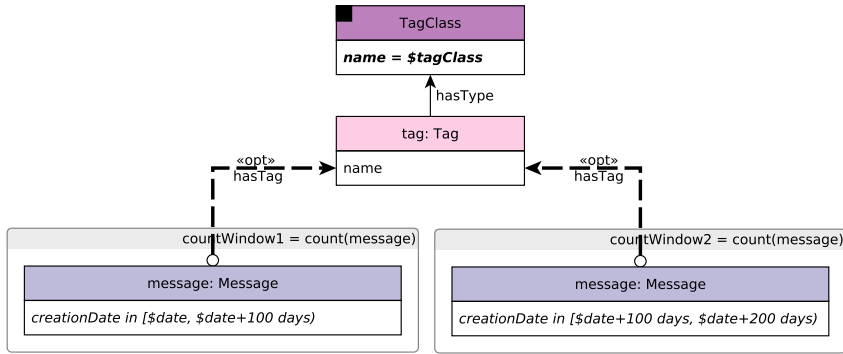


BI / read / 1

BI 1	query	BI / read / 1						
BI 2	title	Posting summary						
BI 3	pattern	<table><tr><td>message: Message</td></tr><tr><td>creationDate < \$dateTime</td></tr><tr><td>length year(creationDate)</td></tr></table>				message: Message	creationDate < \$dateTime	length year(creationDate)
message: Message								
creationDate < \$dateTime								
length year(creationDate)								
BI 4								
BI 5								
BI 6								
BI 7	desc.	Given a datetime, find all Messages created before that moment. Group them by a 3-level grouping:						
BI 8		1. by year of creation						
BI 9		2. for each year, group into Message types: is Comment or not						
BI 10		3. for each year-type group, split into four groups based on length of their content						
BI 11		• 0: $0 \leq \text{length} < 40$ (short)						
BI 12		• 1: $40 \leq \text{length} < 80$ (one liner)						
BI 13		• 2: $80 \leq \text{length} < 160$ (tweet)						
BI 14		• 3: $160 \leq \text{length}$ (long)						
BI 15	params	1	datetime	DateTime	For later microbatches, later datetime parameters are selected keep the variance low (<0.5%)			
BI 16								
BI 17	result	1	year	32-bit Integer	R	year(message.creationDate)		
BI 18		2	isComment	Boolean	M	True for Comments, False for Posts		
BI 19		3	lengthCategory	32-bit Integer	C	0 for short, 1 for one-liner, 2 for tweet, 3 for long		
BI 20		4	messageCount	32-bit Integer	A	Total number of Messages in that group		
		5	averageMessageLength	32-bit Integer	A	Average length of the Message content in that group		
		6	sumMessageLength	32-bit Integer	A	Sum of all Message content lengths		
		7	percentageOfMessages	32-bit Float	A	Number of Messages in group as a percentage of all messages created before the given date		
	sort	1	year	↓				
		2	isComment	↑	False < True, i.e. Posts come first and Comments second			
		3	lengthCategory	↑	order based on the lengthCategory value			
	CPs	1.2, 3.2, 4.1, 4.2, 8.5						

BI / read / 2

query	BI / read / 2																								
title	Tag evolution																								
pattern																									
desc.	Find the Tags under a given TagClass that were used in Messages during in the 100-day period starting at date and compare it with the 100-day period that follows. For the Tags and for both months, compute the count of Messages.																								
params	<table><tr><td>1</td><td>date</td><td>Date</td><td colspan="2"></td></tr><tr><td>2</td><td>tagClass</td><td>Long String</td><td colspan="2">TagClasses with a similar amount of Messages are selected</td></tr></table>					1	date	Date			2	tagClass	Long String	TagClasses with a similar amount of Messages are selected											
1	date	Date																							
2	tagClass	Long String	TagClasses with a similar amount of Messages are selected																						
result	<table><tr><td>1</td><td>tag.name</td><td>Long String</td><td>R</td><td></td></tr><tr><td>2</td><td>countWindow1</td><td>32-bit Integer</td><td>A</td><td>Occurrences of the tagClass during the first time window</td></tr><tr><td>3</td><td>countWindow2</td><td>32-bit Integer</td><td>A</td><td>Occurrences of the tagClass during the second time window</td></tr><tr><td>4</td><td>diff</td><td>32-bit Integer</td><td>A</td><td>Absolute difference of countWindow1 and countWindow2</td></tr></table>					1	tag.name	Long String	R		2	countWindow1	32-bit Integer	A	Occurrences of the tagClass during the first time window	3	countWindow2	32-bit Integer	A	Occurrences of the tagClass during the second time window	4	diff	32-bit Integer	A	Absolute difference of countWindow1 and countWindow2
1	tag.name	Long String	R																						
2	countWindow1	32-bit Integer	A	Occurrences of the tagClass during the first time window																					
3	countWindow2	32-bit Integer	A	Occurrences of the tagClass during the second time window																					
4	diff	32-bit Integer	A	Absolute difference of countWindow1 and countWindow2																					
sort	<table><tr><td>1</td><td>diff</td><td>↓</td><td colspan="2"></td></tr><tr><td>2</td><td>tag.name</td><td>↑</td><td colspan="2"></td></tr></table>					1	diff	↓			2	tag.name	↑												
1	diff	↓																							
2	tag.name	↑																							
limit	100																								
CPs	2.4, 3.1, 3.2, 4.1, 4.2, 4.3, 5.3, 6.1, 8.2, 8.5																								

BI / read / 3

query	BI / read / 3				
title	Popular topics in a country				
pattern					
desc.	<p>Given a TagClass and a Country, find all the Forums created in the given Country, containing at least one Message with Tags belonging directly to the given TagClass, and count the Messages by the Person who created it and by the Forum which contains them.</p> <p>The location of a Forum is identified by the location of the Forum’s moderator.</p>				
params	1	tagClass	Long String	TagClasses with a similar amount of Messages are selected	
	2	country	Long String	Big Countries are selected	
result	1	forum.id	ID	R	
	2	forum.title	Long String	R	
	3	forum.creationDate	DateTime	R	
	4	person.id	ID	R	
	5	messageCount	32-bit Integer	A	
sort	1	messageCount	↓		
	2	forum.id	↑		
limit	20				
CPs	1.1, 1.2, 1.3, 2.1, 2.2, 2.4, 3.3, 8.2				

BI / read / 4

BI 1	query	BI / read / 4			
BI 2	title	Top message creators by country			
BI 3	pattern	<p>The diagram illustrates the query pattern for finding top message creators by country. It is divided into two main steps:</p> <ol style="list-style-type: none"> Step 1: Select top 100 forums based on memberCount in country. This involves finding the top 100 forums (topForum) based on their memberCount. The relationships shown are: Country (name) isPartOf City (isLocatedIn member: Person). The memberCount is calculated as count(member). Step 2: For each country, for each of the top 100 forums (topForum), find the top posters. This involves finding the top posters (topPosters) for each forum. The relationships shown are: topForum (Forum) containerOf Post (replyOf*0..). The messageCount is calculated as count(message). The topPosters are those with creationDate > \$date. The relationships shown are: Message (hasCreator person: Person) and Forum (hasMember person: Person). 			
BI 16	desc.	<p>Find the most popular Forums by Country, where the popularity of a Forum is measured by the number of members that Forum has from a given Country.</p> <p>Calculate the top 100 most popular Forums. In case of a tie, the Forum(s) with the smaller id value(s) should be selected.</p> <p>For each member Person of the 100 most popular Forums, count the number of Messages (messageCount) they made in any of those (most popular) Forums. Also include those member Persons who have not posted any Messages (have a messageCount of 0).</p>			
BI 17	params	1	date	Date	Selected from the first 30 days of the network
BI 18	result	1	person.id	ID	R
BI 19		2	person.firstName	String	R
BI 20		3	person.lastName	String	R
		4	person.creationDate	DateTime	R
		5	messageCount	32-bit Integer	A
	sort	1	messageCount	↓	
		2	person.id	↑	
	limit	100			
	CPs	1.2, 1.3, 2.1, 2.2, 2.3, 2.4, 3.3, 5.3, 6.1, 8.2, 8.4			

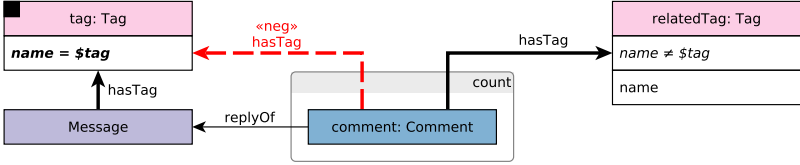
BI / read / 5

query	BI / read / 5			
title	Most active posters of a given topic			
pattern				
desc.	<p>Get each Person (person) who has created a Message (message) with a given Tag (direct relation, not transitive). Considering only these Messages, for each Person node:</p> <ul style="list-style-type: none"> Count its messages (messageCount). Count likes (likeCount) to its messages. Count Comments (replyCount) in reply to it messages. <p>The score is calculated according to the following formula: $1 \times \text{messageCount} + 2 \times \text{replyCount} + 10 \times \text{likeCount}$.</p>			
params	1	tag	Long String	Tags with a similar amount of Messages are selected
result	1	person.id	ID	R
	2	replyCount	32-bit Integer	A
	3	likeCount	32-bit Integer	A
	4	messageCount	32-bit Integer	A
	5	score	32-bit Integer	A
sort	1	score	↓	
	2	person.id	↑	
limit	100			
CPs	1.2, 2.3, 8.2			

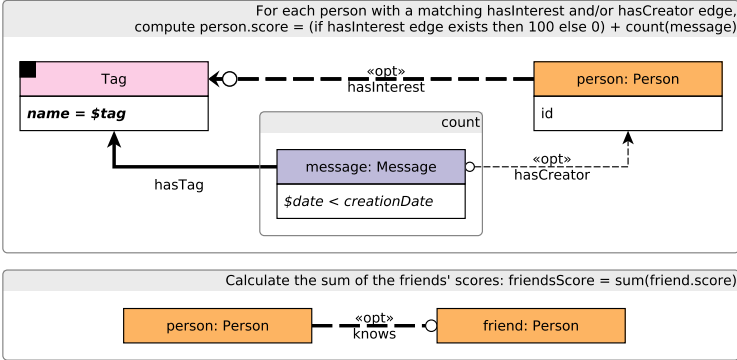
BI / read / 6

BI 1	query	BI / read / 6			
BI 2	title	Most authoritative users on a given topic			
BI 3	pattern	<pre> graph TD Tag[Tag] -- hasTag --> message1[message1: Message] message1 -- hasCreator --> person1[person: Person] message1 -- "«opt» likes" --> p2[p2: Person] p2 -- "person.authorityScore = sum(p2.popularityScore)" --> message1 p2 -- "hasCreator" --> message2[message2: Message] message2 -- "«opt» likes" --> p3[p3: Person] p3 -- "p2.popularityScore = count(p3)" --> message2 subgraph Compute_p2_popularityScore p2 message2 p3 end </pre>			
BI 6	desc.	<p>Given a Tag (tag), find all Persons (person) that ever created a Message with the Tag. For each of these Persons (person) compute their “authority score” as follows:</p> <ul style="list-style-type: none"> The “authority score” is the sum of “popularity scores” of the Persons (p2) that liked any of that Person’s Messages with the given Tag (same criterion as for message1). A Person’s (p2) “popularity score” is defined as the total number of likes on all of their Messages (message2). 			
BI 10	params	1	tag	Long String	Tags with a similar amount of Messages are selected
BI 12	result	1	person.id	ID	R
BI 13		2	authorityScore	32-bit Integer	A
BI 14	sort	1	authorityScore	↓	
BI 15		2	person1.id	↑	
BI 16	limit	100			
BI 17	CPs	1.2, 2.3, 3.3, 6.1, 8.2			
BI 18	relevance	Computing the authority scores might involve computing the popularity score for the same Person multiple times. Implementations are advised to avoid such redundant computations.			

BI / read / 7

BI 1	query	BI / read / 7			
BI 2	title	Related topics			
BI 3	pattern				
BI 8	desc.	Find all Messages that have a given Tag. Find the related Tags attached to (direct) reply Comments of these Messages, but only of those reply Comments that do not have the given Tag. Group the Tags by name, and get the count of replies in each group.			
BI 12	params	1	tag	Long String	Tags with a similar amount of Messages are selected
BI 14	result	1	relatedTag.name	Long String	R
BI 15		2	count	32-bit Integer	A
BI 17	sort	1	count	↓	
BI 18		2	relatedTag.name	↑	
BI 19	limit	100			
BI 20	CPs	1.4, 3.3, 5.2, 8.1			

BI / read / 8

query	BI / read / 8				
title	Central person for a tag				
pattern	<div><div>For each person with a matching hasInterest and/or hasCreator edge, compute person.score = ((if hasInterest edge exists then 100 else 0) + count(message))</div><div>Calculate the sum of the friends' scores: friendsScore = sum(friend.score)</div></div>				
desc.	<p>Given a Tag, find all Persons that are interested in the Tag and/or have written a Message (Post or Comment) with a creationDate after a given date and that has a given Tag. For each Person, compute the score as the sum of the following two aspects:</p> <ul style="list-style-type: none">• 100, if the Person has this Tag as their interest, or 0 otherwise• number of Messages by this Person with the given Tag <p>Also, for each Person, compute the sum of the score of the Person’s friends (friendsScore).</p>				
params	<div><div>1</div><div>tag</div></div>	<div><div>Long String</div></div>	<div>Tags with a similar amount of Messages are selected</div>		
	<div><div>2</div><div>date</div></div>	<div><div>Date</div></div>	<div>Dates from around the same day are selected. (TODO - how exactly? what distribution?)</div>		
result	<div><div>1</div><div>person.id</div></div>	<div><div>ID</div></div>	<div><div>R</div></div>		
	<div><div>2</div><div>score</div></div>	<div><div>32-bit Integer</div></div>	<div><div>A</div></div>		
	<div><div>3</div><div>friendsScore</div></div>	<div><div>32-bit Integer</div></div>	<div><div>A</div></div>	<div>The sum of the score of the person’s friends</div>	
sort	<div><div>1</div><div>score + friendsScore</div></div>	<div><div>↓</div></div>			
	<div><div>2</div><div>person.id</div></div>	<div><div>↑</div></div>			
limit	100				
CPs	1.2, 2.1, 2.3, 3.2, 5.3, 8.2, 8.4, 8.5				
relevance	Similarly to BI 16, there are two major ways to compute this query: (1) creating an induced subgraph of the interested Persons and their friends and performing the scoring on this graph or (2) performing the scoring without creating an induced subgraph and scoring the friends of a Person on-the-fly. The first approach is more efficient as it avoids redundant computations, however, specifying it needs support for composable graph queries.				

BI / read / 9

BI 1	query	BI / read / 9			
BI 2	title	Top thread initiators			
BI 3	pattern				
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	
	desc.	<p>For each Person, count the number of Posts they created in the time interval [startDate, endDate] (equivalent to the number of threads they initiated) and the number of Messages in each of their (transitive) reply trees, including the root Post of each tree. When calculating Message counts only consider Messages created within the given time interval.</p> <p>Return each Person, number of Posts they created, and the count of all Messages that appeared in the reply trees (including the Post at the root of tree) they created.</p>			
	params	1	startDate	Date	TODO
		2	endDate	Date	8-10 days after the startDate
	result	1	person.id	ID	R
		2	person.firstName	String	R
		3	person.lastName	String	R
		4	threadCount	32-bit Integer	A
		5	messageCount	32-bit Integer	A
		<p>The number of Posts created by that Person (the number of threads initiated)</p> <p>The number of Messages created in all the threads this Person initiated</p>			
	sort	1	messageCount	↓	
		2	person.id	↑	
	limit	100			
	CPs	1.2, 2.2, 2.3, 3.2, 7.2, 7.3, 7.4, 8.1, 8.5			

BI / read / 10

query	BI / read / 10				
title	Experts in social circle				
pattern					
desc.	<p>Given a Person (startPerson), find all other Persons (expertCandidatePerson) that live in a given Country and are connected to given Person by a <i>shortest path</i> with length in range [minPathDistance, maxPathDistance] through the knows relation.</p> <p>For each of these expertCandidatePerson nodes, retrieve all of their Messages that contain at least one Tag belonging to a given TagClass (direct relation not transitive). For each Message, retrieve all of its Tags.</p> <p>Group the results by Persons and Tags, then count the Messages by a certain Person having a certain Tag.</p>				
params	<div>1</div>	personId	ID	The ID of the startPerson. Persons with a similar degree of knows edges are selected	
	<div>2</div>	country	String	Countries with a similar number of Persons are selected	
	<div>3</div>	tagClass	Long String	TagClasses with a similar degree of hasType edges are selected	
	<div>4</div>	minPathDistance	32-bit Integer	1 or 2	
	<div>5</div>	maxPathDistance	32-bit Integer	2 or 3	
result	<div>1</div>	expertCandidatePerson.id	ID	R	
	<div>2</div>	tag.name	Long String	R	
	<div>3</div>	messageCount	32-bit Integer	A	Number of Messages created by that Person containing that Tag
sort	<div>1</div>	messageCount	↓		
	<div>2</div>	tag.name	↑		
	<div>3</div>	expertCandidatePerson.id	↑		
limit	100				
CPs	1.2, 1.3, 2.3, 2.4, 3.3, 5.3, 7.1, 7.2, 7.3, 8.1, 8.6				

BI / read / 11

query	BI / read / 11											
title	Friend triangles											
pattern	<pre>graph TD Country[Country] City1[City] City2[City] City3[City] PersonA[a: Person] PersonB[b: Person] PersonC[c: Person] Country -- isPartOf --> City1 Country -- isPartOf --> City2 Country -- isPartOf --> City3 City1 -- isLocatedIn --> PersonA City2 -- isLocatedIn --> PersonB City3 -- isLocatedIn --> PersonC PersonA -- knows --> PersonB PersonB -- knows --> PersonC PersonC -- knows --> PersonA linkStyle 6 stroke-dasharray: 5 5 linkStyle 7 stroke-dasharray: 5 5 linkStyle 8 stroke-dasharray: 5 5</pre>											
desc.	<p>For a given country, count all the distinct triples of Persons such that:</p> <ul style="list-style-type: none">• a is friend of b,• b is friend of c,• c is friend of a, <p>and these friendships were created after a given startDate.</p> <p>Distinct means that given a triple t_1 in the result set R of all qualified triples, there is no triple t_2 in R such that t_1 and t_2 have the same set of elements.</p>											
params	<table><tr><td>1</td><td>country</td><td>Long String</td><td></td></tr><tr><td>2</td><td>startDate</td><td>Date</td><td></td></tr></table>	1	country	Long String		2	startDate	Date				
1	country	Long String										
2	startDate	Date										
result	<table><tr><td>1</td><td>count</td><td>32-bit Integer</td><td>A</td><td></td></tr></table>	1	count	32-bit Integer	A							
1	count	32-bit Integer	A									
CPs	1.1, 2.3, 2.5											

BI / read / 12

query	BI / read / 12				
title	How many persons have a given number of messages				
pattern					
desc.	<p>For each Person, count the number of Messages they made (messageCount). Only count Messages with the following attributes:</p> <ul style="list-style-type: none">• Its content is not empty (and consequently, the imageFile attribute is empty for Posts).• Its length is below the lengthThreshold (exclusive, equality is not allowed).• Its creationDate is after date (exclusive, equality is not allowed).• It is written in any of the given languages. <ul style="list-style-type: none">– The language of a Post is defined by its language attribute.– The language of a Comment is that of the Post that initiates the thread where the Comment replies to. <p>The Post and Comments in the reply tree’s path (from the Message to the Post) do not have to satisfy the constraints for content, length and creationDate.</p> <p>For each messageCount value, count the number of Persons with exactly messageCount Messages (with the required attributes).</p>				
params	<div>1</div>	date	Date		
	<div>2</div>	lengthThreshold	32-bit Integer	Selected as balanced against date to filter around 30% of the Messages within a language and keep the variance low	
	<div>3</div>	languages	{String}	Only the most frequently used languages are selected	
result	<div>1</div>	messageCount	32-bit Integer	A	Number of Messages created
	<div>2</div>	personCount	32-bit Integer	A	Number of Persons with messageCount Messages
sort	<div>1</div>	personCount	↓		
	<div>2</div>	messageCount	↓		
CPs	1.1, 1.2, 1.4, 3.2, 4.2, 4.3, 8.1, 8.2, 8.3, 8.4, 8.5				

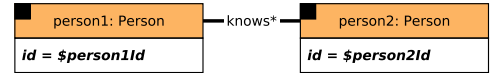
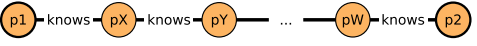
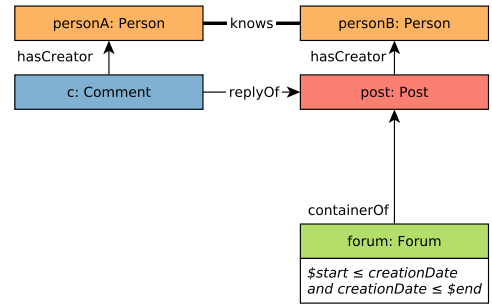
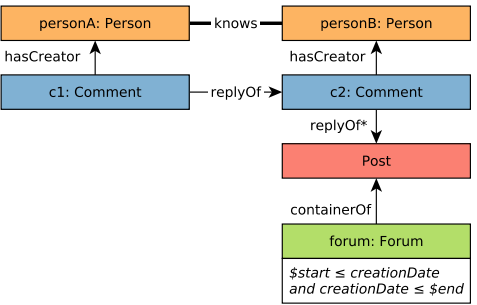
BI / read / 13

query	BI / read / 13				
title	Zombies in a country				
pattern	<div><div>1. zombies = collect(zombie)</div><div><div><div>Country</div><div>name = \$country</div></div><div><div>City</div><div>isPartOf</div></div><div><div>zombie: Person</div><div>creationDate < \$endDate and (messageCount / months) < 1</div><div>isLocatedIn</div></div><div><div>message: Message</div><div>messageCount = count(message) creationDate < \$endDate</div><div><<opt>> hasCreator</div></div></div></div> <div><div>2. For each zombie IN zombies, calculate: zombieScore = zombieLikeCount / totalLikeCount</div><div><div><div>likerPerson: Person</div><div>creationDate < \$endDate</div><div>totalLikeCount = count(likerPerson)</div></div><div><div>zombie: Person</div><div>hasCreator</div></div><div><div>Message</div><div><<opt>> likes</div></div><div><div>likerZombie: Person</div><div>creationDate < \$endDate and likerZombie IN zombies</div><div>zombieLikeCount = count(likerZombie)</div></div><div><<opt>> likes</div></div></div>				
desc.	<p>Find zombies within the given country, and return their zombie scores. A zombie is a Person created before the given endDate, which has created an average of [0, 1) Messages per month, during the time range between profile’s creationDate and the given endDate. The number of months spans the time range from the creationDate of the profile to the endDate with partial months on both end counting as one month (e.g. a creationDate of Jan 31 and an endDate of Mar 1 result in 3 months).</p> <p>For each zombie, calculate the following:</p> <ul style="list-style-type: none">• zombieLikeCount: the number of likes received from other zombies.• totalLikeCount: the total number of likes received.• zombieScore: zombieLikeCount / totalLikeCount. If the value of totalLikeCount is 0, the zombieScore of the zombie should be 0.0. <p>For both zombieLikeCount and totalLikeCount, only consider likes received from profiles that were created before the given endDate.</p>				
params	1	country	Long String	Only the largest Countries are selected	
	2	endDate	Date	Selected from the last days of the initial data set	
result	1	zombie.id	ID	R	
	2	zombieLikeCount	32-bit Integer	A	
	3	totalLikeCount	32-bit Integer	A	
	4	zombieScore	64-bit Float	A	Determined as zombieLikeCount / totalLikeCount
sort	1	zombieScore	↓		
	2	zombie.id	↑		
limit	100				
CPs	1.2, 2.1, 2.3, 2.4, 3.2, 3.3, 4.2, 5.1, 5.3, 8.2, 8.4, 8.5				

BI / read / 14

BI 1	query	BI / read / 14			
BI 2	title	International dialog			
BI 3	pattern	<p>For each pair of countries, calculate the cost as a sum of cases #1-5. Cases that have a match add to the final score with the specified value. Each case only counts once, multiple matches do not increase to the score.</p>			
BI 4					
BI 5					
BI 6					
BI 7					
BI 8					
BI 9					
BI 10					
BI 11					
BI 12					
BI 13					
BI 14	desc.	<p>Consider all pairs of people (person1, person2) such that one is located in a City of Country country1 and the other is located in a City of Country country2. For each City of Country country1, return the highest scoring pair. The score of a pair is defined as the sum of the subscores awarded for the following kinds of interaction. The initial value is score = 0.</p> <ol style="list-style-type: none"> 1. person1 has created a reply Comment to at least one Message by person2: score += 4 2. person1 has created at least one Message that person2 has created a reply to: score += 1 3. person1 and person2 know each other: score += 15 4. person1 liked at least one Message by person2: score += 10 5. person1 has created at least one Message that was liked by person2: score += 1 <p>Consequently, the maximum score a pair can obtain is: 4 + 1 + 15 + 10 + 1 = 31.</p> <p>This query has two variants based on whether the input parameters are selected as correlated (close countries) or uncorrelated (far countries).</p>			
BI 15					
BI 16					
BI 17					
BI 18					
BI 19					
BI 20					
	params	1	country1	Long String	<p>A: correlated with parameter country2, i.e. the countries are close and there are many Persons visiting both Countries.</p> <p>B: uncorrelated with parameter country2, i.e. the countries are afar and there are few Persons visiting both Countries.</p>
		2	country2	Long String	
	result	1	person1.id	ID	R
		2	person2.id	ID	R
		3	city1.name	Long String	R
		4	score	32-bit Integer	C
	sort	1	score	↓	
		2	person1.id	↑	
		3	person2.id	↑	
	CPs	1.3, 1.4, 2.1, 3.1, 3.3, 5.1, 5.2, 5.3, 8.3, 8.4			

BI / read / 15

BI 1	query	BI / read / 15			
BI 2	title	Trusted connection paths through forums created in a given timeframe			
BI 3	pattern	<p>Enumerate all unweighted shortest paths on knows edges between person1 to person2.</p> 			
BI 4		<p>For each knows edge in the path, calculate a weight based on interactions between the pair of Persons of the edge, calculated as a sum of cases #1 and #2 for the Persons (both ways), and the sum of these weights determine the total weight of each path.</p> 			
BI 5		<p>Case 1: Replies on Posts, weight += 1.0 × count(c)</p> 			
BI 6		<p>Case 2: Replies on Comments, weight += 0.5 × count(c1)</p> 			
BI 7					
BI 8					
BI 9					
BI 10					
BI 11					
BI 12					
BI 13	desc.	<p>Given two Persons, find all (unweighted) shortest paths between these two Persons, in the subgraph induced by the knows relationship.</p> <p>Then, for each path calculate a weight. The nodes in the path are Persons, and the weight of a path is the sum of weights between every pair of consecutive Person nodes in the path.</p> <p>The weight for a pair of Persons is calculated based on their interactions:</p> <ul style="list-style-type: none"> • Every direct reply (by one of the Persons) to a Post (by the other Person) contributes 1.0. • Every direct reply (by one of the Persons) to a Comment (by the other Person) contributes 0.5. <p>Only consider Messages that were created in a Forum that was created within the timeframe (interval) [startDate, endDate]. Note that for Comments, the containing Forum is that of the Post that the comment (transitively) replies to. Also note that interactions are counted both ways.</p> <p>Return all paths with the Person IDs ordered by their weights descending.</p>			
BI 14					
BI 15					
BI 16					
BI 17					
BI 18					
BI 19					
BI 20					
	params	1	person1Id	ID	
		2	person2Id	ID	
		3	startDate	Date	
		4	endDate	Date	
	result	1	person.id	[ID]	C Ordered sequence of the Person IDs in the path
		2	weight	64-bit Float	C
	sort	1	weight	↓	The order of paths with the same weight is unspecified
		2	personIds	↑	The IDs in the paths are used for lexicographical sorting
	CPs	1.2, 2.1, 2.2, 2.4, 3.3, 5.1, 5.3, 7.2, 7.3, 7.5, 7.7, 8.1, 8.2, 8.3, 8.4, 8.5, 8.6			

BI / read / 16

BI 1	query	BI / read / 16			
BI 2	title	Fake news detection			
BI 3	pattern	<p>For \$tagX/\$dayX in [tagA/dateA, tagB/dateB], compute scoreX = count(messageX)</p>			
BI 4					
BI 5					
BI 6					
BI 7					
BI 8					
BI 9					
BI 10					
BI 11					
BI 12					
BI 13	desc.	<p>Given two Tag/date pairs (tagA/dateA and tagB/dateB), for each pair tagX/dateX:</p> <ul style="list-style-type: none"> • 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. <p>Return Persons who had at least one Messages for both tagA/dateA and tagB/dateB ranked by their total number of Messages (descending).</p>			
BI 14					
BI 15					
BI 16					
BI 17	params	1	tagA	Long String	
BI 18		2	dateA	Date	
BI 19		3	tagB	Long String	
BI 20		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
		3	messageCountB	32-bit Integer	A
	sort	1	messageCountA + messageCountB	↓	
		2	person.id	↑	
	limit	20			
	CPs	5.3, 8.4, 8.5			
	relevance	<p>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 satisfies the conditions).</p>			

BI / read / 17

query	BI / read / 17										
title	Information propagation analysis										
pattern	<pre>graph TD person1[person1: Person] -- hasCreator --> message1[message1: Message] message1 -- hasTag --> tag[tag: Tag] message1 -- replyOf*0.. --> post1[post1: Post] post1 -- containerOf --> forum1[forum1: Forum] forum1 -- hasMember --> person2[person2: Person] forum1 -- hasMember --> person3[person3: Person] person2 -- hasCreator --> tag person3 -- hasCreator --> tag tag -- hasTag --> message2[message2: Message] message2 -- replyOf --> comment[comment: Comment] comment -- replyOf*0.. --> post2[post2: Post] post2 -- containerOf --> forum2[forum2: Forum] forum2 -- hasMember --> person1 forum2 -- «neg» hasMember --> person1 message2 -- count --> count[count]</pre>										
desc.	<p>This query aims to identify instances of “information propagation” when a Person (person1) submits a Message (message1) with a given Tag (tag) to a Forum (forum1). This is read by other members of forum1, Persons person2 and person3. Some time later (specified by the delta parameter), these persons have a discussion with the same tag in a different Forum (forum2) where person1 is not a member. The discussion consists of a Message (message2) by person2 and a direct reply Comment (comment) by person3.</p> <p>Return IDs of person1 with the number of interactions their Messages (might have) caused.</p>										
params	<table><tr><td>1</td><td>tag</td><td>Long String</td><td>Tags with a similar amount of Messages are selected</td></tr><tr><td>2</td><td>delta</td><td>32-bit Integer</td><td>Measured in hours, selected to be between 8 and 16 hours.</td></tr></table>	1	tag	Long String	Tags with a similar amount of Messages are selected	2	delta	32-bit Integer	Measured in hours, selected to be between 8 and 16 hours.		
1	tag	Long String	Tags with a similar amount of Messages are selected								
2	delta	32-bit Integer	Measured in hours, selected to be between 8 and 16 hours.								
result	<table><tr><td>1</td><td>person1.id</td><td>ID</td><td>R</td><td></td></tr><tr><td>2</td><td>messageCount</td><td>32-bit Integer</td><td>A</td><td></td></tr></table>	1	person1.id	ID	R		2	messageCount	32-bit Integer	A	
1	person1.id	ID	R								
2	messageCount	32-bit Integer	A								
sort	<table><tr><td>1</td><td>person1.id</td><td>↑</td><td></td></tr></table>	1	person1.id	↑							
1	person1.id	↑									
CPs	2.1, 2.3, 8.1										

BI / read / 18

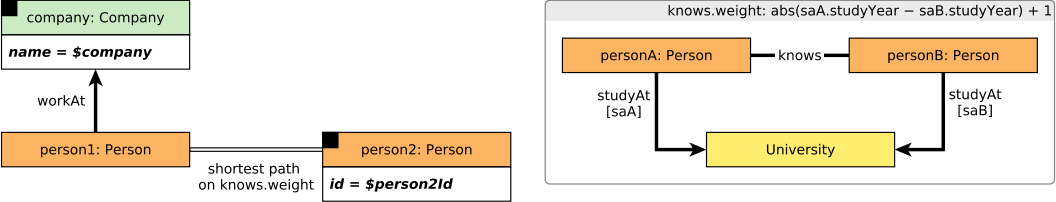
BI 1
BI 2
BI 3
BI 4
BI 5
BI 6
BI 7
BI 8
BI 9
BI 10
BI 11
BI 12
BI 13
BI 14
BI 15
BI 16
BI 17
BI 18
BI 19
BI 20

query	BI / read / 18				
title	Friend recommendation				
pattern	<div><div>For each person1 compute top-k(person2) based on mutualFriendCount</div><div><div><div>person1: Person</div><div>id = \$person1Id</div><div>«neg» knows</div></div><div>knows</div><div><div>Person</div><div>mutualFriendCount = count(*)</div></div><div>knows</div><div><div>person2: Person</div><div>≠ person1</div><div>id</div></div><div><div>tag: Tag</div><div>name = \$tag</div></div><div>hasInterest</div></div></div>				
desc.	<div>For a given Person (person1) and a Tag (tag), recommend new friends (person2) who</div> <ul style="list-style-type: none">• do not yet know person1• have many mutual friends with person1• are interested in tag. <div>Rank Persons person2 based on the number of mutual friends.</div>				
params	<div><div>1</div><div>person1Id</div><div>ID</div><div>Persons with a similar amount of friends are selected</div></div>				
	<div><div>2</div><div>tag</div><div>Long String</div><div>Tags with a similar amount of Messages are selected</div></div>				
result	<div><div>1</div><div>person2.id</div><div>ID</div><div>R</div></div>				
	<div><div>2</div><div>mutualFriendCount</div><div>32-bit Integer</div><div>A</div></div>				
sort	<div><div>1</div><div>mutualFriendCount</div><div>↓</div></div>				
	<div><div>2</div><div>person2.id</div><div>↑</div></div>				
limit	20				
CPs	2.5, 8.1				

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>Case i2: Reply from personB to personA's Message</p>			
BI 14	desc.	<p>Given two Cities <i>city1</i>, <i>city2</i>, find Persons <i>person1</i>, <i>person2</i> 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 15	params	1	city1Id	ID	Small Cities within the same Country are selected
BI 16		2	city2Id	ID	
BI 17	result	1	person1.id	ID	R
BI 18		2	person2.id	ID	R
BI 19		3	totalWeight	64-bit Float	C
BI 20	sort	1	totalWeight	↓	
		2	person1.id	↑	
		3	person2.id	↑	
	limit	20			
	CPs	3.3, 7.6, 7.7, 8.4, 8.6			
	relevance	<p>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).</p>			

BI / read / 20

BI 1	query	BI / read / 20			
BI 2	title	Recruitment			
BI 3	pattern				
BI 10	desc.	<p>Given a Company company and a Person person2 (who is known to be working at another Company), find a Person (person1) working the in the company who have the top-20 shortest path to person2 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>).</p> <p>If there are multiple Person person1 nodes with the same shortest path, return all of them.</p>			
BI 16	params	1	company	Long String	Companies with a similar number of employees (former or current) are selected
BI 17		2	person2Id	ID	
BI 18	result	1	person1.id	ID	R
BI 19		2	totalWeight	64-bit Integer	C
BI 20	sort	1	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.			