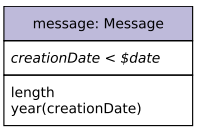
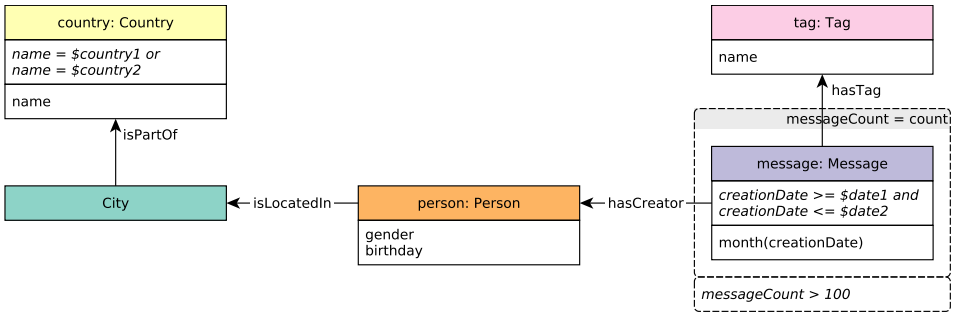
BI / read / 1

BI 1	query	BI / read / 1
BI 2	title	Posting summary
BI 3	pattern	
BI 4	desc.	Given a date, find all Messages created before that date. Group them by a 3-level grouping: 1. by year of creation 2. for each year, group into Message types: is Comment or not 3. for each year-type group, split into four groups based on length of their content • `0`: 0 <= length < 40 (short) • `1`: 40 <= length < 80 (one liner) • `2`: 80 <= length < 160 (tweet) • `3`: 160 <= length (long)
BI 5	params	1 `date` Date
BI 6	result	1 `year` 32-bit Integer R `year(message.creationDate)` 2 `isComment` Boolean M `true` for Comments, `false` for Posts 3 `lengthCategory` String C `0` for short, `1` for one-liner, `2` for tweet, `3` for long 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
BI 7	sort	1 `year` ↓ 2 `isComment` ↑ `false` < `true`, i.e. the ordering puts Posts first, and Comments second 3 `lengthCategory` ↑ order based on the length of the category, `0` (short), `1` (one liner), etc.
BI 8	CPs	1.2, 3.2, 4.1, 8.5
BI 9		
BI 10		
BI 11		
BI 12		
BI 13		
BI 14		
BI 15		
BI 16		
BI 17		
BI 18		
BI 19		
BI 20		
BI 21		
BI 22		
BI 23		
BI 24		
BI 25		

BI / read / 2

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
BI 21
BI 22
BI 23
BI 24
BI 25

query	BI / read / 2
title	Top tags for country, age, gender, time
pattern	
desc.	Select all Messages created in the range of `[startDate, endDate]` by Persons located in `country1` or `country2`. Select the creator Persons and the Tags of these Messages. Split these Persons, Tags and Messages into a 5-level grouping: 1. name of country of Person, 2. month the Message was created, 3. gender of Person, 4. age group of Person, defined as years between person's birthday and end of simulation (2013-01-01), divided by 5, rounded down (partial years do not count), 5. name of tag attached to Message. Consider only those groups where number of Messages is greater than 100.

params

1	`startDate`	Date	
2	`endDate`	Date	
3	`country1`	String	
4	`country2`	String	

result

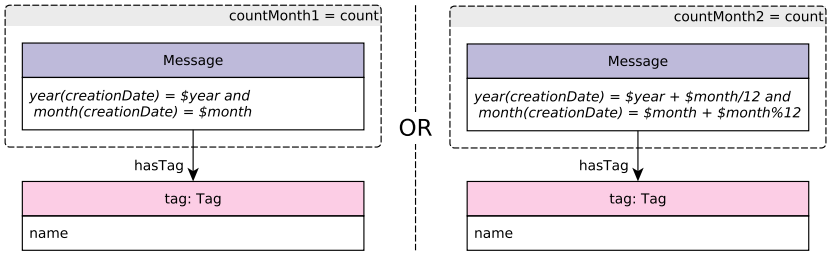
1	`country.name`	String	R	
2	`messageMonth`	32-bit Integer	R	1–12
3	`person.gender`	String	R	`male` or `female`
4	`ageGroup`	32-bit Integer	C	
5	`tag.name`	String	R	
6	`messageCount`	64-bit Integer	A	The number of messages in the group

sort

1	`messageCount`	↓
2	`tag.name`	↑
3	`ageGroup`	↑
4	`person.gender`	↑
5	`messageMonth`	↑
6	`country.name`	↑

limit	100
CPs	1.1, 1.2, 1.3, 2.1, 2.3, 3.1, 3.2, 8.2, 8.5

BI / read / 3

query	BI / read / 3
title	Tag evolution
pattern	
desc.	Find the Tags that were used in Messages during the given `month` of the given `year` and the Tags that were used during the next month. For the Tags and for both months, compute the count of Messages.

params

1	`year`	32-bit Integer
2	`month`	32-bit Integer

result

1	`tag.name`	String	R	
2	`countMonth1`	32-bit Integer	A	Occurrences of the tag during the given `year` and `month`
3	`countMonth2`	32-bit Integer	A	Occurrences of the tag during the next month after the given `year` and `month`
4	`diff`	32-bit Integer	A	Absolute difference of `countMonth1` and `countMonth2`

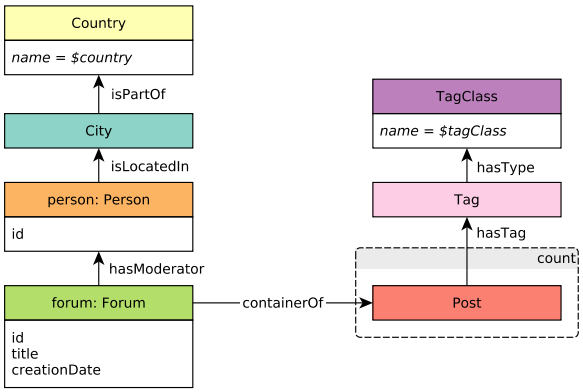
sort

1	`diff`	↓
2	`tag.name`	↑

limit	100
CPs	2.4, 3.1, 3.2, 4.1, 4.3, 5.3, 6.1, 8.2, 8.5

BI / read / 4

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
BI 21
BI 22
BI 23
BI 24
BI 25

query	BI / read / 4
title	Popular topics in a country
pattern	
desc.	Given a TagClass and a Country, find all the Forums created in the given Country, containing at least one Post with Tags belonging directly to the given TagClass. The location of a Forum is identified by the location of the Forum's moderator.

params

1	tagClass	String
2	country	String

result

1	forum.id	64-bit Integer	R
2	forum.title	String	R
3	forum.creationDate	DateTime	R
4	person.id	64-bit Integer	R
5	postCount	32-bit Integer	A

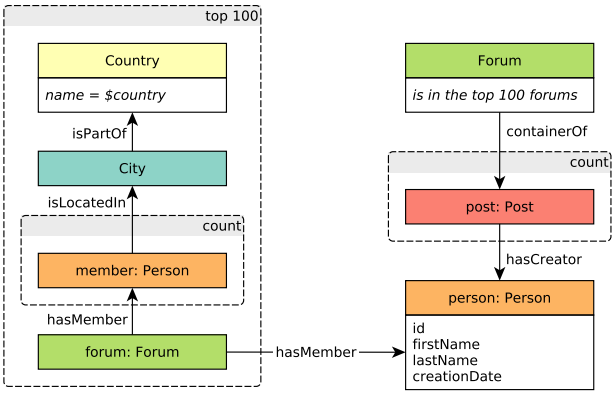
sort

1	postCount	↓
2	forum.id	↑

limit	20
CPs	1.1, 1.2, 1.3, 2.1, 2.2, 2.4, 3.3, 8.2

BI / read / 5

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
BI 21
BI 22
BI 23
BI 24
BI 25

query	BI / read / 5
title	Top posters in a country
pattern	
desc.	Find the most popular Forums for a given Country, where the popularity of a Forum is measured by the number of members that Forum has from the given Country. Calculate the top 100 most popular Forums. In case of a tie, the forum(s) with the smaller id value(s) should be selected. For each member Person of the 100 most popular Forums, count the number of Posts (`postCount`) they made in any of those (most popular) Forums. Also include those member Persons who have not posted any messages (have a `postCount` of 0).

params

1	`country`	String

result

1	`person.id`	64-bit Integer	R	
2	`person.firstName`	String	R	
3	`person.lastName`	String	R	
4	`person.creationDate`	DateTime	R	
5	`postCount`	32-bit Integer	A	

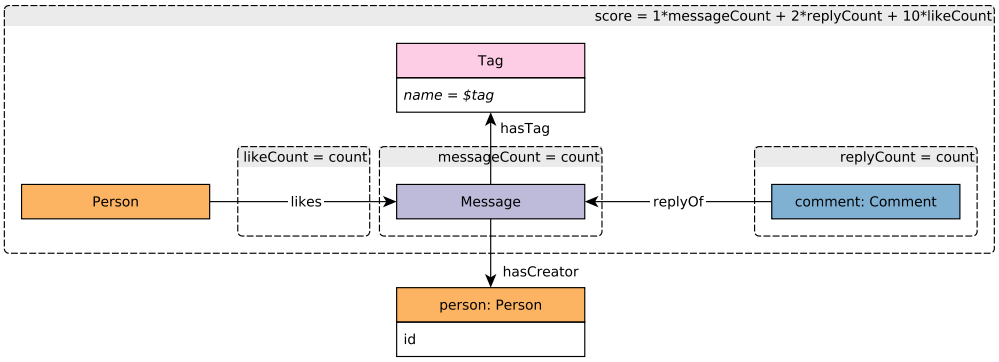
sort

1	`postCount`	↓
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 / 6

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
BI 21
BI 22
BI 23
BI 24
BI 25

query	BI / read / 6
title	Most active posters of a given topic
pattern	
desc.	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: • Count its `messages` (`messageCount`). • Count likes (`likeCount`) to its `messages`. • Count Comments (`replyCount`) in reply to it `messages`. The `score` is calculated according to the following formula: `1 * messageCount + 2 * replyCount + 10 * likeCount`.

params

1	tag	String

result

1	person.id	64-bit Integer	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	

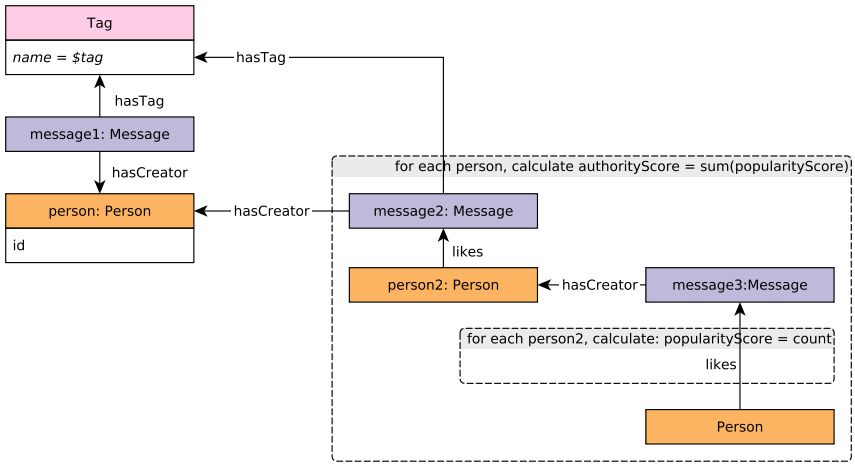
sort

1	score	↓	
2	person.id	↑	

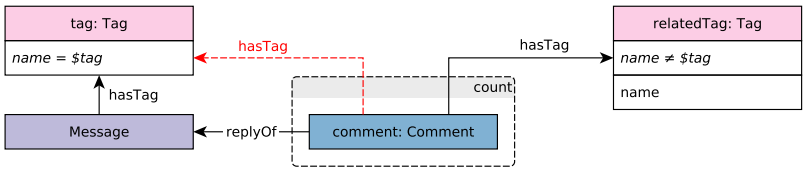
limit	100
CPs	1.2, 2.3, 8.2

BI / read / 7

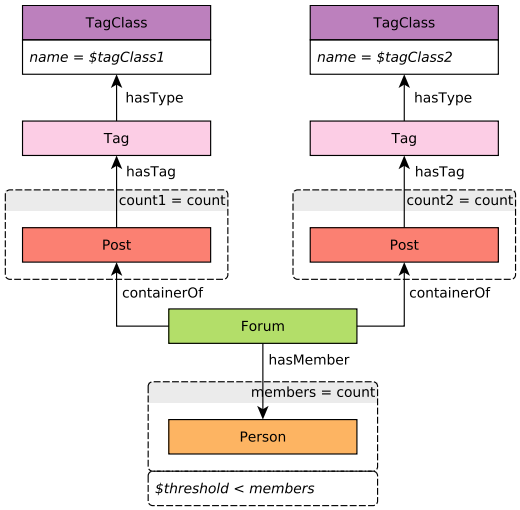
BI 1
BI 2
BI 3
BI 4
BI 5
BI 6
BI 7
BI 8
BI 9
BI 10
BI 11
BI 12
BI 13
BI 14
BI 15
BI 16
BI 17
BI 18
BI 19
BI 20
BI 21
BI 22
BI 23
BI 24
BI 25

query	BI / read / 7
title	Most authoritative users on a given topic
pattern	
desc.	Given a Tag, find all Persons (`person`) that ever created a Message (`message1`) with the given Tag. For each of these Persons (`person`) compute their "authority score" as follows: • The "authority score" is the sum of "popularity scores" of the Persons (`person2`) that liked any of that Person's Messages (`message2`) with the given Tag. • A Person's (`person2`) "popularity score" is defined as the total number of likes on all of their Messages (`message3`).
params	1 `tag` String
result	1 `person.id` 64-bit Integer R 2 `authorityScore` 32-bit Integer A
sort	1 `authorityScore` ↓ 2 `person1.id` ↑
limit	100
CPs	1.2, 2.3, 3.2, 3.3, 6.1, 8.2

BI / read / 8

query	BI / read / 8
title	Related topics
pattern	
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.
params	1 | tag | String
result	1 | relatedTag.name | String | R 2 | count | 32-bit Integer | A
sort	1 | count | ↓ 2 | relatedTag.name | ↑
limit	100
CPs	1.4, 3.3, 5.2, 8.1

BI / read / 9

query	BI / read / 9
title	Forum with related tags
pattern	
desc.	Given two TagClasses (`tagClass1` and `tagClass2`), find Forums that contain • at least one Post (`post1`) with a Tag with a (direct) type of `tagClass1` and • at least one Post (`post2`) with a Tag with a (direct) type of `tagClass2`. The `post1` and `post2` nodes may be the same Post. Consider the Forums with a number of members greater than a given `threshold`. For every such Forum, count the number of `post1` nodes (`count1`) and the number of `post2` nodes (`count2`).

params

1	`tagClass1`	String
2	`tagClass2`	String
3	`threshold`	32-bit Integer

result

1	`forum.id`	64-bit Integer	R	
2	`count1`	32-bit Integer	A	Number of `post1` nodes
3	`count2`	32-bit Integer	A	Number of `post2` nodes

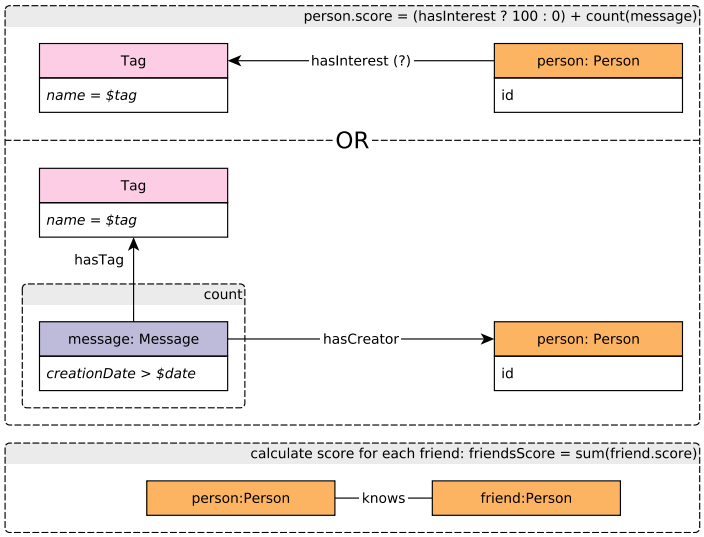
sort

1	`abs(count2 - count1)`	↓
2	`forum.id`	↑

limit	100
CPs	1.2, 1.3, 2.1, 2.3, 2.4, 8.2

BI / read / 10

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
BI 21
BI 22
BI 23
BI 24
BI 25

query	BI / read / 10
title	Central person for a tag
pattern	
desc.	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: • 100, if the Person has this Tag as their interest, or 0 otherwise • number of Messages by this Person with the given Tag Also, for each Person, compute the sum of the score of the Person's friends (`friendsScore`).

params

1	`tag`	String
2	`date`	Date

result

1	`person.id`	64-bit Integer	R	
2	`score`	32-bit Integer	A	
3	`friendsScore`	32-bit Integer	A	The sum of the score of the Person's friends

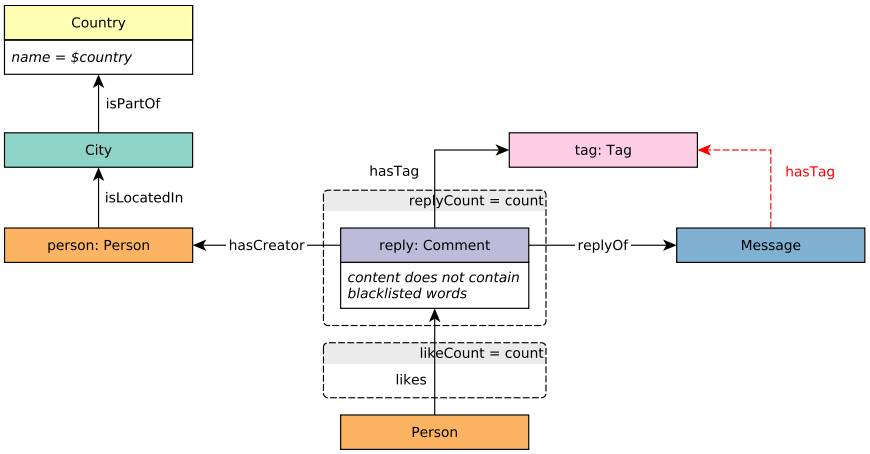
sort

1	`score + friendsScore`	↓
2	`person.id`	↑

limit	100
CPs	1.2, 2.1, 2.3, 3.2, 8.2, 8.4, 8.5

BI / read / 11

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
BI 21
BI 22
BI 23
BI 24
BI 25

query	BI / read / 11
title	Unrelated replies
pattern	
desc.	Find those Persons of a given Country that replied to any Message, such that the reply does not have any Tag in common with the Message (only direct replies are considered, transitive ones are not). Consider only those replies that do no contain any word from a given `blacklist`. For each Person and valid reply, retrieve the Tags associated with the reply, and retrieve the number of likes on the reply. The detailed conditions for checking blacklisted words are currently as follows. Words do not have to stand separately, i.e. if the word "Green" is blacklisted, "South-Greenland" cannot be included in the results. Also, comparison should be done in a case-sensitive way. These conditions are preliminary and might be changed in later versions of the benchmark.

params

1	`country`	String
2	`blacklist`	String[]

result

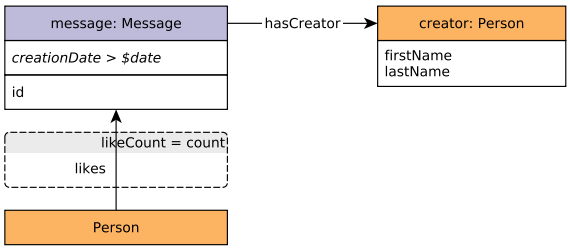
1	`person.id`	64-bit Integer	R	
2	`tag.name`	String	R	
3	`likeCount`	32-bit Integer	A	The count of likes to replies with that Tag
4	`replyCount`	32-bit Integer	A	The count of replies with that Tag

sort

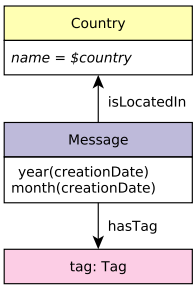
1	`likeCount`	↓
2	`person.id`	↑
3	`tag.name`	↑

limit	100
CPs	1.1, 2.1, 2.2, 2.3, 3.1, 3.2, 6.1, 8.1, 8.3

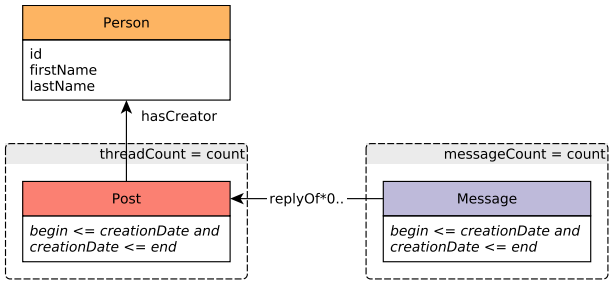
BI / read / 12

BI 1	query	BI / read / 12
BI 2	title	Trending posts
BI 3	pattern	
BI 4	desc.	Find all Messages created after a given `date` (exclusive), that received more than a given number of likes (`likeThreshold`).
BI 5	params	1 `date` — Date 2 `likeThreshold` — 32-bit Integer
BI 6	result	1 `message.id` — 64-bit Integer — R 2 `message.creationDate` — DateTime — R 3 `creator.firstName` — String — R — The first name of the Post's `creator` 4 `creator.lastName` — String — R — The last name of the Post's `creator` 5 `likeCount` — 32-bit Integer — A — The number of likes the Post received
BI 7	sort	1 `likeCount` ↓ 2 `message.id` ↑
BI 8	limit	100
BI 9	CPs	1.2, 2.2, 3.1, 6.1, 8.5

BI / read / 13

query	BI / read / 13
title	Popular tags per month in a country
pattern	
desc.	Find all Messages in a given Country, as well as their Tags. Group Messages by creation `year` and `month`. For each group, find the 5 most popular Tags, where popularity is the number of Messages (from within the same group) where the Tag appears. Note: even if there are no Tags for Messages in a given `year` and `month`, the result should include the `year` and `month` with an empty `popularTags` list.
params	1 `country` String
result	1 `year` 32-bit Integer C `year(message.creationDate)` 2 `month` 32-bit Integer C `month(message.creationDate)` 3 `popularTags` TagPairs C `(tag.name [String], popularity [32-bit Integer])` pairs, sorted descending by `popularity`, then ascending by `tag.name`
sort	1 `year` ↓ 2 `month` ↑
limit	100
CPs	1.2, 2.2, 2.3, 3.2, 6.1, 8.3, 8.5

BI / read / 14

query	BI / read / 14
title	Top thread initiators
pattern	
desc.	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. 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.

params

1	`startDate`	Date
2	`endDate`	Date

result

1	`person.id`	64-bit Integer	R	
2	`person.firstName`	String	R	
3	`person.lastName`	String	R	
4	`threadCount`	32-bit Integer	A	The number of Posts created by that Person (the number of threads initiated)
5	`messageCount`	32-bit Integer	A	The number of Messages created in all the threads this Person initiated

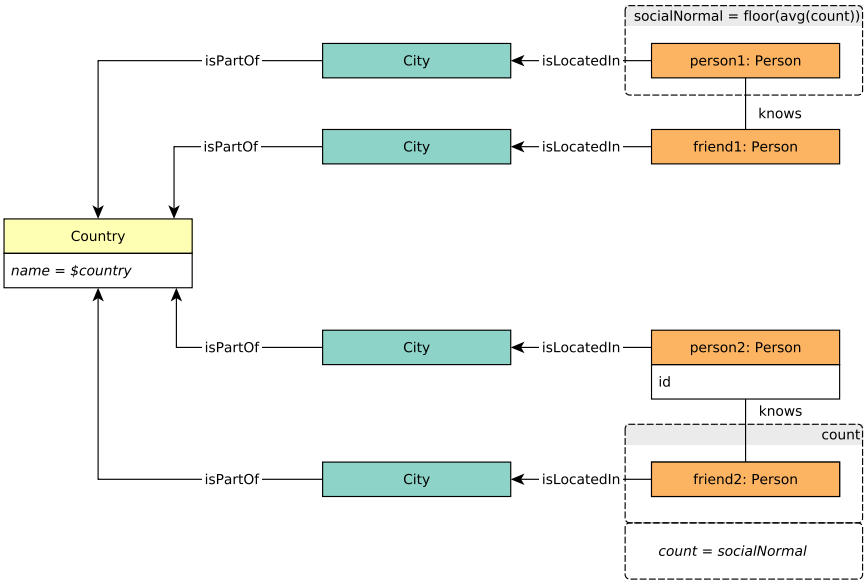
sort

1	`messageCount`	↓
2	`person.id`	↑

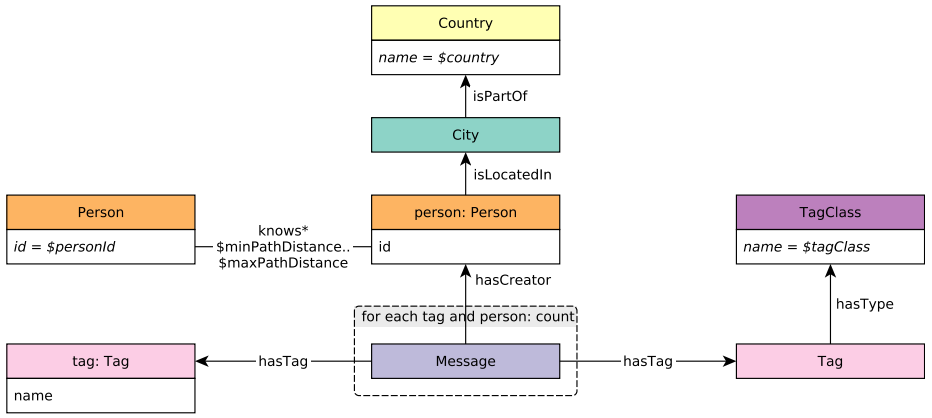
limit	100
CPs	1.2, 2.2, 2.3, 3.2, 7.2, 7.3, 7.4, 8.1, 8.5

BI 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
BI 21
BI 22
BI 23
BI 24
BI 25

BI / read / 15

query	BI / read / 15
title	Social normals
pattern	
desc.	Given a Country `country`, determine the "social normal", i.e. the floor of average number of friends that Persons of `country` have in `country`. Then, find all Persons in `country`, whose number of friends in `country` equals the social normal value.
params	1 `country` String
result	1 `person.id` 64-bit Integer R 2 `count` 32-bit Integer A
sort	1 `person.id` ↑
limit	100
CPs	1.2, 2.3, 3.2, 3.3, 5.3, 6.1, 8.2, 8.4

BI / read / 16

query	BI / read / 16
title	Experts in social circle
pattern	
desc.	Given a Person, find all other Persons that live in a given Country and are connected to given Person by a transitive trail with length in range `[minPathDistance, maxPathDistance]` through the knows relation. In the trail, an edge can be only traversed once while nodes can be traversed multiple times (as opposed to a path which allows repetitions of both nodes and edges). For each of these Persons, 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. Group the results by Persons and Tags, then count the Messages by a certain Person having a certain Tag. (Note: it is not yet decided whether a Person connected to the start Person on a trail with a length smaller than `minPathDistance`, but also on a trail with the length in `[minPathDistance, maxPathDistance]` should be included. The current reference implementations allow such Persons, but this might be subject to change in the future.)

params

1	`personId`	64-bit Integer	
2	`country`	String	
3	`tagClass`	String	
4	`minPathDistance`	32-bit Integer	
5	`maxPathDistance`	32-bit Integer	

result

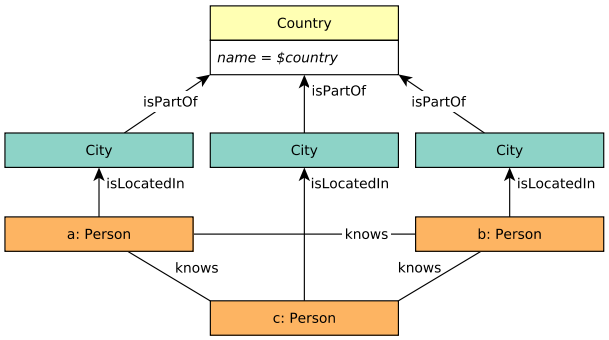
1	`person.id`	64-bit Integer	R	
2	`tag.name`	String	R	
3	`messageCount`	32-bit Integer	A	Number of Messages created by that Person containing that Tag

sort

1	`messageCount`	↓
2	`tag.name`	↑
3	`person.id`	↑

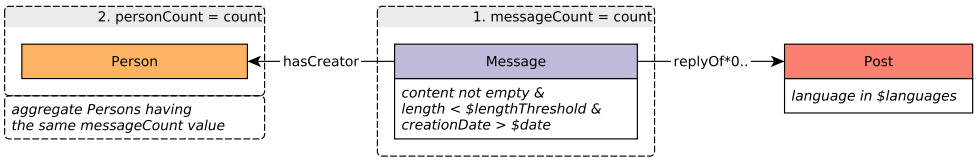
limit	100
CPs	1.2, 1.3, 2.3, 2.4, 3.3, 5.3, 7.1, 7.2, 7.3, 8.1, 8.6

BI / read / 17

query	BI / read / 17
title	Friend triangles
pattern	
desc.	For a given `country`, count all the distinct triples of Persons such that: • `a` is friend of `b`, • `b` is friend of `c`, • `c` is friend of `a`. 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.
params	1 `country` String
result	1 `count` 32-bit Integer A
CPs	1.1, 2.3

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 BI 21 BI 22 BI 23 BI 24 BI 25

query	BI / read / 18
title	How many persons have a given number of messages
pattern	
desc.	For each Person, count the number of Messages they made (`messageCount`). Only count Messages with the following attributes: • Its `content` is not empty (and consequently, `imageFile` 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`. – 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. 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`. For each `messageCount` value, count the number of Persons with exactly `messageCount` Messages (with the required attributes).

params

1	`date`	Date
2	`lengthThreshold`	32-bit Integer
3	`languages`	String[]

result

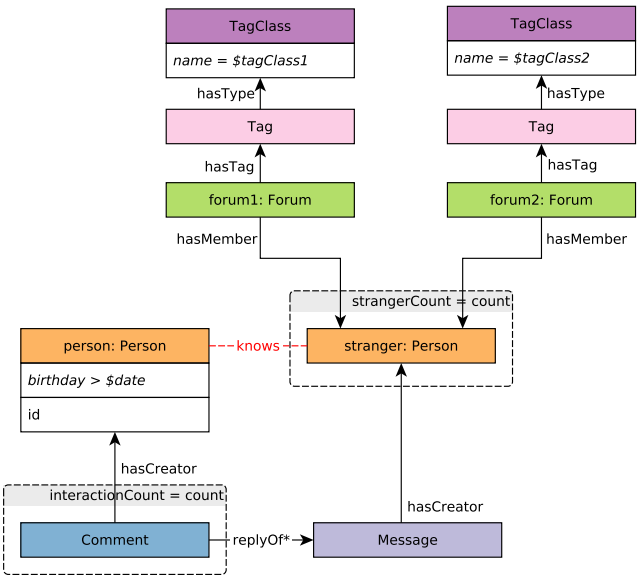
1	`messageCount`	32-bit Integer	A	Number of Messages created
2	`personCount`	32-bit Integer	A	Number of Persons with `messageCount` messages

sort

1	`personCount`	↓
2	`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 / 19

query	BI / read / 19
title	Stranger's interaction
pattern	
desc.	For all the Persons (`person`) born after a certain `date`, find all the `strangers` they interacted with, where `strangers` are Persons that do not know `person`. There is no restriction on the date that `strangers` were born. (Of course, `person` and `stranger` are required to be two different Persons.) Consider only `strangers` that are • members of Forums tagged with a Tag with a (direct) type of `tagClass1` and • members of Forums tagged with a Tag with a (direct) type of `tagClass2`. The Tags may be attached to the same Forum or they may be attached to different Forums. Interaction is defined as follows: the `person` has replied to a Message by the `stranger` B (the reply might be a transitive one). For each `person`, count the number of `strangers` they interacted with (`strangerCount`) and total number of times they interacted with them (`interactionCount`).

params

1	`date`	Date
2	`tagClass1`	String
3	`tagClass2`	String

result

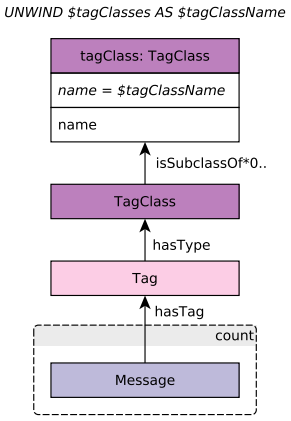
1	`person.id`	64-bit Integer	R
2	`strangerCount`	32-bit Integer	A
3	`interactionCount`	32-bit Integer	A

sort

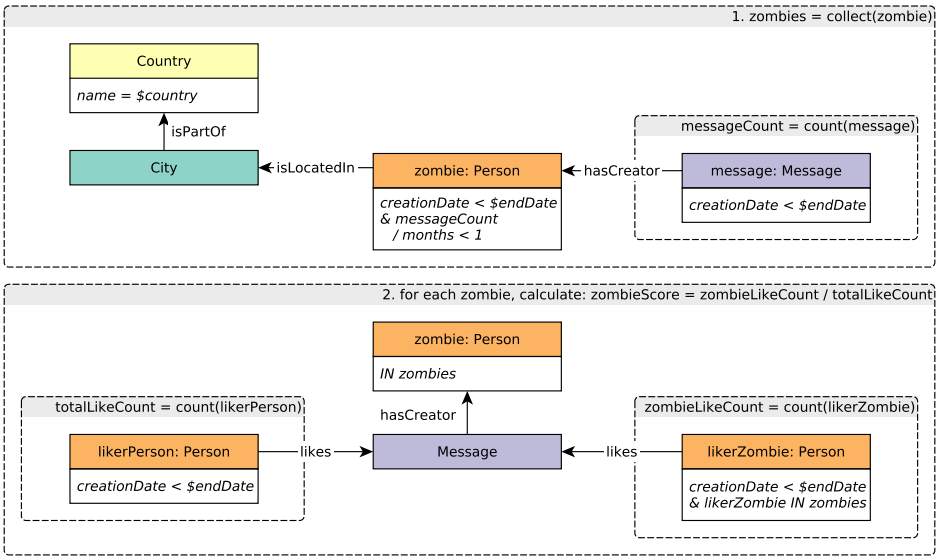
1	`interactionCount`	↓
2	`person.id`	↑

limit	100
CPs	1.1, 1.3, 2.1, 2.3, 2.4, 3.3, 5.1, 7.3, 7.4, 8.1, 8.5

BI / read / 20

query	BI / read / 20
title	High-level topics
pattern	
desc.	For all given TagClasses, count number of Messages that have a Tag that belongs to that TagClass or any of its children (all descendants through a transitive relation).
params	1 `tagClasses` String[]
result	1 `tagClass.name` String R The TagClass of the root 2 `messageCount` 32-bit Integer A
sort	1 `messageCount` ↓ 2 `tagClass.name` ↑
limit	100
CPs	1.4, 2.1, 6.1, 8.1

BI / read / 21

query	BI / read / 21
title	Zombies in a country
pattern	
desc.	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). For each `zombie`, calculate the following: • `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. For both `zombieLikeCount` and `totalLikeCount`, only consider likes received from profiles that were created before the given `endDate`.

params

1	`country`	String
2	`endDate`	Date

result

1	`zombie.id`	64-bit Integer	R	
2	`zombieLikeCount`	32-bit Integer	A	
3	`totalLikeCount`	32-bit Integer	A	
4	`zombieScore`	64-bit Float	A	`zombieLikeCount` / `totalLikeCount`

sort

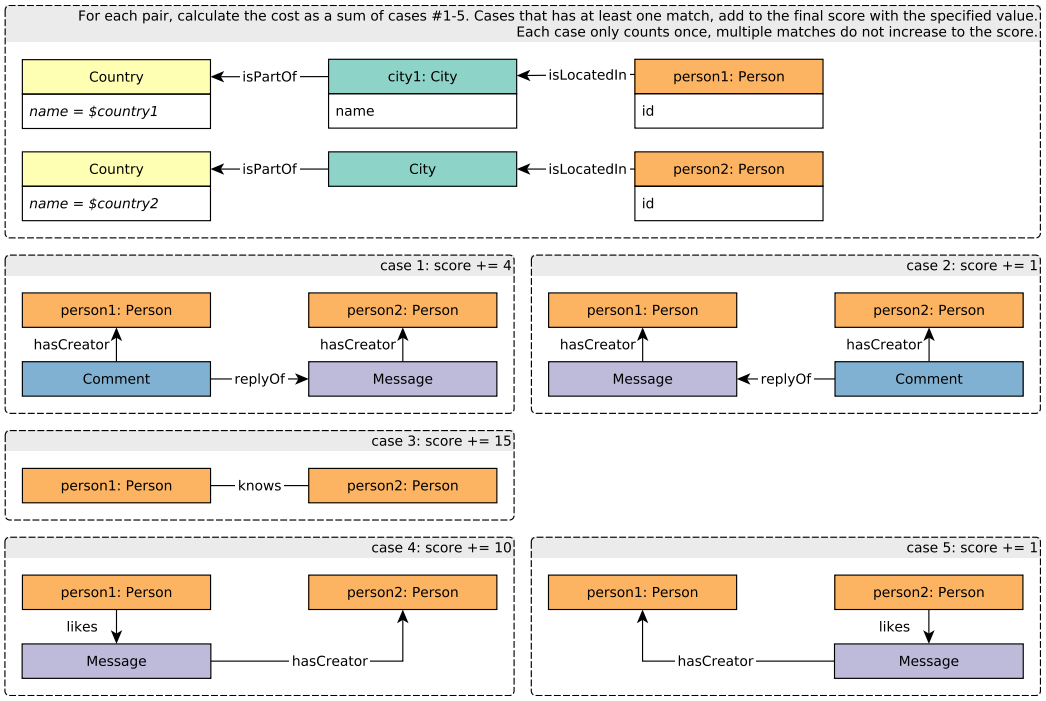
1	`zombieScore`	↓
2	`zombie.id`	↑

limit	100
CPs	1.2, 2.1, 2.3, 2.4, 3.2, 3.3, 5.1, 5.3, 8.2, 8.4, 8.5

BI / read / 22

query	BI / read / 22
title	International dialog

pattern



desc.

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`.

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 Comment 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`

Consequently, the maximum score a pair can obtain is: `4 + 1 + 15 + 10 + 1 = 31`.
To break ties, order by (1) `person1.id` ascending and (2) `person2.id` ascending.

params

1	country1	String
2	country2	String

result

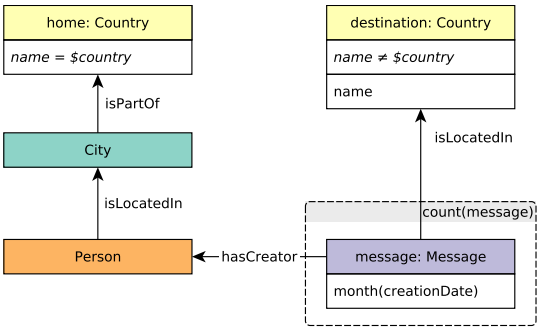
1	person1.id	64-bit Integer	R
2	person2.id	64-bit Integer	R
3	city1.name	String	R
4	score	32-bit Integer	C

sort

1	score	↓
2	person1.id	↑
3	person2.id	↑

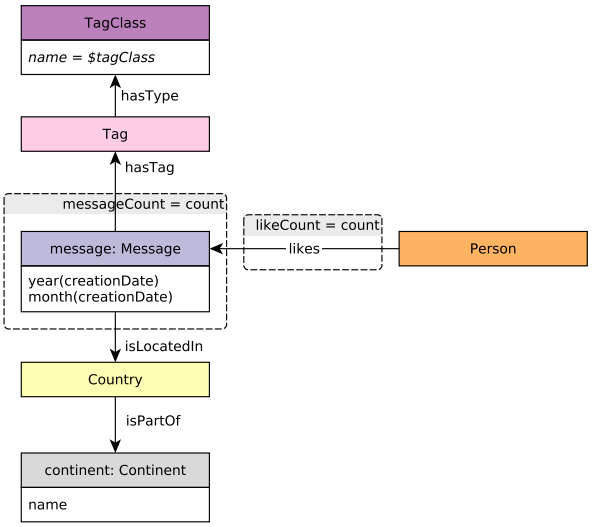
CPs	1.3, 1.4, 2.1, 3.1, 3.3, 5.1, 5.2, 5.3, 8.3, 8.4

BI / read / 23

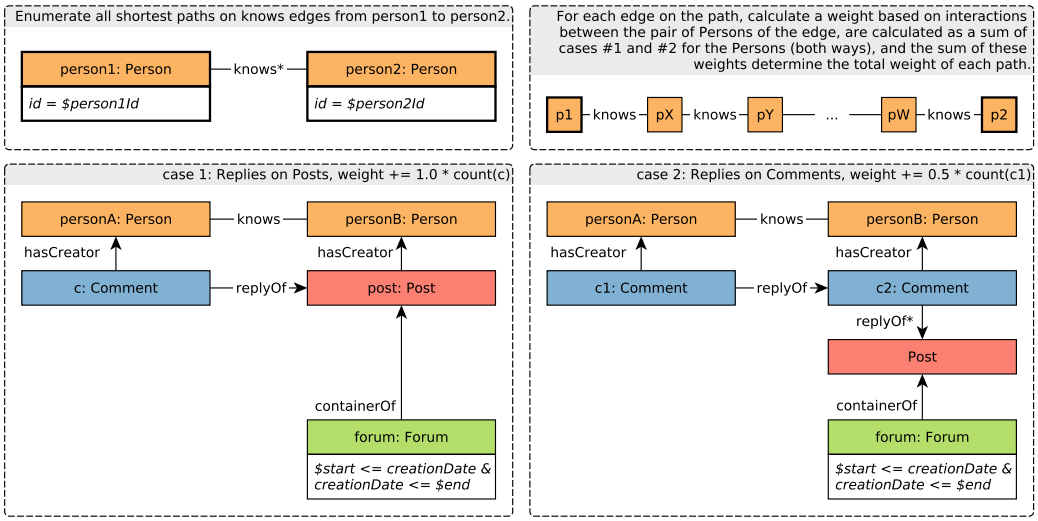
BI 1	query	BI / read / 23
BI 2	title	Holiday destinations
BI 3 BI 4 BI 5 BI 6 BI 7 BI 8 BI 9	pattern	
BI 10 BI 11 BI 12	desc.	Count the Messages of all residents of a given Country (`home`), where the message was written abroad. Group the messages by `month` and `destination`. A Message was written abroad if it is located in a Country (`destination`) different than `home`.
BI 13 BI 14 BI 15 BI 16	params	1 `country` String
BI 17 BI 18 BI 19 BI 20 BI 21	result	1 `messageCount` 32-bit Integer A The number of Messages in each group 2 `destination.name` String R The name of the destination Country 3 `month` 32-bit Integer C month(message.creationDate)
BI 22 BI 23 BI 24 BI 25	sort	1 `messageCount` ↓ 2 `desination.name` ↑ 3 `month` ↑
	limit	100
	CPs	1.4, 2.3, 2.4, 3.3, 4.3, 8.5

BI / read / 24

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
BI 21
BI 22
BI 23
BI 24
BI 25

query	BI / read / 24
title	Messages by topic and continent
pattern	
desc.	Find all Messages tagged with a Tag that has the (direct) type of the given `tagClass`. Count all Messages and their likes grouped by Continent, `year`, and `month`.
params	1 `tagClass` String
result	1 `messageCount` 32-bit Integer A 2 `likeCount` 32-bit Integer A 3 `year` 32-bit Integer C `year(message.creationDate)` 4 `month` 32-bit Integer C `month(message.creationDate)` 5 `continent.name` String R
sort	1 `year` ↑ 2 `month` ↑ 3 `continent.name` ↓
limit	100
CPs	1.4, 2.1, 2.3, 2.4, 3.2, 4.3, 8.5

BI / read / 25

query	BI / read / 25
title	Trusted connection paths through forums created in a given timeframe
pattern	
desc.	Given two Persons, find all (unweighted) shortest paths between these two Persons, in the sub-graph induced by the knows relationship. 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. The weight for a pair of Persons is calculated based on their interactions: • 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. Only consider Messages that were created in a Forum that was created within the timeframe `[startDate, endDate]`. Note that for Comments, the containing Forum is that of the Post that the comment (transitively) replies to. Return all paths with the Person ids ordered by their weights descending.

params

1	`person1Id`	64-bit Integer	
2	`person2Id`	64-bit Integer	
3	`startDate`	Date	
4	`endDate`	Date	

result

1	`person.id`	64-bit Integer[]	R	Identifiers representing an ordered sequence of the Persons in the path weight
2	`weight`	64-bit Double	R	

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, 8.1, 8.2, 8.3, 8.4, 8.5, 8.6