The LDBC Social Network Benchmark Interactive workload v1

Gábor Szárnyas

Linked Data Benchmark Council

Overview

The LDBC Social Network Benchmark is a state-of-the-art benchmark suite for database management systems with a focus on graph processing.

- The *Interactive workload* focuses on transactional systems
- The *BI workload* targets OLAP systems

The workloads operate a social graph which is highly connected and has correlations on attribute values (e.g. names) and structure (e.g. friendship). The queries make use of graph features, e.g. traversing Message trees, finding the k-hop neighbourhoods of Persons, and computing shortest paths between Persons.

This slide deck presents the Interactive v1 workload. Interactive v2 is under development.

Overview of SNB Interactive v1

Transactional workload

Queries start in 1–2 person nodes

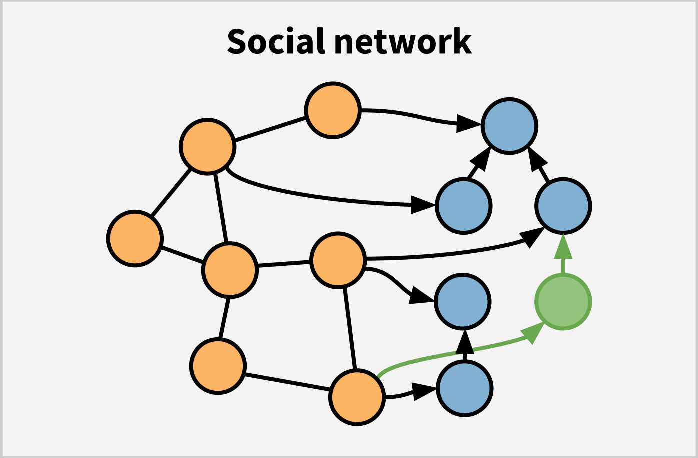
14 complex reads, 7 short reads

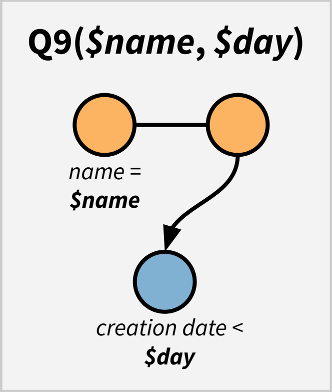
8 insert operations run concurrently

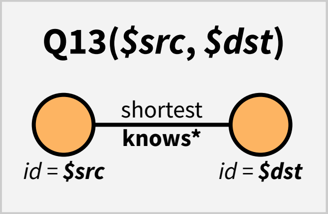
Goal: High throughput (ops/s)

Paper (SIGMOD 2015)

Specification







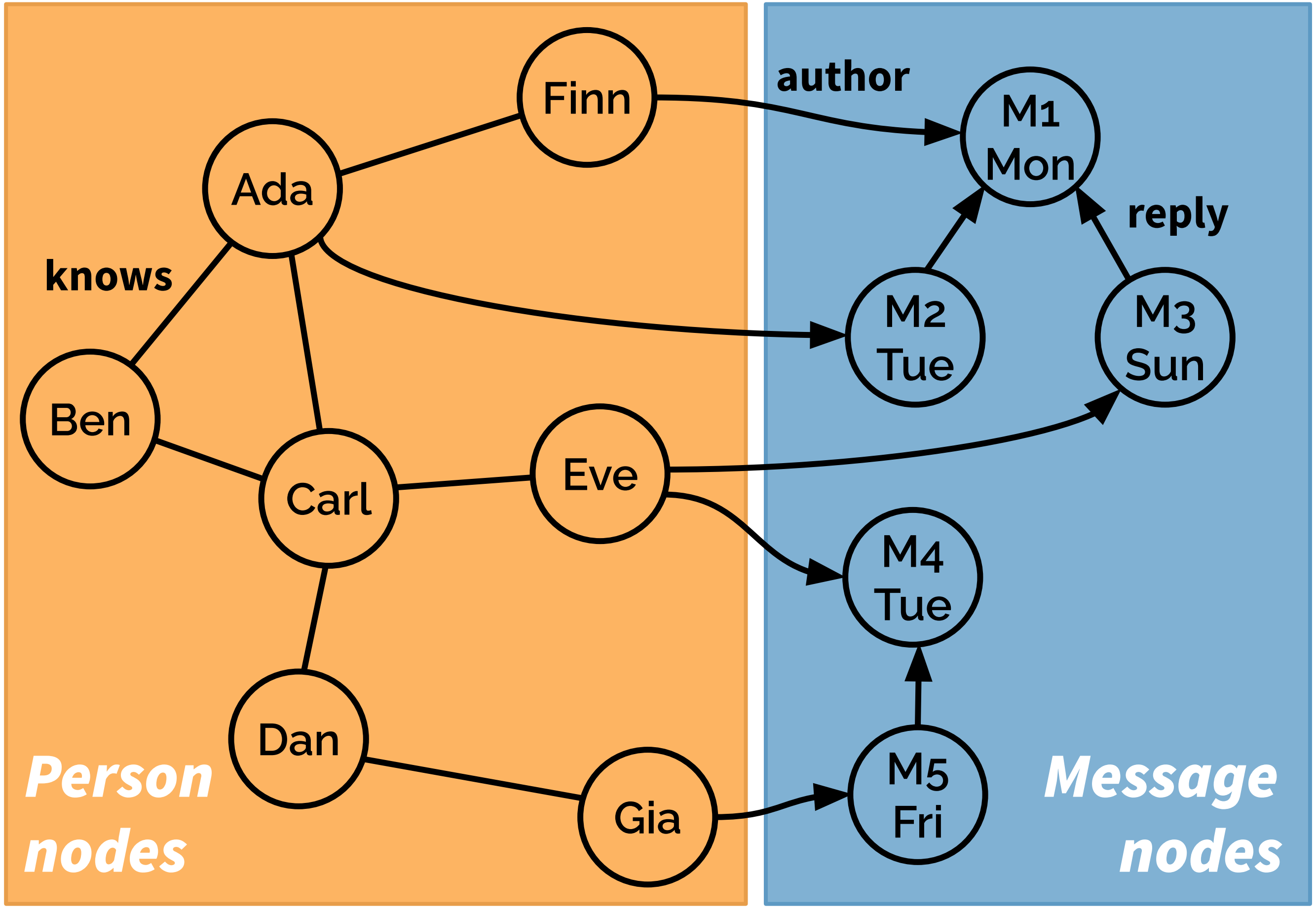
Data set and queries

Data set

Queries

Updates

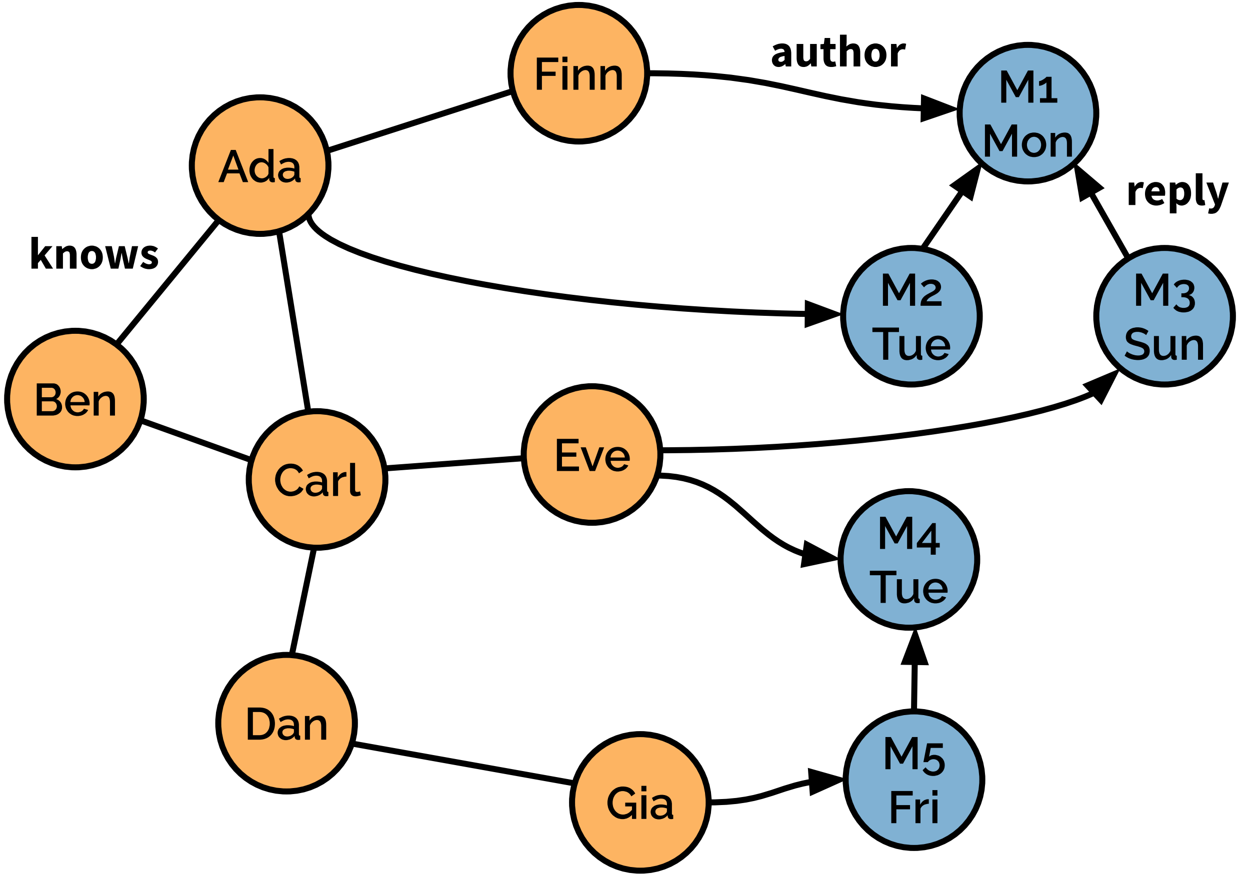
Data set | Queries | Updates



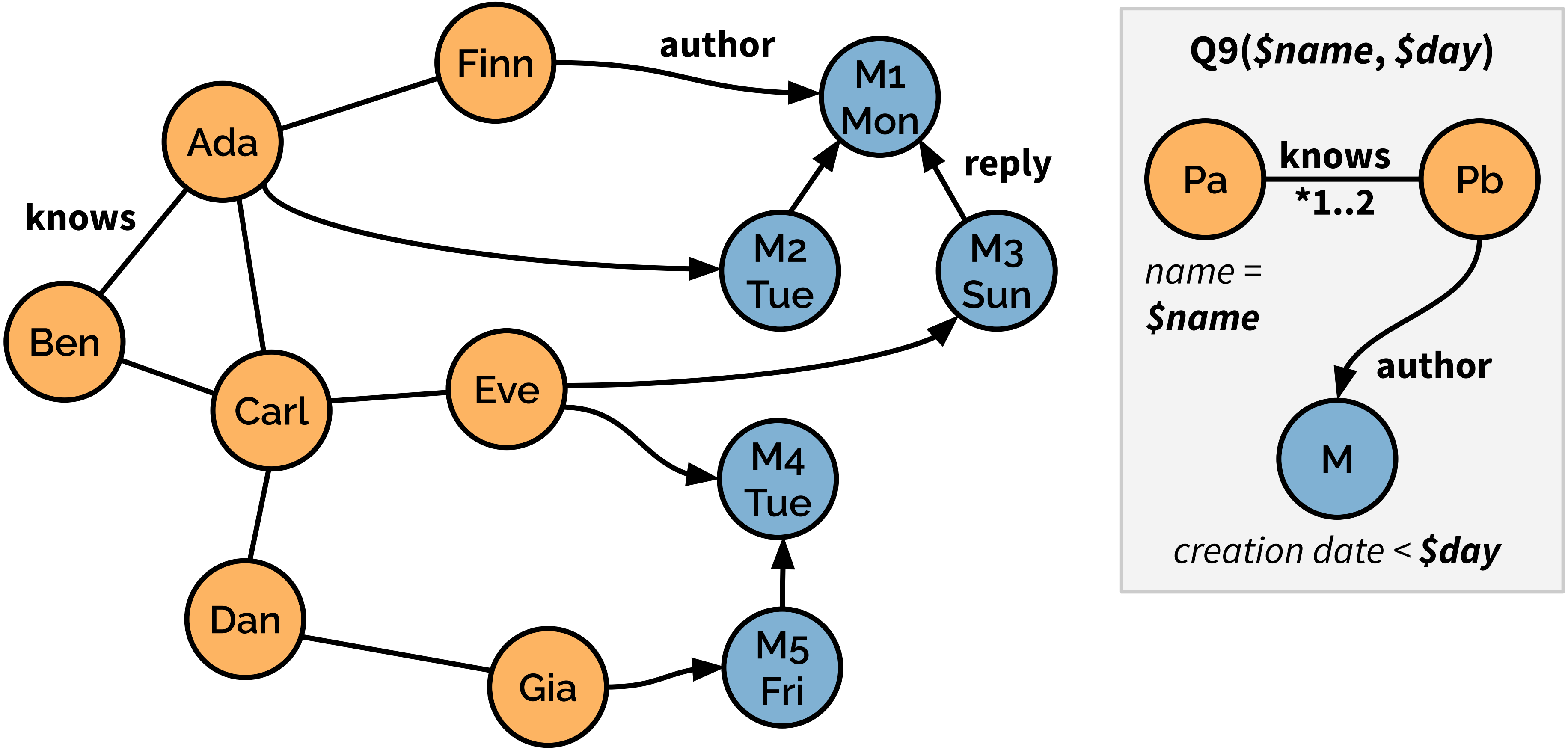
Data set

Queries

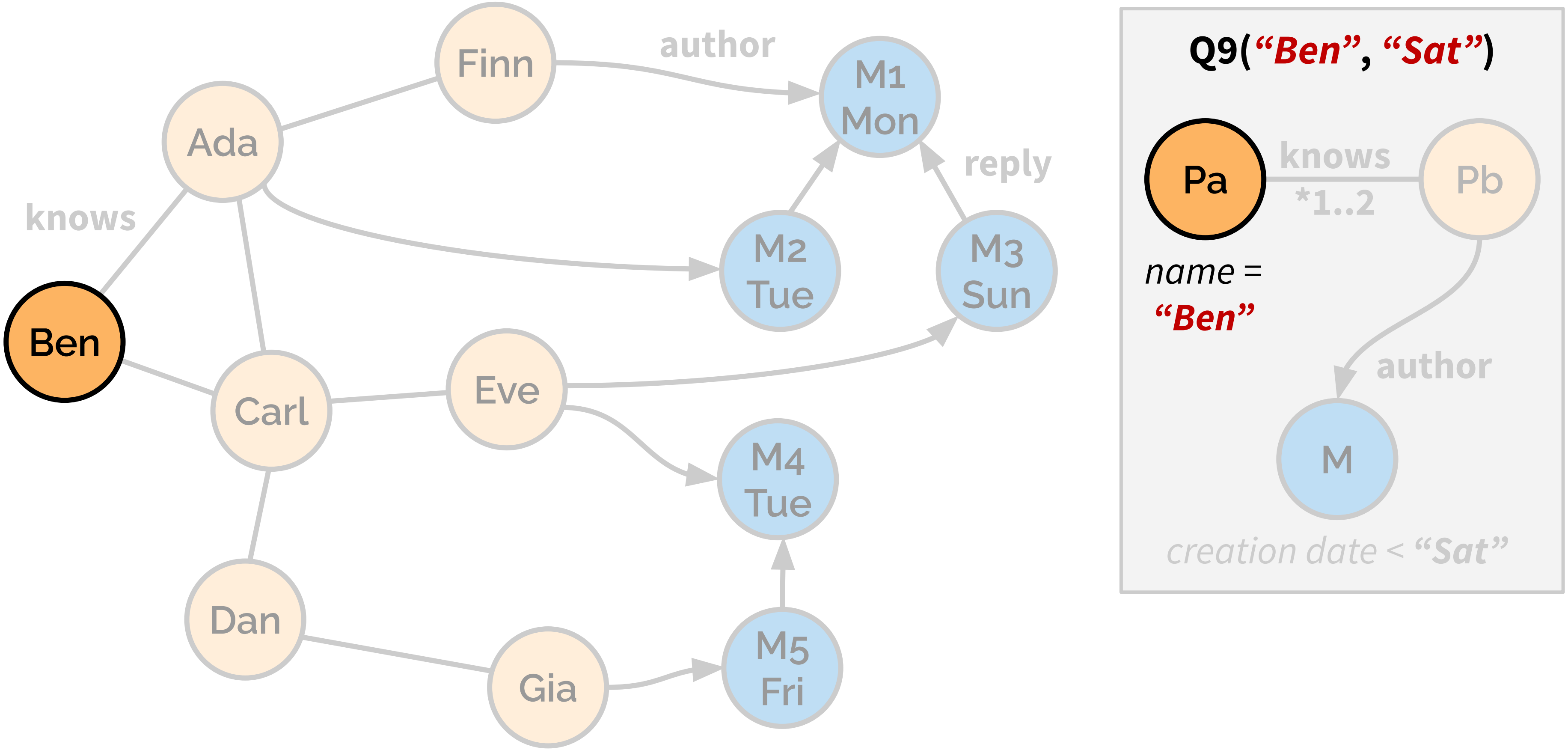
Updates



Data set | **Queries** | Updates



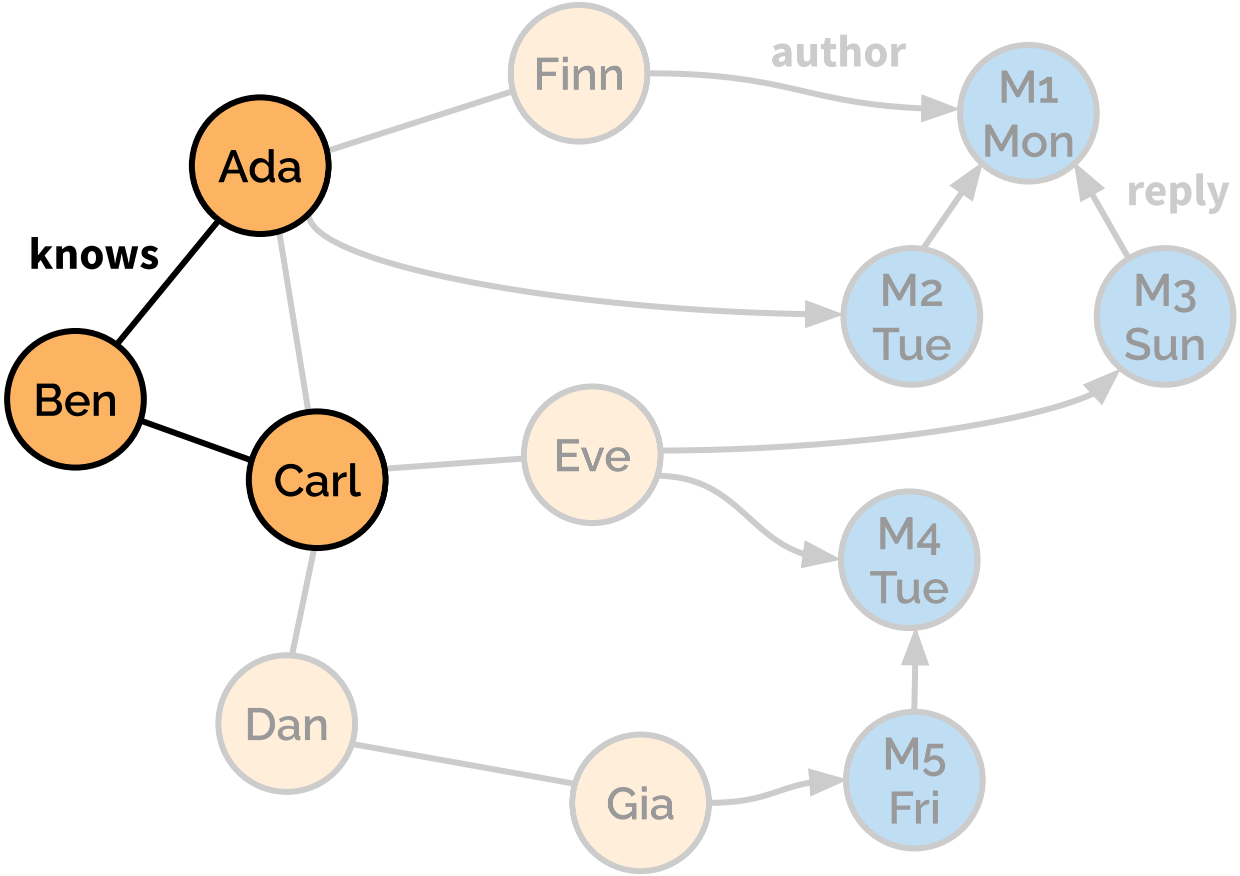
Data set | **Queries** | Updates

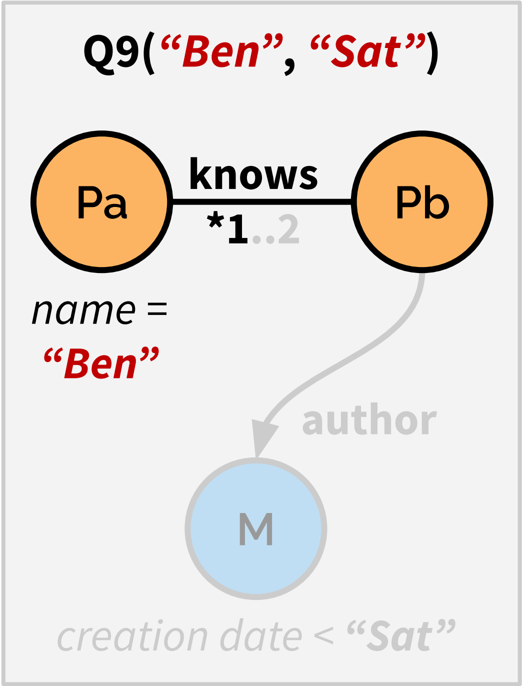


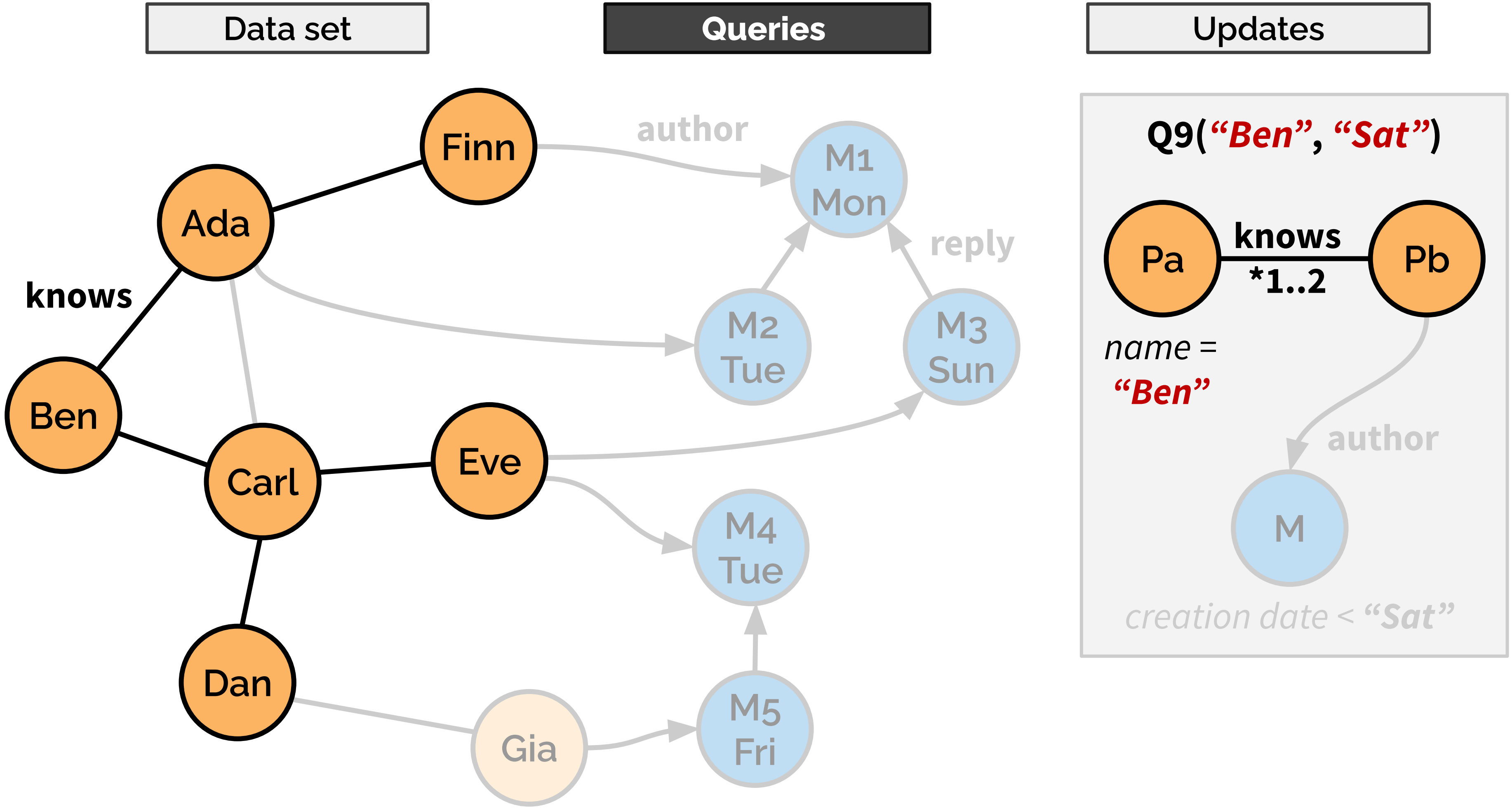
Data set

Queries

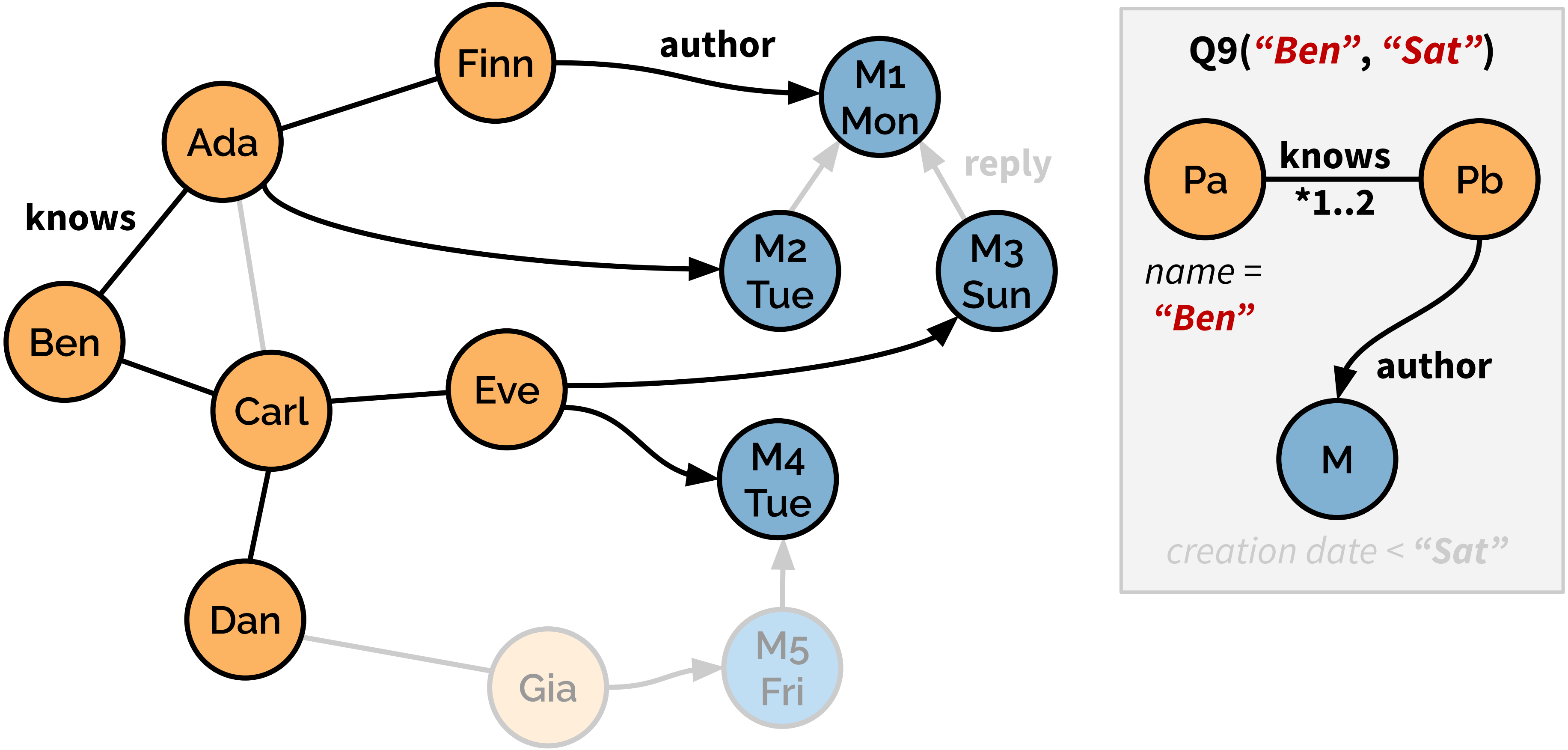
Updates

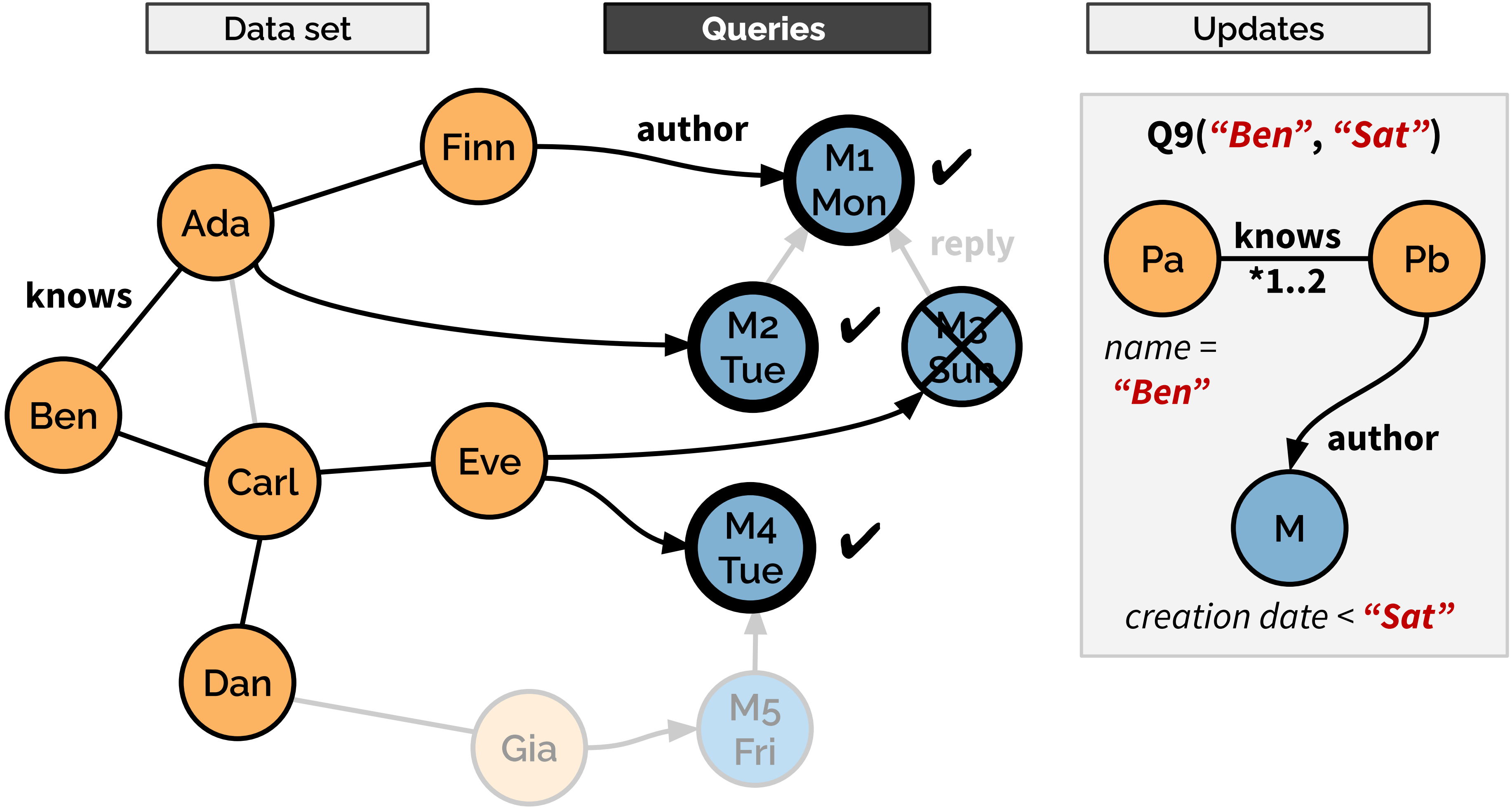





Data set
Queries
Updates
author
Finn
M1
Mon
Ada
reply
knows
M2
Tue
M3
Sun
Ben
Carl
Eve
M4
Tue
Dan
Gia
M5
Fri
Q9("Ben", "Sat")
Pa
knows
*1..2
Pb
name =
"Ben"
author
M
creation date < "Sat"

Data set | **Queries** | Updates



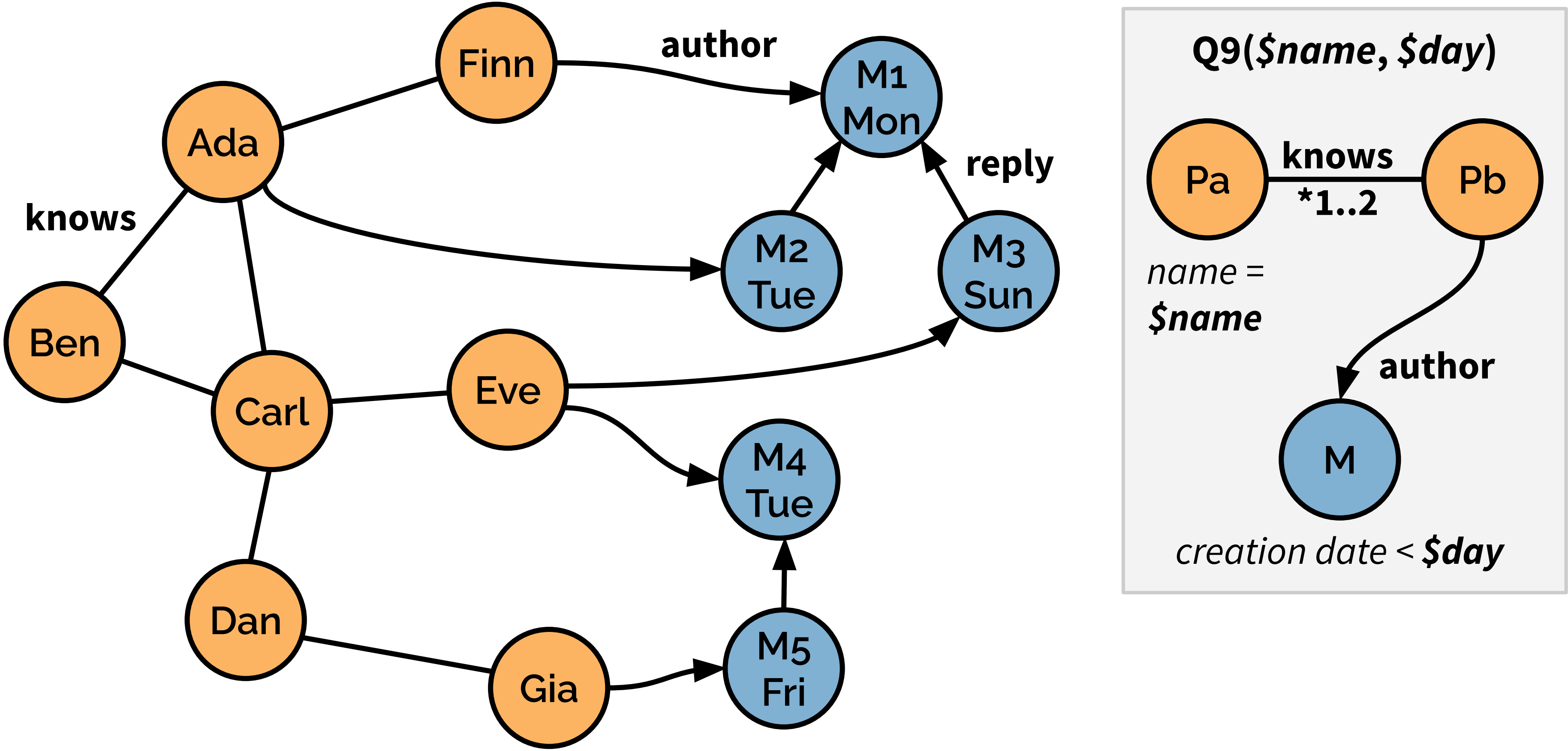


Data set
Queries
Updates
author
knows
reply
Finn
Ada
Ben
Carl
Eve
Dan
Gia
M1 Mon
M2 Tue
M3 Sun
M4 Tue
M5 Fri
✔
✔
✔
Q9("Ben", "Sat")
Pa
knows
*1..2
Pb
name = "Ben"
author
M
creation date < "Sat"

Data set

Queries

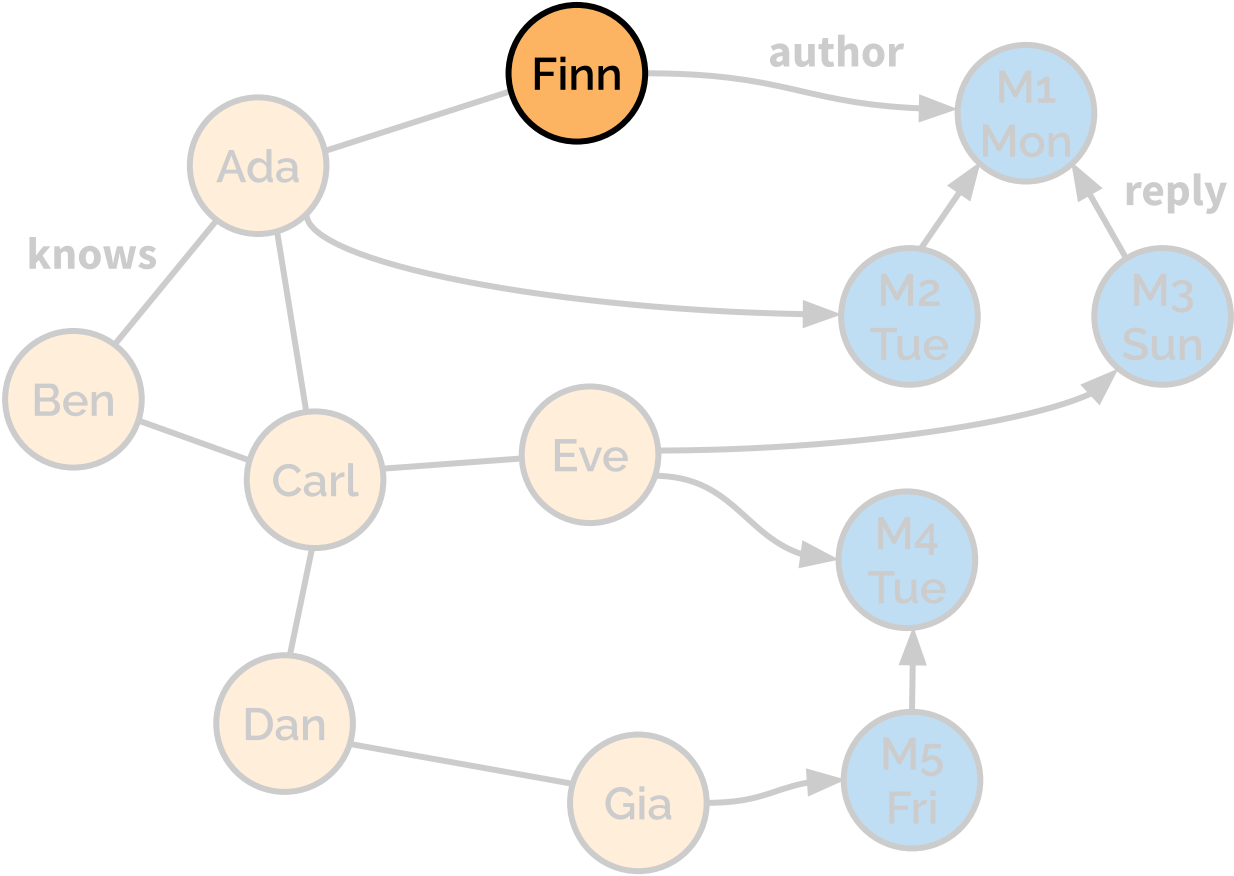
Updates

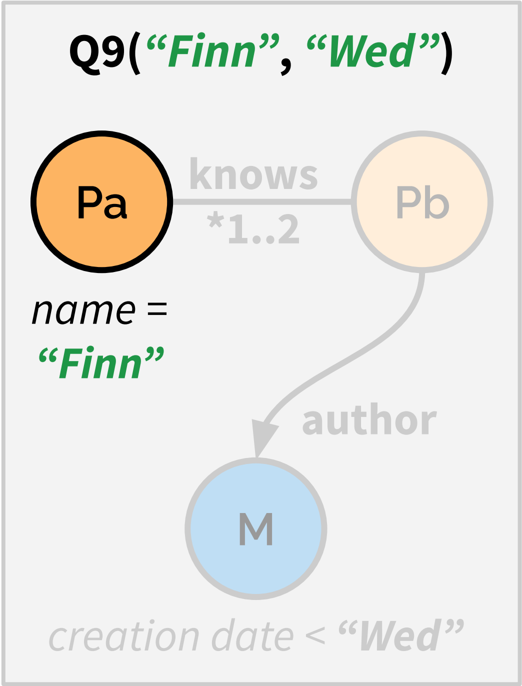


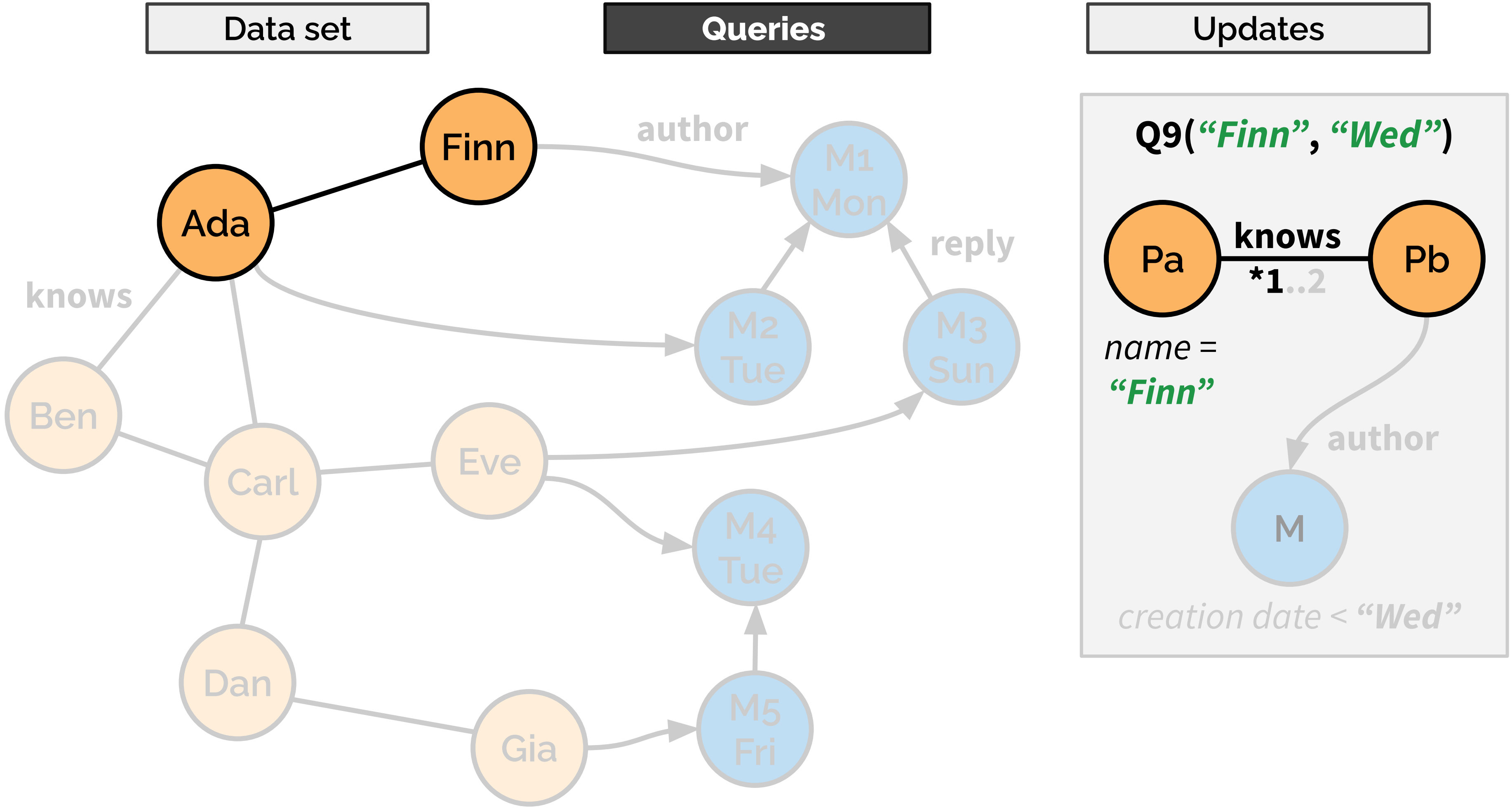
Data set

Queries

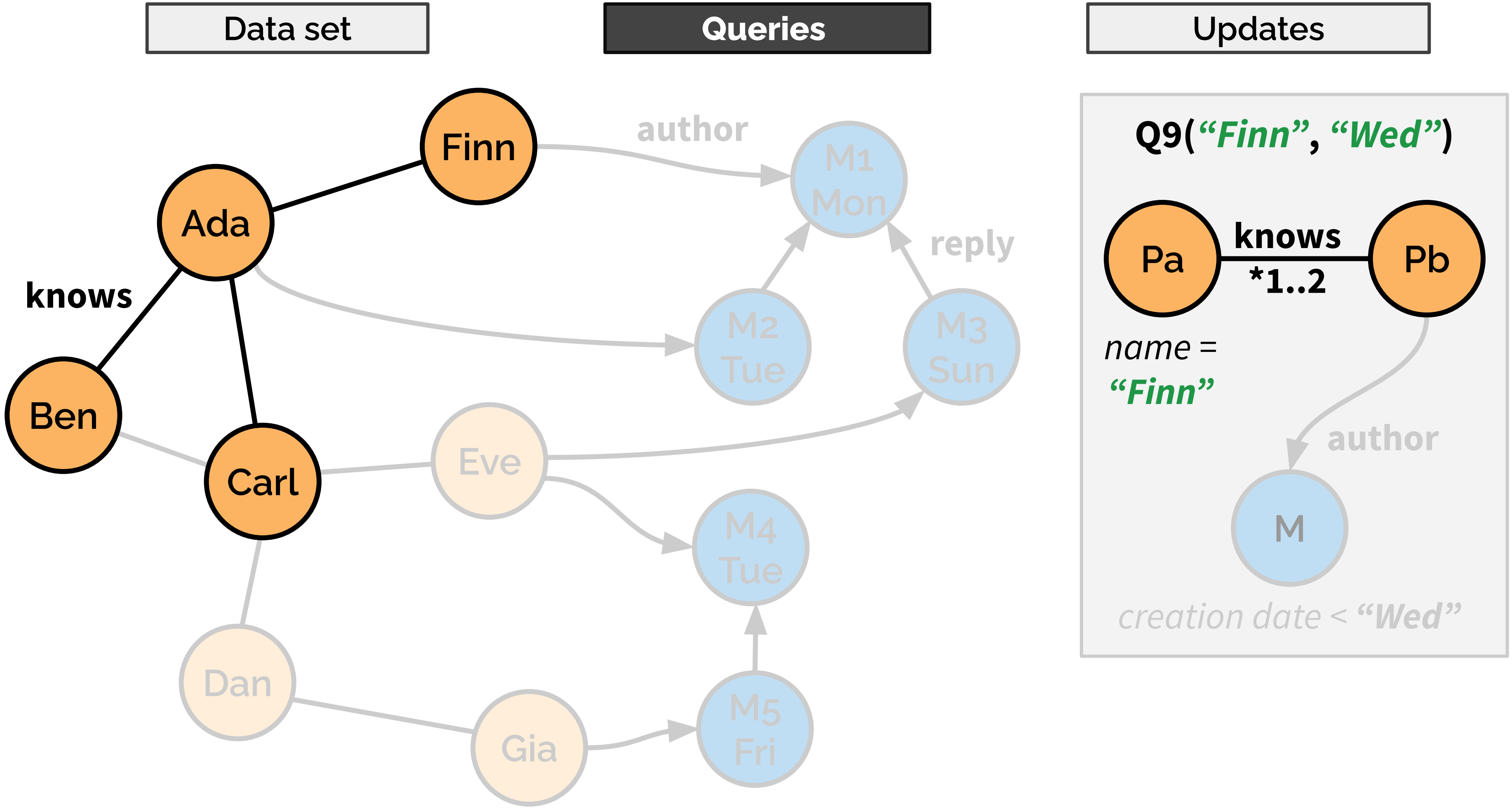
Updates



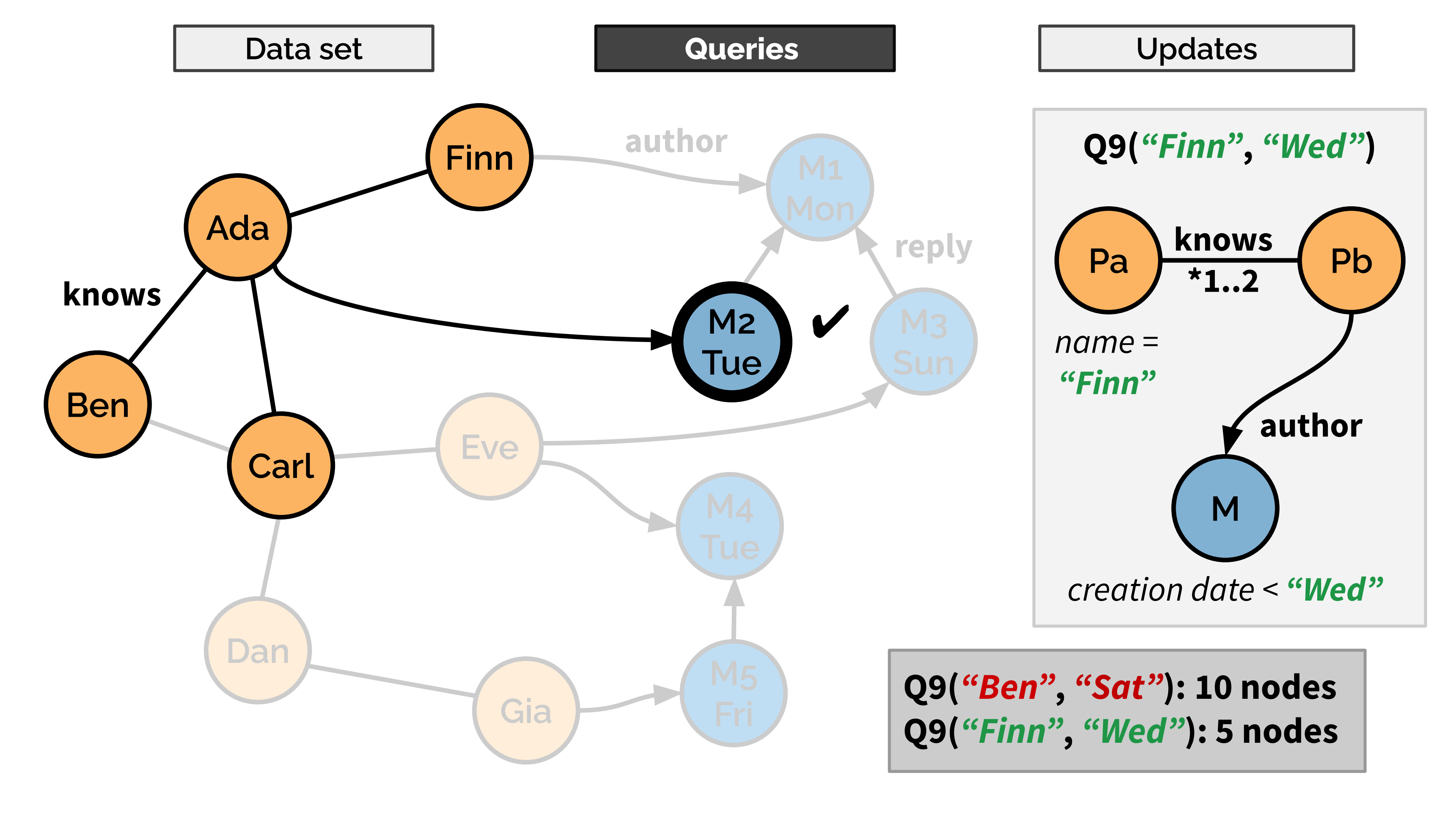




Data set
Queries
Updates
Finn
Ada
author
M1 Mon
reply
knows
M2 Tue
M3 Sun
Ben
Carl
Eve
M4 Tue
Dan
Gia
M5 Fri
Q9("Finn", "Wed")
Pa
knows
*1..2
Pb
name = "Finn"
author
M
creation date < "Wed"

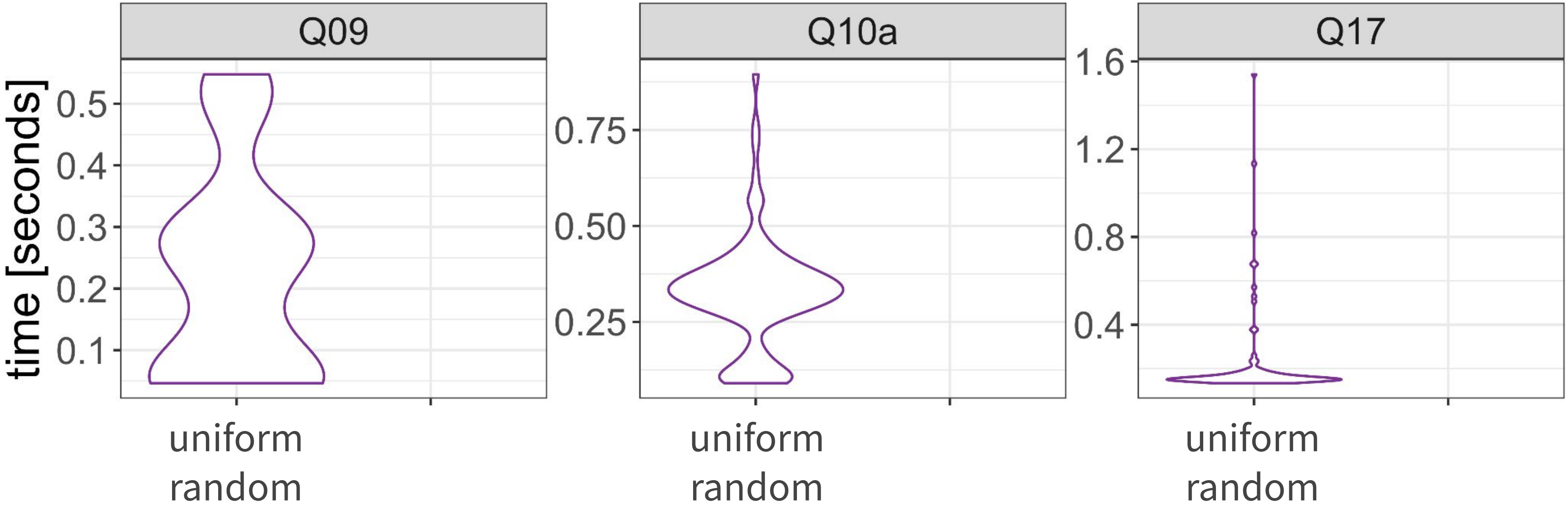


Data set
Queries
Updates
Finn
Ada
knows
Ben
Carl
author
M1
Mon
reply
M2
Tue
M3
Sun
Eve
M4
Tue
Dan
Gia
M5
Fri
Q9("Finn", "Wed")
Pa
knows
*1..2
Pb
name =
"Finn"
author
M
creation date < "Wed"


Data set
Queries
Updates
Finn
Ada
knows
Ben
Carl
author
M1 Mon
reply
M2 Tue
M3 Sun
Eve
M4 Tue
Dan
Gia
M5 Fri
Q9("Finn", "Wed")
Pa
knows *1..2
Pb
name = "Finn"
author
M
creation date < "Wed"
Q9("Ben", "Sat"): 10 nodes
Q9("Finn", "Wed"): 5 nodes

Parameter selection

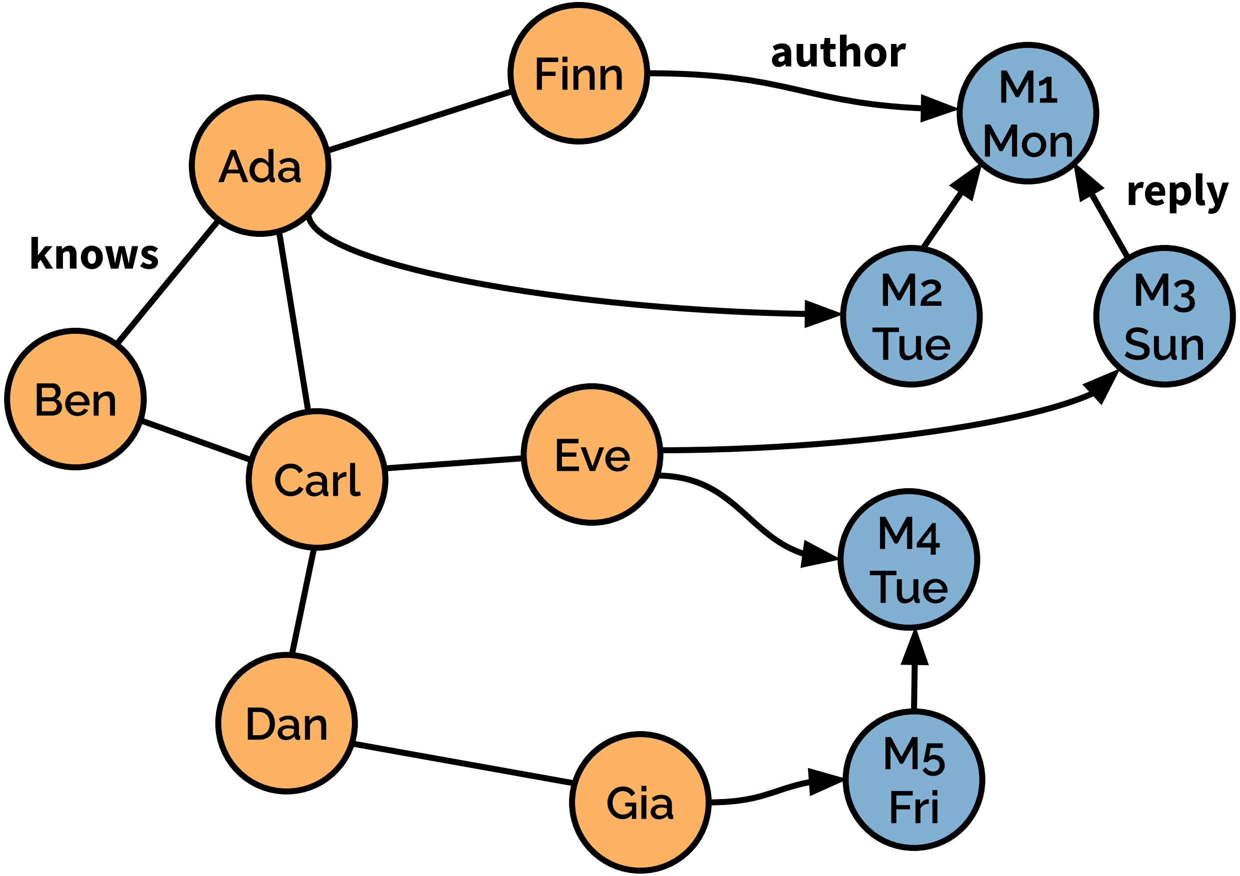
- ***Uniform random parameters*** → unstable distributions



Parameter curation

A. Gubichev, P. Boncz
TPCTC 2014

Data set | **Queries** | Updates



Statistics ("factors")

numFriendsOfFriends

name	#1-hop	#2-hop
Ben	2	3
Carl	4	2
Ada	3	2
...		

numMessagesPerDay

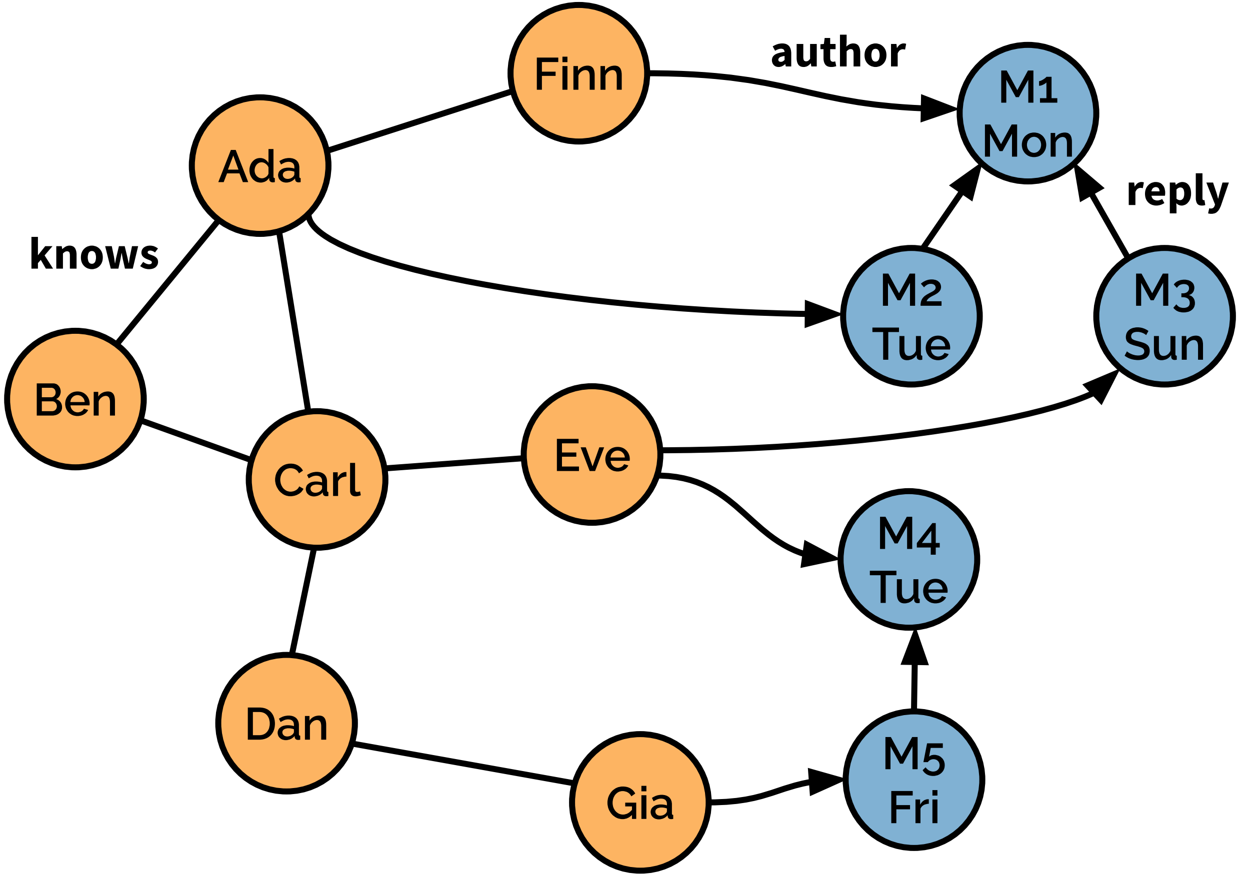
day	#
Mon	1
Tue	2
...	

Inserts

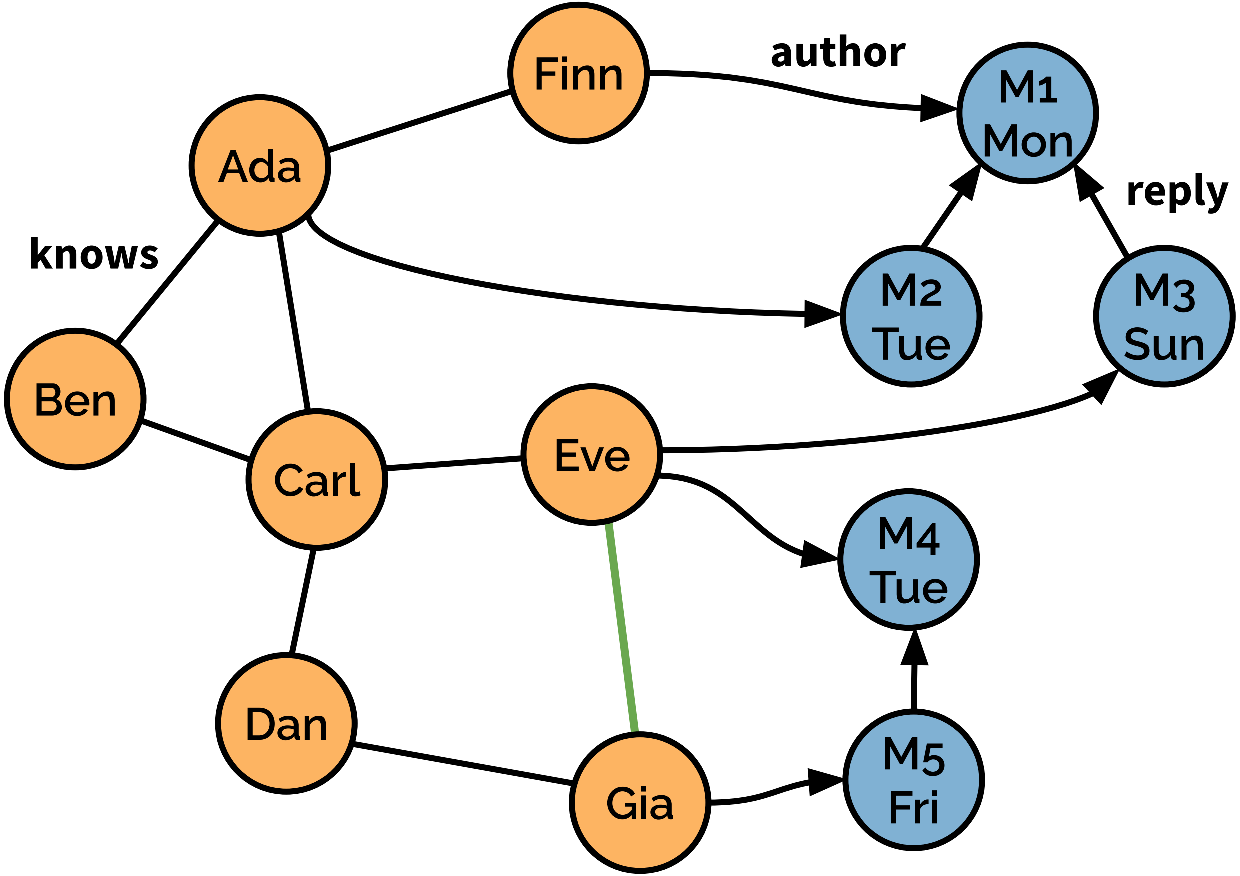
Data set

Queries

Updates



Data set | Queries | **Updates**



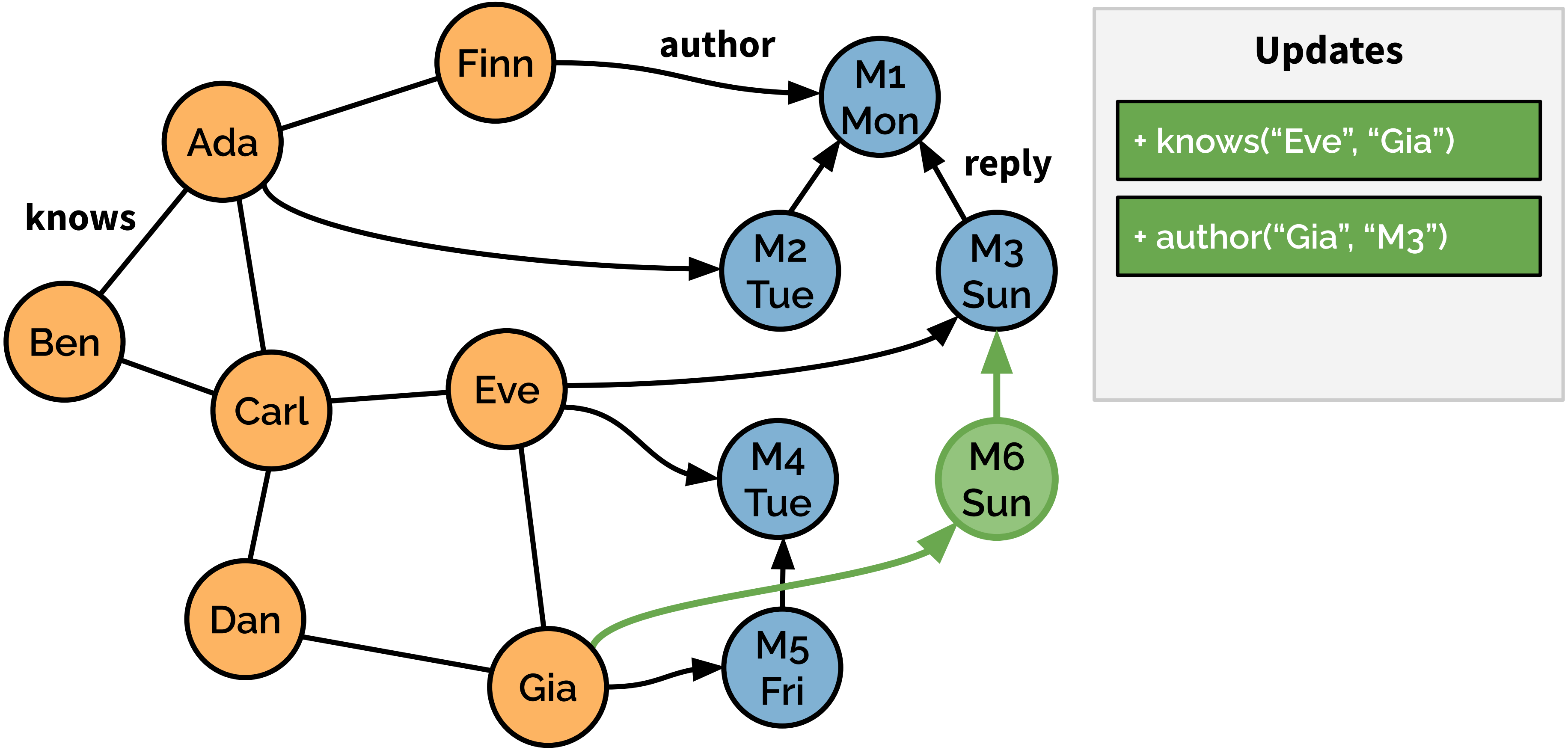
Updates

+ knows(“Eve”, “Gia”)

Data set

Queries

Updates



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*”

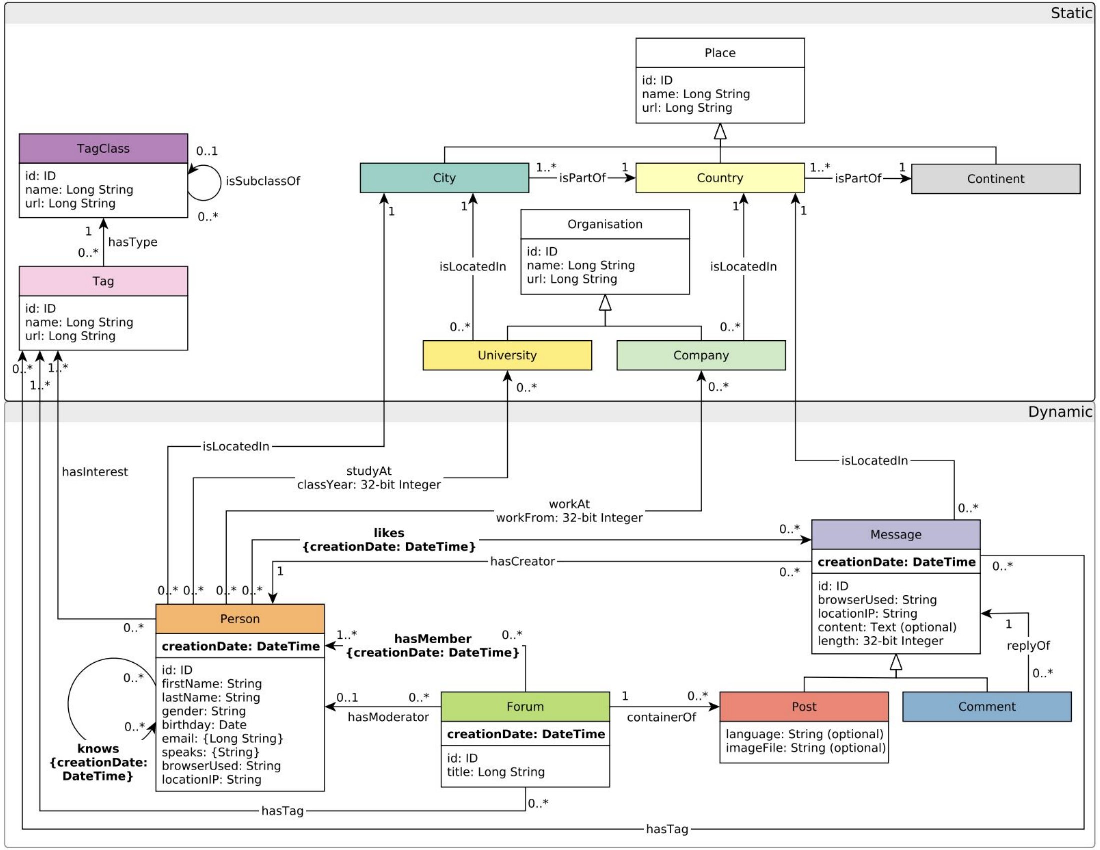
The generated graphs are realistic *to some extent* but not fully. The goal is to add some realistic correlations which query engines can exploit when optimizing the queries.

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

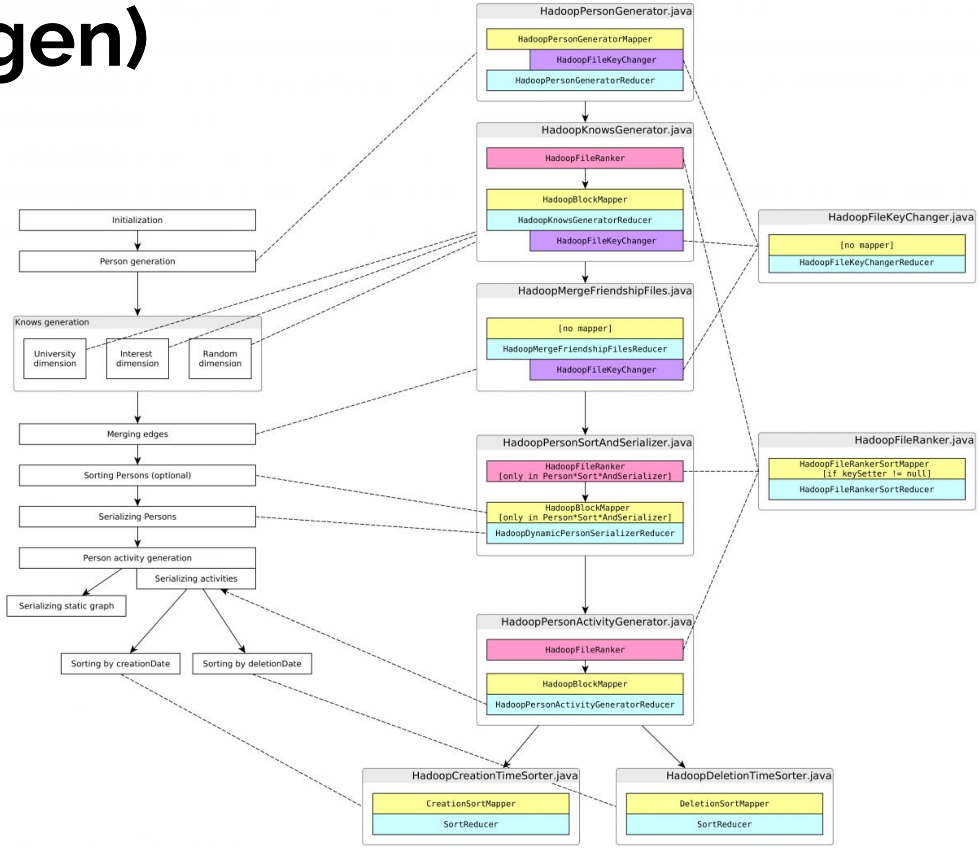
The Datagen for SNB Interactive v1 uses Hadoop

📃 S3G2: a Scalable Structure-correlated Social Graph Generator, TPCTC 2012
📺 LDBC SNB Datagen: Under the hood by Arnau Prat, 9th LDBC TUC meeting, 2017

Data generator (Datagen)

Graphs are produced using a Hadoop-based distributed generator

The generator is capable of producing output with different serializers (CSV variants, Turtle).



Update 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.

In **Interactive v1,** new Persons/Forums/Messages are *inserted* along with their edges

Data sets

SNB Interactive data sets of SF0.1 to SF1000 are published at the SURF/CWI repository.

These data sets were generated using different serializers and partition numbers:

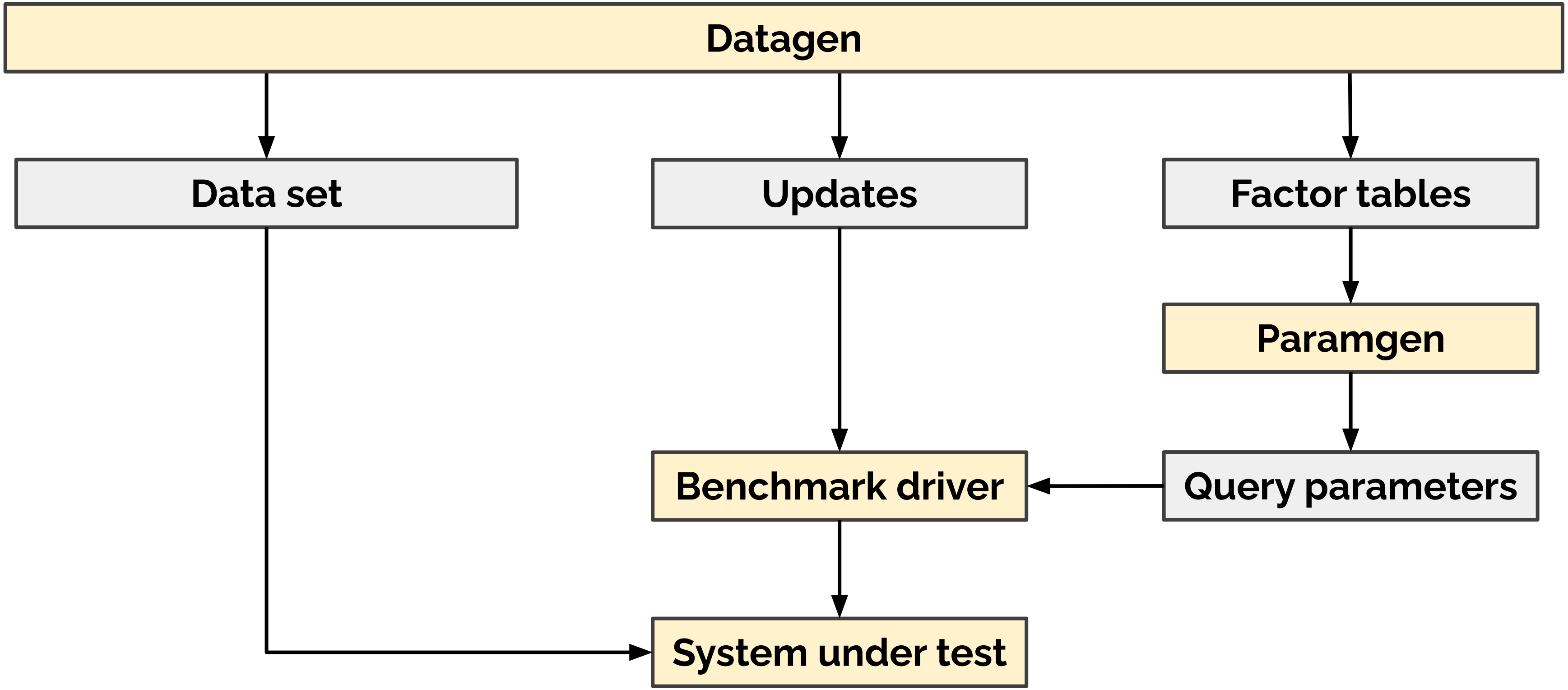
- Serializers:
  - csv_basic, csv_basic-longdateformatter
  - csv_composite, csv_composite-longdateformatter
  - csv_composite_merge_foreign, csv_composite_merge_foreign-longdateformatter
  - csv_merge_foreign, csv_merge_foreign-longdateformatter
  - ttl
- Partition numbers:
  - 2^k (1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024)
  - 12×2^k (12, 24, 48, 96, 192, 384, 768)

⚠️ The data sets are stored on tape and have to be staged to disks before downloading.

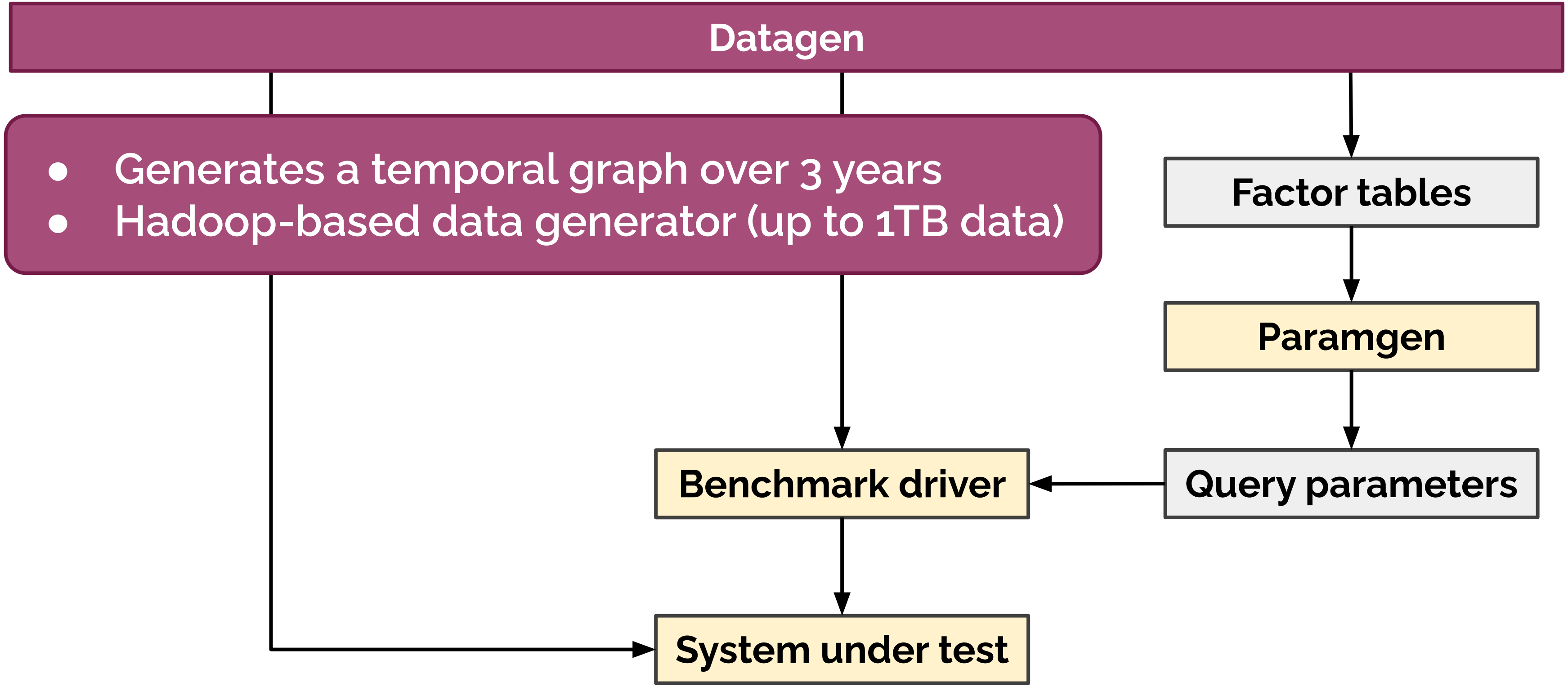
See download instructions.

Benchmark framework

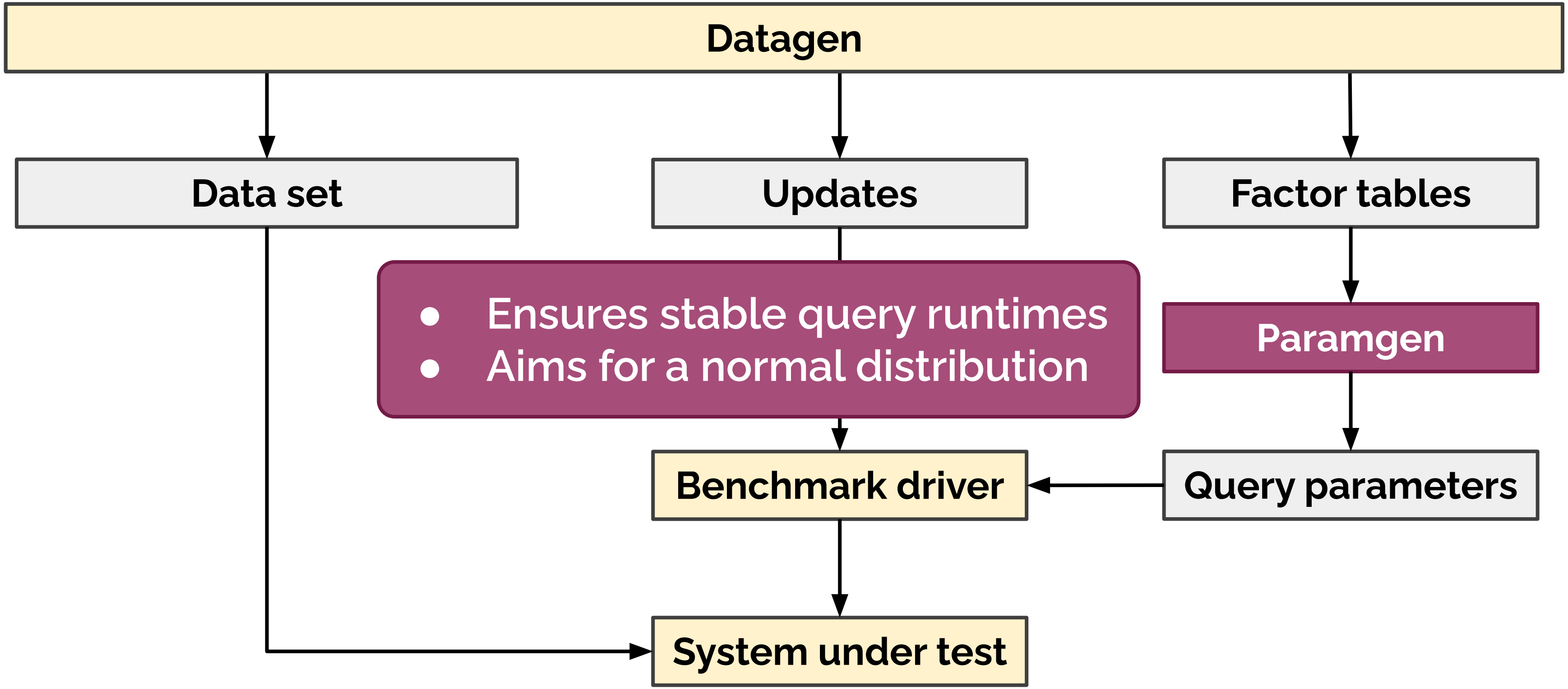
Benchmark workflow



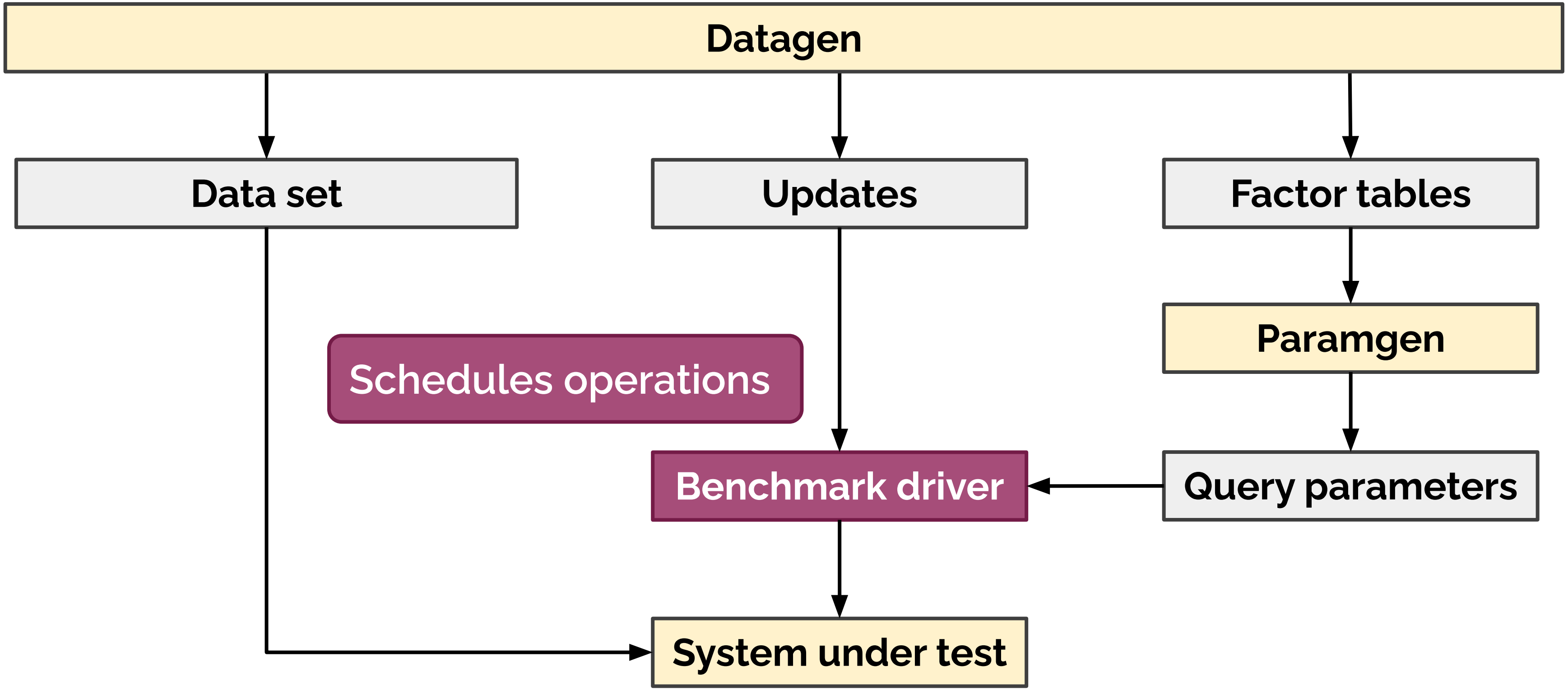
Benchmark workflow



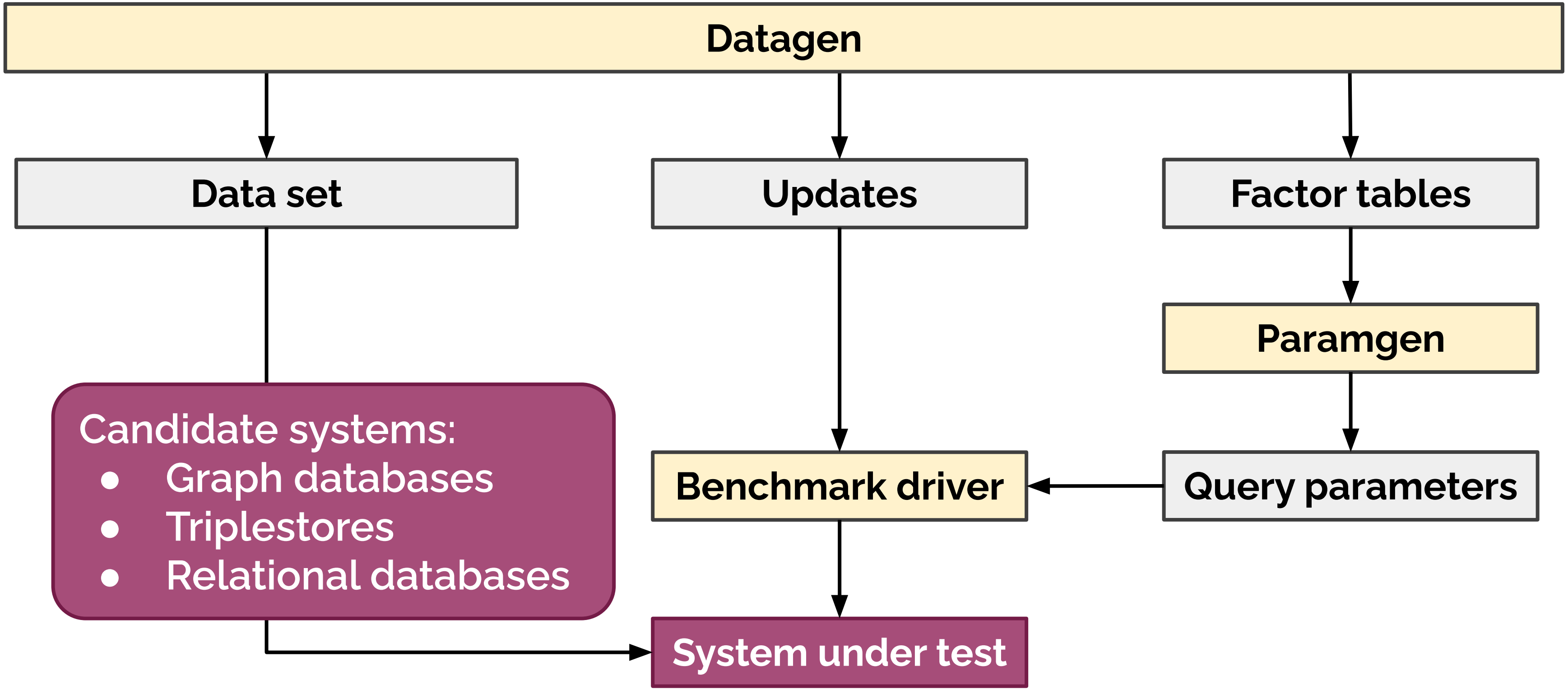
Benchmark workflow



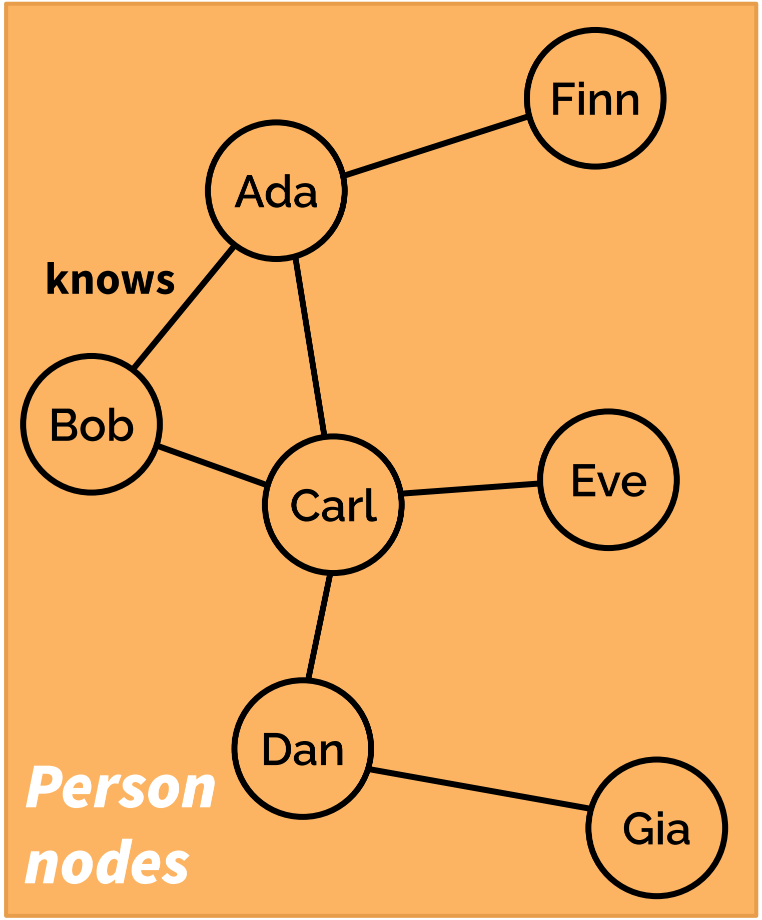
Benchmark workflow



Benchmark workflow



Data generator



Person–knows–Person

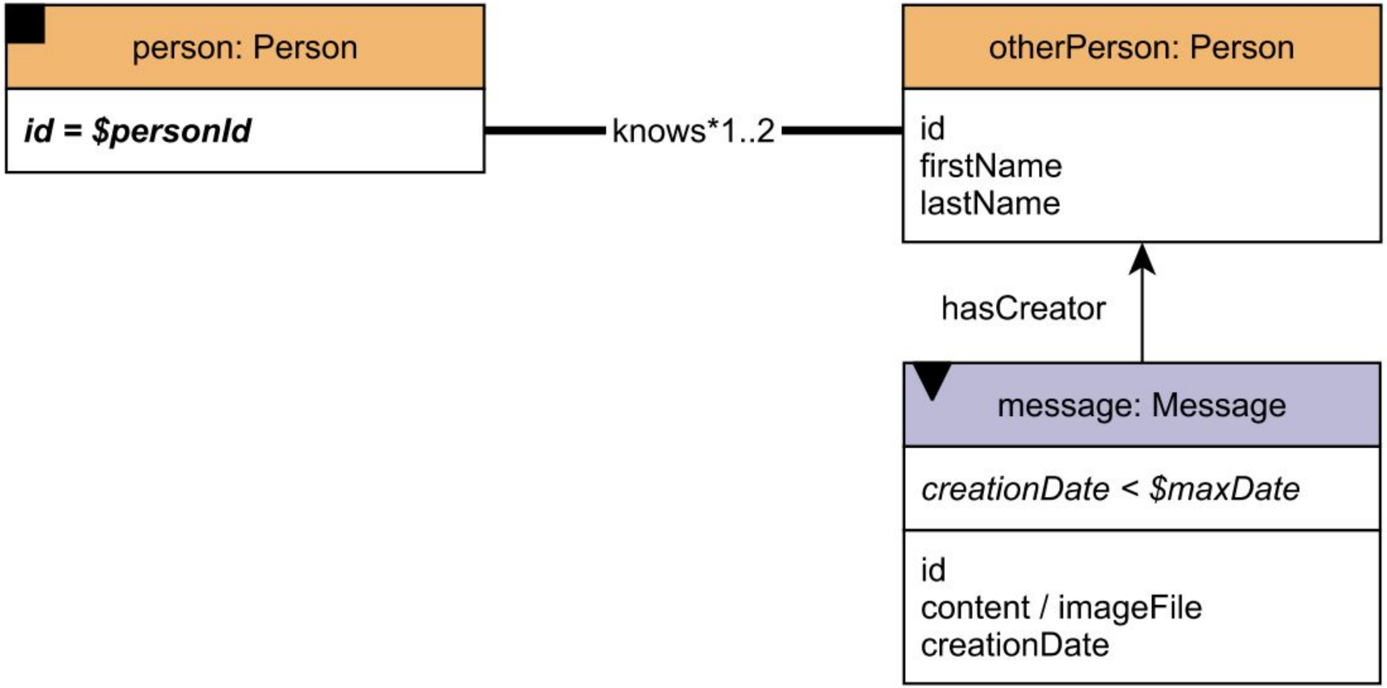
- Degree distribution: Ugander et al. *“The Anatomy of the Facebook Social Graph”* (2011)
- “knows” edges are added along 3 dimensions:
  - university attendance
  - geographical location
  - random

Operations

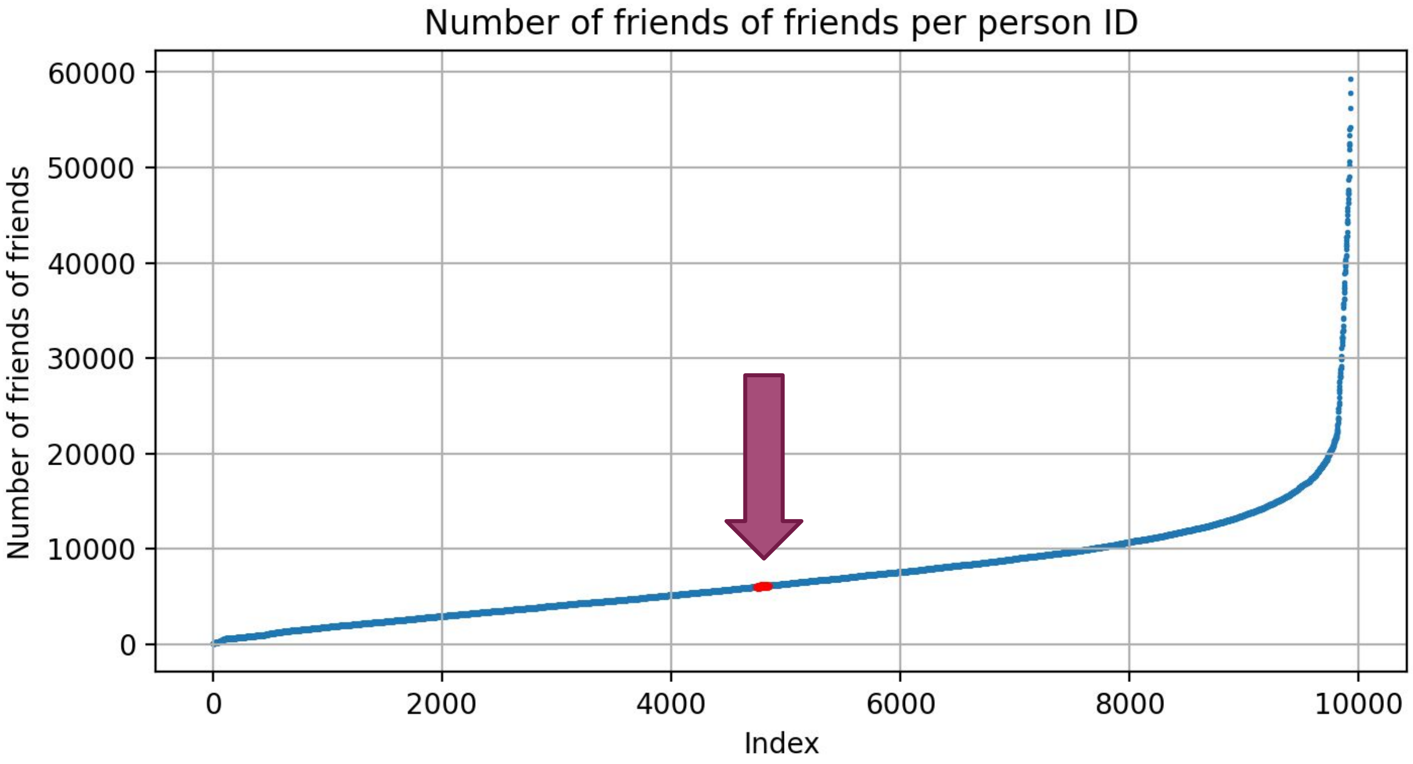
Workload mix

CR	SR	INS
complex reads	short reads	inserts
1–500 ms	0.1–75 ms	0.1–100 ms

Complex read Q9: Recent messages by F/FoaF

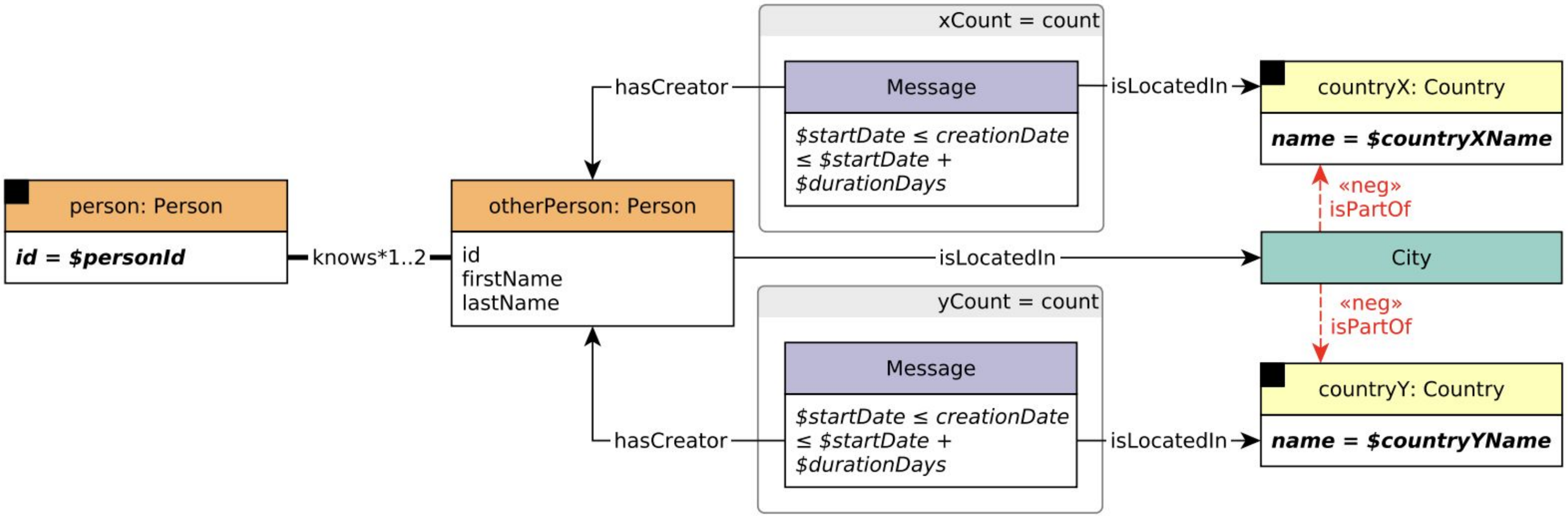


Q9 parameter selection: Window

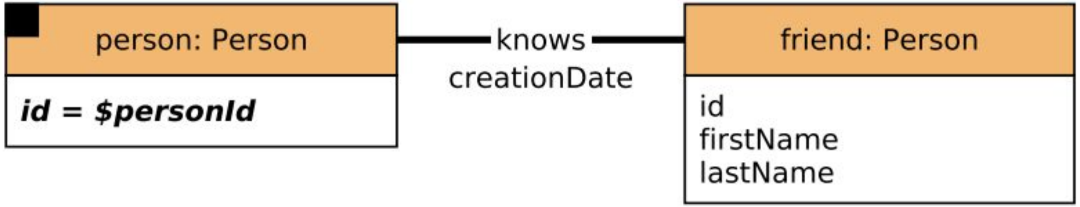


Complex read Q3: Travelling abroad

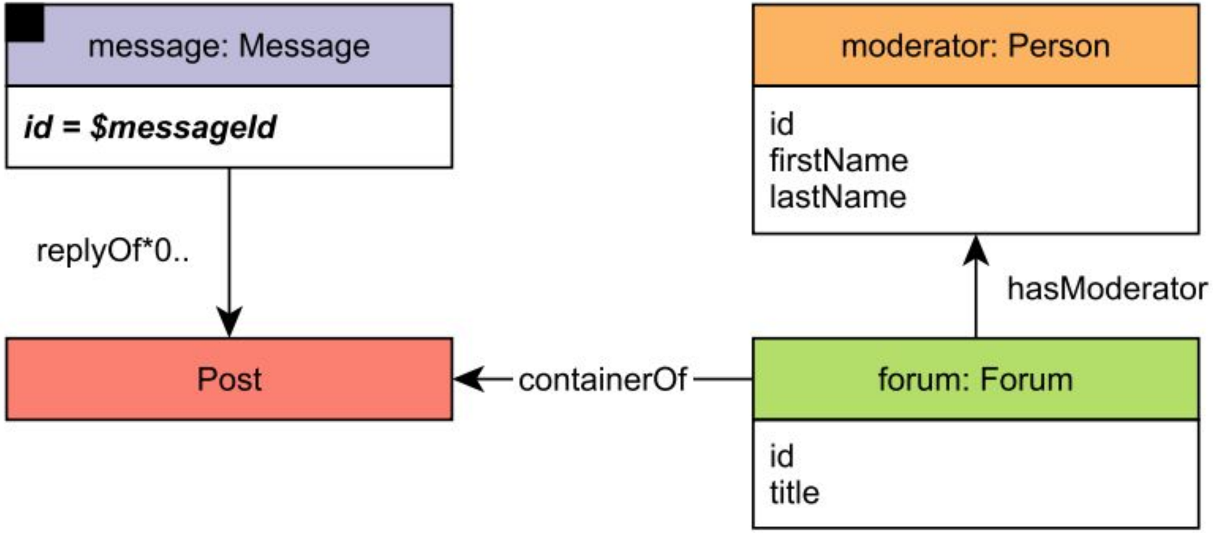
Friends and FoaFs that created Messages from given Countries but do not live there



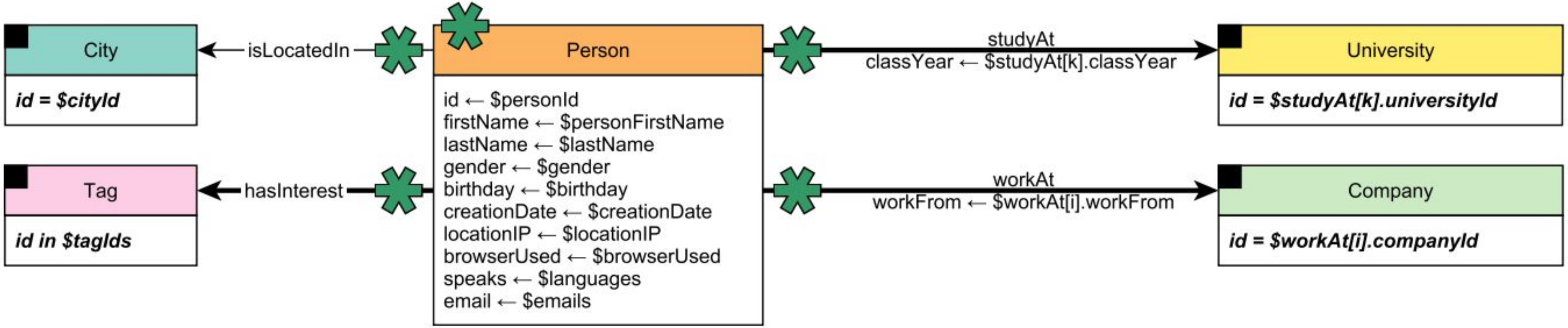
Short read Q3: Friends of a Person



Short read Q6: Forum of a Message

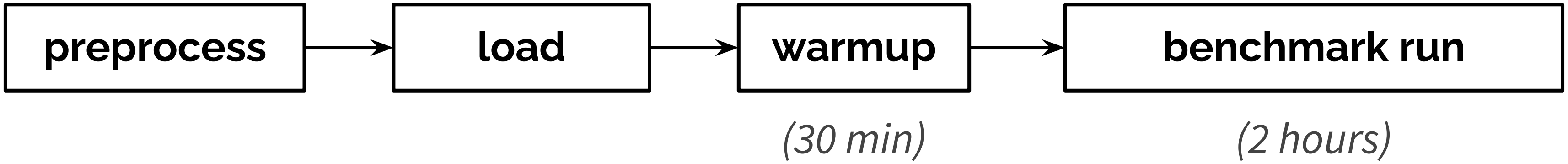


Insert query INS1: Add Person



Scheduling

Benchmark execution



- Collect individual query runtimes
- Check 95% on-time requirement

Driver execution modes

The driver has 3 modes of operation, all start with the initial data set loaded.

1-2) Generate validation data set / Validate implementation

- single-threaded
- deterministic

3) Run benchmark

- multi-threaded
- calculates throughput
- pass/fail schedule

Scheduling operations: Theory

Updates: replayed as they happen in the social network

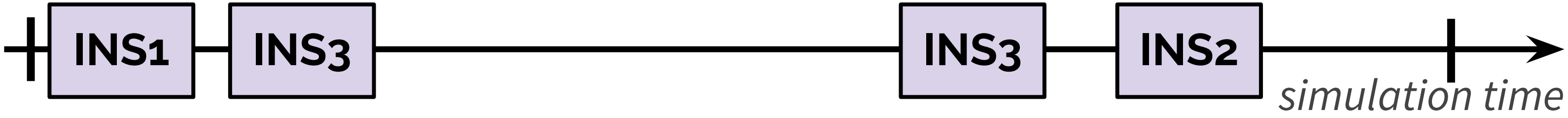
Complex reads: a given complex read query is scheduled for X update operations

For each complex read instance, a sequence of **short reads** is triggered, short reads can trigger other short reads

	IS 1	IS 2	IS 3	IS 4	IS 5	IS 6	IS 7
IC 1	$\otimes$	$\otimes$	$\otimes$				
IC 2	$\otimes$	$\otimes$	$\otimes$	$\otimes$	$\otimes$	$\otimes$	$\otimes$
IC 3	$\otimes$	$\otimes$	$\otimes$				
IC 7	$\otimes$	$\otimes$	$\otimes$	$\otimes$	$\otimes$	$\otimes$	$\otimes$
IC 8	$\otimes$	$\otimes$	$\otimes$	$\otimes$	$\otimes$	$\otimes$	$\otimes$
IC 9	$\otimes$	$\otimes$	$\otimes$	$\otimes$	$\otimes$	$\otimes$	$\otimes$
IC 10	$\otimes$	$\otimes$	$\otimes$				
IC 11	$\otimes$	$\otimes$	$\otimes$				
IC 12	$\otimes$	$\otimes$	$\otimes$				
IC 14	$\otimes$	$\otimes$	$\otimes$				
IS 2	$\otimes$	$\otimes$	$\otimes$	$\otimes$	$\otimes$	$\otimes$	$\otimes$
IS 3	$\otimes$	$\otimes$	$\otimes$				
IS 5	$\otimes$	$\otimes$	$\otimes$				
IS 6	$\otimes$	$\otimes$	$\otimes$				
IS 7	$\otimes$	$\otimes$	$\otimes$	$\otimes$	$\otimes$	$\otimes$	$\otimes$

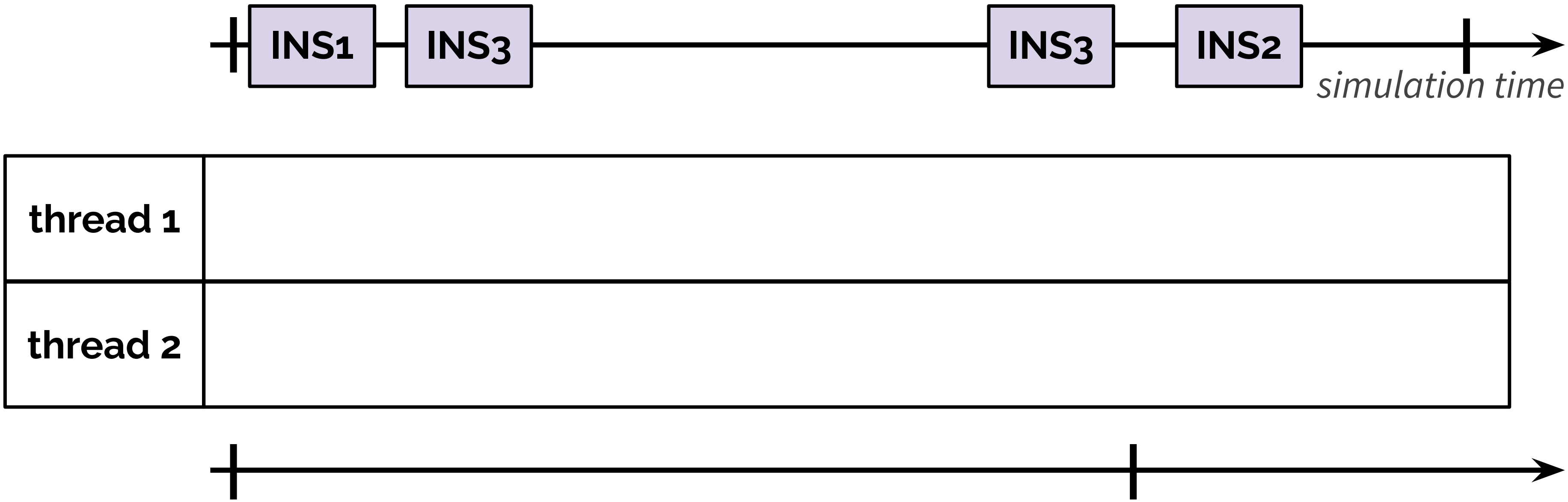
Scheduling operations: Example

Replay speed is determined by the TCR (total compression ratio)



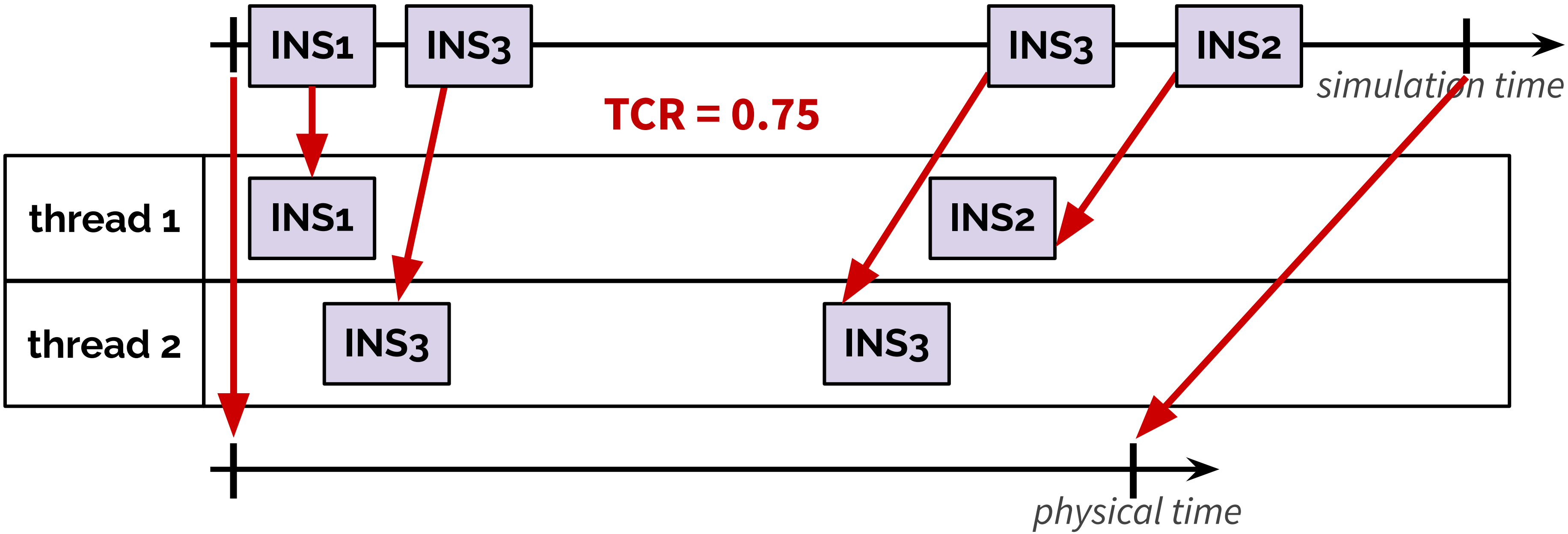
Scheduling operations: Example

Replay speed is determined by the TCR (total compression ratio)



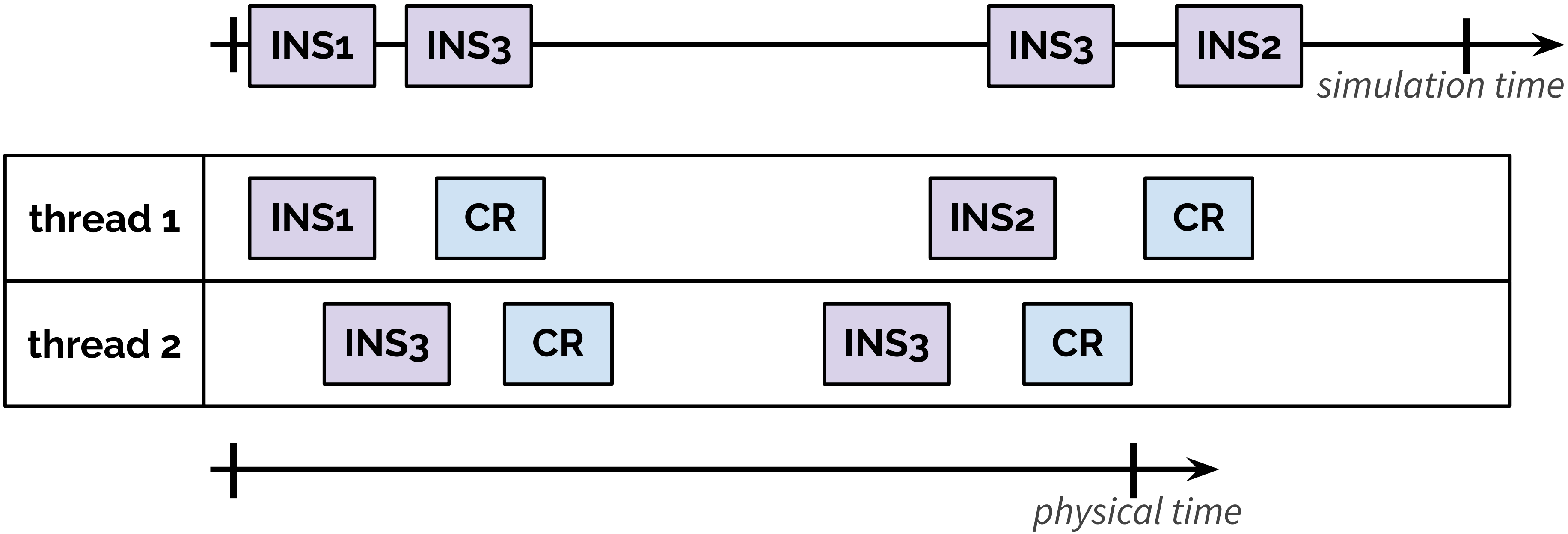
Scheduling operations: Example

Replay speed is determined by the TCR (total compression ratio)



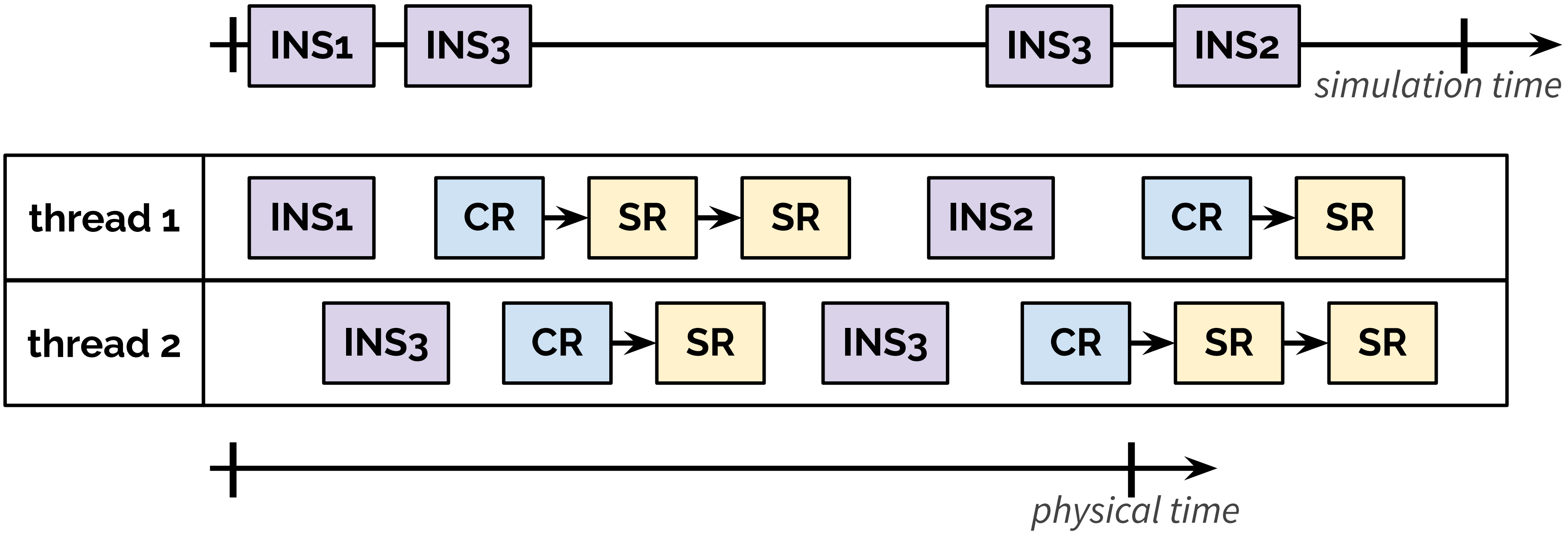
Scheduling operations: Example

Replay speed is determined by the TCR (total compression ratio)



Scheduling operations: Example

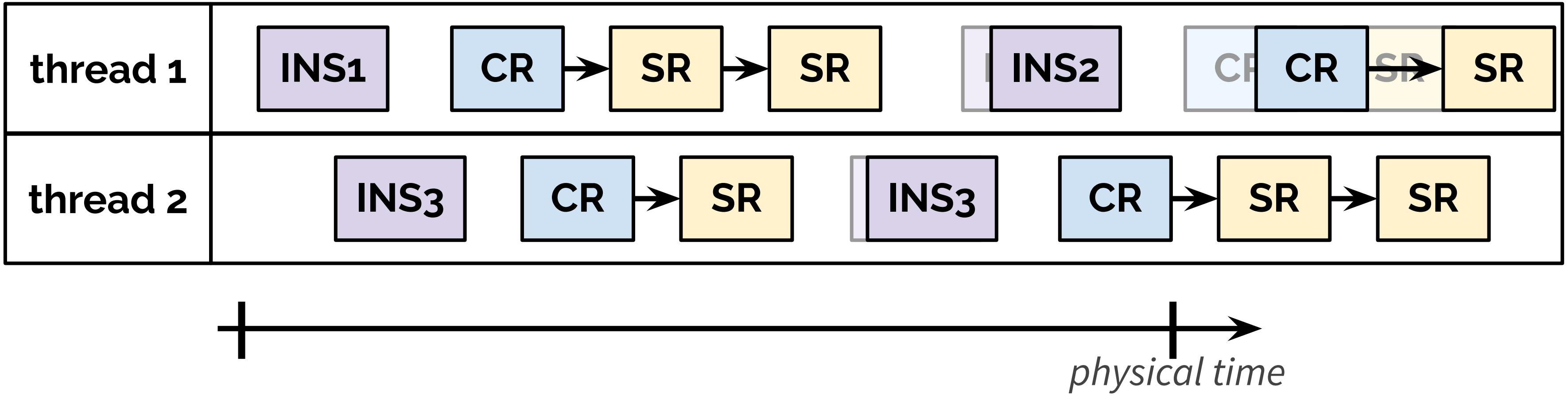
Replay speed is determined by the TCR (total compression ratio)



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



Creating a new
SNB Interactive 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 or the TigerGraph/GSQL implementation
- Relational DBMSs: use the PostgreSQL or the Microsoft SQL Server 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 or download the required data sets and query substitution parameters.
    a. Use SF10 for cross-validation.
    b. For benchmarks, SF30+ is required.
2. Fork the **SNB Interactive repository** and create a new Maven subproject.
3. Add a **Java client** to the DBMS as a Maven dependency (e.g. `org.postgresql:postgresql`)
4. 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), then proceed to larger data sets.
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 SF10 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 that 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.

Summary

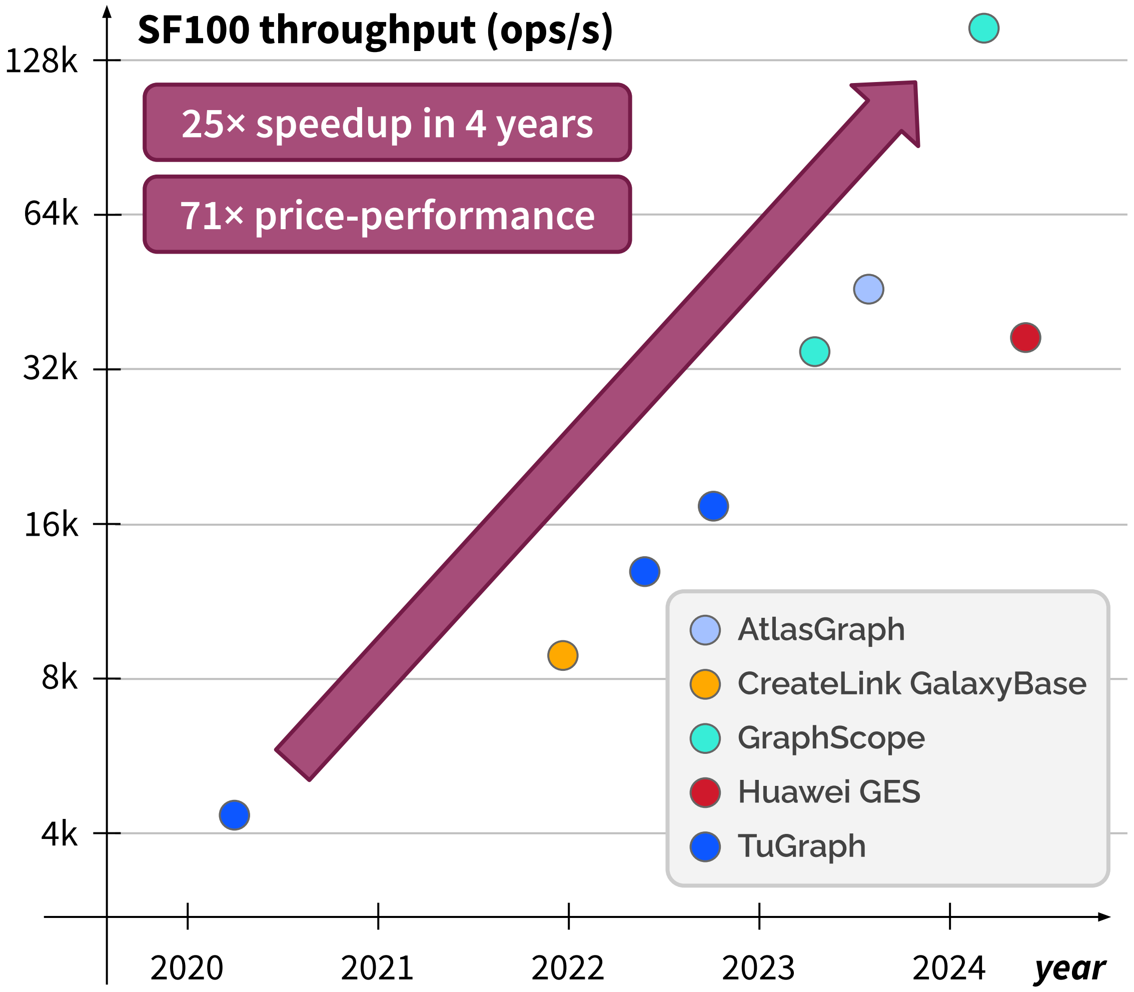
Implementations

system	data model	language
neo4j	graph	Cypher
PostgreSQL	relational	SQL
Microsoft® SQL Server®	relational	SQL + graph extension
UMBRA	relational	SQL

Audited results

As of September 2024, there are 33 audited results on scale factors between 30 and 1,000.

See the SNB Interactive site for the results.



Future work: SNB Interactive v2

- Larger data sets: SF10,000 and beyond
- Deep delete operations
- Improved parameter selection
- Fine-tuning ongoing

Please reach out if you would like to implement the benchmark

LDBC

The graph & RDF
benchmark reference