

1 BUSINESS INTELLIGENCE WORKLOAD

The Business Intelligence (BI) workload is the SNB’s analytical (OLAP) workload. As such, it defines complex read queries that touch a significant portion of the data (see Section 1.4). Additionally, it defines daily batches of updates over a 33-day period (see Section 1.5 for inserts and Section 1.6 for deletes).

Related Publications

The BI workload was published in PVLDB 2022 [17].

Related Software Components

- Datagen (Spark-based): https://github.com/ldbc/ldbc_snb_datagen_spark
- Driver and reference implementations: https://github.com/ldbc/ldbc_snb_bi

1.1 Overview

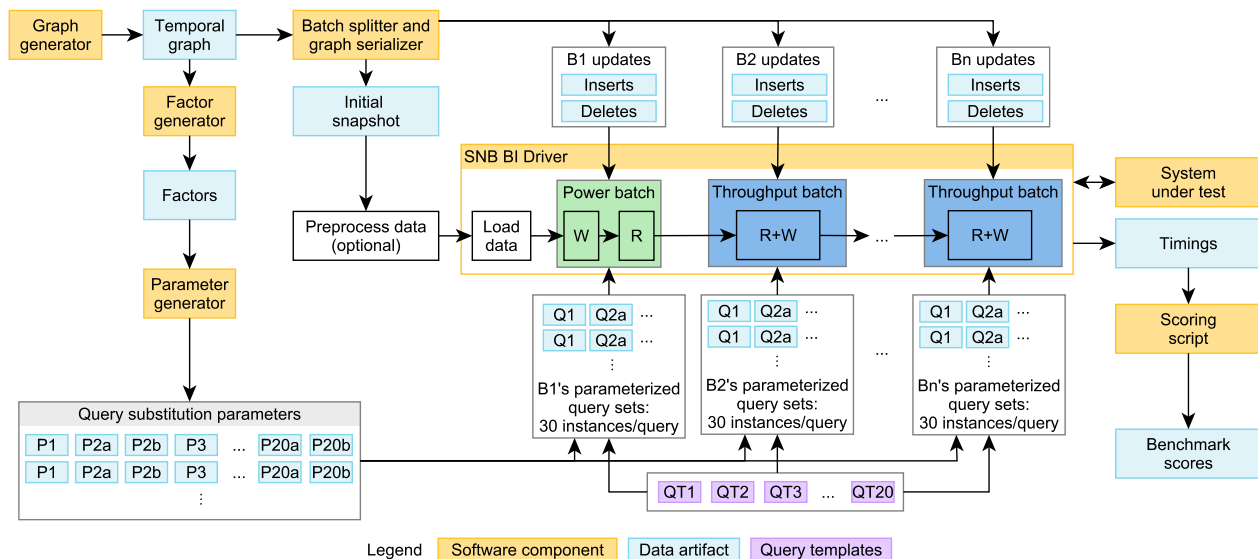


Figure 1.1: Main software components and data artifacts of the benchmark and their connection to the workflow executed by the BI benchmark driver.

An overview of the BI workload is shown in Figure 1.1. The rules for auditing workload implementations are given in ??.

1.2 Read Query Templates

SNB BI consists of 20 parameterized *read query templates*, referred to as *queries*. These search for graph patterns (often implying join-heavy operations on many-to-many edges), traverse hierarchies, and compute cheapest paths (a.k.a. weighted shortest paths). Additionally, they include filtering, grouping, aggregation, and sorting operators. While all queries explore a large portion of the graph, they only return the top-*k* (typically 20 or 100) results, keeping their result sizes compact to avoid emphasizing the client-server network protocol’s role in the benchmark [13].

1.2.1 Choke Point-Based Design Methodology

LDBC’s query design process relies on the use of *choke points* (??), i.e. challenging aspects of query processing. SNB BI includes 38 choke points divided into 9 categories: aggregation performance, join performance, data access locality, expression calculation, correlated subqueries, parallelism and concurrency, graph specifics, language features, and update operations. Their coverage is shown in ???. In the following, we discuss two challenges that are particularly prevalent in graph workloads.

1.2.1.1 Explosive and redundant multi-joins

In recent years it has become clear that graph pattern matching, or equivalent multi-join queries over many-to-many relationships, typically generate very large intermediate results when executed with traditional join algorithms. This is especially the case for cyclical join graphs (corresponding to cyclic graph queries). It was proven in theory [11] and shown in practice [18, 9, 5] that “worst-case optimal” *multi-join* algorithms can avoid these large intermediates and outperform traditional joins. Following this, there has been increased attention on *redundancy* in join results (even when produced by worst-case optimal joins), which can be eliminated using *factorized* query processing techniques [2, 12, 8]. Graph pattern matching queries that contain large join patterns will trigger these phenomena.

1.2.1.2 Expressive path finding

SNB BI contains queries that require an efficient implementation of shortest path finding between many pairs. Expressing such queries requires a query language which supports either path finding or recursion. The underlying system implementation must then handle this with an optimized execution strategy, as recursing to try all paths will not scale. As some of this path finding includes on-the-fly computed edges (joins) between nodes, the queries can benefit from *path expressions*, as proposed in Oracle’s PGQL language [15] and as part of the GQL and SQL/PQ languages [3]. The path finding required by SNB BI not only tests connectivity (as supported in SPARQL), but also requires returning the *cheapest cost* along weighted paths (necessitating SPARQL extensions [10]).

1.2.2 Analysis of Selected Queries

In order to defeat trivializing complex query performance by query caching, benchmarks can use both frequent updates (which require invalidating caches or maintaining cached intermediates) as well as parameterized query templates. The BI workload features update batches, so parametrized *read query templates* are necessary to guard against this between the batches. In this section, we analyze four read query templates.

Notation: We denote the query parameters with the \$ symbol and discuss their generation in Section 1.3.

1.2.2.1 Q11: Friend triangles

BI 11 imposes two key difficulties. First, systems should efficiently filter the knows edges based on the location of their endpoint Persons (Country \$country) and the date range. Second, given a large number of knows edges even after filtering, efficient enumeration of personA–personB–personC triangles (a cyclic subgraph query) requires worst-case optimal multi-joins.

1.2.2.2 Q14: International dialog

BI 14 imposes different challenges depending on whether Countries \$country1 and \$country2 are correlated or anti-correlated (Section 1.3.3.1). For the ranking, *top-k pushdown* can be exploited: once a result for a City in \$country1 is obtained, extra restrictions in a selection can be added based on the value

of this element. As the score of two Persons does not depend on any query parameters, precomputing and maintaining it as an attribute on the knows edge can be beneficial.

1.2.2.3 Q18: Friend recommendation

BI 18 is inspired by Twitter’s recommendation algorithm [7]. Implementations of this query can exploit factorization: systems can count the number of mutual friends without explicitly enumerating all $\langle \text{person1}, \text{personM}, \text{person2} \rangle$ tuples.

1.2.2.4 Q20: Recruitment

BI 20 performs *graph projection* [1]. Instead of materializing this graph in the database, systems may represent it using a compact in-memory structure such as CSR (Compressed Sparse Row) [16]. To perform the cheapest path computation, a single-source shortest path algorithm (starting from $\text{\$person2}$), such as Dijkstra’s algorithm, can be used. As the projected graph is independent of query parameters, precomputing and maintaining it can be beneficial.

1.3 Parameter Curation for BI Queries

1.3.1 The Need for Parameter Curation

A disadvantage of executing the same read query template with different parameters is that the intermediate results and runtimes can be severely influenced by the parameter values. This is particularly the case in SNB BI with its explosive joins, skewed out-degrees, skewed value distributions, correlated value distributions, and structural correlations. Moreover, the updates (including cascading deletes) can significantly change the portion of the graph reached by the same query executed at different times. In order to keep query performance understandable we need to actively *curate* parameters, such that different parameters executed at different logical times still lead to stable and, therefore, understandable results. We achieve this through *parameter curation* [6, 4], a data mining process of looking for parameter values with suitably similar characteristics.

1.3.2 Parameter Generation Steps

Our parameter curation process is a two-step process: we first generate *factors* followed by the *parameters* (Figure 1.1). These components are executed for each scale factor and are independent of the serialization format/layout of the data set.

1.3.2.1 Factor Generator

The factor generator produces 21 *factor tables* containing summary statistics from the temporal graph, e.g. the number of Persons per City or the number of Messages per day for each Tag.

1.3.2.2 Parameter Generation

To find suitable substitution parameters that (presumably) lead to the same amount of data access and thus similar runtimes, we first identify the factor table containing the summary statistics of the query’s parameters. For example, Q14’s template uses the parameters Country $\text{\$country1}$ and Country $\text{\$country2}$. Therefore, we use the `countryPairsNumFriends` factor table which contains $\text{\$country1}$, $\text{\$country2}$ pairs and the number of friendships on Person lives in $\text{\$country1}$ and the other lives in $\text{\$country2}$. Using this table, we select the p th percentile from the distribution as the *anchor*, then rank the rest of the distribution based on their absolute difference from the anchor and take the top- k values. We shuffle the values using a hash function to avoid introducing artificial locality, where e.g. subsequent queries start in nodes from the same ID range. Listing 1.1 shows the SQL query implementing the parameter generation for Q14a.

1.3.3 Parameter Curation for Graph Queries

We discuss two parameter curation cases that are particularly important in graph data management.

1.3.3.1 Correlated vs. Anti-Correlated Parameters

Our parameter curation provides a straightforward way of selecting start entities which are affected by (structural or attribute-level) correlation vs. anti-correlation: corresponding parameters can be found by selecting a high vs. low percentile as the anchor in the parameter generation query. For example, for Q14 (Section 1.4), we selected variant a to $p = 0.98$ (correlated) and variant b to $p = 0.03$ (anti-correlated).

1.3.3.2 Path Queries

SNB BI queries Q15, Q19, Q20 include cheapest path finding queries computed between given (sets of) Persons. These queries are particularly challenging for parameter curation: if there is no path between the two endpoints, query runtimes are significantly higher as the search has to traverse an entire connected component to ensure that no path exists. Moreover, the presence of a path between two nodes *at a given time* does not guarantee that it will always exist during the benchmark execution as deletions can render the endpoints of a path unreachable.

1.3.4 Query Variants

12 queries have a single variant, while 8 queries have two variants, yielding a total of 28 query variants. As a rule of thumb, variants a are expected to produce a longer runtime while variants b are expected to be simpler. Variants of Q2, Q8, Q16 are parametrized with a flashmob vs. a non-flashmob date. Variants of Q14 and Q19 select correlated vs. non-correlated Countries/Cities. Q10's variants differ in degree (a start Person with an average number of friends vs. only a few friends), while Q15's variants have different path lengths and time intervals (4 hops and one week vs. 2 hops and one month). Q20a selects endpoints where it is guaranteed that *no path exists*, while Q20b selects ones where there is guaranteed that a path exists.

1.3.5 Scalability and Reproducibility

1.3.5.1 Scalability

The *factor generator* is part of the SNB Datagen and runs after the *temporal graph* has been created. It is implemented in Spark for distributed execution. While its computations use expensive, aggregation-heavy queries, the derived factor tables are *compact*, e.g. SF10 000 has only 20 GiB of factors in compressed Parquet format, the equivalent of approximately 100 GiB in CSV format, i.e. 1% of the total data set size. The *parameter generator* queries are executed in DuckDB [14], which supports vertical scalability and is capable of running the parameter generation for SF10 000 using less than 512 GiB memory.

```
SELECT country1, country2
FROM (
  SELECT
    country1,
    country2,
    abs(frequency - (
      SELECT percentile_disc(0.98) WITHIN GROUP (ORDER BY frequency) AS anchor FROM countryPairsNumFriends
    )) AS diff
  FROM countryPairsNumFriends
  ORDER BY diff, country1, country2
)
ORDER BY md5(concat(country1, country2))
```

```
LIMIT 50
```

Listing 1.1: Parameter generation SQL query for Q14a.

1.3.5.2 Reproducibility

It is important to guarantee that the parameter curation process is reproducible. To this end, we leverage that the Datagen and, consequently, the factor generator are reproducible. To ensure that the parameter generation queries yield deterministic results we define a total ordering in each query. To provide deterministic shuffling we base the ordering on MD5 hashes (instead of the actual attribute values), see Listing 1.1.

1.4 Reads

BI / read / 1

BI 1	query	BI / read / 1				
BI 2	title	Posting summary				
BI 3	pattern	<div style="border: 1px solid black; padding: 2px;"> message: Message </div>				
BI 4		<div style="border: 1px solid black; padding: 2px;"> <code>creationDate < \$datetime</code> </div>				
BI 5		<div style="border: 1px solid black; padding: 2px;"> <code>length</code> </div>				
BI 6		<div style="border: 1px solid black; padding: 2px;"> <code>year(creationDate)</code> </div>				
BI 7	description	<p>Given a <code>\$datetime</code>, find all Messages created before that moment. Group them by a 3-level grouping:</p> <ol style="list-style-type: none"> 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 <ul style="list-style-type: none"> • 0: $0 \leq \text{length} < 40$ (short) • 1: $40 \leq \text{length} < 80$ (one liner) • 2: $80 \leq \text{length} < 160$ (tweet) • 3: $160 \leq \text{length}$ (long) 				
BI 8						
BI 9						
BI 10						
BI 11						
BI 12						
BI 13	params	1	<code>\$datetime</code>	DateTime		
BI 14						
BI 15	result	1	year	32-bit Integer	R	<code>year(message.creationDate)</code>
BI 16		2	isComment	Boolean	M	True for Comments, False for Posts
BI 17		3	lengthCategory	32-bit Integer	C	0 for short, 1 for one-liner, 2 for tweet, 3 for long
BI 18		4	messageCount	64-bit Integer	A	Total number of Messages in that group
BI 19		5	averageMessageLength	32-bit Float	A	Average length of the Message content in that group
BI 20		6	sumMessageLength	64-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	↑		
	limit	n/a				
	CPs	1.2, 3.2, 4.1, 4.2, 8.5				

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

query	BI / read / 2																						
title	Tag evolution																						
pattern	<pre> classDiagram class TagClass { name = \$tagClass } class Tag { name } class Message1 { message: Message creationDate in [\$date, \$date+100 days) } class Message2 { message: Message creationDate in [\$date+100 days, \$date+200 days) } TagClass -- > Tag : hasType Tag --> Message1 : «cql» hasTag Tag --> Message2 : «cql» hasTag Message1 --> TagClass : countWindow1 = count(message) Message2 --> TagClass : countWindow2 = count(message) </pre>																						
description	Find the Tags under a given \$tagClass that were used in Messages during in the 100-day time window starting at \$date and compare it with the 100-day time window that follows. For the Tags and for both time windows, compute the count of Messages.																						
params	<table border="1"> <tr> <td style="background-color: red; color: white;">1</td> <td>\$date</td> <td>Date</td> <td rowspan="2">Based on the creation day – TagClass – number of Messages factor table: (a) A flashmob date (b) A non-flashmob date</td> </tr> <tr> <td style="background-color: red; color: white;">2</td> <td>\$tagClass</td> <td>Long String</td> </tr> </table>	1	\$date	Date	Based on the creation day – TagClass – number of Messages factor table: (a) A flashmob date (b) A non-flashmob date	2	\$tagClass	Long String															
1	\$date	Date	Based on the creation day – TagClass – number of Messages factor table: (a) A flashmob date (b) A non-flashmob date																				
2	\$tagClass	Long String																					
result	<table border="1"> <tr> <td style="background-color: blue; color: white;">1</td> <td>tag.name</td> <td>Long String</td> <td>R</td> <td></td> </tr> <tr> <td style="background-color: blue; color: white;">2</td> <td>countWindow1</td> <td>32-bit Integer</td> <td>A</td> <td>Occurrences of the tag during the first time window</td> </tr> <tr> <td style="background-color: blue; color: white;">3</td> <td>countWindow2</td> <td>32-bit Integer</td> <td>A</td> <td>Occurrences of the tag during the second time window</td> </tr> <tr> <td style="background-color: blue; color: white;">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 tag during the first time window	3	countWindow2	32-bit Integer	A	Occurrences of the tag 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 tag during the first time window																			
3	countWindow2	32-bit Integer	A	Occurrences of the tag during the second time window																			
4	diff	32-bit Integer	A	Absolute difference of countWindow1 and countWindow2																			
sort	<table border="1"> <tr> <td style="background-color: green; color: white;">1</td> <td>diff</td> <td>↓</td> <td></td> </tr> <tr> <td style="background-color: green; color: white;">2</td> <td>tag.name</td> <td>↑</td> <td></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

BI 1	query	BI / read / 3																									
BI 2	title	Popular topics in a country																									
BI 3	pattern	<pre> classDiagram class Country { name = \$country } class City { } class Person { id } class Forum { id title creationDate } class TagClass { name = \$tagClass } class Tag { } class Message { count(message) } class Post { } Country < -- City City --> Person : isLocatedIn Forum --> Person : hasModerator Forum --> Post : containerOf TagClass --> Tag : hasType Tag --> Message : hasTag Message --> Post : replyOf*0.. </pre>																									
BI 4																											
BI 5																											
BI 6																											
BI 7																											
BI 8																											
BI 9																											
BI 10																											
BI 11																											
BI 12																											
BI 13	description	<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 Forum which contains them.</p> <p>The location of a Forum is identified by the location of the Forum's moderator.</p>																									
BI 14																											
BI 15	params	<table border="1"> <tr> <td>1</td> <td>\$tagClass</td> <td>Long String</td> <td>TagClasses with a similar amount of Messages are selected</td> </tr> <tr> <td>2</td> <td>\$country</td> <td>Long String</td> <td>Big Countries are selected</td> </tr> </table>	1	\$tagClass	Long String	TagClasses with a similar amount of Messages are selected	2	\$country	Long String	Big Countries are selected																	
1		\$tagClass	Long String	TagClasses with a similar amount of Messages are selected																							
2	\$country	Long String	Big Countries are selected																								
BI 16	result	<table border="1"> <tr> <td>1</td> <td>forum.id</td> <td>ID</td> <td>R</td> <td></td> </tr> <tr> <td>2</td> <td>forum.title</td> <td>Long String</td> <td>R</td> <td></td> </tr> <tr> <td>3</td> <td>forum.creationDate</td> <td>DateTime</td> <td>R</td> <td></td> </tr> <tr> <td>4</td> <td>person.id</td> <td>ID</td> <td>R</td> <td></td> </tr> <tr> <td>5</td> <td>messageCount</td> <td>32-bit Integer</td> <td>A</td> <td></td> </tr> </table>	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	
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																								
BI 17	sort	<table border="1"> <tr> <td>1</td> <td>messageCount</td> <td>↓</td> <td></td> </tr> <tr> <td>2</td> <td>forum.id</td> <td>↑</td> <td></td> </tr> </table>	1	messageCount	↓		2	forum.id	↑																		
1		messageCount	↓																								
2	forum.id	↑																									
BI 18	limit	20																									
BI 19	CPs	1.1, 1.2, 1.3, 2.1, 2.2, 2.4, 3.3, 8.2																									
BI 20																											

BI / read / 4

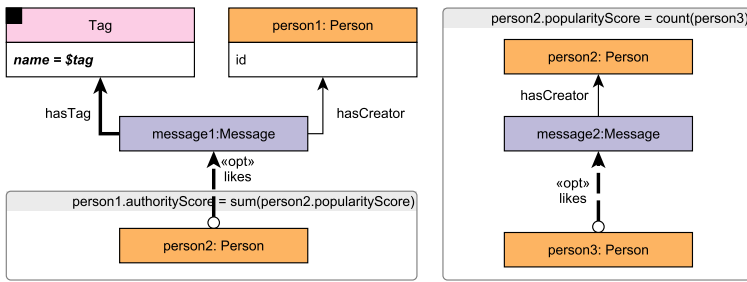
- BI 1
- BI 2
- BI 3
- BI 4
- BI 5
- BI 6
- BI 7
- BI 8
- BI 9
- BI 10
- BI 11
- BI 12
- BI 13
- BI 14
- BI 15
- BI 16
- BI 17
- BI 18
- BI 19
- BI 20

query	BI / read / 4																									
title	Top message creators by country																									
pattern	<p>The diagram illustrates the query patterns for BI / read / 4. It is divided into two main steps:</p> <ul style="list-style-type: none"> Step 1: "select top 100 forums based on memberCount in country". This involves a hierarchy where a Country (yellow) contains Cities (teal). A Forum (green) is located in a City (isLocatedIn) and has a memberCount = count(member). A Person (orange) is a member of a Forum (hasMember). Step 2: "for each country, for each of the top 100 forums (topForum1), count the Messages made by Persons who are members of any of the top 100 forums (topForum2)". This involves a Forum (topForum1) containing Posts (red). A Person (topForum2) is a member of a Forum (hasMember) and creates Messages (purple). A Message is a reply to a Post (replyOf*0..). A Message has a messageCount = count(message) and is created by a Person (hasCreator). 																									
description	<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 and the Forum was created after a given \$date.</p> <p>Calculate the top 100 most popular Forums. If a Forum is popular in multiple countries, it should only be calculated once with its largest membership. In case of a tie, the Forum with the smaller id value 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>																									
params	<table border="1"> <tr> <td>1</td> <td>\$date</td> <td>Date</td> <td>Selected from the first 30 days of the network</td> </tr> </table>	1	\$date	Date	Selected from the first 30 days of the network																					
1	\$date	Date	Selected from the first 30 days of the network																							
result	<table border="1"> <tr> <td>1</td> <td>person.id</td> <td>ID</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>person.creationDate</td> <td>DateTime</td> <td>R</td> <td></td> </tr> <tr> <td>5</td> <td>messageCount</td> <td>32-bit Integer</td> <td>A</td> <td></td> </tr> </table>	1	person.id	ID	R		2	person.firstName	String	R		3	person.lastName	String	R		4	person.creationDate	DateTime	R		5	messageCount	32-bit Integer	A	
1	person.id	ID	R																							
2	person.firstName	String	R																							
3	person.lastName	String	R																							
4	person.creationDate	DateTime	R																							
5	messageCount	32-bit Integer	A																							
sort	<table border="1"> <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>	1	messageCount	↓		2	person.id	↑																		
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

BI 1	query	BI / read / 5									
BI 2	title	Most active posters of a given topic									
BI 3	pattern	<pre> classDiagram class Tag { name = \$tag } class person_Person { id } class liker_Person { } class m_Message { } class comment_Comment { } Tag --> person_Person : hasCreator Tag --> m_Message : hasTag person_Person --> m_Message : hasCreator liker_Person --> m_Message : «opt» likes comment_Comment --> m_Message : «opt» replyOf subgraph likeCount_calculation [likeCount = count(liker)] liker_Person end subgraph messageCount_calculation [messageCount = count(m)] m_Message end subgraph replyCount_calculation [replyCount = count(comment)] comment_Comment end person_Person --> score : person.score = 1*messageCount + 2*replyCount + 10*likeCount </pre>									
BI 4					description	<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 its Messages. <p>The score is calculated according to the following formula: $1 \times \text{messageCount} + 2 \times \text{replyCount} + 10 \times \text{likeCount}$.</p>					
BI 5						params	1	\$tag	Long String	Tags with a similar amount of Messages are selected. To avoid caching, different Tags should be used than the ones in Q6 and Q7.	
BI 6							result	1	person.id	ID	R
BI 7								2	replyCount	32-bit Integer	A
BI 8								3	likeCount	32-bit Integer	A
BI 9								4	messageCount	32-bit Integer	A
BI 10								5	score	32-bit Integer	A
BI 11							sort	1	score	↓	
BI 12								2	person.id	↑	
BI 13	limit	100									
BI 14	CPs	1.2, 2.3, 2.6, 8.2									

BI / read / 6

BI 1	query	BI / read / 6				
BI 2	title	Most authoritative users on a given topic				
BI 3	pattern					
BI 4						
BI 5						
BI 6						
BI 7						
BI 8						
BI 9						
BI 10						
BI 11						
BI 12						description
BI 13						
BI 14						
BI 15						
BI 16						
BI 17						
BI 18						
BI 19						
BI 20						
BI 20	params	1	\$tag	Long String	Tags with a similar amount of Messages are selected. To avoid caching, different Tags should be used than the ones in Q5 and Q7.	
BI 1		result	1	person1.id	ID	R
BI 2			2	authorityScore	32-bit Integer	A
BI 1		sort	1	authorityScore	↓	
BI 2			2	person1.id	↑	
BI 1		limit	100			
BI 1		CPs	1.2, 2.3, 2.6, 3.3, 6.1, 8.2			
BI 1		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 4	description	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 related Tags by name, and get the count of replies in each group.			
BI 5	params	1	\$tag	Long String	Tags with a similar amount of Messages are selected. To avoid caching, different Tags should be used than the ones in Q5 and Q6.
BI 6	result	1	relatedTag.name	Long String	R
BI 7		2	count	32-bit Integer	A
BI 8	sort	1	count	↓	
BI 9		2	relatedTag.name	↑	
BI 10	limit	100			
BI 11	CPs	1.4, 3.3, 5.2, 8.1			

BI / read / 8

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 / 8																	
title	Central person for a tag																	
pattern	<p>For each person with a matching hasInterest and/or hasCreator edge, compute person.score = (if hasInterest edge exists then 100 else 0) + count(message)</p> <p>Calculate the sum of the friends' scores: friendsScore = sum(friend.score)</p>																	
description	<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 \$startDate 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	<table border="1"> <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>\$startDate</td> <td>Date</td> <td>(a): A range during which a flashmob event happened (it should yield at least a 5× difference) (b): A regular range (does not include a flashmob event)</td> </tr> <tr> <td>3</td> <td>\$endDate</td> <td>Date</td> <td></td> </tr> </table>	1	\$tag	Long String	Tags with a similar amount of Messages are selected	2	\$startDate	Date	(a): A range during which a flashmob event happened (it should yield at least a 5× difference) (b): A regular range (does not include a flashmob event)	3	\$endDate	Date						
1	\$tag	Long String	Tags with a similar amount of Messages are selected															
2	\$startDate	Date	(a): A range during which a flashmob event happened (it should yield at least a 5× difference) (b): A regular range (does not include a flashmob event)															
3	\$endDate	Date																
result	<table border="1"> <tr> <td>1</td> <td>person.id</td> <td>ID</td> <td>R</td> <td></td> </tr> <tr> <td>2</td> <td>score</td> <td>32-bit Integer</td> <td>A</td> <td></td> </tr> <tr> <td>3</td> <td>friendsScore</td> <td>32-bit Integer</td> <td>A</td> <td>The sum of the score of the person's friends</td> </tr> </table>	1	person.id	ID	R		2	score	32-bit Integer	A		3	friendsScore	32-bit Integer	A	The sum of the score of the person's friends		
1	person.id	ID	R															
2	score	32-bit Integer	A															
3	friendsScore	32-bit Integer	A	The sum of the score of the person's friends														
sort	<table border="1"> <tr> <td>1</td> <td>score + friendsScore</td> <td>↓</td> <td></td> </tr> <tr> <td>2</td> <td>person.id</td> <td>↑</td> <td></td> </tr> </table>	1	score + friendsScore	↓		2	person.id	↑										
1	score + friendsScore	↓																
2	person.id	↑																
limit	100																	
CPs	1.2, 2.1, 2.3, 3.2, 5.3, 8.2, 8.4, 8.5																	
relevance	<p>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.</p>																	

BI / read / 9

BI 1	query	BI / read / 9				
BI 2	title	Top thread initiators				
BI 3	pattern	<pre> classDiagram class Person { id firstName lastName } class Post { creationDate in [startDate, endDate] } class Message { creationDate in [startDate, endDate] } Person --> Post : hasCreator Post --> Message : replyOf*0.. </pre>				
BI 8	description	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.				
BI 13		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).				
BI 15	params	1	startDate	Date	Selected around the same date	
BI 16		2	endDate	Date	80-100 days after the startDate	
BI 17	result	1	person.id	ID	R	
BI 18		2	person.firstName	String	R	
BI 19		3	person.lastName	String	R	
BI 20		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, 2.6, 3.2, 7.2, 7.3, 7.4, 8.1, 8.5				

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

query	BI / read / 10																						
title	Experts in social circle																						
pattern																							
description	<p>Given a Person startPerson with ID \$personID, find all other Persons (expertCandidatePerson) that live in a given \$country and are connected to the startPerson on 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	<table border="1"> <tr> <td>1</td> <td>\$personId</td> <td>ID</td> <td>(a) Persons with an average degree of knows edges are selected (b) Persons who have only one friend and that Person has two friends in total (including the original Person)</td> </tr> <tr> <td>2</td> <td>\$country</td> <td>String</td> <td>Select mid-sized Countries</td> </tr> <tr> <td>3</td> <td>\$tagClass</td> <td>Long String</td> <td>TagClasses with a similar degree of hasType edges are selected</td> </tr> <tr> <td>4</td> <td>\$minPathDistance</td> <td>32-bit Integer</td> <td>3</td> </tr> <tr> <td>5</td> <td>\$maxPathDistance</td> <td>32-bit Integer</td> <td>4</td> </tr> </table>	1	\$personId	ID	(a) Persons with an average degree of knows edges are selected (b) Persons who have only one friend and that Person has two friends in total (including the original Person)	2	\$country	String	Select mid-sized Countries	3	\$tagClass	Long String	TagClasses with a similar degree of hasType edges are selected	4	\$minPathDistance	32-bit Integer	3	5	\$maxPathDistance	32-bit Integer	4		
1	\$personId	ID	(a) Persons with an average degree of knows edges are selected (b) Persons who have only one friend and that Person has two friends in total (including the original Person)																				
2	\$country	String	Select mid-sized Countries																				
3	\$tagClass	Long String	TagClasses with a similar degree of hasType edges are selected																				
4	\$minPathDistance	32-bit Integer	3																				
5	\$maxPathDistance	32-bit Integer	4																				
result	<table border="1"> <tr> <td>1</td> <td>expertCandidatePerson.id</td> <td>ID</td> <td>R</td> <td></td> </tr> <tr> <td>2</td> <td>tag.name</td> <td>Long 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>	1	expertCandidatePerson.id	ID	R		2	tag.name	Long String	R		3	messageCount	32-bit Integer	A	Number of Messages created by that Person containing that Tag							
1	expertCandidatePerson.id	ID	R																				
2	tag.name	Long String	R																				
3	messageCount	32-bit Integer	A	Number of Messages created by that Person containing that Tag																			
sort	<table border="1"> <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>expertCandidatePerson.id</td> <td>↑</td> <td></td> </tr> </table>	1	messageCount	↓		2	tag.name	↑		3	expertCandidatePerson.id	↑											
1	messageCount	↓																					
2	tag.name	↑																					
3	expertCandidatePerson.id	↑																					
limit	100																						
CPs	1.2, 1.3, 2.3, 2.4, 2.6, 3.3, 5.3, 7.1, 7.2, 7.3, 8.1, 8.6																						

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

query	BI / read / 11														
title	Friend triangles														
pattern															
description	<p>For a given \$country, count all the distinct triples of Persons such that:</p> <ul style="list-style-type: none"> • personA is friend of personB, • personB is friend of personC, • personC is friend of personA, <p>and these friendships were created in the range [\$startDate, \$endDate]. 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 border="1"> <tr> <td>1</td> <td>\$country</td> <td>Long String</td> <td>Selected from the largest Countries (India, China)</td> </tr> <tr> <td>2</td> <td>\$startDate</td> <td>Date</td> <td>Selected from a 30-day interval towards the end of the simulation time</td> </tr> <tr> <td>3</td> <td>\$endDate</td> <td>Date</td> <td>Selected to yield around a 100-day interval</td> </tr> </table>	1	\$country	Long String	Selected from the largest Countries (India, China)	2	\$startDate	Date	Selected from a 30-day interval towards the end of the simulation time	3	\$endDate	Date	Selected to yield around a 100-day interval		
1	\$country	Long String	Selected from the largest Countries (India, China)												
2	\$startDate	Date	Selected from a 30-day interval towards the end of the simulation time												
3	\$endDate	Date	Selected to yield around a 100-day interval												
result	<table border="1"> <tr> <td>1</td> <td>count</td> <td>64-bit Integer</td> <td>A</td> </tr> </table>	1	count	64-bit Integer	A										
1	count	64-bit Integer	A												
limit	n/a														
CPs	2.3, 2.5, 3.2														

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

query	BI / read / 12														
title	How many persons have a given number of messages														
pattern															
description	<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 creationDate is after \$startDate (exclusive, equality is not allowed). • Its length is below the \$lengthThreshold (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	<table border="1"> <tr> <td>1</td> <td>\$startDate</td> <td>Date</td> <td>Selected randomly from a 60-day interval.</td> </tr> <tr> <td>2</td> <td>\$lengthThreshold</td> <td>32-bit Integer</td> <td>Balanced against startDate to filter around 30% of the Messages within a language and keep the variance low. The selection of this parameter uses a factor table of bucketed Message lengths and creation dates.</td> </tr> <tr> <td>3</td> <td>\$languages</td> <td>{String}</td> <td>Only the most frequently used languages</td> </tr> </table>	1	\$startDate	Date	Selected randomly from a 60-day interval.	2	\$lengthThreshold	32-bit Integer	Balanced against startDate to filter around 30% of the Messages within a language and keep the variance low. The selection of this parameter uses a factor table of bucketed Message lengths and creation dates.	3	\$languages	{String}	Only the most frequently used languages		
1	\$startDate	Date	Selected randomly from a 60-day interval.												
2	\$lengthThreshold	32-bit Integer	Balanced against startDate to filter around 30% of the Messages within a language and keep the variance low. The selection of this parameter uses a factor table of bucketed Message lengths and creation dates.												
3	\$languages	{String}	Only the most frequently used languages												
result	<table border="1"> <tr> <td>1</td> <td>messageCount</td> <td>32-bit Integer</td> <td>A</td> <td>Number of Messages created</td> </tr> <tr> <td>2</td> <td>personCount</td> <td>32-bit Integer</td> <td>A</td> <td>Number of Persons with messageCount Messages</td> </tr> </table>	1	messageCount	32-bit Integer	A	Number of Messages created	2	personCount	32-bit Integer	A	Number of Persons with messageCount Messages				
1	messageCount	32-bit Integer	A	Number of Messages created											
2	personCount	32-bit Integer	A	Number of Persons with messageCount Messages											
sort	<table border="1"> <tr> <td>1</td> <td>personCount</td> <td>↓</td> <td></td> </tr> <tr> <td>2</td> <td>messageCount</td> <td>↓</td> <td></td> </tr> </table>	1	personCount	↓		2	messageCount	↓							
1	personCount	↓													
2	messageCount	↓													
limit	n/a														
CPs	1.1, 1.2, 1.4, 2.6, 3.2, 4.2, 4.3, 8.1, 8.2, 8.3, 8.4, 8.5														

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

query	BI / read / 13				
title	Zombies in a country				
pattern					
description	<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: $\text{zombieLikeCount} / \text{totalLikeCount}$. 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	Selected from the largest Countries (India, China)	
	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	32-bit Float	A	Determined as $\text{zombieLikeCount} / \text{totalLikeCount}$
sort	1	zombieScore	↓		
	2	zombie.id	↑		
limit	100				
CPs	1.2, 2.1, 2.3, 2.4, 2.6, 3.2, 3.3, 4.2, 5.1, 5.3, 8.2, 8.4, 8.5				

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

query	BI / read / 14			
title	International dialog			
pattern	<p>For each pair of countries, calculate the cost as a sum of cases #1-4. 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> <p>Case 1: score += 4</p> <p>Case 2: score += 1</p> <p>Case 3: score += 10</p> <p>Case 4: score += 1</p>			
description	<p>Consider all pairs of people ($person1$, $person2$) such that (1) they know each other, (2) one is located in a City of $\\$country1$, and (3) the other is located in a City of $\\$country2$. For each City of $\\$country1$, return the highest scoring pair. If there are multiple top-scoring pairs in a city, return the pair with the lowest ($person1.id$, $person2.id$) using a lexicographical ordering.</p> <p>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. $person1$ liked at least one Message by $person2$: score += 10 4. $person1$ has created at least one Message that was liked by $person2$: score += 1 <p>Consequently, the maximum score a pair can obtain is: $4 + 1 + 10 + 1 = 16$.</p>			
params	1	$\$country1$	Long String	(a) Correlated with parameter $country2$, i.e. the Countries are close and there are many Persons knowing each other (b) Uncorrelated with parameter $country2$, i.e. the Countries are afar and there are few Persons knowing each other
	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	$person2.id$	↑	
limit	100			
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
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 / 15																		
title	Trusted connection paths through forums created in a given timeframe																		
pattern	<p>Calculate the weight of the shortest path on knows edges between person1 and person2. Edge weights are determined as $1 / (\text{interaction score} + 1)$, where interaction score is the sum of cases #1 and #2 for the Person endpoints of the edge (tried both ways).</p> <p>Example for finding a path between person1 and person2</p>																		
description	<p>Given two Persons with IDs $\\$person1Id$ and $\\$person2Id$, calculate the cost of the weighted shortest path between these two Persons, in the subgraph induced by the knows relationship. The interaction score of a knows edge is calculated based on the interactions of its Person endpoints:</p> <ul style="list-style-type: none"> • Every direct reply (by one of the Persons) to a Post (by the other Person) is 1.0 point. • Every direct reply (by one of the Persons) to a Comment (by the other Person) is 0.5 points. <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>The weight for the shortest path algorithm is determined as $\frac{1}{\text{interaction score} + 1}$.</p> <p>The result of the query is a single number, the cost of the weighted shortest path. If no such path exists, the query should return -1.0.</p>																		
params	<table border="1"> <tr> <td>1</td> <td>$\\$person1Id$</td> <td>ID</td> <td>(a) $\\$person1Id - \\$person2Id$ pair with a distance of 4 hops (b) $\\$person1Id - \\$person2Id$ pair with a distance of 2 hops</td> </tr> <tr> <td>2</td> <td>$\\$person2Id$</td> <td>ID</td> <td></td> </tr> <tr> <td>3</td> <td>$\\$startDate$</td> <td>Date</td> <td>(a) Small interval (approx. one week) (b) Big interval (approx. one month)</td> </tr> <tr> <td>4</td> <td>$\\$endDate$</td> <td>Date</td> <td></td> </tr> </table>	1	$\$person1Id$	ID	(a) $\$person1Id - \$person2Id$ pair with a distance of 4 hops (b) $\$person1Id - \$person2Id$ pair with a distance of 2 hops	2	$\$person2Id$	ID		3	$\$startDate$	Date	(a) Small interval (approx. one week) (b) Big interval (approx. one month)	4	$\$endDate$	Date			
1	$\$person1Id$	ID	(a) $\$person1Id - \$person2Id$ pair with a distance of 4 hops (b) $\$person1Id - \$person2Id$ pair with a distance of 2 hops																
2	$\$person2Id$	ID																	
3	$\$startDate$	Date	(a) Small interval (approx. one week) (b) Big interval (approx. one month)																
4	$\$endDate$	Date																	
result	1	weight	32-bit Float C																
limit	n/a																		
CPs	1.2, 2.1, 2.2, 2.4, 3.3, 5.1, 5.3, 7.2, 7.3, 7.6, 7.7, 8.1, 8.2, 8.3, 8.4, 8.5, 8.6																		

BI / read / 16

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

query	BI / read / 16																						
title	Fake news detection																						
pattern	<p>The diagram illustrates the query pattern in two steps. Step 1 shows an induced subgraph where a Tag (name = \$tagX) is associated with a Message (day(creationDate) = \$dateX) via a 'hasTag' relationship, and the Message is created by a Person (person: Person) via a 'hasCreator' relationship. Step 2 shows the same subgraph but with a 'count(messageX)' operation on the Message entity. Additionally, a Person entity is shown with a 'count <= \$maxKnowsLimit' constraint on its 'knows' relationship to other Person entities.</p>																						
description	<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>																						
params	<table border="1"> <tr> <td>1</td> <td>\$tagA</td> <td>Long String</td> <td>(a) \$tagA/\$dateA, \$tagB/\$dateB are both selected to be a flashmob Tag/date combination (b) \$tagA/\$dateA, \$tagB/\$dateB are both selected to be a non-flashmob Tag/date combination</td> </tr> <tr> <td>2</td> <td>\$dateA</td> <td>Date</td> <td></td> </tr> <tr> <td>3</td> <td>\$tagB</td> <td>Long String</td> <td></td> </tr> <tr> <td>4</td> <td>\$dateB</td> <td>Date</td> <td></td> </tr> <tr> <td>5</td> <td>\$maxKnowsLimit</td> <td>32-bit Integer</td> <td>Selected between 3 and 6</td> </tr> </table>	1	\$tagA	Long String	(a) \$tagA/\$dateA, \$tagB/\$dateB are both selected to be a flashmob Tag/date combination (b) \$tagA/\$dateA, \$tagB/\$dateB are both selected to be a non-flashmob Tag/date combination	2	\$dateA	Date		3	\$tagB	Long String		4	\$dateB	Date		5	\$maxKnowsLimit	32-bit Integer	Selected between 3 and 6		
1	\$tagA	Long String	(a) \$tagA/\$dateA, \$tagB/\$dateB are both selected to be a flashmob Tag/date combination (b) \$tagA/\$dateA, \$tagB/\$dateB are both selected to be a non-flashmob Tag/date combination																				
2	\$dateA	Date																					
3	\$tagB	Long String																					
4	\$dateB	Date																					
5	\$maxKnowsLimit	32-bit Integer	Selected between 3 and 6																				
result	<table border="1"> <tr> <td>1</td> <td>person.id</td> <td>ID</td> <td>R</td> <td></td> </tr> <tr> <td>2</td> <td>messageCountA</td> <td>32-bit Integer</td> <td>A</td> <td>Message count for \$tagA/\$dateA</td> </tr> <tr> <td>3</td> <td>messageCountB</td> <td>32-bit Integer</td> <td>A</td> <td>Message count for \$tagB/\$dateB</td> </tr> </table>	1	person.id	ID	R		2	messageCountA	32-bit Integer	A	Message count for \$tagA/\$dateA	3	messageCountB	32-bit Integer	A	Message count for \$tagB/\$dateB							
1	person.id	ID	R																				
2	messageCountA	32-bit Integer	A	Message count for \$tagA/\$dateA																			
3	messageCountB	32-bit Integer	A	Message count for \$tagB/\$dateB																			
sort	<table border="1"> <tr> <td>1</td> <td>messageCountA + messageCountB</td> <td>↓</td> <td></td> </tr> <tr> <td>2</td> <td>person.id</td> <td>↑</td> <td></td> </tr> </table>	1	messageCountA + messageCountB	↓		2	person.id	↑															
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 might repeatedly check whether a Person has at least one Message that satisfies the conditions).</p>																						

BI / read / 17

BI 1	query	BI / read / 17										
BI 2	title	Information propagation analysis										
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			description	<p>This query aims to identify instances of “information propagation” when a Person (person1) submits a Message (message1) with a given \$tag to a Forum (forum1). This is read by other members of forum1, Persons person2 and person3 (who must be different Persons). 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>								
BI 20												
BI 1	params	<table border="1"> <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.									
BI 1	result	<table border="1"> <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									
BI 1	sort	<table border="1"> <tr> <td>1</td> <td>messageCount</td> <td>↓</td> <td></td> </tr> <tr> <td>2</td> <td>person1.id</td> <td>↑</td> <td></td> </tr> </table>	1	messageCount	↓		2	person1.id	↑			
1		messageCount	↓									
2	person1.id	↑										
BI 1	limit	10										
BI 1	CPs	2.1, 2.3, 2.5, 2.6, 8.1										

BI / read / 18

BI 1	query	BI / read / 18															
BI 2	title	Friend recommendation															
BI 3	pattern																
BI 4																	
BI 5																	
BI 6																	
BI 7																	
BI 8																	
BI 9																	
BI 10																	
BI 11																	
BI 12																	
BI 13	description	For a given \$tag, for each person1 interested in \$tag, recommend new friends (person2) who															
BI 14		<ul style="list-style-type: none"> do not yet know person1 have at least one mutual friend with person1 are also interested in \$tag. 															
BI 15	params	Rank Persons person2 based on the number of mutual friends with person1.															
BI 16		<table border="1"> <tr> <td>1</td> <td>\$tag</td> <td>Long String</td> <td>Tags with a similar amount of Persons are selected</td> </tr> </table>	1	\$tag	Long String	Tags with a similar amount of Persons are selected											
1	\$tag	Long String	Tags with a similar amount of Persons are selected														
BI 17	result	<table border="1"> <tr> <td>1</td> <td>person1.id</td> <td>ID</td> <td>R</td> <td></td> </tr> <tr> <td>2</td> <td>person2.id</td> <td>ID</td> <td>R</td> <td></td> </tr> <tr> <td>3</td> <td>mutualFriendCount</td> <td>32-bit Integer</td> <td>A</td> <td></td> </tr> </table>	1	person1.id	ID	R		2	person2.id	ID	R		3	mutualFriendCount	32-bit Integer	A	
1		person1.id	ID	R													
2		person2.id	ID	R													
3	mutualFriendCount	32-bit Integer	A														
BI 18	sort	<table border="1"> <tr> <td>1</td> <td>mutualFriendCount</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>	1	mutualFriendCount	↓		2	person1.id	↑		3	person2.id	↑				
1		mutualFriendCount	↓														
2	person1.id	↑															
3	person2.id	↑															
BI 19	limit	20															
BI 20	CPs	2.5, 2.6, 8.1															

BI / read / 19

- 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 / 19			
title	Interaction path between cities			
pattern	<div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid black; padding: 5px; width: 45%;"> <p>Find the shortest paths between all pairs of Persons in city1 and city2. The weight of a knows edge is based on the number of interactions between its Persons: $knows.weight = \max(\text{round}(40 - \sqrt{\text{numInteractions}}), 1)$</p> </div> <div style="border: 1px solid black; padding: 5px; width: 45%;"> <p>Reply from personA to personB's Message, or personB to personA's Message</p> </div> </div> <div style="margin-top: 10px;"> <p>Example for finding a path between person1 and person2</p> </div>			
description	<p>Given two Cities with IDs $\\$city1Id$, $\\$city2Id$, find Persons $person1$, $person2$ living in these Cities (respectively) with the <i>cheapest</i> interaction path between them.</p> <p>The cheapest path is equivalent to the <i>weighted shortest</i> path. It is computed on a subgraph of the Person-knows-Person graph with the edge weights based on the number of interactions. An <i>interaction</i> is a direct reply Comments from one Person to Messages by the other Person. Only knows edges with at least one interaction between their endpoint Persons are considered. For these, the weight of a knows edge is defined as: $\max(\text{round}(40 - \sqrt{\text{numInteractions}}), 1)$</p> <p>If there are multiple pairs of people with cheapest paths that have the same total weight, return all of them.</p> <p><i>Note:</i> Interactions are counted both ways, e.g. if Alice knows Bob, 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 and the weight of the knows edge is 38.</p> <p><i>Remark:</i> Determinism is ensured by using square root followed by rounding. For all integers between 1 and 100 000, the square root's fractional part is more than 10e-5 from 0.5, where the rounding could be non-deterministic based on floating point inaccuracies. As 10e-5 is significantly larger than the machine epsilon of IEEE 754 floats (both 32- and 64-bit), the floating point inaccuracies have no chance to affect the derived integer edge weights.</p>			
params	1	$\$city1Id$	ID	(a) Small Cities within the same Country (b) Larger Cities from different Countries
	2	$\$city2Id$	ID	
result	1	person1.id	ID	R
	2	person2.id	ID	R
	3	totalWeight	32-bit Integer	C
sort	1	person1.id	↑	
	2	person2.id	↑	
limit	n/a			
CPs	3.3, 7.6, 7.7, 8.4, 8.6			
relevance	To find the weighted shortest paths efficiently, the system can use e.g. a bidirectional Dijkstra algorithm. As the edge weights do not depend on any parameter, systems can pre-compute them (if they do not interleave reads and writes).			

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

query	BI / read / 20			
title	Recruitment			
pattern	<div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid black; padding: 5px;"> <p>Compute weighted shortest path between all Persons who work in the Company to Person person2 on knows.weight</p> <pre> graph TD C[company: Company] -- workAt --> P1[person1: Person] P1 -- "compute weighted shortest path on knows.weight" --> P2[person2: Person] </pre> </div> <div style="border: 1px solid black; padding: 5px;"> <p>knows.weight: $\min(\text{abs}(\text{studyAtA.classYear} - \text{studyAtB.classYear}) + 1$</p> <pre> graph TD PA[personA: Person] -- knows --> PB[personB: Person] PA -- studyAtA: studyAt --> U[University] PB -- studyAtB: studyAt --> U </pre> </div> </div> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>Example for finding a path between person1 and person2</p> <pre> graph LR P1((p1)) -- knows --> PX((pX)) PX -- knows --> PY((pY)) PY -- knows --> PW((pW)) PW -- knows --> P2((p2)) P1 -- studyAt --> U1(()) PX -- studyAt --> U1 PX -- studyAt --> U2(()) PY -- studyAt --> U2 PW -- studyAt --> U3(()) P2 -- studyAt --> U3 </pre> </div>			
description	<p>Consider knows edges where the endpoint Persons attended the same University and set the weight of the edge to the absolute difference between the year of enrolment plus 1. If the Persons attended multiple universities, we select the smallest (min) value. Formally:</p> $w = \min_{\text{studyAt}_A, \text{studyAt}_B} \text{studyAt}_A.\text{classYear} - \text{studyAt}_B.\text{classYear} + 1$ <p>Given a \$company and a Person person2 with ID \$person2Id (who is not working and has not worked at \$company), find a different Person (person1) who works or at some point worked in \$company and is reachable from person2 through people who have studied together through the shortest weighted path.</p> <p>If there are multiple Person person1 nodes with the same shortest path length, return all of them.</p>			
params	1	\$company	Long String	Companies with a similar number of employees (former or current) are selected
	2	\$person2Id	ID	(a) There is guaranteed to be no path between any person1 working at company and person2 (b) There is guaranteed to be a 2-hop path between at least one person1 working at company and person
result	1	person1.id	ID	R
	2	totalWeight	32-bit Integer	C
sort	1	totalWeight	↑	
	2	person1.id	↑	
limit	20			
CPs	3.3, 7.6, 7.7, 7.8, 8.4, 8.6			
relevance	To find the weighted shortest paths efficiently, the system can use e.g. a bidirectional Dijkstra algorithm. As the edge weights do not depend on any parameter, systems can pre-compute them (if they do not interleave reads and writes).			

1.5 Insert Operations

Insert operations consist of individual inserts for each entity type. Implementations typically use the same format as the one for loading the initial snapshot of the data set.

1.6 Delete Operations

Updates / delete / 1

DEL 1
DEL 2
DEL 3
DEL 4
DEL 5
DEL 6
DEL 7
DEL 8

query	Updates / delete / 1				
title	Remove person and its personal forums and message (sub)threads				
pattern					
description	Remove a Person with ID \$personId and its edges (isLocatedIn, studyAt, workAt, hasInterest, likes, knows, hasMember, hasModerator, hasCreator). Additionally, remove the Album and Wall Forums whose moderator is the Person and remove all Messages the Person has created in the rest of the Forums (Groups).				
params	<table border="1"> <tr> <td>1</td> <td>\$personId</td> <td>ID</td> <td></td> </tr> </table>	1	\$personId	ID	
1	\$personId	ID			
CPs	9.3, 9.4, 9.5				
relevance	<ul style="list-style-type: none"> Removal of a Person removes Forums of type “Walls” and “Albums” but not “Groups”, which can continue if even the founder has left the network. For Groups, the hasModerator edge is deleted. We have discussed various approaches to appoint a new moderator, e.g. <ol style="list-style-type: none"> choose member at random from the set of existing group members or the member with the oldest group join date becomes the moderator. However, to keep the generator and the workload simple, currently no moderator is selected, leaving the group without a moderator. Removal of a Person removes all Posts/Comments they are creator of this could result in the removal of a Comment in the middle of a thread. 				

Updates / delete / 2

DEL 1	query	Updates / delete / 2		
DEL 2	title	Remove post like		
DEL 3	pattern			
DEL 6	description	Given a Person with ID \$personId and a Post with ID \$postId, remove the likes edge between them.		
DEL 7	params	1	\$personId	ID
DEL 8		2	\$postId	ID
	CPs	9.4		
	relevance	Removal of a likes edge is a rare event, e.g. people accidentally liking a Post, this can be reflected by the relative frequency of the operation.		

Updates / delete / 3

DEL 1	query	Updates / delete / 3		
DEL 2	title	Remove comment like		
DEL 3	pattern			
DEL 6	description	Given a Person with ID \$personId and a Comment with ID \$commentId, remove the likes edge between them.		
DEL 7	params	1	\$personId	ID
DEL 8		2	\$commentId	ID
	CPs	9.4		
	relevance	Removal of a likes edge is a rare event, e.g. people accidentally liking a Comment, this can be reflected by the relative frequency of the operation.		

Updates / delete / 4

DEL 1	query	Updates / delete / 4				
DEL 2	title	Remove forum and its content				
DEL 3	pattern					
DEL 4						
DEL 5						
DEL 6						
DEL 7						
DEL 8	description	Remove a Forum with ID <code>\$forumId</code> and its edges (<code>hasModerator</code> , <code>hasMember</code> , <code>hasTag</code>) and all Posts in the Forum (connected by <code>containerOf</code> edges) and their direct and transitive Comments.				
	params	<table border="1"> <tr> <td>1</td> <td><code>\$forumId</code></td> <td>ID</td> <td></td> </tr> </table>	1	<code>\$forumId</code>	ID	
1	<code>\$forumId</code>	ID				
	CPs	9.3, 9.4, 9.5				
	relevance	n/a				

Updates / delete / 5

DEL 1	query	Updates / delete / 5								
DEL 2	title	Remove forum membership								
DEL 3	pattern									
DEL 4										
DEL 5	description	Given a Forum with ID <code>\$forumId</code> and a Person with ID <code>\$personId</code> , remove the <code>hasMember</code> edge between them.								
DEL 6	params	<table border="1"> <tr> <td>1</td> <td><code>\$forumId</code></td> <td>ID</td> <td></td> </tr> <tr> <td>2</td> <td><code>\$personId</code></td> <td>ID</td> <td></td> </tr> </table>	1	<code>\$forumId</code>	ID		2	<code>\$personId</code>	ID	
1			<code>\$forumId</code>	ID						
2	<code>\$personId</code>	ID								
DEL 7	CPs	9.4								
DEL 8	relevance	n/a								

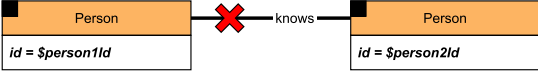
Updates / delete / 6

DEL 1	query	Updates / delete / 6
DEL 2	title	Remove post thread
DEL 3	pattern	
DEL 4		
DEL 5		
DEL 6		
DEL 7		
DEL 8	<div style="border: 1px solid black; padding: 2px; width: fit-content;"> invoke delete operation 7 </div>	
description	Remove a Post node with ID <code>\$postId</code> and its edges (<code>isLocatedIn</code> , <code>likes</code> , <code>hasCreator</code> , <code>hasTag</code> , <code>containerOf</code>). Remove all replies to the Post and the connecting <code>replyOf</code> edges. In addition, remove all transitive reply Comments to the Post and their edges.	
params	<div style="border: 1px solid black; padding: 2px;"> 1 <code>\$postId</code> ID </div>	
CPs	9.3, 9.4, 9.5	
relevance	n/a	

Updates / delete / 7

DEL 1	query	Updates / delete / 7
DEL 2	title	Remove comment subthread
DEL 3	pattern	
DEL 4		
DEL 5		
DEL 6		
DEL 7		
DEL 8	<div style="border: 1px solid black; padding: 2px; width: fit-content;"> delete recursively </div>	
description	Remove a Comment node with ID <code>\$commentId</code> and its <i>edges</i> (<code>isLocatedIn</code> , <code>likes</code> , <code>hasCreator</code> , <code>hasTag</code>). In addition, remove all replies to the Comment connected by <code>replyOf</code> and their <i>edges</i> .	
params	<div style="border: 1px solid black; padding: 2px;"> 1 <code>\$commentId</code> ID </div>	
CPs	9.3, 9.4, 9.5	
relevance	n/a	

Updates / delete / 8

DEL 1	query	Updates / delete / 8		
DEL 2	title	Remove friendship		
DEL 3	pattern			
DEL 6	description	Given two Person nodes with IDs \$person1Id and \$person2Id, remove the knows edge between them.		
DEL 7	params	1	\$person1Id	ID
DEL 8		2	\$person2Id	ID
	CPs	9.4		
	relevance	n/a		

BIBLIOGRAPHY

- [1] Renzo Angles et al. “G-CORE: A Core for Future Graph Query Languages”. In: *SIGMOD*. ACM, 2018, pp. 1421–1432. DOI: 10.1145/3183713.3190654.
- [2] Nurzhan Bakibayev, Dan Olteanu, and Jakub Zavodny. “FDB: A Query Engine for Factorised Relational Databases”. In: *Proc. VLDB Endow.* 5.11 (2012), pp. 1232–1243. DOI: 10.14778/2350229.2350242.
- [3] Alin Deutsch et al. “Graph Pattern Matching in GQL and SQL/PQ”. In: *SIGMOD*. ACM, 2022, pp. 2246–2258. DOI: 10.1145/3514221.3526057.
- [4] Orri Erling et al. “The LDBC Social Network Benchmark: Interactive Workload”. In: *SIGMOD*. 2015, pp. 619–630. DOI: 10.1145/2723372.2742786.
- [5] Michael J. Freitag et al. “Adopting Worst-Case Optimal Joins in Relational Database Systems”. In: *VLDB* 13.11 (2020), pp. 1891–1904. URL: <http://www.vldb.org/pvldb/vol13/p1891-freitag.pdf>.
- [6] Andrey Gubichev and Peter A. Boncz. “Parameter Curation for Benchmark Queries”. In: *TPCTC*. Vol. 8904. Lecture Notes in Computer Science. Springer, 2014, pp. 113–129.
- [7] Pankaj Gupta et al. “WTF: The who to follow service at Twitter”. In: *WWW*. International World Wide Web Conferences Steering Committee / ACM, 2013, pp. 505–514. DOI: 10.1145/2488388.2488433.
- [8] Pranjal Gupta, Amine Mhedhbi, and Semih Salihoglu. “Columnar Storage and List-based Processing for Graph Database Management Systems”. In: *Proc. VLDB Endow.* 14.11 (2021), pp. 2491–2504. DOI: 10.14778/3476249.3476297.
- [9] Amine Mhedhbi and Semih Salihoglu. “Optimizing Subgraph Queries by Combining Binary and Worst-Case Optimal Joins”. In: *Proc. VLDB Endow.* 12.11 (2019), pp. 1692–1704. DOI: 10.14778/3342263.3342643.
- [10] David Mizell, Kristyn J. Maschhoff, and Steven P. Reinhardt. “Extending SPARQL with graph functions”. In: *BigData*. IEEE Computer Society, 2014, pp. 46–53. DOI: 10.1109/BigData.2014.7004371.
- [11] Hung Q. Ngo, Christopher Ré, and Atri Rudra. “Skew strikes back: New developments in the theory of join algorithms”. In: *SIGMOD Rec.* 42.4 (2013), pp. 5–16. DOI: 10.1145/2590989.2590991.
- [12] Dan Olteanu and Maximilian Schleich. “Factorized Databases”. In: *SIGMOD Rec.* 45.2 (2016), pp. 5–16. DOI: 10.1145/3003665.3003667.
- [13] Mark Raasveldt and Hannes Mühleisen. “Don’t Hold My Data Hostage - A Case For Client Protocol Redesign”. In: *Proc. VLDB Endow.* 10.10 (2017), pp. 1022–1033. DOI: 10.14778/3115404.3115408.
- [14] Mark Raasveldt and Hannes Mühleisen. “DuckDB: An Embeddable Analytical Database”. In: *SIGMOD*. ACM, 2019, pp. 1981–1984. DOI: 10.1145/3299869.3320212.
- [15] Oskar van Rest et al. “PGQL: a property graph query language”. In: *GRADES at SIGMOD*. 2016. DOI: 10.1145/2960414.2960421.
- [16] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003. ISBN: 978-0-89871-534-7. DOI: 10.1137/1.9780898718003.
- [17] Gábor Szárnyas et al. “The LDBC Social Network Benchmark: Business Intelligence Workload”. In: *Proc. VLDB Endow.* 16.4 (2022), pp. 877–890. URL: <https://ldbouncil.org/docs/papers/ldbcsnb-bi-vldb-2022.pdf>.
- [18] Todd L. Veldhuizen. “Leapfrog Triejoin: a worst-case optimal join algorithm”. In: *CoRR* abs/1210.0481 (2012). URL: <http://arxiv.org/abs/1210.0481>.