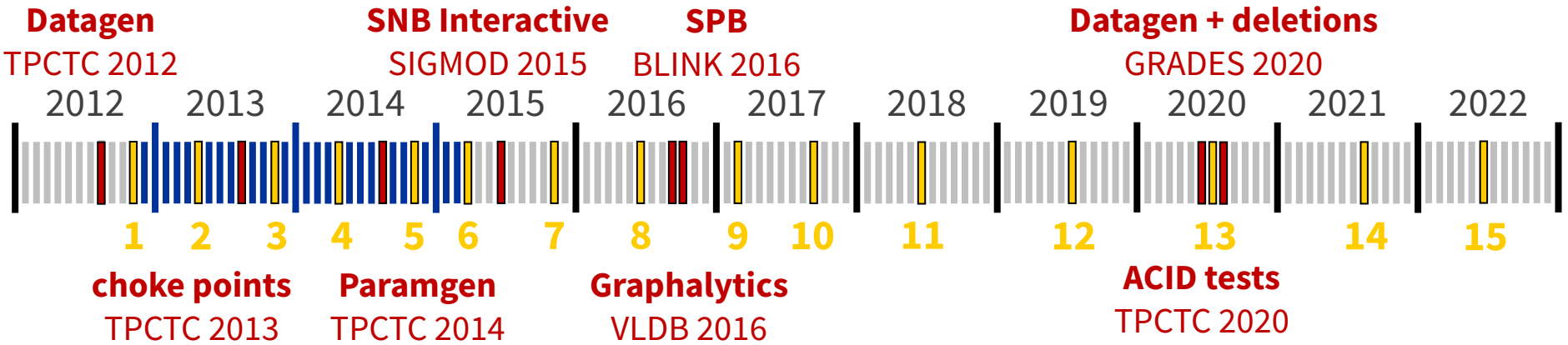
The LDBC Social Network Benchmark

Gábor Szárnyas, Jack Waudby, Benjamin Steer, Peter Boncz

with contributions from former members of the SNB Task Force

LDBC project, benchmark papers & meetings



EU FP7 project | **TUC meetings** | **Benchmark papers**

The LDBC Social Network Benchmark

- Initial mission during the EU project (2012-2015): develop a benchmark suite
  - Continued after the project, now in the making for almost 10 years
  - Influential in both academia and industry
- From 2015: new LDBC groups around query languages
  - Graph Query Language Task Force (G-CORE)
  - GQL Formal Semantics Working Group
  - Existing Languages Working Group
  - Property Graph Schema Working Group
- LDBC has a growing membership of individuals and organizations

This presentation is a summary of the LDBC Social Network Benchmark. We provide an overview of the benchmark and codify some lessons learnt.

For an overview of the LDBC, see the talk titled “The Linked Data Benchmark Council”

Data sets

- Graph schema
- Correlated data
- Deletions
- The Datagen project

Social network domain

Disclaimer: It is now established that serving as the primary database for a social network is *not the primary use case* of graph databases.

That said: It is a widely understood domain with interesting graph data structures. Additionally, it makes it easy to argue about correlations in the graph such as:

- “People are *Germany* are more likely to be called *Joachim* than in *Italy*”
- “People in the *France* make more trips to *Belgium* than people in *Mexico* to *Japan*”

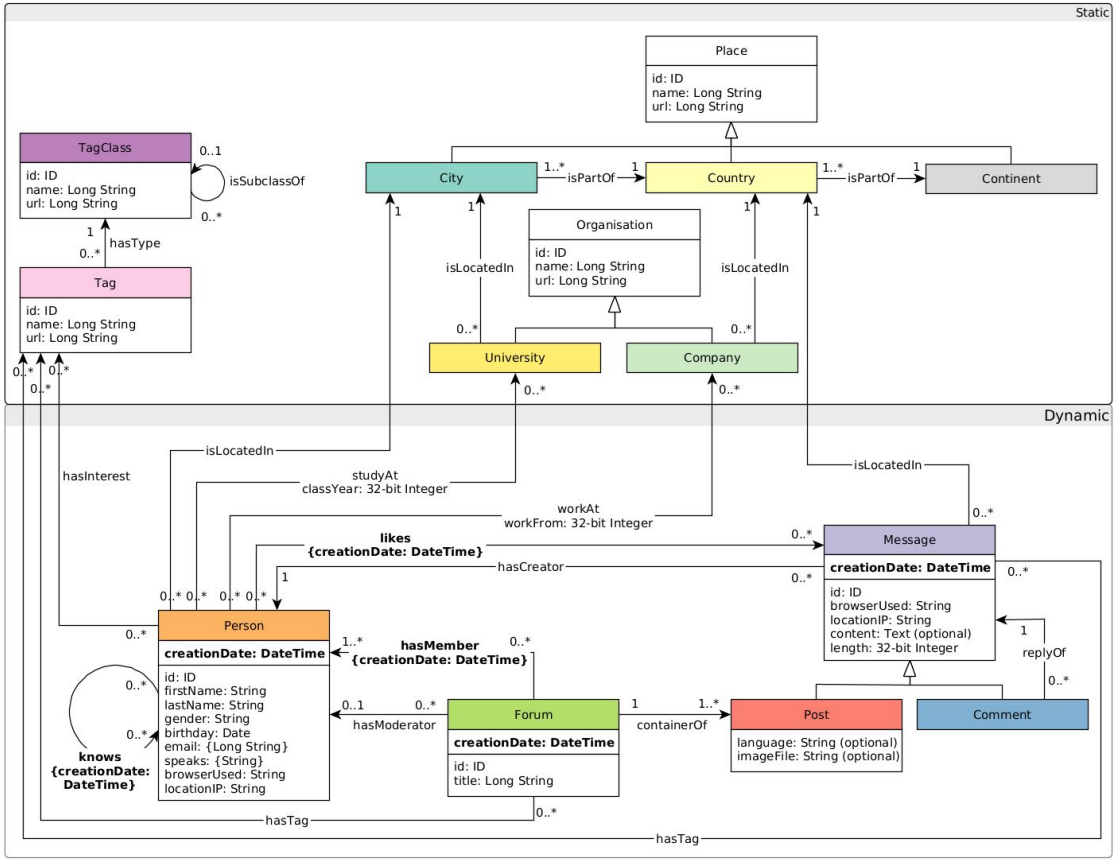
The generated graphs are realistic *to some extent* but not fully. The goal is to add some realistic correlations which query engines can exploit when optimizing the queries.

Statistics

Network of Person nodes, trees of Messages/TagClasses/Places

Statistics for scale factor 1:

- 3M nodes, 17M edges
- 11k Persons, avg. degree of knows edges: 39.4
- Branching factors
  - Message tree: 3.2
  - TagClass tree: 3.7
  - Place tree: 12.4



Graph schema

The graph is a **labelled property graph**. All edges are directed except the Person-knows-Person edges, which are *undirected*.

Edge types (between node types) can be categorized as follows:

- Bipartite: most edge types form a bipartite subgraph, e.g. Forum-hasMember-Person
- Network: Person nodes form network along the knows edges
- Hierarchies:
  - TagClasses: a rooted tree of TagClass nodes (root: “Thing”)
  - Places: a non-rooted tree of 3 levels (Continent, Country, City)
  - Messages: each thread is a rooted tree with a Post root node and Comment nodes

Data generator (Datagen)

The Datagen produces a **property graph** data set

The graph is fully dynamic: **inserts** and **deletes** with realistic distributions

Distributed generation for scalability:

- The Hadoop-based Datagen was used for the Interactive workload
- It was migrated to Spark in 2020, which is now used for the BI workload

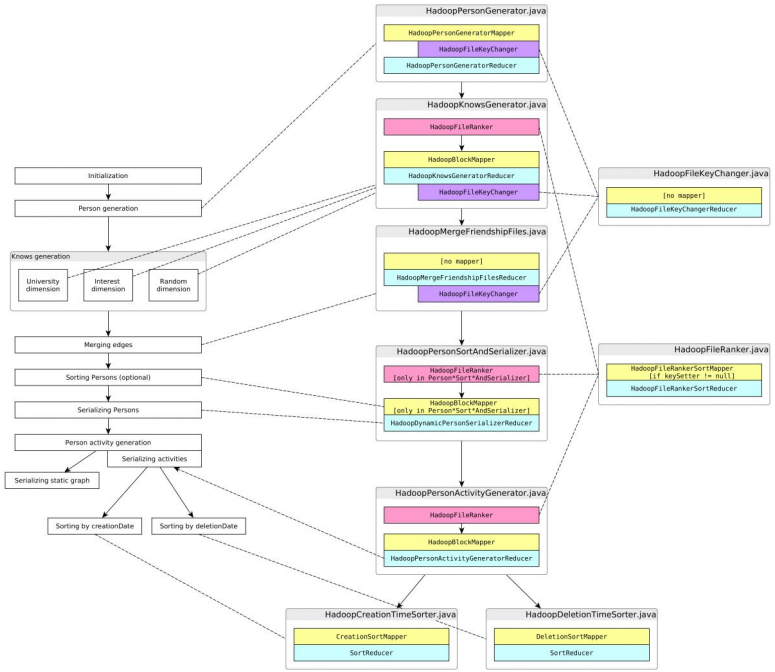
📃 S3G2: a Scalable Structure-correlated Social Graph Generator, TPCTC 2012
📺 LDBC SNB Datagen: Under the hood by Arnau Prat, 9th LDBC TUC meeting, 2017
📃 *Supporting dynamic graphs in SNB Datagen* by J. Waudby et al., GRADES-NDA 2020
📃 *Speeding up LDBC SNB Datagen*, blogpost, 2020

Data generator (Datagen)

Graphs are produced using a distributed data processing framework

- Earlier versions used Hadoop
- Migrated to Spark in 2020

Capable of producing output with different serializers (CSV variants, Turtle).



📃 *Speeding up LDBC SNB Datagen*, blogpost, 2020

Refresh operations

The “dynamic” part of the graph is changing throughout the benchmark. This puts systems using static data structures (such as plain CSR) at a disadvantage.

Depending on the workload of SNB, the refresh operations are different:

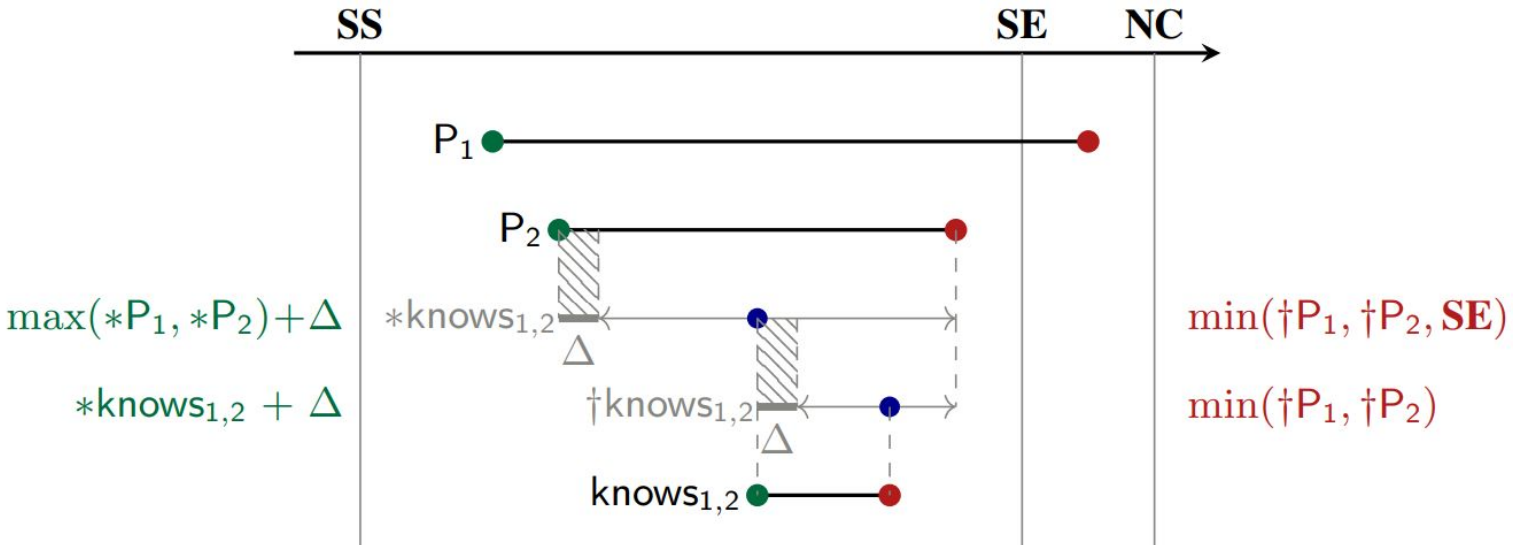
Interactive: New Persons/Forums/Messages are *inserted* along with their edges

BI: Same *inserts* plus the same type of entities are also subject to *deletes*

Generating deletions is challenging as it necessitates assigning a lifespan to each entity during generating, which takes into account how certain deletions are cascading (e.g. deleting an entire Forum or a Message thread) which has a significant impact on the distribution of the data.

Lifespan management, example 1

When can a Person-knows-Person edge exist? Its ∗*creation date* and †*deletion date* values are selected from intervals constrained by those of its Person endpoints.



📃 *Supporting fully dynamic graphs in LDBC SNB* [GRADES-NDA'20]

Lifespan management, example 2

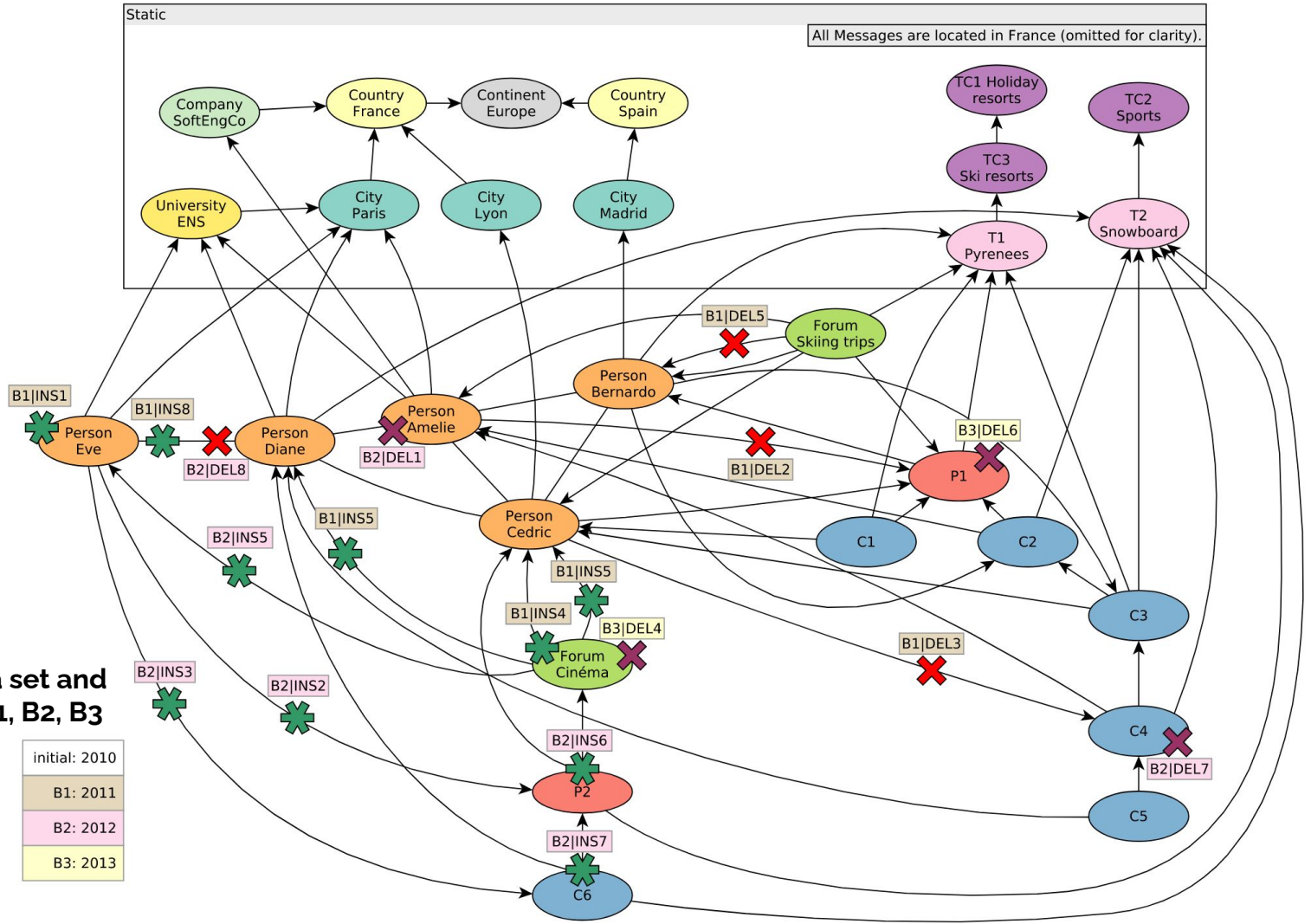
To create a Comment, its parent Message and its create Person has to exist and the person has to be a member (hm) of the Forum where the Message's root Post is located.

3.6.5.2 Comment

A Comment $comm$ is created by Person p as a reply to Message m. Comments are only made in Walls and Groups. Comment always occur within γ days of their parent message following a power-law distribution with mean 6.85 hours.

- $*comm \in \left[\max(*m, *hm) + \Delta,\ \min(\dagger m, \dagger hm, *m + \gamma\,\mathrm{d}, \mathbf{SE}) \right)$
- $\dagger comm \in \left[*comm + \Delta,\ \min(\dagger m, \dagger hm) \right)$

📃 *Supporting fully dynamic graphs in LDBC SNB* [GRADES-NDA'20]


Static
All Messages are located in France (omitted for clarity).
Company SoftEngCo
Country France
Continent Europe
Country Spain
TC1 Holiday resorts
TC2 Sports
TC3 Ski resorts
University ENS
City Paris
City Lyon
City Madrid
T1 Pyrenees
T2 Snowboard
B1|DEL5
Forum Skiing trips
Person Bernardo
B1|INS1
B1|INS8
Person Eve
Person Diane
Person Amelie
B2|DEL8
B2|DEL1
B1|DEL2
B3|DEL6
P1
B1|INS5
Person Cedric
C1
C2
B2|INS5
B1|INS5
B1|INS4
B3|DEL4
Forum Cinéma
C3
B1|DEL3
B2|INS3
B2|INS2
C4
B2|DEL7
B2|INS6
P2
C5
B2|INS7
C6
Initial data set and batches B1, B2, B3
initial: 2010
B1: 2011
B2: 2012
B3: 2013

Workloads

- Interactive workload
- Business Intelligence workload

Comparison of workloads

	Interactive	Business Intelligence
focus	OLTP	OLAP
typical query	2-3 hop neighbourhood queries with filtering	multi-hop/path/subgraph queries with filtering & aggregation
data generator	SNB Hadoop Datagen	SNB Spark Datagen
refresh operations	inserts	inserts and deletes
target metric	total compression ratio, implying the throughput (ops/s)	mean latency (WIP)

Interactive workload

Interactive workload

Scenario: Users browsing a social network and producing content (Forums, Messages)

Queries: 14 complex reads, 7 short reads, 8 insert operations

Audit rules:

- Implementations using imperative code are allowed
- Defining materialized views is allowed if they are constantly maintained

✅ First audited benchmark in 2020, TuGraph by FMA Technologies [FDR]

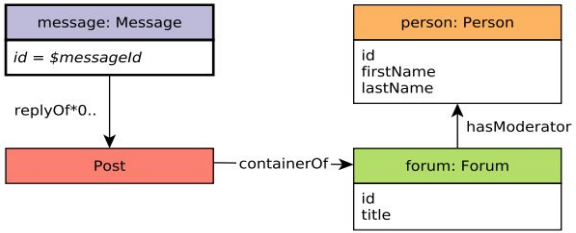
📃 SNB Interactive [SIGMOD'15 paper] [SIGMOD'15 slides]

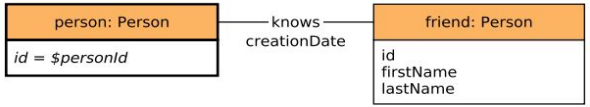
🔗 Benchmark page

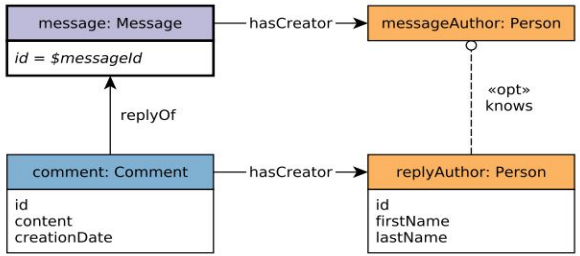
Interactive workload: Queries

Complex queries: Always start from one or two Person nodes, and discover their neighbourhoods (1..2 nodes) or paths between Person nodes.

Short queries: Discover the neighbourhood of a Person or a Message node.



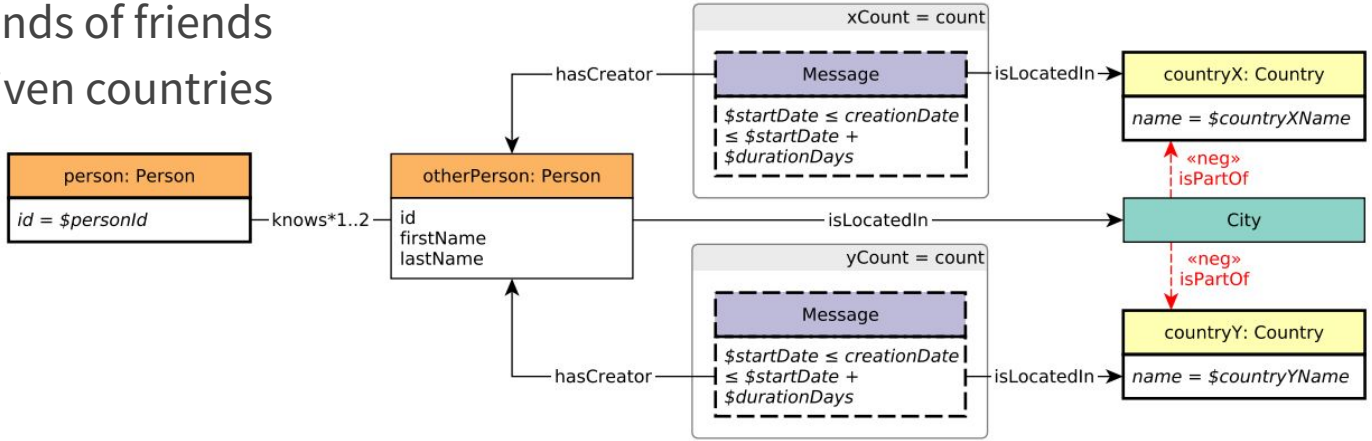




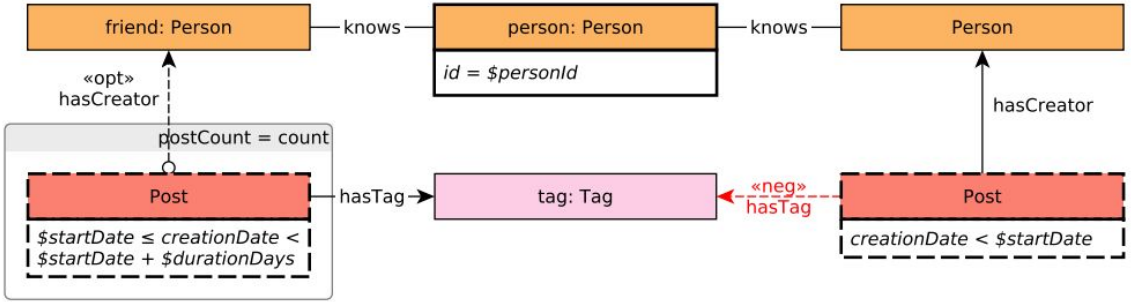
Insert operations: each operations inserts a node (an connects it to its neighbourhood) or an edge between existing nodes.

Interactive workload: Complex queries

Q3: Friends and friends of friends that have been to given countries

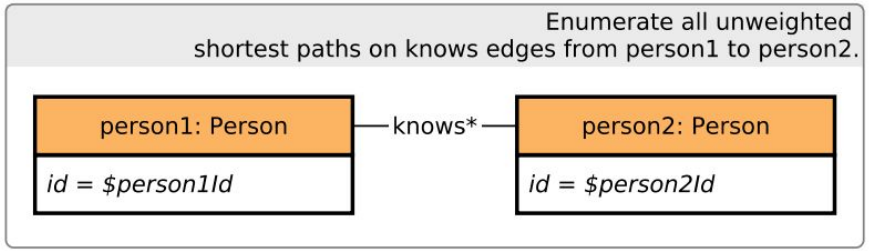


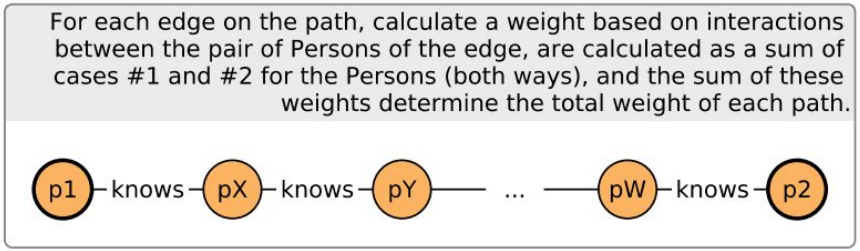
Q4: New topics

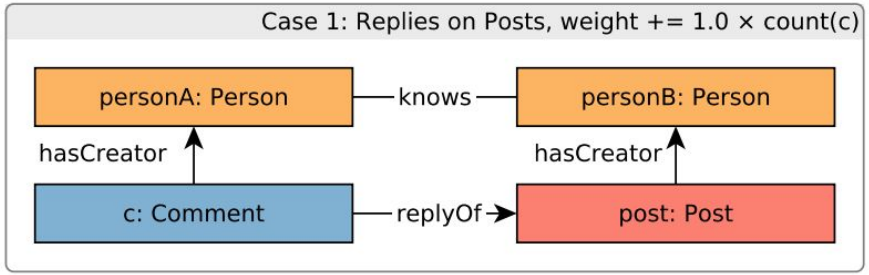


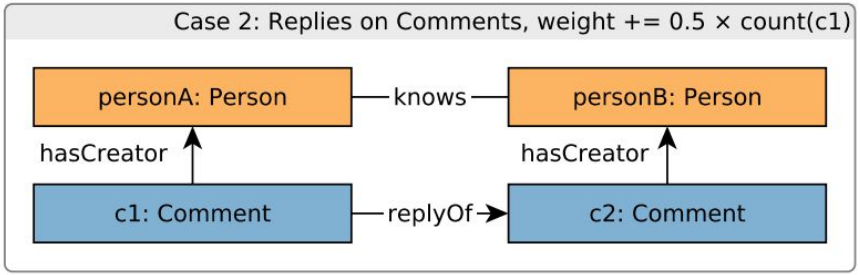
Interactive workload: Complex queries

Q14: Trusted connection paths









Interactive workload: Execution of queries

- **Insert operations' issue times are taken from the update streams generated by the data generator.** These are the times where the actual event happened during the simulation of the social network.
- **Complex reads' times are expressed in terms of update operations.** For each complex read query type, a frequency value is assigned which specifies the relation between the number of updates performed per complex read.
- **For each complex read instance, a sequence of short reads is planned.** [...] The substitution parameters for short reads are taken from the results of previously executed complex reads and short reads. Once a short read sequence is issued (and provided that sufficient substitution parameters exist), there is a probability that another short read sequence is issued. This probability decreases for each new sequence issued. Since the same random number generator seed is used across executions, the workload is deterministic.

(See the specification for more details.)

Interactive workload: Data set

The Datagen produces 3 years worth of data. From this data

- 90% is used the initial data set (separated into static/dynamic directories), and
- 10% is added later in the form of inserts (updates).

These inserts affect the entities in the “dynamic” category (e.g. Person/Message nodes, knows/likes edges). There are 8 insert operations, encoded in a variable-width CSV format:

- insert node: Person, Forum, Comment, Post
- insert edge: knows, hasMember, Comment-hasCreator-Person, Post-hasCreator-Person

t | t_d | 1 | personId | personFirstName | personLastName | gender | birthday | creationDate | locationIP | browserUsed | cityId | languages | emails | tagIds | studyAt | workAt
t | t_d | 2 | personId | postId | creationDate
t | t_d | 3 | personId | commentId | creationDate
t | t_d | 4 | forumId | forumTitle | creationDate | moderatorPersonId | tagIds
t | t_d | 5 | personId | forumId | creationDate
t | t_d | 6 | postId | imageFile | creationDate | locationIP | browserUsed | language | content | length | authorPersonId | forumId | countryId | tagIds
t | t_d | 7 | commentId | creationDate | locationIP | browserUsed | content | length | authorPersonId | countryId | replyToPostId | replyToCommentId | tagIds
t | t_d | 8 | person1Id | person2Id | creationDate

Interactive workload: driver #1

The driver has 3 modes of operation, all starting with a database containing the initial data set.

1. Generate validation data set

- single-threaded, sequential execution
- inputs:
  - query parameters: `substitution_parameters/` dir 🕒
  - update streams: update streams directory with the `updateStream_0_0_{forum,person}.csv` files
- output:
  - `validation_params.csv` file

2. Validate implementation

- single-threaded, sequential execution
- inputs:
  - `validation_params.csv` file
- output:
  - passed/failed validation
  - if failed: expected vs. actual results

Interactive workload: driver #2

3. Execute benchmark

- multi-threaded, concurrent execution
  - some non-deterministic behaviour is possible due to concurrent execution
- inputs:
  - time_compression_ratio value
  - number of read and write threads
  - query parameters: `substitution_parameters/` directory
  - update streams: `updateStream_*.csv` files (for 2*n* write threads, the framework requires *n* `updateStream_*_forum.csv` and *n* `updateStream_*_person.csv` files)
- output:
  - passed/failed schedule audit
  - throughput (operations per second)
  - detailed performance results

Parameter selection

For each generated data set, the Datagen component creates *substitution parameters* (also known as “query parameters” and “query seeds”). Parameters are selected so that the *variance* of the expected execution times is limited. This is a non-trivial task as graph queries are prone to high-variance due to their skewed, power-law degree distribution (exhibited by e.g. the Person-knows-Person subgraph).

Path queries are especially tricky as the execution time has huge variance based on whether the path exist (usually quick to find) or does not exist (usually slow to prove).

There is a `txt` file for each query and each line them corresponds to a query execution.

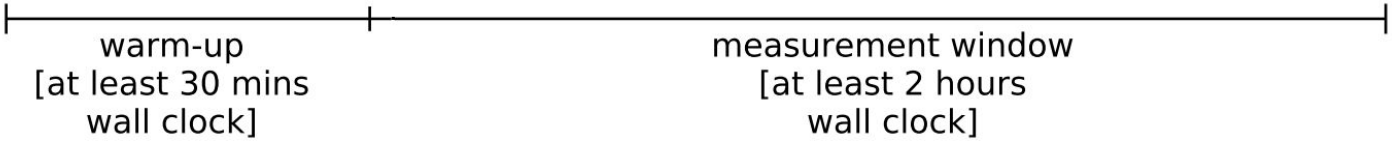
🕒 The datetime values in the `txt` files are represented as UNIX epoch values. The driver converts them into GMT-based timestamps.

📃 Parameter Curation for Benchmark Queries, TPCTC 2014

TCR and valid benchmark runs #1

Implementations compete on the TCR (time compression ratio), i.e. how quickly they can replay a sequence of operations. A TCR of 0.1 means the operations are played at 10x speed. **A lower TCR is better as it indicates a higher throughput.**

For a run to pass the audit, the implementation has to sustain its throughput for 2 hours (after a 30-minute initial warmup whose performance results are discarded). How this is checked is discussed on the next slide.



TCR and valid benchmark runs #2

For a given SF/TCR, implementations have to satisfy the 95% on-time requirement:

In order to pass an audit, 95% of the executed queries must meet the following condition:

$$actual\_start\_time - scheduled\_start\_time < 1\ second$$

That is, 95% of the executed queries have to start in less than 1 second of their originally intended start time. If the system falls behind too much and less than 95% of the queries start on time, the run fails the audit.

In these cases, the test executor is advised to reduce the TCR and start another run.

⚠️ Due to potentially noisy execution environments and slight differences in individual runs (due to multi-threaded execution), it is recommended to leave a bit ‘in reserve’ when calibrating the TCR value.

BI workload

- Analytical queries
- Cyclic subgraphs
- Shortest paths
- Inserts/deletes

Business Intelligence workload

Scenario: Ad-hoc data analytics

Queries: 20 complex reads, 8 insert and 8 delete operations

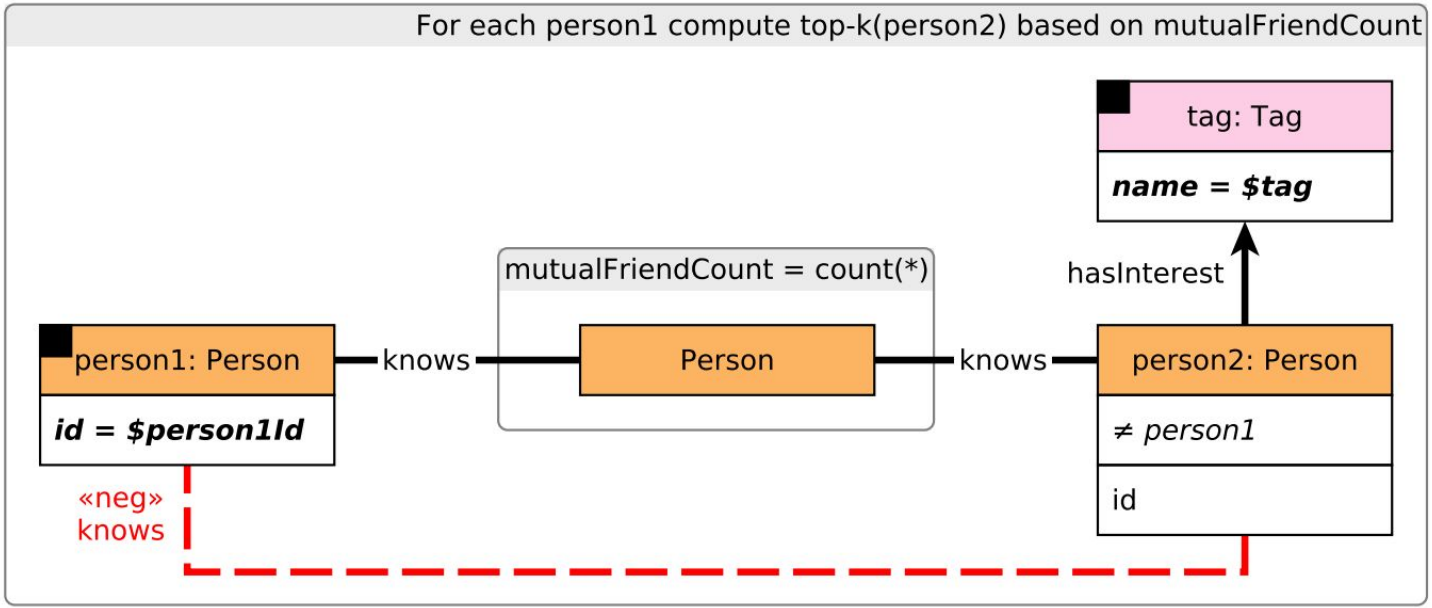
Audit rules:

- Must use a domain-specific query language — unlike Interactive
- Defining views is allowed (if they are maintained) — same as Interactive

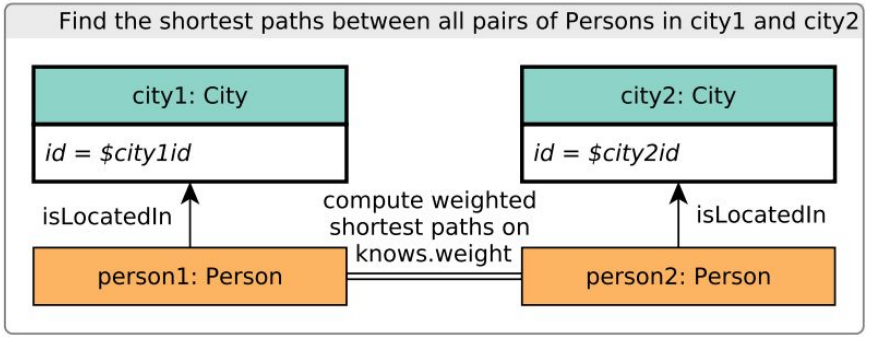
⌛ Recently completed: Deprecated 10 queries and added 5 new ones

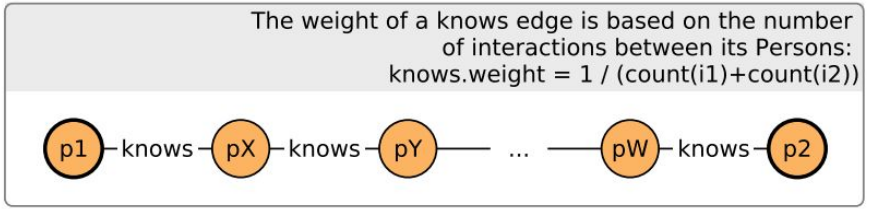
📃 SNB BI [GRADES-NDA'18 paper] [GRADES-NDA'18 slides] [specification]

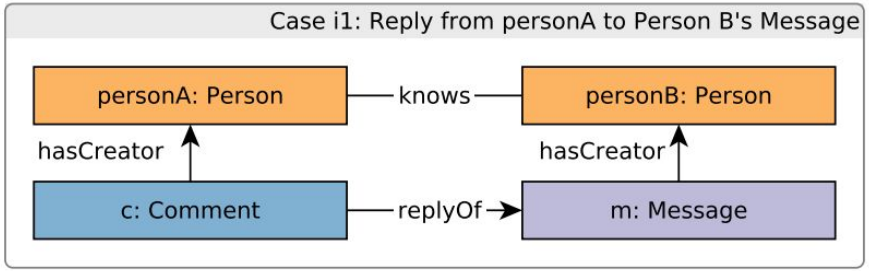
Q18: Dense subgraphs for WCOJ algorithms

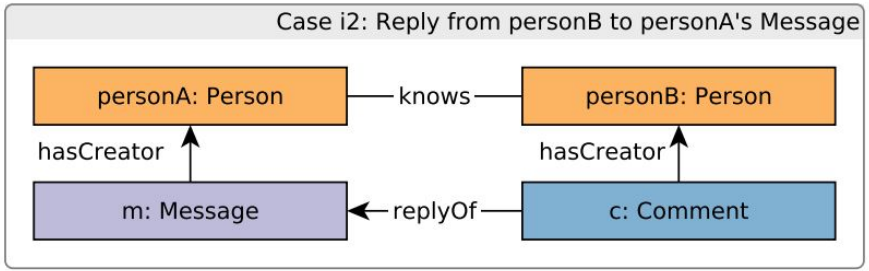


Q19: Weighted shortest paths







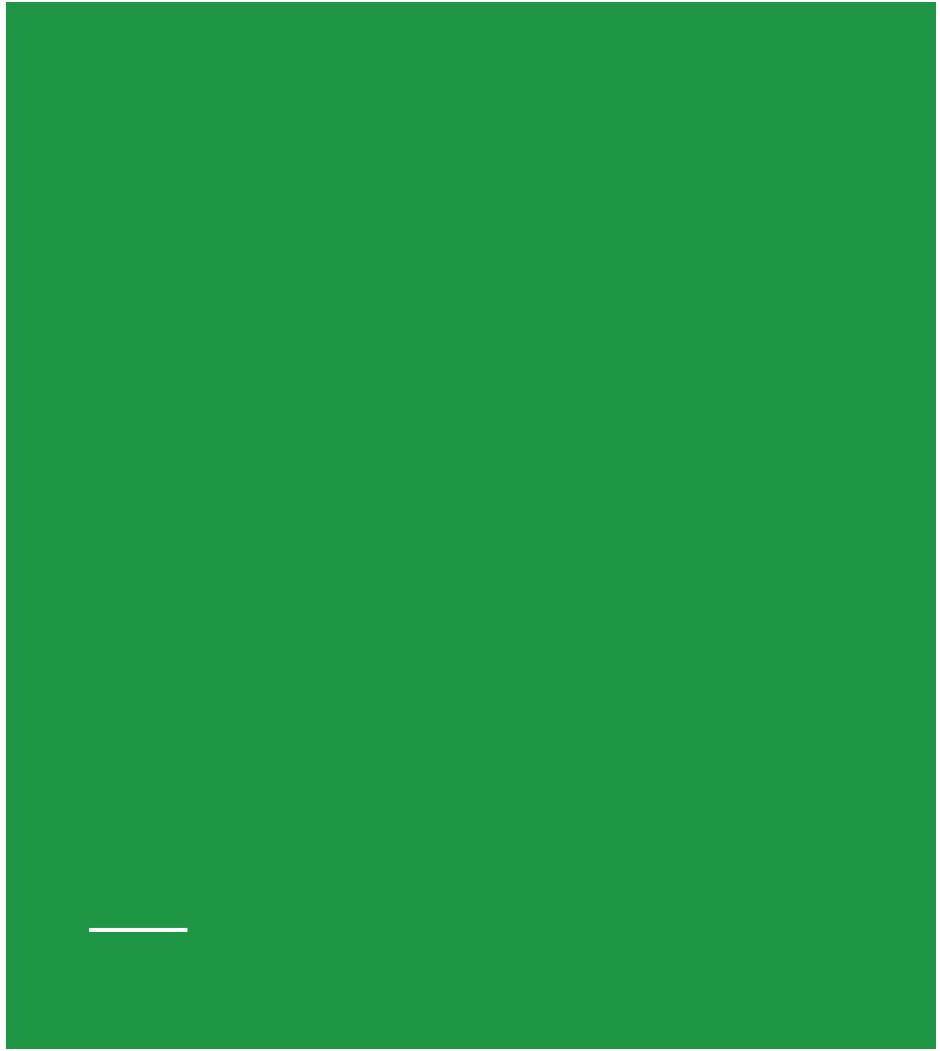


Parameter generator

Implemented in SQL, uses DuckDB 🦆:

- https://github.com/ldbc/ldbc_snb_data_converter/blob/main/paramgen.sh
- https://github.com/ldbc/ldbc_snb_data_converter/blob/main/factor-tables.sql
- https://github.com/ldbc/ldbc_snb_data_converter/blob/main/select-bi-params.sql

Example graph

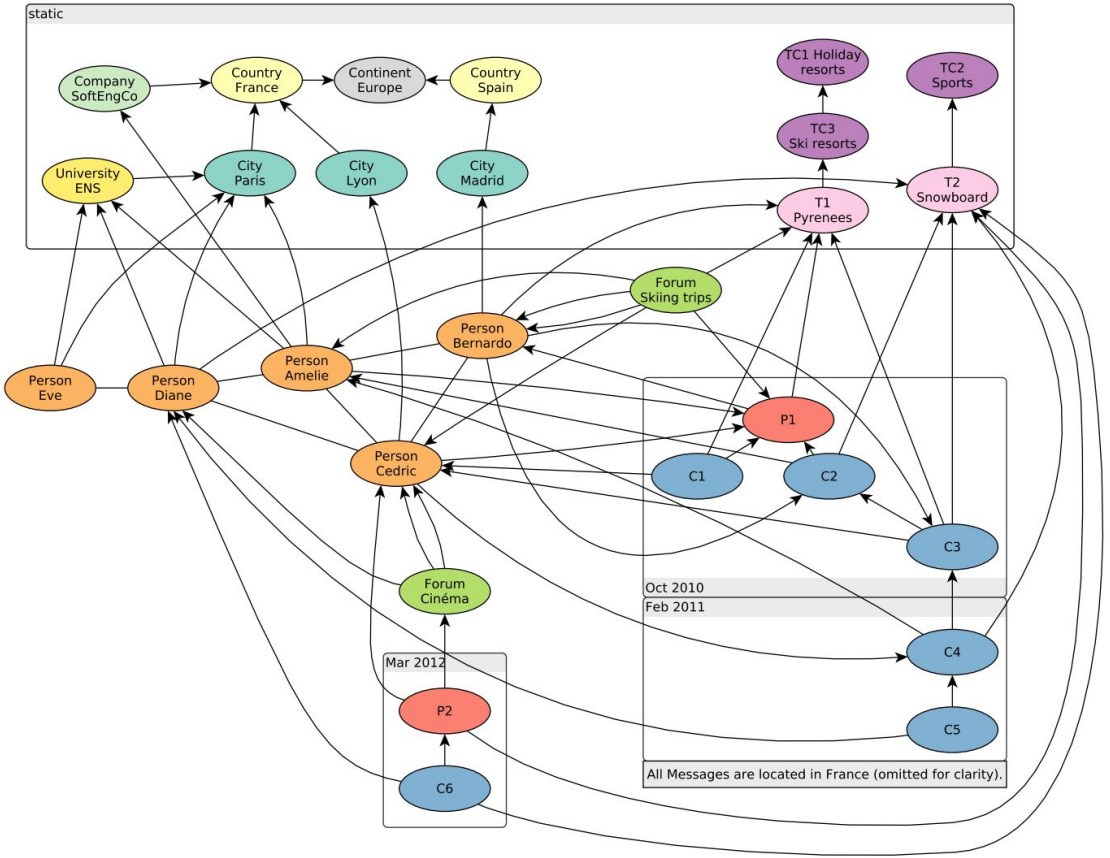


Example graph

~30 nodes and ~60 edges

- Useful for **debugging the toolchain end-to-end.**
- Has at least one match for all BI queries.
- Parametrization for the Interactive workload to be done later.

🔗 Repository



Query design

Choke points and parameters

- Intended query plan
- Choke point analysis
- Parameter curation

Query templates and parameters

Queries are given using a **query template** which can have multiple **input parameters**.

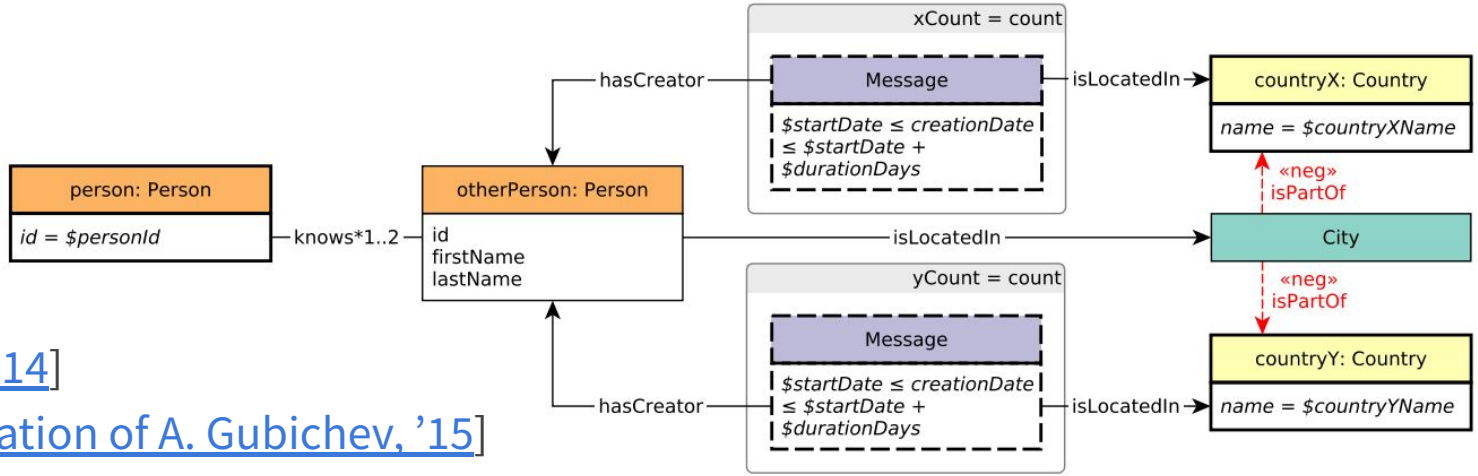
These are **substituted for with different parameters** during execution.

The parameters are produced by the **Paramgen** component of Datagen.

Intended query plan

The **intended query plan** of a query is the ideal execution plan to evaluate that query.
E.g. in Interactive Q3 params can be chosen to produce a large or a small result:

- Neighbouring countries X = Belgium Y = France
- Far away countries X = Mexico Y = Japan



[TPCTC'14]

[Dissertation of A. Gubichev, '15]

Choke points

A choke point is a **difficult aspect of query processing** that has a significant impact on the performance of the query *when evaluated using the intended query plan*.

The TPCTC'12 paper analyzed TPC-H based on the lessons learnt when implementing the benchmark on Vectorwise, Virtuoso, and HyPer.

Examples:

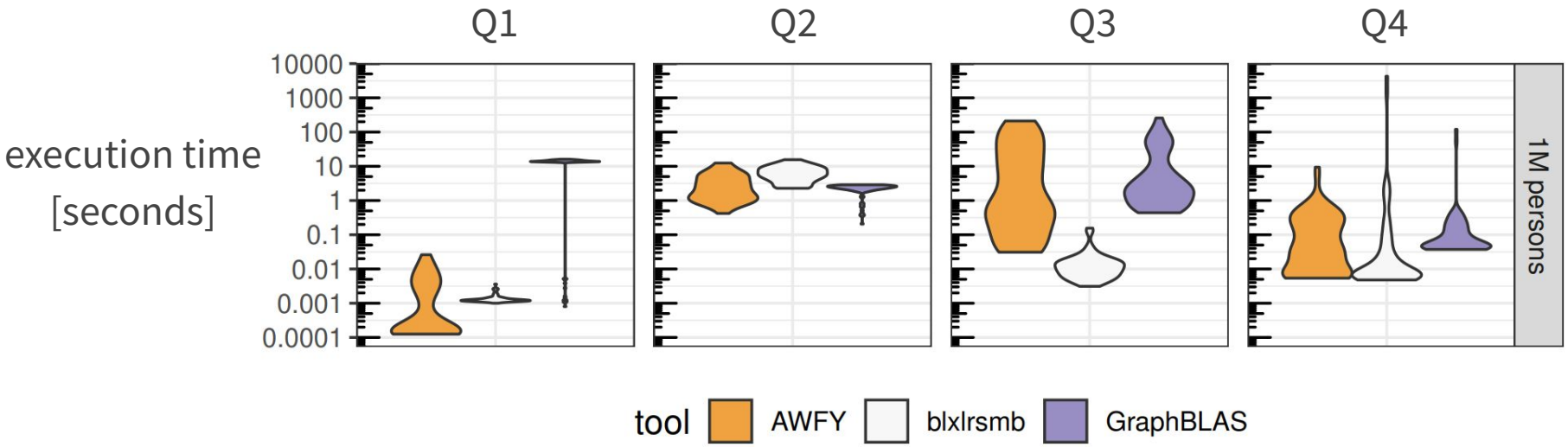
- Join ordering
- Efficient antijoins and outer joins
- Handling paths

📃 TPC-H choke points [TPCTC'12], Quantitative analysis of TPC-H CPs [VLDB'20]

Parameter curation

Goal: Reduce variance of query execution times, make results easier to interpret.

A negative example: Q1-Q4 of the SIGMOD 2014 Contest *without parameter curation*



A GraphBLAS solution to the SIGMOD 2014 Programming Contest [HPEC'20]

Selecting entities for deletes

One can think of this as a special case of parameter curation: based on whether we select a Person

- with many friends and a lot of content or
- with little activity

The cost of performing the delete operation varies significantly.

⏳ This is currently being worked out as part of tuning the distribution of the deletes.

The Ecosystem of the LDBC SNB

- Specification
- Datagen
- Driver
- Implementations

Specification

Challenges:

- Coming up with a representative workload which has the “optimum” difficulty
- Establishing auditing rules (inspired by TPC)
- Specifying queries in an unambiguous way
- Creating a graphical notation (inspired by the graph transformation community)

We believe to have successfully tackled these in the latest specification.

📃 [Latest on GitHub] [Stable on arXiv]

Driver

Lots of challenges regarding concurrent execution: tracking dependencies between refresh operations while maintaining a high throughput.

The driver implements these features in Java and is by far the largest project in SNB.

- main project: 38k LOC
- tests: 22k LOC

The new BI queries and deletes are already supported by the driver.

⏳ Adding support for batched refresh operations is ongoing work.

📃 EU Deliverable “Benchmarking transactions” [D2.2.3]

Implementations

Reference implementations:

- PostgreSQL [SQL]: a row-oriented RDBMS
- Neo4j [Cypher]: a graph database management system
- DuckDB [SQL]: a column-oriented OLAP RDBMS with a vectorized runtime
- Umbra [SQL]: a column-oriented HTAP RDBMS with a compiled runtime, WIP

Audited systems:

- Sparksee (2015)
- Virtuoso (2015)
- FMA TuGraph (2020)
- More coming…

How to create a new SNB implementation?

Steps to create an auditable SNB implementation

Creating a new SNB Interactive implementation #1

It is recommended to base a new implementation on an existing one:

- Graph DBMSs: use the Neo4j/Cypher implementation
- Relational DBMSs: use the PostgreSQL implementation

Pick a data set serializer. In general:

- Graph DBMSs: use data sets produced by the CsvComposite serializer
- Relational DBMSs: use data sets produced by the CsvMergeForeign serializer

Creating a new SNB Interactive implementation #2

1. Generate the required data sets. Use SF1, 3, and 10 for validation. For benchmarks, SF30+ is required.
2. Implement a **bulk loader** which loads the initial data set. Test it with a small data set (available in the `cypher/test-data/` and `postgres/test-data/` directories) and with an SF1 data set.
3. Fork the **SNB Interactive repository** and create a new Maven subproject.
4. Add a **Java client** to the DBMS as a Maven dependency (e.g. `org.postgresql:postgresql`)
5. Implement the **complex read queries**:
    a. Create the query implementations and their glue code in the `*Db` and `*QueryStore` classes.
    b. Turn the update and short operations off, then use the ***create-validation-parameters* mode** to generate the validation data set with an existing implementation.
    c. Use the ***validation* mode** to check the correctness of the queries on the SF1 data set.

Creating a new SNB Interactive implementation #3

6. Implement the **short read queries** and the **insert operations**:
    a. Implement the 7 short queries and 8 insert operations and their glue code.
    b. Create a full validation data set and cross-validate the new implementation against it on SF1 and SF10. Note that the database has to be reset to its initial state between runs: use the `scripts/snapshot-database.sh` and `scripts/restore-database.sh` scripts.
7. Use the ***benchmark* mode** to perform a benchmark run.
8. Determine the **best `total_compression_ratio` value** for benchmarks.
    a. The `driver/determine-best-tcr.sh` script can help find this value.
    b. Ensure that the warmup plus benchmark runs execute for 2.5h+ in total.
9. Implement the **ACID test suite** and ensure the system passes it.
10. Perform a **recovery test** by killing the system during a benchmark run (e.g. `kill -9`, `reboot`) and checking whether the inserted entities are in the database after restarting.

Implementation guidelines

- For best performance, using multiple read and write threads is a must. These are configurable separately (see the Interactive repository's README).
- For the Interactive workload, using imperative code is allowed for all queries, including complex, short, update, and ACID test queries.
- For BI, all queries and insert/delete operations must use a domain-specific language.

Auditing process

- Ensure objective comparison
- Drive competition among vendors

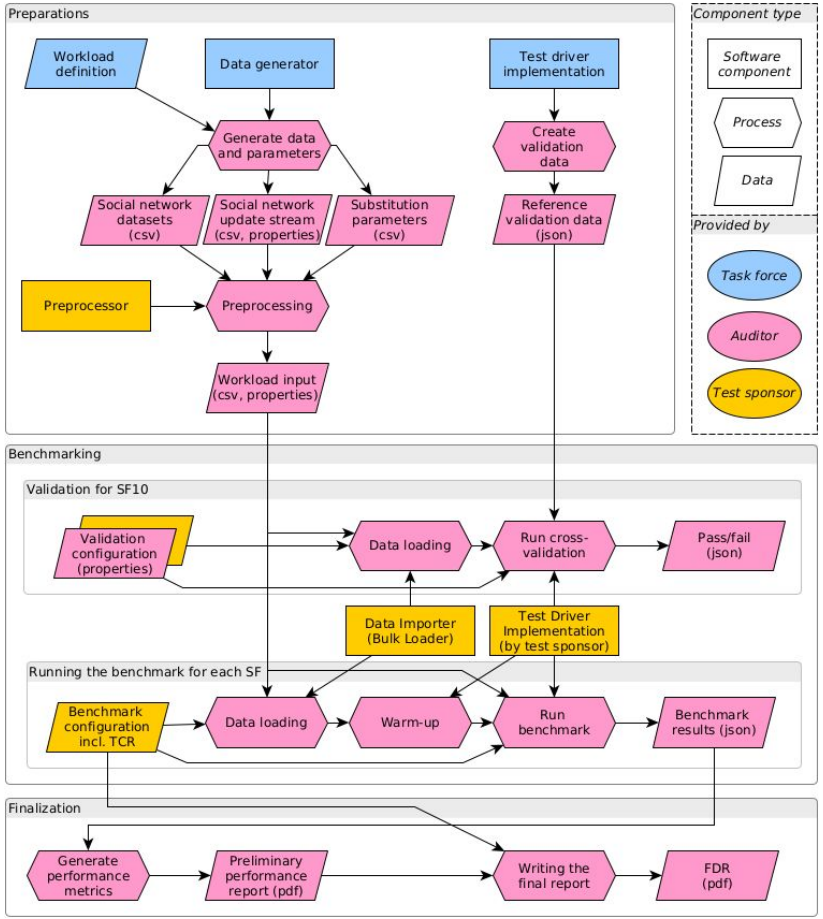
Auditing guidelines

Complex workflow to ensure fair comparison.

TPC has lots of rules to prevent cheating (including the use of “benchmark specials”), sometimes going as far as deprecating entire benchmarks such as TPC-D.

For **LDBC**, audited benchmark results:

- Are produced by an independent auditor
- Can be published as “LDBC benchmark results”

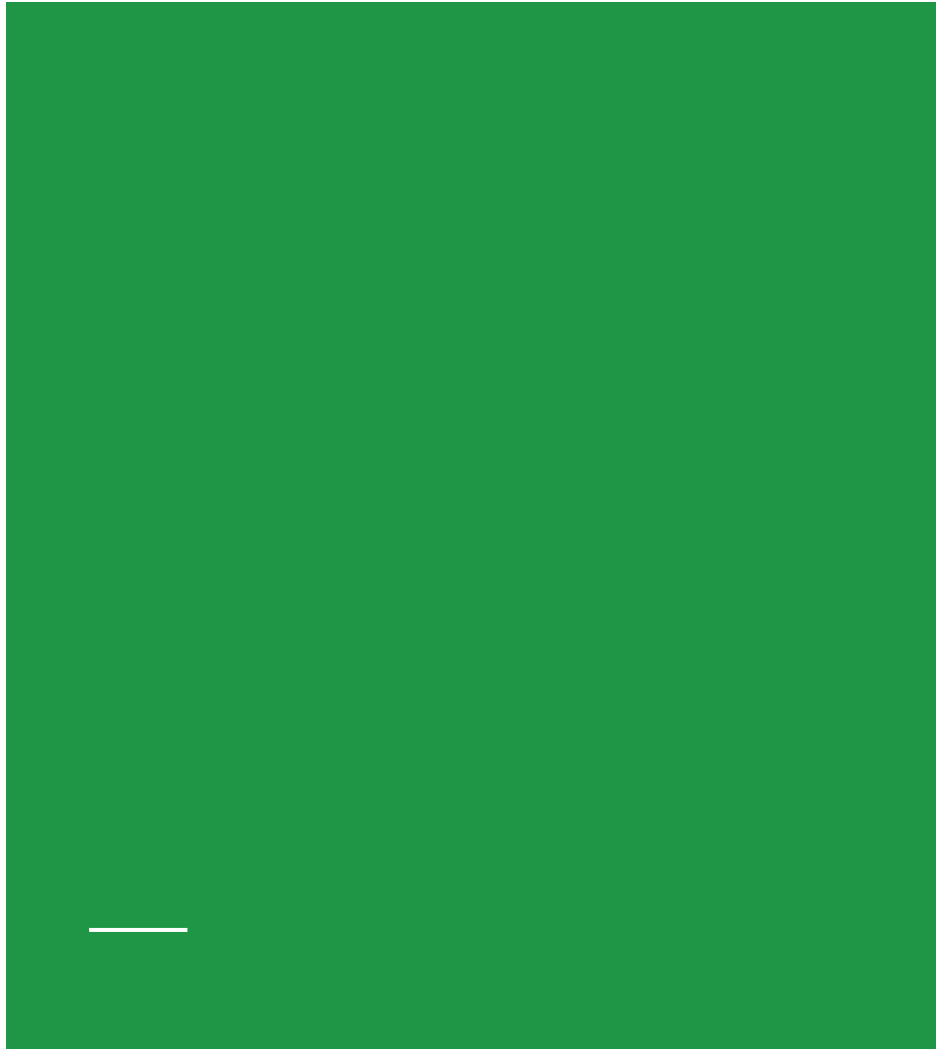


ACID tests

Ensure that the isolation level claimed by the DBMS is enforced. 📜 [TPCTC'20]

Database	C	RB	Isolation Level	G0	G1a	G1b	G1c	OTV	FR	IMP	PMP	LU	WS
Neo4j 3.5	⊗	⊗	Read Committed	⊗	⊗	⊗	⊗	⊗	↯	↯	↯	⊗	↯
Neo4j 4.0	⊗	⊗	Read Committed	⊗	⊗	⊗	⊗	⊗	↯	↯	↯	↯	↯
Memgraph	⊗	⊗	Snapshot Isolation	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	↯
Dgraph	⊗	⊗	Snapshot Isolation	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗
JG/BerkeleyDB	⊗	⊗	Read Uncommitted	⊗	⊗	⊛	⊗	↯	↯	⊗	↯	↯	⊖
JG/BerkeleyDB	⊗	⊗	Read Committed	⊖	⊗	⊛	⊖	↯	⊖	⊗	⊖	↯	⊖
JG/BerkeleyDB	⊗	⊗	Repeatable Read	⊖	⊗	⊛	⊖	⊖	⊖	⊗	⊖	⊖	⊖
JG/BerkeleyDB	⊗	⊗	Serializable	⊖	⊗	⊛	⊖	⊖	⊖	⊗	⊖	⊖	⊖
JG/Cassandra	⊗	⊗	Read Uncommitted	⊗	⊗	⊛	⊗	↯	↯	⊛	↯	↯	↯
PostgreSQL	⊗	⊗	Read Committed	⊗	⊗	⊗	⊗	⊗	⊗	⊗	↯	⊗	⊗
PostgreSQL	⊗	⊗	Repeatable Read	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊘	⊗
PostgreSQL	⊗	⊗	Serializable	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	◎	⊗

The essential complexity of graph DB benchmarks

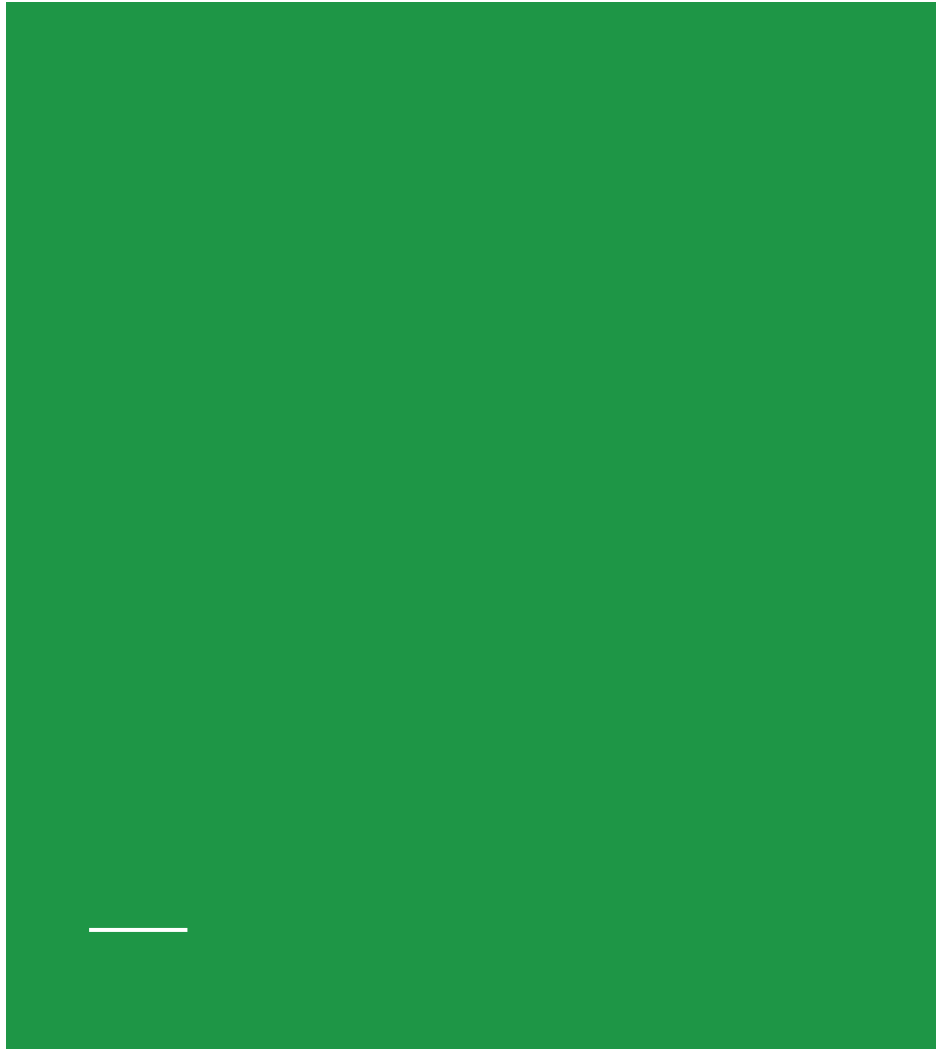


Why are the SNB Interactive/BI workloads so complex?

- Real graph data is correlated [TPCTC'12]
  - Graph data generator with correlations
  - Scalability is important -> distributed generator
  - Need to support multiple layouts (merged FK/projected FK)
- A mature database system has dozens of intertwined optimizations [TPCTC'13]
  - Characterized by choke points in the context of TPC-H
  - ~30 choke points (aggregation, join, data access locality, expressions, correlation, parallelism)
- Benchmark needs parameterized queries [TPCTC'14]
  - Some warmup is required but many systems cache results -> queries need to be parameterized
  - Parameter selection needs to be done carefully to make query times predictable
- Issuing updates needs a sophisticated driver [SIGMOD'15]
  - Update streams need to be able to run concurrently without cross-stream dependencies
- Updates are required to discourage read-only data structures [GRADES-NDA'20]
  - Without updates, materialization of partial results could give an unfair advantage
  - Introducing deletions needs lifespan management
- ACID compliance is required [TPCTC'20]
  - It is difficult to test within the full benchmark, needs a separate benchmark suite

LSQB

Labelled Subgraph Query Benchmark

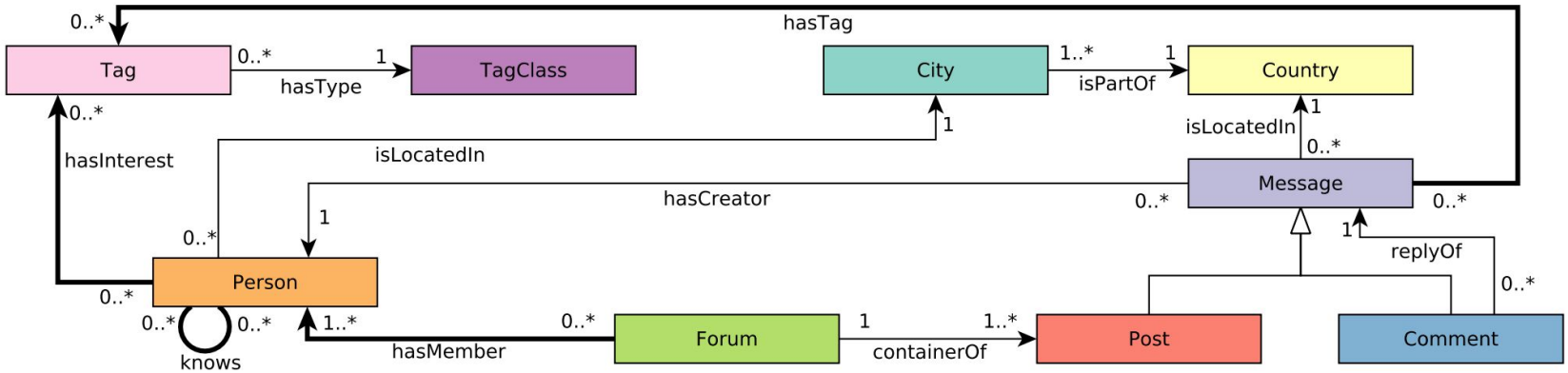


LSQB: Labelled Subgraph Query Benchmark

Note: This in not an official LDBC benchmark but a microbenchmark for developers

Reuse Datagen from the LDBC SNB:

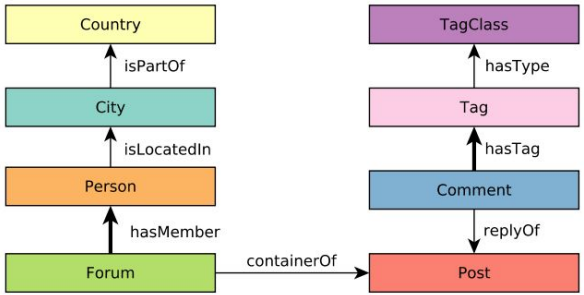
- Same scale factors, same vertex and edges labels
- Lots of many-to-many cardinality edges with interesting distributions
- No updates, no properties, just INT64 identifiers

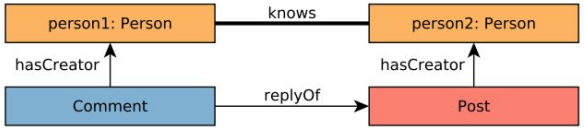


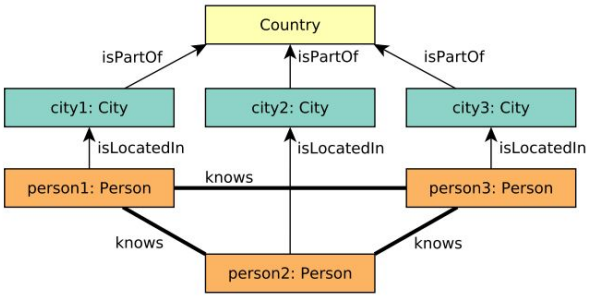
Basic graph patterns

Simplified the queries from the BI workload

All queries are global and use `count(*)` aggregation





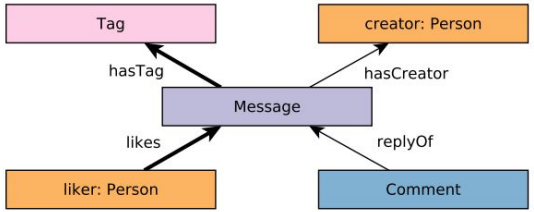


Q1: long path

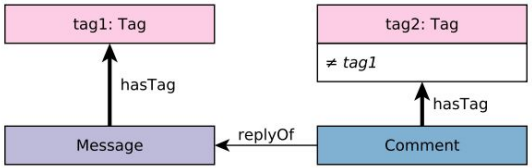
Q2: simple cycle

Q3: triangle

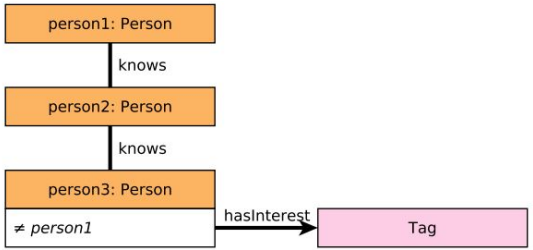
Basic and complex graph patterns



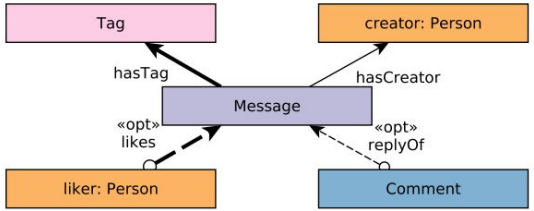
Q4: star



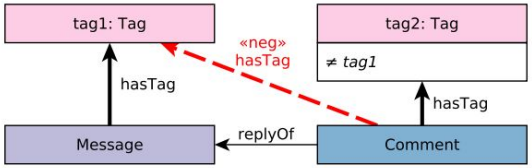
Q5: low-cardinality path



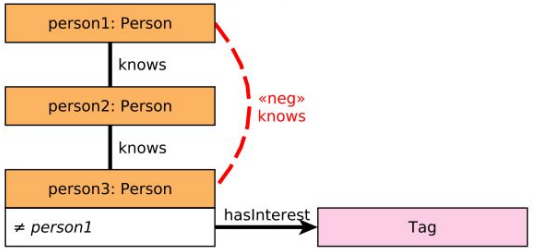
Q6: high-cardinality path



Q7: star
with optional edges



Q8: low-cardinality path
with negative condition



Q9: high-cardinality path
with negative condition

Future work

Future work

⚙️ **Continuous:** We support the adoption of this benchmark and help audits

💡 **New benchmarks:** There are many possibilities to discover, including

- a benchmark with “financial fraud detection”-like queries
- streaming/temporal graph queries
- machine learning (embeddings, GNNs)

We are happy to discuss proposed new graph benchmarks. Feel free to reach out at info@ldbcouncil.org


LDBC
The graph & RDF
benchmark reference