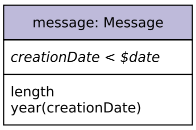
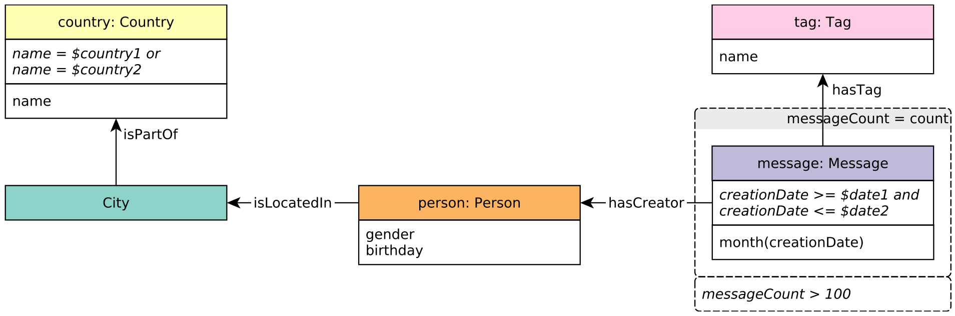
BI 1
BI 2
BI 3
BI 4
BI 5
BI 6
BI 7
BI 8
BI 9
BI 10
BI 11
BI 12
BI 13
BI 14
BI 15
BI 16
BI 17
BI 18
BI 19
BI 20
BI 21
BI 22
BI 23
BI 24
BI 25

BI / read / 1

<table>
<tr><td>query</td><td>BI / read / 1</td></tr>
<tr><td>title</td><td>Posting summary</td></tr>
<tr><td>pattern</td><td>
</td></tr>
<tr><td>desc.</td><td>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)</td></tr>
<tr><td>params</td><td><table>
<tr><td>1</td><td>date</td><td>Date</td><td></td></tr>
</table></td></tr>
<tr><td>result</td><td><table>
<tr><td>1</td><td>year</td><td>32-bit Integer</td><td>R</td><td>year(message.creationDate)</td></tr>
<tr><td>2</td><td>isComment</td><td>Boolean</td><td>M</td><td>true for Comments, false for Posts</td></tr>
<tr><td>3</td><td>lengthCategory</td><td>String</td><td>C</td><td>0 for short, 1 for one-liner, 2 for tweet, 3 for long</td></tr>
<tr><td>4</td><td>messageCount</td><td>32-bit Integer</td><td>A</td><td>Total number of Messages in that group</td></tr>
<tr><td>5</td><td>averageMessageLength</td><td>32-bit Integer</td><td>A</td><td>Average length of the Message content in that group</td></tr>
<tr><td>6</td><td>sumMessageLength</td><td>32-bit Integer</td><td>A</td><td>Sum of all Message content lengths</td></tr>
<tr><td>7</td><td>percentageOfMessages</td><td>32-bit Float</td><td>A</td><td>Number of Messages in group as a percentage of all messages created before the given date</td></tr>
</table></td></tr>
<tr><td>sort</td><td><table>
<tr><td>1</td><td>year</td><td>↓</td><td></td></tr>
<tr><td>2</td><td>isComment</td><td>↑</td><td>false < true, i.e. the ordering puts Posts first, and Comments second</td></tr>
<tr><td>3</td><td>lengthCategory</td><td>↑</td><td>order based on the length of the category, 0 (short), 1 (one liner), etc.</td></tr>
</table></td></tr>
<tr><td>CPs</td><td>1.2, 3.2, 4.1, 8.5</td></tr>
</table>

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 / 2

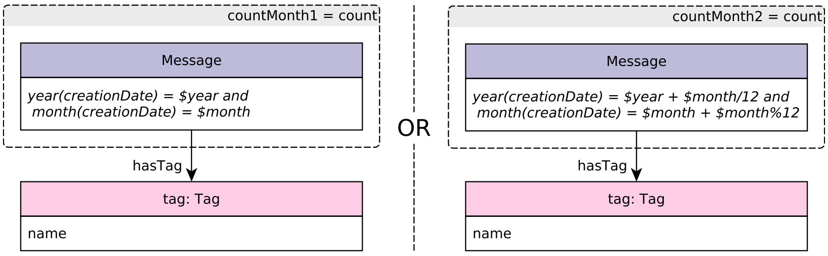
<table>
<tr><td>query</td><td>BI / read / 2</td></tr>
<tr><td>title</td><td>Top tags for country, age, gender, time</td></tr>
<tr><td>pattern</td><td>
</td></tr>
<tr><td>desc.</td><td>Select all Messages created in the range of [startDate, endDate] by Persons located in country1 or country2. Select the 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.</td></tr>
<tr><td>params</td><td><table>
<tr><td>1</td><td>startDate</td><td>Date</td><td></td></tr>
<tr><td>2</td><td>endDate</td><td>Date</td><td></td></tr>
<tr><td>3</td><td>country1</td><td>String</td><td></td></tr>
<tr><td>4</td><td>country2</td><td>String</td><td></td></tr>
</table></td></tr>
<tr><td>result</td><td><table>
<tr><td>1</td><td>country.name</td><td>String</td><td>R</td><td></td></tr>
<tr><td>2</td><td>messageMonth</td><td>32-bit Integer</td><td>R</td><td>1–12</td></tr>
<tr><td>3</td><td>person.gender</td><td>String</td><td>R</td><td>male or female</td></tr>
<tr><td>4</td><td>ageGroup</td><td>32-bit Integer</td><td>C</td><td></td></tr>
<tr><td>5</td><td>tag.name</td><td>String</td><td>R</td><td></td></tr>
<tr><td>6</td><td>messageCount</td><td>64-bit Integer</td><td>A</td><td>The number of messages in the group</td></tr>
</table></td></tr>
<tr><td>sort</td><td><table>
<tr><td>1</td><td>messageCount</td><td>↓</td><td></td></tr>
<tr><td>2</td><td>tag.name</td><td>↑</td><td></td></tr>
<tr><td>3</td><td>ageGroup</td><td>↑</td><td></td></tr>
<tr><td>4</td><td>person.gender</td><td>↑</td><td></td></tr>
<tr><td>5</td><td>messageMonth</td><td>↑</td><td></td></tr>
<tr><td>6</td><td>country.name</td><td>↑</td><td></td></tr>
</table></td></tr>
<tr><td>limit</td><td>100</td></tr>
<tr><td>CPs</td><td>1.1, 1.2, 1.3, 2.1, 2.3, 3.1, 3.2, 8.2, 8.5</td></tr>
</table>

BI / read / 3

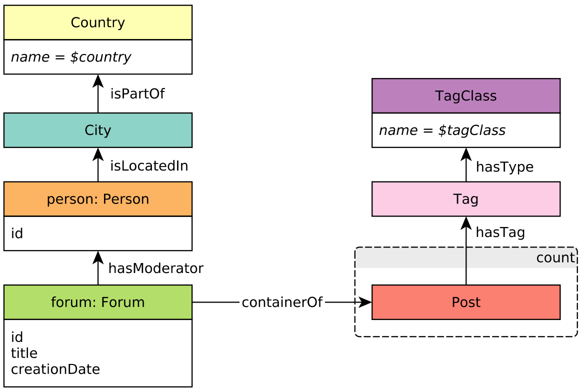
BI 1
BI 2
BI 3
BI 4
BI 5
BI 6
BI 7
BI 8
BI 9
BI 10
BI 11
BI 12
BI 13
BI 14
BI 15
BI 16
BI 17
BI 18
BI 19
BI 20
BI 21
BI 22
BI 23
BI 24
BI 25

<table>
<tr><td>query</td><td>BI / read / 3</td></tr>
<tr><td>title</td><td>Tag evolution</td></tr>
<tr><td>pattern</td><td>
countMonth1 = count
Message
year(creationDate) = $year and
month(creationDate) = $month
hasTag
tag: Tag
name
OR
countMonth2 = count
Message
year(creationDate) = $year + $month/12 and
month(creationDate) = $month + $month%12
hasTag
tag: Tag
name

</td></tr>
<tr><td>desc.</td><td>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.</td></tr>
<tr><td>params</td><td><table>
<tr><td>1</td><td>year</td><td>32-bit Integer</td><td></td></tr>
<tr><td>2</td><td>month</td><td>32-bit Integer</td><td></td></tr>
</table></td></tr>
<tr><td>result</td><td><table>
<tr><td>1</td><td>tag.name</td><td>String</td><td>R</td><td></td></tr>
<tr><td>2</td><td>countMonth1</td><td>32-bit Integer</td><td>A</td><td>Occurrences of the tag during the given year and month</td></tr>
<tr><td>3</td><td>countMonth2</td><td>32-bit Integer</td><td>A</td><td>Occurrences of the tag during the next month after the given year and month</td></tr>
<tr><td>4</td><td>diff</td><td>32-bit Integer</td><td>A</td><td>Absolute difference of countMonth1 and countMonth2</td></tr>
</table></td></tr>
<tr><td>sort</td><td><table>
<tr><td>1</td><td>diff</td><td>↓</td><td></td></tr>
<tr><td>2</td><td>tag.name</td><td>↑</td><td></td></tr>
</table></td></tr>
<tr><td>limit</td><td>100</td></tr>
<tr><td>CPs</td><td>2.4, 3.1, 3.2, 4.1, 4.3, 5.3, 6.1, 8.2, 8.5</td></tr>
</table>

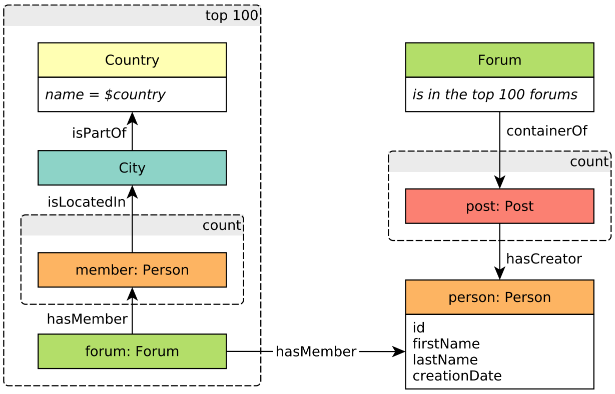
BI / read / 4

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	 Country name = $country isPartOf City isLocatedIn person: Person id hasModerator forum: Forum id title creationDate containerOf TagClass name = $tagClass hasType Tag hasTag count Post
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

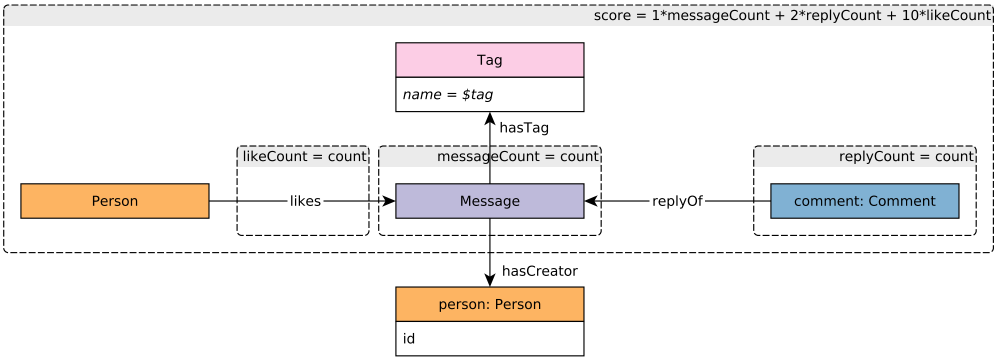
BI / read / 5

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	 top 100 Country name = $country isPartOf City isLocatedIn count member: Person hasMember forum: Forum hasMember Forum is in the top 100 forums containerOf count post: Post hasCreator person: Person id firstName lastName creationDate
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

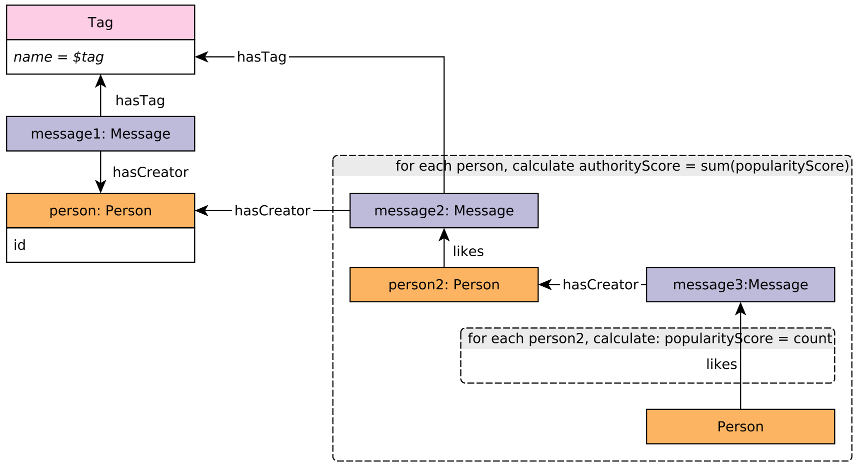
BI 1
BI 2
BI 3
BI 4
BI 5
BI 6
BI 7
BI 8
BI 9
BI 10
BI 11
BI 12
BI 13
BI 14
BI 15
BI 16
BI 17
BI 18
BI 19
BI 20
BI 21
BI 22
BI 23
BI 24
BI 25

BI / read / 6

query	BI / read / 6
title	Most active posters of a given topic
pattern	 score = 1*messageCount + 2*replyCount + 10*likeCount Tag name = $tag hasTag likeCount = count messageCount = count replyCount = count Person likes Message replyOf comment: Comment hasCreator person: Person id
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

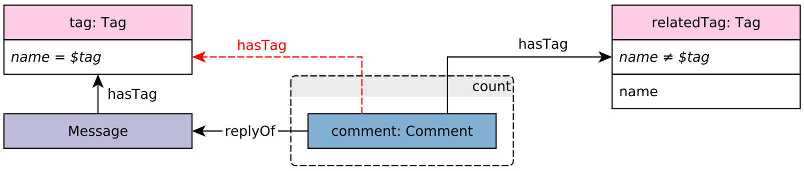
BI 1
BI 2
BI 3
BI 4
BI 5
BI 6
BI 7
BI 8
BI 9
BI 10
BI 11
BI 12
BI 13
BI 14
BI 15
BI 16
BI 17
BI 18
BI 19
BI 20
BI 21
BI 22
BI 23
BI 24
BI 25

BI / read / 7

query	BI / read / 7
title	Most authoritative users on a given topic
pattern	 Tag name = $tag hasTag hasTag message1: Message hasCreator person: Person id hasCreator for each person, calculate authorityScore = sum(popularityScore) message2: Message likes person2: Person hasCreator message3:Message for each person2, calculate: popularityScore = count likes Person
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

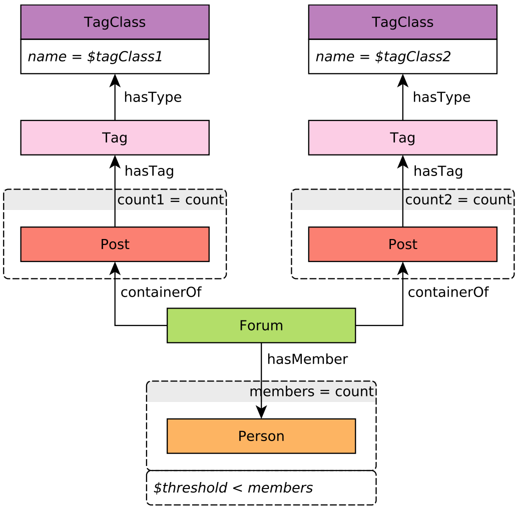
BI 1
BI 2
BI 3
BI 4
BI 5
BI 6
BI 7
BI 8
BI 9
BI 10
BI 11
BI 12
BI 13
BI 14
BI 15
BI 16
BI 17
BI 18
BI 19
BI 20
BI 21
BI 22
BI 23
BI 24
BI 25

BI / read / 8

query	BI / read / 8
title	Related topics
pattern	 tag: Tag name = $tag hasTag hasTag Message replyOf comment: Comment count hasTag relatedTag: Tag name ≠ $tag name
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

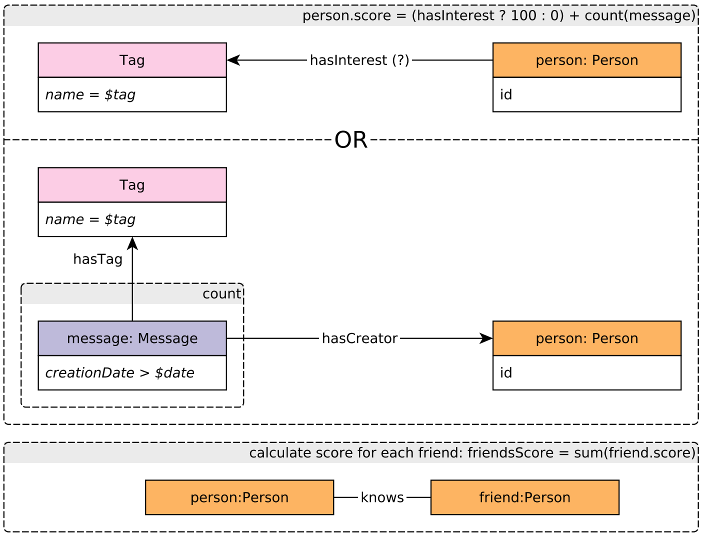
BI / read / 9

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 / 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				

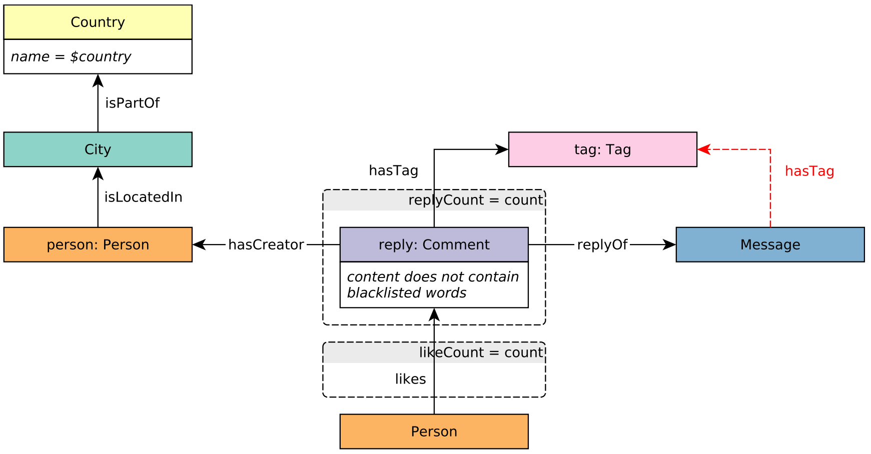
BI / read / 10

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	 person.score = (hasInterest ? 100 : 0) + count(message) Tag name = $tag hasInterest (?) person: Person id OR Tag name = $tag hasTag count message: Message creationDate > $date hasCreator person: Person id calculate score for each friend: friendsScore = sum(friend.score) person:Person knows friend:Person
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

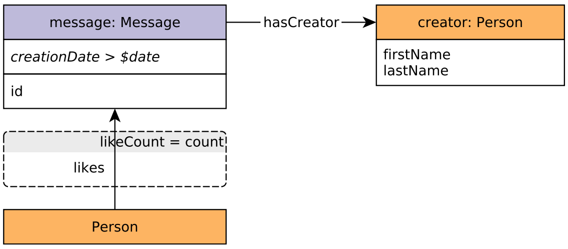
BI 1
BI 2
BI 3
BI 4
BI 5
BI 6
BI 7
BI 8
BI 9
BI 10
BI 11
BI 12
BI 13
BI 14
BI 15
BI 16
BI 17
BI 18
BI 19
BI 20
BI 21
BI 22
BI 23
BI 24
BI 25

<table>
<tr><td>query</td><td>BI / read / 11</td></tr>
<tr><td>title</td><td>Unrelated replies</td></tr>
<tr><td>pattern</td><td>


</td></tr>
<tr><td>desc.</td><td>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.</td></tr>
<tr><td>params</td><td><table>
<tr><td>1</td><td>country</td><td>String</td><td></td></tr>
<tr><td>2</td><td>blacklist</td><td>String[]</td><td></td></tr>
</table></td></tr>
<tr><td>result</td><td><table>
<tr><td>1</td><td>person.id</td><td>64-bit Integer</td><td>R</td><td></td></tr>
<tr><td>2</td><td>tag.name</td><td>String</td><td>R</td><td></td></tr>
<tr><td>3</td><td>likeCount</td><td>32-bit Integer</td><td>A</td><td>The count of likes to replies with that Tag</td></tr>
<tr><td>4</td><td>replyCount</td><td>32-bit Integer</td><td>A</td><td>The count of replies with that Tag</td></tr>
</table></td></tr>
<tr><td>sort</td><td><table>
<tr><td>1</td><td>likeCount</td><td>↓</td><td></td></tr>
<tr><td>2</td><td>person.id</td><td>↑</td><td></td></tr>
<tr><td>3</td><td>tag.name</td><td>↑</td><td></td></tr>
</table></td></tr>
<tr><td>limit</td><td>100</td></tr>
<tr><td>CPs</td><td>1.1, 2.1, 2.2, 2.3, 3.1, 3.2, 6.1, 8.1, 8.3</td></tr>
</table>

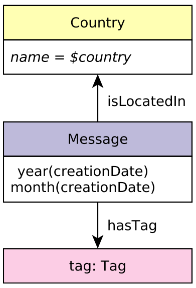
BI / read / 12

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

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

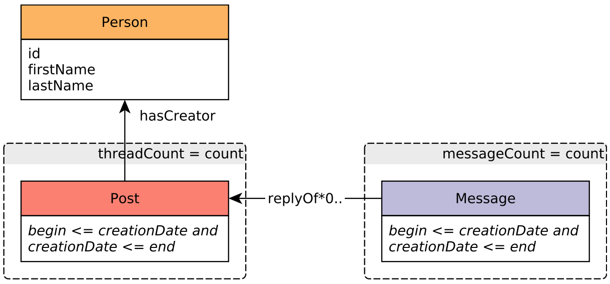
BI / read / 13

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

<table>
<tr><td>query</td><td>BI / read / 13</td></tr>
<tr><td>title</td><td>Popular tags per month in a country</td></tr>
<tr><td>pattern</td><td>
</td></tr>
<tr><td>desc.</td><td>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.</td></tr>
<tr><td>params</td><td><table><tr><td>1</td><td>country</td><td>String</td></tr></table></td></tr>
<tr><td>result</td><td><table>
<tr><td>1</td><td>year</td><td>32-bit Integer</td><td>C</td><td>year(message.creationDate)</td></tr>
<tr><td>2</td><td>month</td><td>32-bit Integer</td><td>C</td><td>month(message.creationDate)</td></tr>
<tr><td>3</td><td>popularTags</td><td>TagPairs</td><td>C</td><td>(tag.name [String], popularity [32-bit Integer]) pairs, sorted descending by popularity, then ascending by tag.name</td></tr>
</table></td></tr>
<tr><td>sort</td><td><table>
<tr><td>1</td><td>year</td><td>↓</td></tr>
<tr><td>2</td><td>month</td><td>↑</td></tr>
</table></td></tr>
<tr><td>limit</td><td>100</td></tr>
<tr><td>CPs</td><td>1.2, 2.2, 2.3, 3.2, 6.1, 8.3, 8.5</td></tr>
</table>

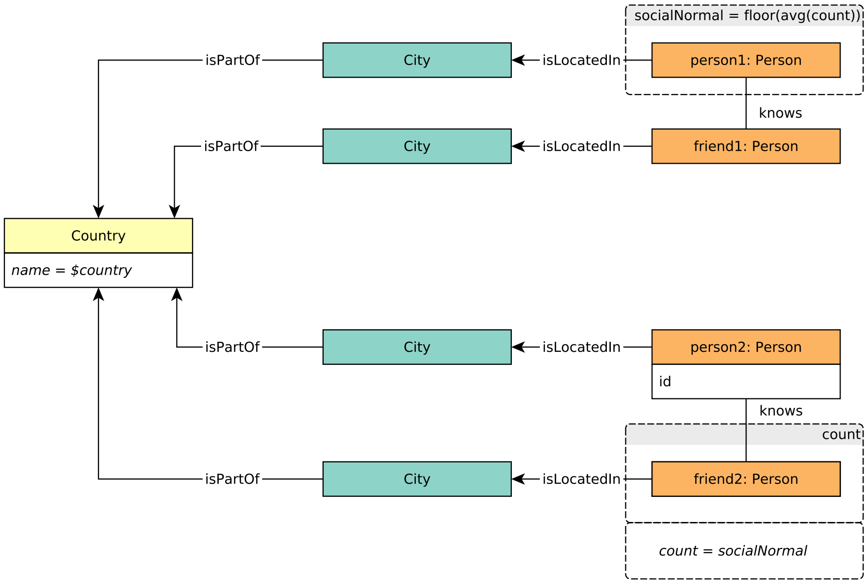
BI / read / 14

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

<table>
<tr><td>query</td><td>BI / read / 14</td></tr>
<tr><td>title</td><td>Top thread initiators</td></tr>
<tr><td>pattern</td><td>
</td></tr>
<tr><td>desc.</td><td>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.</td></tr>
<tr><td>params</td><td><table>
<tr><td>1</td><td>startDate</td><td>Date</td><td></td></tr>
<tr><td>2</td><td>endDate</td><td>Date</td><td></td></tr>
</table></td></tr>
<tr><td>result</td><td><table>
<tr><td>1</td><td>person.id</td><td>64-bit Integer</td><td>R</td><td></td></tr>
<tr><td>2</td><td>person.firstName</td><td>String</td><td>R</td><td></td></tr>
<tr><td>3</td><td>person.lastName</td><td>String</td><td>R</td><td></td></tr>
<tr><td>4</td><td>threadCount</td><td>32-bit Integer</td><td>A</td><td>The number of Posts created by that Person (the number of threads initiated)</td></tr>
<tr><td>5</td><td>messageCount</td><td>32-bit Integer</td><td>A</td><td>The number of Messages created in all the threads this Person initiated</td></tr>
</table></td></tr>
<tr><td>sort</td><td><table>
<tr><td>1</td><td>messageCount</td><td>↓</td><td></td></tr>
<tr><td>2</td><td>person.id</td><td>↑</td><td></td></tr>
</table></td></tr>
<tr><td>limit</td><td>100</td></tr>
<tr><td>CPs</td><td>1.2, 2.2, 2.3, 3.2, 7.2, 7.3, 7.4, 8.1, 8.5</td></tr>
</table>

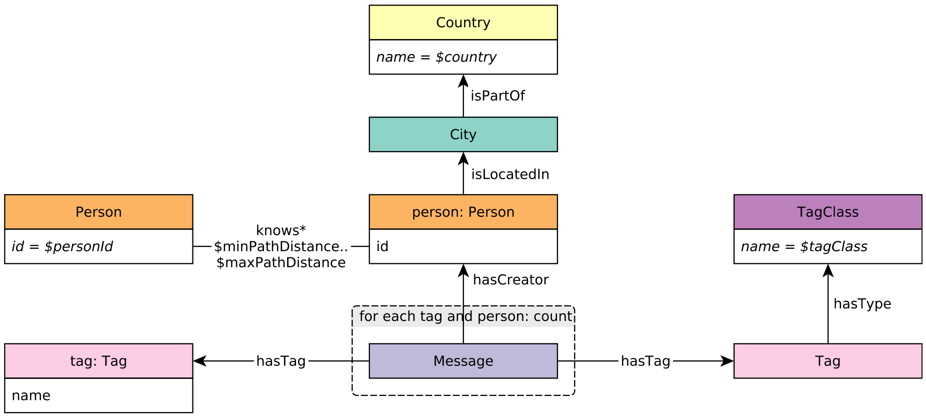
BI / read / 15

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 / 15
title	Social normals
pattern	 socialNormal = floor(avg(count)) isPartOf City isLocatedIn person1: Person knows isPartOf City isLocatedIn friend1: Person Country name = $country isPartOf City isLocatedIn person2: Person id knows count isPartOf City isLocatedIn friend2: Person count = socialNormal
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

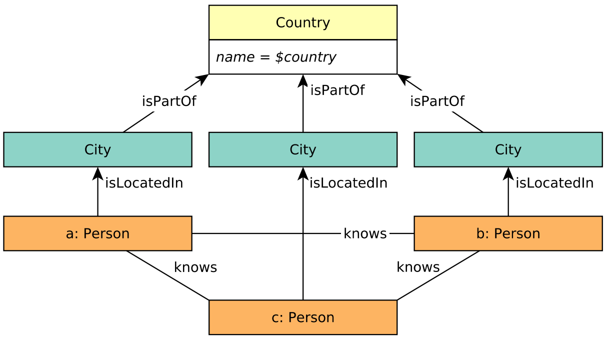
BI 1
BI 2
BI 3
BI 4
BI 5
BI 6
BI 7
BI 8
BI 9
BI 10
BI 11
BI 12
BI 13
BI 14
BI 15
BI 16
BI 17
BI 18
BI 19
BI 20
BI 21
BI 22
BI 23
BI 24
BI 25

BI / read / 16

<table>
<tr><td>query</td><td>BI / read / 16</td></tr>
<tr><td>title</td><td>Experts in social circle</td></tr>
<tr><td>pattern</td><td>
</td></tr>
<tr><td>desc.</td><td>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.)</td></tr>
<tr><td>params</td><td><table>
<tr><td>1</td><td>personId</td><td>64-bit Integer</td><td></td></tr>
<tr><td>2</td><td>country</td><td>String</td><td></td></tr>
<tr><td>3</td><td>tagClass</td><td>String</td><td></td></tr>
<tr><td>4</td><td>minPathDistance</td><td>32-bit Integer</td><td></td></tr>
<tr><td>5</td><td>maxPathDistance</td><td>32-bit Integer</td><td></td></tr>
</table></td></tr>
<tr><td>result</td><td><table>
<tr><td>1</td><td>person.id</td><td>64-bit Integer</td><td>R</td><td></td></tr>
<tr><td>2</td><td>tag.name</td><td>String</td><td>R</td><td></td></tr>
<tr><td>3</td><td>messageCount</td><td>32-bit Integer</td><td>A</td><td>Number of Messages created by that Person containing that Tag</td></tr>
</table></td></tr>
<tr><td>sort</td><td><table>
<tr><td>1</td><td>messageCount</td><td>↓</td><td></td></tr>
<tr><td>2</td><td>tag.name</td><td>↑</td><td></td></tr>
<tr><td>3</td><td>person.id</td><td>↑</td><td></td></tr>
</table></td></tr>
<tr><td>limit</td><td>100</td></tr>
<tr><td>CPs</td><td>1.2, 1.3, 2.3, 2.4, 3.3, 5.3, 7.1, 7.2, 7.3, 8.1, 8.6</td></tr>
</table>

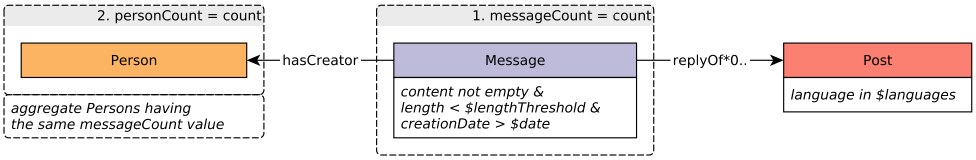
BI 1
BI 2
BI 3
BI 4
BI 5
BI 6
BI 7
BI 8
BI 9
BI 10
BI 11
BI 12
BI 13
BI 14
BI 15
BI 16
BI 17
BI 18
BI 19
BI 20
BI 21
BI 22
BI 23
BI 24
BI 25

BI / read / 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

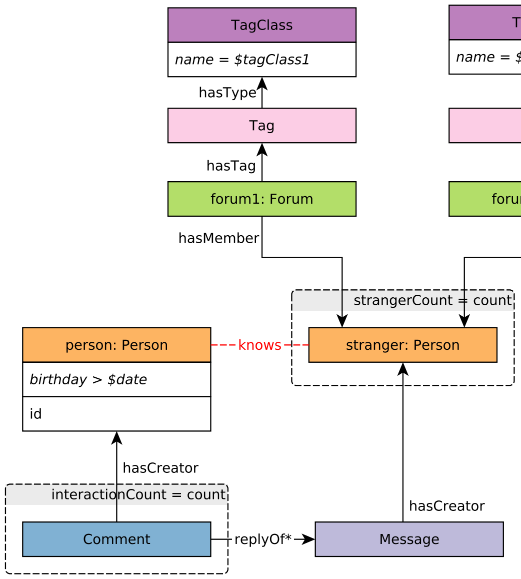
BI / read / 18

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	 2. personCount = count Person aggregate Persons having the same messageCount value hasCreator 1. messageCount = count Message content not empty & length < $lengthThreshold & creationDate > $date replyOf*0.. Post language in $languages
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 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 / 19

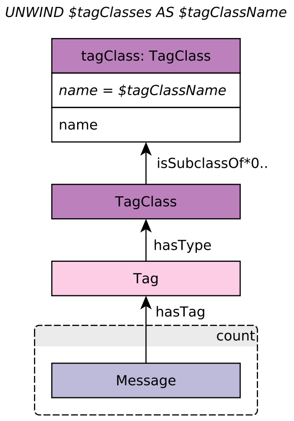
<table>
<tr><td>query</td><td>BI / read / 19</td></tr>
<tr><td>title</td><td>Stranger’s interaction</td></tr>
<tr><td>pattern</td><td>
</td></tr>
<tr><td>desc.</td><td>For all the Persons (person) born after a certain date, find all the strangers they interacted with, where strangers are Persons that 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).</td></tr>
<tr><td>params</td><td><table>
<tr><td>1</td><td>date</td><td>Date</td><td></td></tr>
<tr><td>2</td><td>tagClass1</td><td>String</td><td></td></tr>
<tr><td>3</td><td>tagClass2</td><td>String</td><td></td></tr>
</table></td></tr>
<tr><td>result</td><td><table>
<tr><td>1</td><td>person.id</td><td>64-bit Integer</td><td>R</td><td></td></tr>
<tr><td>2</td><td>strangerCount</td><td>32-bit Integer</td><td>A</td><td></td></tr>
<tr><td>3</td><td>interactionCount</td><td>32-bit Integer</td><td>A</td><td></td></tr>
</table></td></tr>
<tr><td>sort</td><td><table>
<tr><td>1</td><td>interactionCount</td><td>↓</td><td></td></tr>
<tr><td>2</td><td>person.id</td><td>↑</td><td></td></tr>
</table></td></tr>
<tr><td>limit</td><td>100</td></tr>
<tr><td>CPs</td><td>1.1, 1.3, 2.1, 2.3, 2.4, 3.3, 5.1, 7.3, 7.4, 8.1, 8.5</td></tr>
</table>

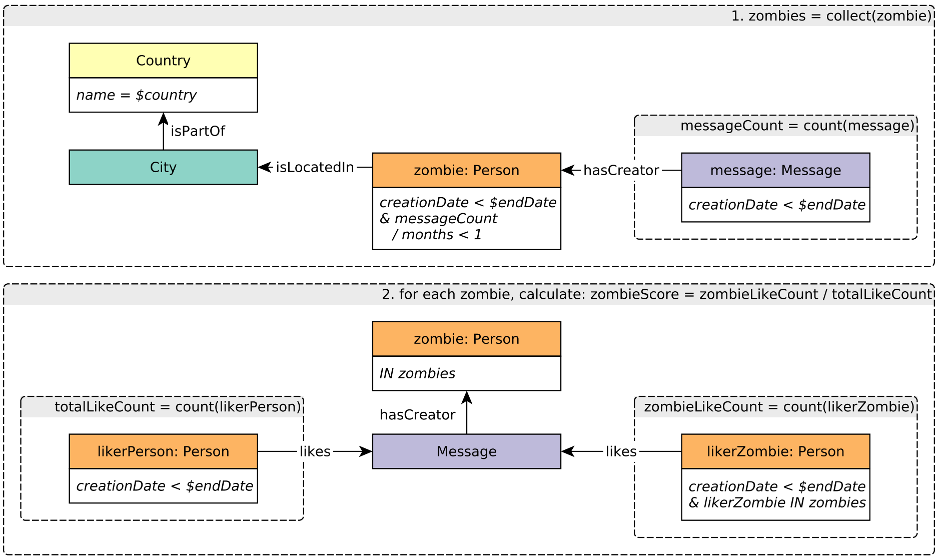
BI / read / 20

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 / 20
title	High-level topics
pattern	 UNWIND $tagClasses AS $tagClassName tagClass: TagClass name = $tagClassName name isSubclassOf*0.. TagClass hasType Tag hasTag count Message
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

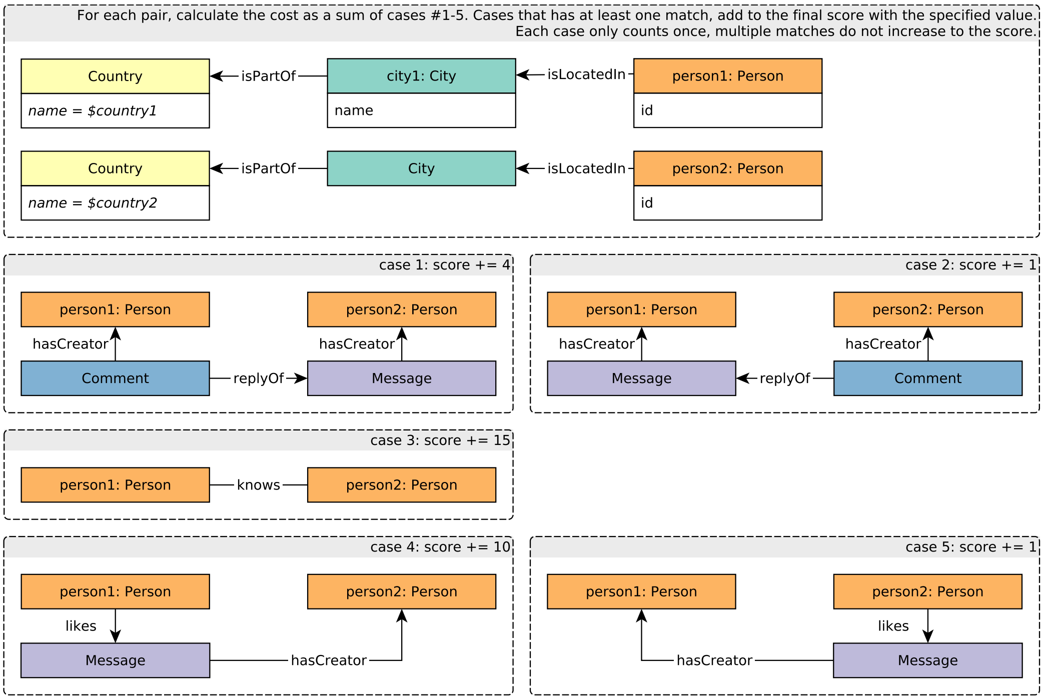
BI 1
BI 2
BI 3
BI 4
BI 5
BI 6
BI 7
BI 8
BI 9
BI 10
BI 11
BI 12
BI 13
BI 14
BI 15
BI 16
BI 17
BI 18
BI 19
BI 20
BI 21
BI 22
BI 23
BI 24
BI 25

BI / read / 21

<table>
<tr><td>query</td><td>BI / read / 21</td></tr>
<tr><td>title</td><td>Zombies in a country</td></tr>
<tr><td>pattern</td><td>

1. zombies = collect(zombie)
Country
name = $country
isPartOf
City
isLocatedIn
zombie: Person
creationDate < $endDate & messageCount / months &lt; 1
hasCreator
messageCount = count(message)
message: Message
creationDate < $endDate
2. for each zombie, calculate: zombieScore = zombieLikeCount / totalLikeCount
zombie: Person
IN zombies
hasCreator
totalLikeCount = count(likerPerson)
likerPerson: Person
creationDate < $endDate
likes
Message
likes
zombieLikeCount = count(likerZombie)
likerZombie: Person
creationDate < $endDate & likerZombie IN zombies
</td></tr>
<tr><td>desc.</td><td>
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:
<ul>
<li>zombieLikeCount: the number of likes received from other zombies.</li>
<li>totalLikeCount: the total number of likes received.</li>
<li>zombieScore: zombieLikeCount / totalLikeCount. If the value of totalLikeCount is 0, the zombieScore of the zombie should be 0.0.</li>
</ul>
For both zombieLikeCount and totalLikeCount, only consider likes received from profiles that were created before the given endDate.
</td></tr>
<tr><td>params</td><td>
<table>
<tr><td>1</td><td>country</td><td>String</td><td></td></tr>
<tr><td>2</td><td>endDate</td><td>Date</td><td></td></tr>
</table>
</td></tr>
<tr><td>result</td><td>
<table>
<tr><td>1</td><td>zombie.id</td><td>64-bit Integer</td><td>R</td><td></td></tr>
<tr><td>2</td><td>zombieLikeCount</td><td>32-bit Integer</td><td>A</td><td></td></tr>
<tr><td>3</td><td>totalLikeCount</td><td>32-bit Integer</td><td>A</td><td></td></tr>
<tr><td>4</td><td>zombieScore</td><td>64-bit Float</td><td>A</td><td>zombieLikeCount / totalLikeCount</td></tr>
</table>
</td></tr>
<tr><td>sort</td><td>
<table>
<tr><td>1</td><td>zombieScore</td><td>↓</td><td></td></tr>
<tr><td>2</td><td>zombie.id</td><td>↑</td><td></td></tr>
</table>
</td></tr>
<tr><td>limit</td><td>100</td></tr>
<tr><td>CPs</td><td>1.2, 2.1, 2.3, 2.4, 3.2, 3.3, 5.1, 5.3, 8.2, 8.4, 8.5</td></tr>
</table>

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 / 22

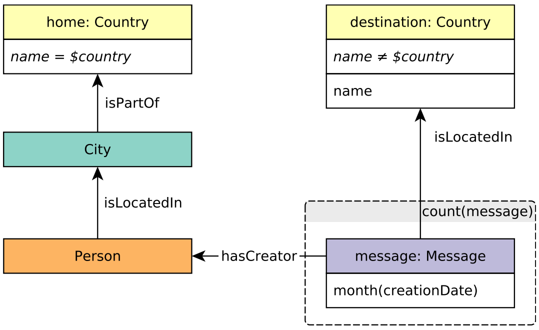
<table>
<tr><td>query</td><td>BI / read / 22</td></tr>
<tr><td>title</td><td>International dialog</td></tr>
<tr><td>pattern</td><td>
</td></tr>
<tr><td>desc.</td><td>Consider all pairs of people (person1, person2) such that one is located in a City of Country country1 and the other is located in a City 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.</td></tr>
<tr><td>params</td><td><table>
<tr><td>1</td><td>country1</td><td>String</td><td></td></tr>
<tr><td>2</td><td>country2</td><td>String</td><td></td></tr>
</table></td></tr>
<tr><td>result</td><td><table>
<tr><td>1</td><td>person1.id</td><td>64-bit Integer</td><td>R</td><td></td></tr>
<tr><td>2</td><td>person2.id</td><td>64-bit Integer</td><td>R</td><td></td></tr>
<tr><td>3</td><td>city1.name</td><td>String</td><td>R</td><td></td></tr>
<tr><td>4</td><td>score</td><td>32-bit Integer</td><td>C</td><td></td></tr>
</table></td></tr>
<tr><td>sort</td><td><table>
<tr><td>1</td><td>score</td><td>↓</td><td></td></tr>
<tr><td>2</td><td>person1.id</td><td>↑</td><td></td></tr>
<tr><td>3</td><td>person2.id</td><td>↑</td><td></td></tr>
</table></td></tr>
<tr><td>CPs</td><td>1.3, 1.4, 2.1, 3.1, 3.3, 5.1, 5.2, 5.3, 8.3, 8.4</td></tr>
</table>

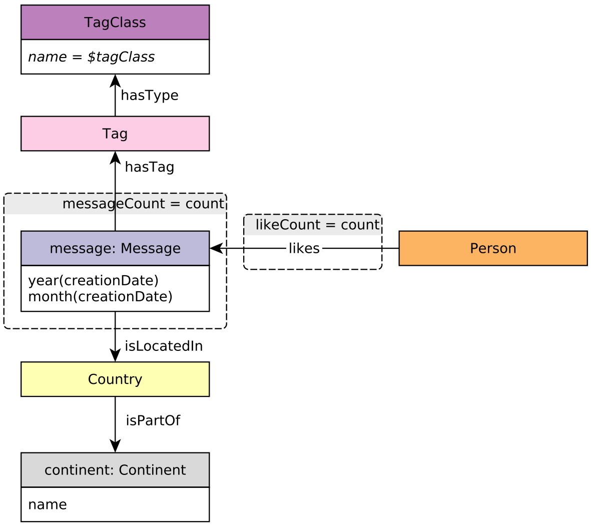
BI / read / 23

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

<table>
<tr><td>query</td><td>BI / read / 23</td></tr>
<tr><td>title</td><td>Holiday destinations</td></tr>
<tr><td>pattern</td><td>
</td></tr>
<tr><td>desc.</td><td>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.</td></tr>
<tr><td>params</td><td><table><tr><td>1</td><td>country</td><td>String</td><td></td></tr></table></td></tr>
<tr><td>result</td><td><table>
<tr><td>1</td><td>messageCount</td><td>32-bit Integer</td><td>A</td><td>The number of Messages in each group</td></tr>
<tr><td>2</td><td>destination.name</td><td>String</td><td>R</td><td>The name of the destination Country</td></tr>
<tr><td>3</td><td>month</td><td>32-bit Integer</td><td>C</td><td>month(message.creationDate)</td></tr>
</table></td></tr>
<tr><td>sort</td><td><table>
<tr><td>1</td><td>messageCount</td><td>↓</td><td></td></tr>
<tr><td>2</td><td>desination.name</td><td>↑</td><td></td></tr>
<tr><td>3</td><td>month</td><td>↑</td><td></td></tr>
</table></td></tr>
<tr><td>limit</td><td>100</td></tr>
<tr><td>CPs</td><td>1.4, 2.3, 2.4, 3.3, 4.3, 8.5</td></tr>
</table>

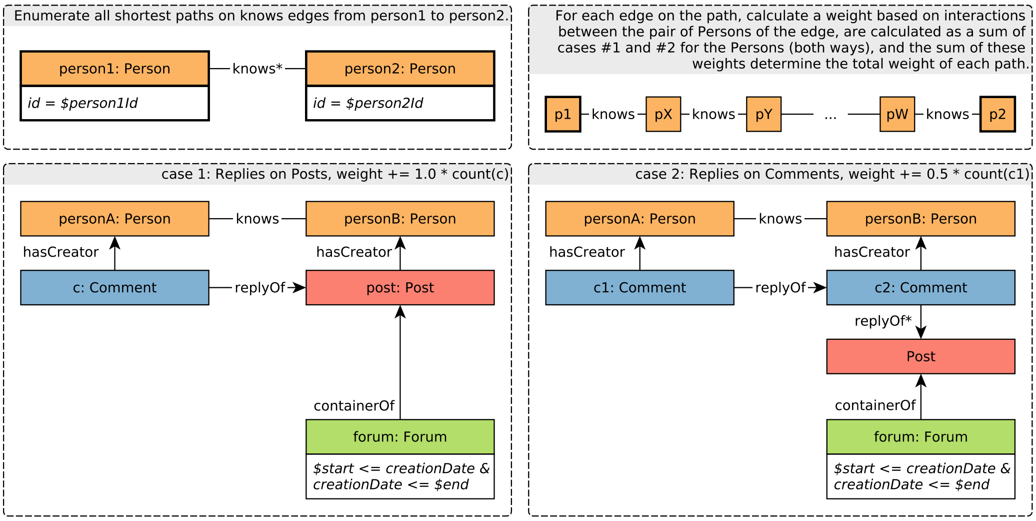
BI / read / 24

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

<table>
<tr><td>query</td><td>BI / read / 24</td></tr>
<tr><td>title</td><td>Messages by topic and continent</td></tr>
<tr><td>pattern</td><td>
</td></tr>
<tr><td>desc.</td><td>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.</td></tr>
<tr><td>params</td><td><table><tr><td>1</td><td>tagClass</td><td>String</td><td></td></tr></table></td></tr>
<tr><td>result</td><td><table>
<tr><td>1</td><td>messageCount</td><td>32-bit Integer</td><td>A</td><td></td></tr>
<tr><td>2</td><td>likeCount</td><td>32-bit Integer</td><td>A</td><td></td></tr>
<tr><td>3</td><td>year</td><td>32-bit Integer</td><td>C</td><td>year(message.creationDate)</td></tr>
<tr><td>4</td><td>month</td><td>32-bit Integer</td><td>C</td><td>month(message.creationDate)</td></tr>
<tr><td>5</td><td>continent.name</td><td>String</td><td>R</td><td></td></tr>
</table></td></tr>
<tr><td>sort</td><td><table>
<tr><td>1</td><td>year</td><td>↑</td><td></td></tr>
<tr><td>2</td><td>month</td><td>↑</td><td></td></tr>
<tr><td>3</td><td>continent.name</td><td>↓</td><td></td></tr>
</table></td></tr>
<tr><td>limit</td><td>100</td></tr>
<tr><td>CPs</td><td>1.4, 2.1, 2.3, 2.4, 3.2, 4.3, 8.5</td></tr>
</table>

BI / read / 25

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

<table>
<tr><td>query</td><td>BI / read / 25</td></tr>
<tr><td>title</td><td>Trusted connection paths through forums created in a given timeframe</td></tr>
<tr><td>pattern</td><td>
Enumerate all shortest paths on knows edges from person1 to person2.
person1: Person
id = $person1Id
knows*
person2: Person
id = $person2Id
For each edge on the path, calculate a weight based on interactions between the pair of Persons of the edge, are 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.
p1
knows
pX
knows
pY
...
pW
knows
p2
case 1: Replies on Posts, weight += 1.0 * count(c)
personA: Person
knows
personB: Person
hasCreator
hasCreator
c: Comment
replyOf
post: Post
containerOf
forum: Forum
$start <= creationDate & creationDate <= $end
case 2: Replies on Comments, weight += 0.5 * count(c1)
personA: Person
knows
personB: Person
hasCreator
hasCreator
c1: Comment
replyOf
c2: Comment
replyOf*
Post
containerOf
forum: Forum
$start <= creationDate & creationDate <= $end

</td></tr>
<tr><td>desc.</td><td>Given two Persons, find all (unweighted) shortest paths between these two Persons, in the subgraph 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.</td></tr>
<tr><td>params</td><td><table>
<tr><td>1</td><td>person1Id</td><td>64-bit Integer</td><td></td></tr>
<tr><td>2</td><td>person2Id</td><td>64-bit Integer</td><td></td></tr>
<tr><td>3</td><td>startDate</td><td>Date</td><td></td></tr>
<tr><td>4</td><td>endDate</td><td>Date</td><td></td></tr>
</table></td></tr>
<tr><td>result</td><td><table>
<tr><td>1</td><td>person.id</td><td>64-bit Integer[]</td><td>R</td><td>Identifiers representing an ordered sequence of the Persons in the path weight</td></tr>
<tr><td>2</td><td>weight</td><td>64-bit Double</td><td>R</td><td></td></tr>
</table></td></tr>
<tr><td>sort</td><td><table>
<tr><td>1</td><td>weight</td><td>↓</td><td>The order of paths with the same weight is unspecified</td></tr>
<tr><td>2</td><td>personIds</td><td>↑</td><td>The ids in the paths are used for lexicographical sorting</td></tr>
</table></td></tr>
<tr><td>CPs</td><td>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</td></tr>
</table>