LDBC Social Network Benchmark and Graphalytics

Gábor Szárnyas
(CWI Amsterdam, LDBC)

16th LDBC TUC meeting | 2023-06-24 | Seattle

The Linked Data Benchmark Council (LDBC): Driving competition and collaboration in the graph data management space

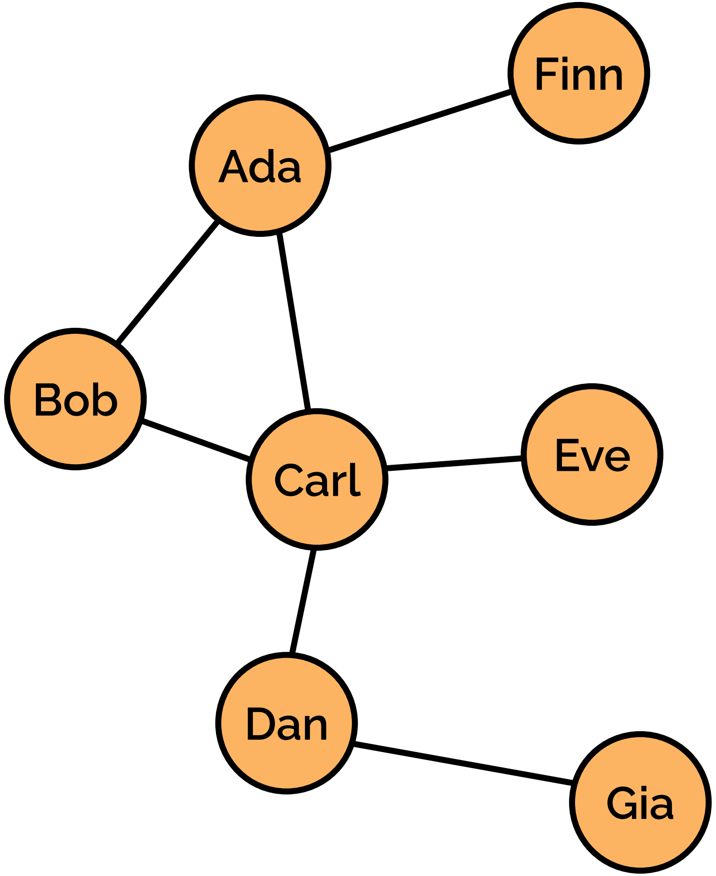
Gábor Szárnyas[1]*, Brad Bebee[2], Altan Birler[3], Alin Deutsch[4,5], George Fletcher[6], Henry A. Gabb[7], Denise Gosnell[2], Alastair Green[8], Zhihui Guo[9], Keith W. Hare[8], Jan Hidders[10], Alexandru Iosup[11], Atanas Kiryakov[12], Tomas Kovatchev[12], Xinsheng Li[13], Leonid Libkin[14], Heng Lin[9], Xiaojian Luo[15], Arnau Prat-Pérez[16], David Püroja[1], Shipeng Qi[9], Oskar van Rest[17], Benjamin A. Steer[18], Dávid Szakállas[19], Bing Tong[20], Jack Waudby[21], Mingxi Wu[5], Bin Yang[13], Wenyuan Yu[15], Chen Zhang[20], Jason Zhang[13], Yan Zhou[20], and Peter Boncz[1]

[1] CWI, the Netherlands, [2] Amazon Web Services, [3] Technische Universität München, Germany, [4] UC San Diego, [5] TigerGraph, [6] TU Eindhoven, [7] Intel Corporation, [8] JCC Consulting, [9] Ant Group, [10] Birkbeck, University of London, [11] VU Amsterdam, the Netherlands, [12] Ontotext AD, [13] Ultipa, [14] University of Edinburgh; RelationalAI; ENS, PSL University, [15] Alibaba Damo Academy, [16] *work done while at UPC Barcelona and Sparsity*, [17] Oracle, USA, [18] Pometry Ltd., [19] *individual contributor*, [20] CreateLink, [21] Newcastle University, School of Computing

* Corresponding author, `gabor.szarnyas@ldbcouncil.org`

Abstract. Graph data management is instrumental for several use cases such as recommendation, root cause analysis, financial fraud detection, and enterprise knowledge representation. Efficiently supporting these use cases yields a number of unique requirements, including the need for a concise query language and graph-aware query optimization techniques. The goal of the Linked Data Benchmark Council (LDBC) is to design a set of standard benchmarks that capture representative categories of graph data management problems, making the performance of systems comparable and facilitating competition among vendors. LDBC also conducts research on graph schemas and graph query languages. This paper introduces the LDBC organization and its work over the last decade.

LDBC Graphalytics



The Graphalytics data sets consist of **untyped, unattributed graphs,** which are *either directed or undirected* and *optionally have edge weights*

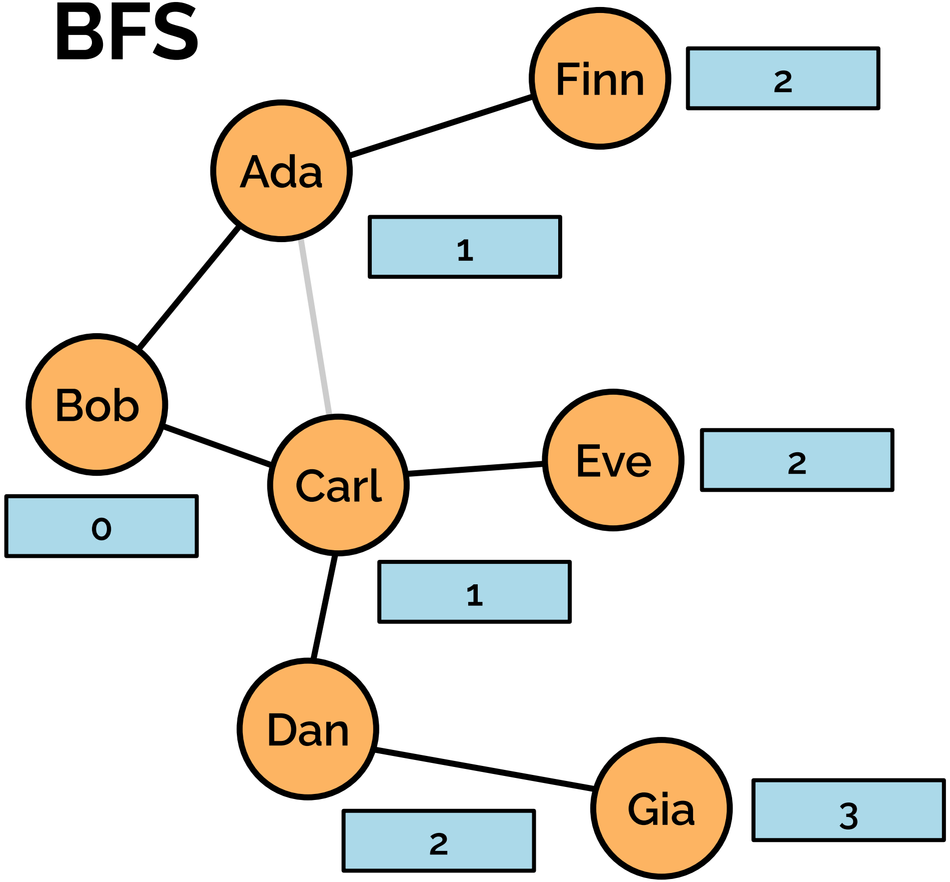
LDBC SNB Datagen
Graph500
Twitter
Friendster
Patents
wiki-Talk
…

Largest graphs

graph	|V|	|E|
datagen-9_3-zf	555M	1.3B
datagen-sf10k-fb	100M	18.8B
graph500-30	450M	34.0B

Algorithms

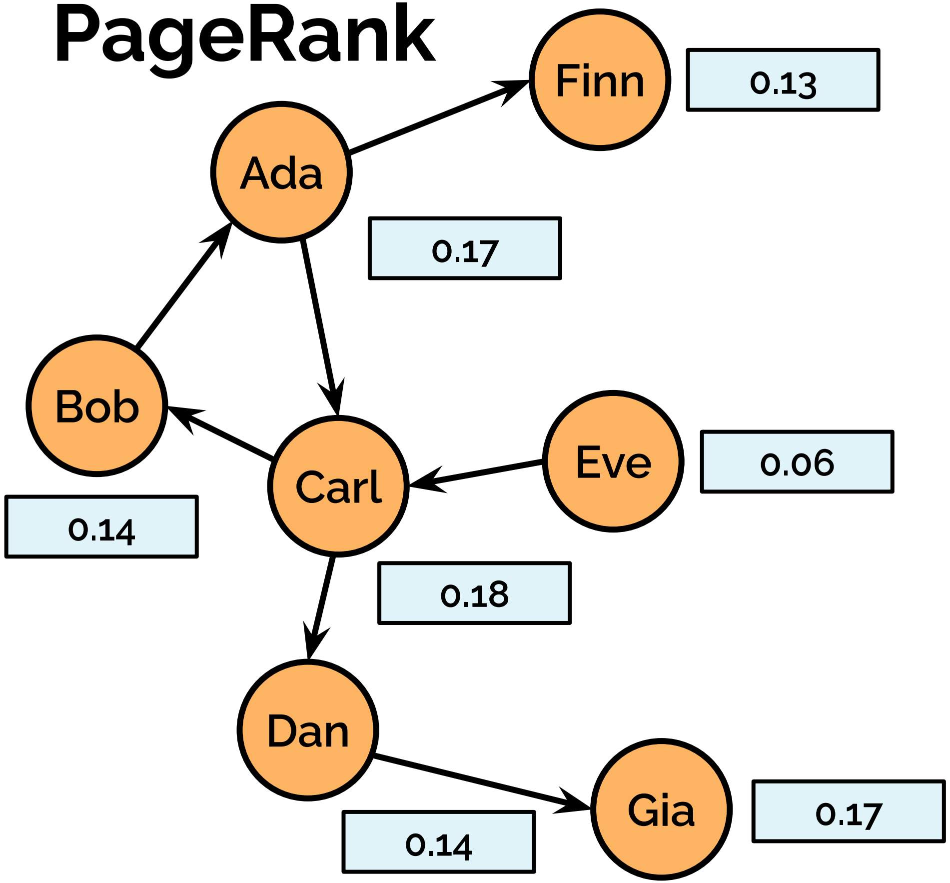
BFS



Breadth-first search(*source*: "Bob")

Assign the level of traversal for each vertex starting from the source (level = 0).

PageRank

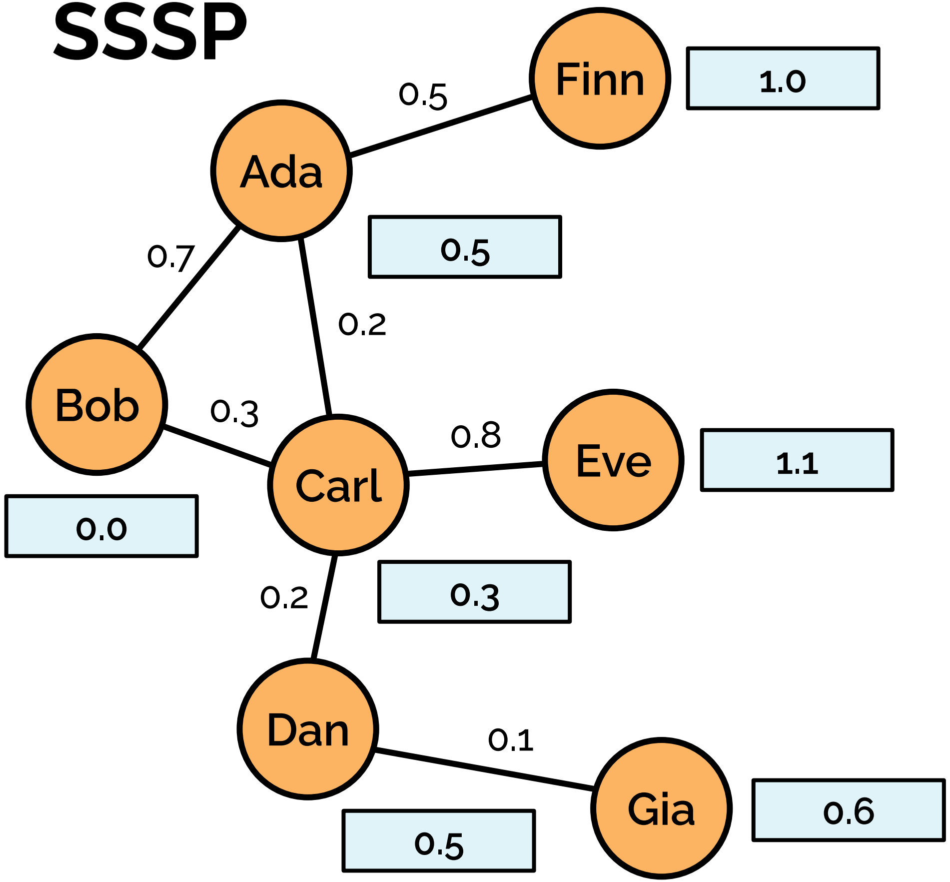


PageRank(*damping factor*: 0.85, *iterations*: 5)

$$PR_i(v) = \underbrace{\frac{1-d}{|V|}}_{\textit{teleport}} + \underbrace{d \cdot \sum_{u \in N_{\mathrm{in}}(v)} \frac{PR_{i-1}(u)}{|N_{\mathrm{out}}(u)|}}_{\textit{importance}} + \underbrace{\frac{d}{|V|} \cdot \sum_{w \in D} PR_{i-1}(w)}_{\textit{redistributed from dangling}}$$

The PageRank variant in Graphalytics redistributes the PageRank values from sinks among all vertices to avoid "leaking" the PageRank out of the network.

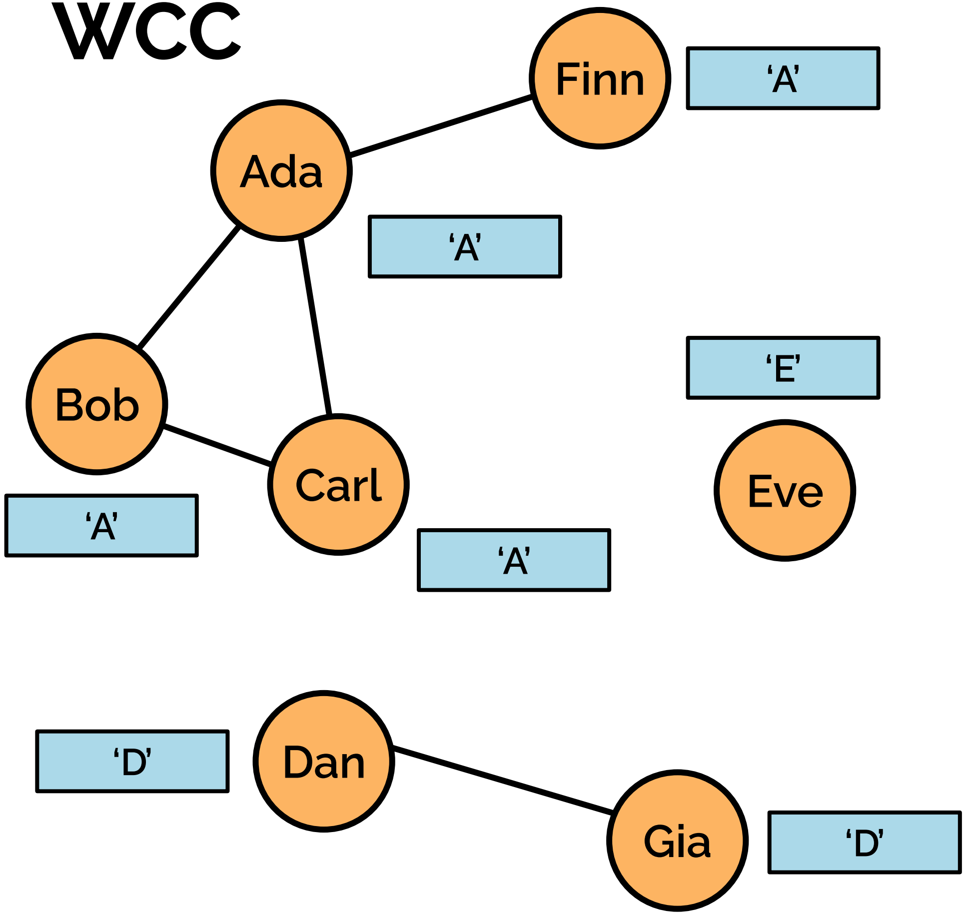
SSSP



Single-source shortest paths(*source*: “Bob”)

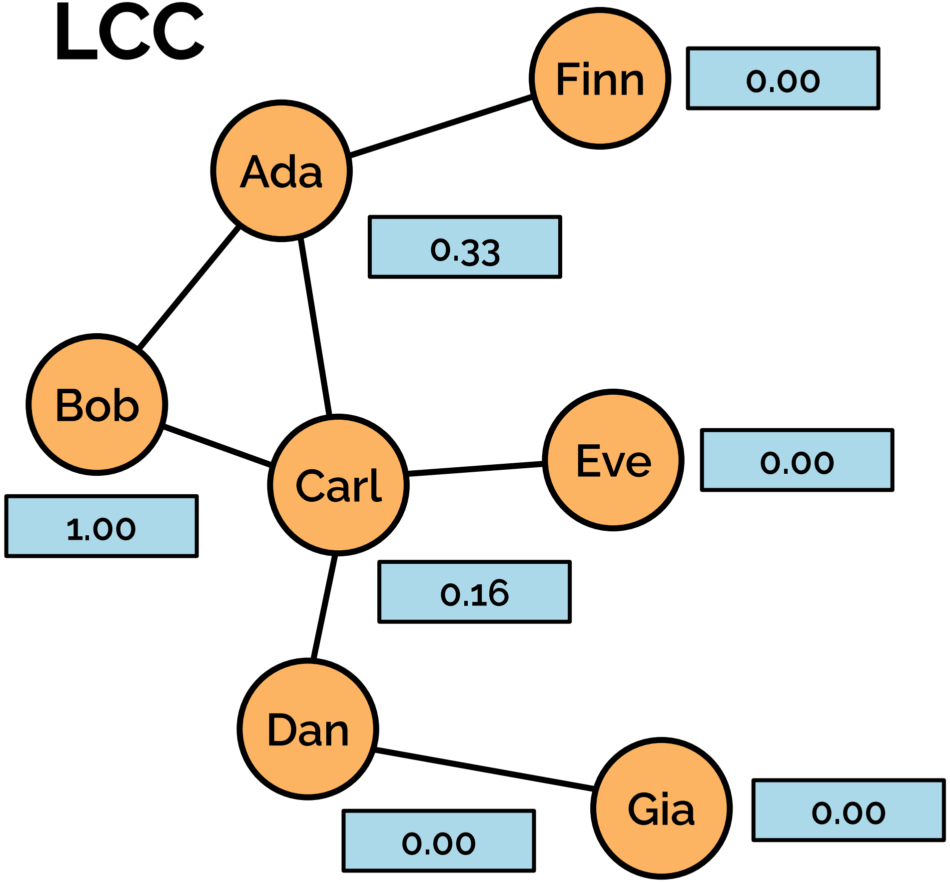
This is the only algorithm that uses edge weights. Many implementations use the delta-stepping SSSP algorithm. These are allowed to specify the delta value for each graph.

WCC



Weakly connected components

LCC



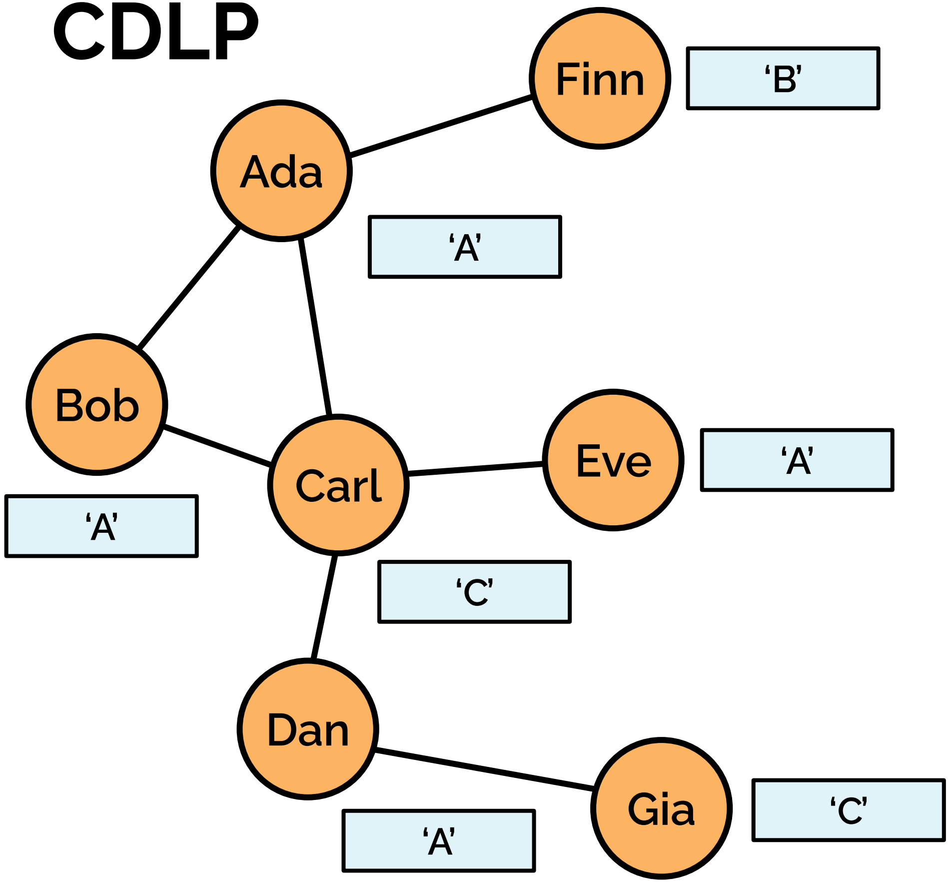
Local clustering coefficient

$$LCC(v) = \begin{cases} 0 & \text{If } |N(v)| \leq 1 \\ \frac{|\{(u,w) | u,w \in N(v) \wedge (u,w) \in E\}|}{|(u,w) | u,w \in N(v)|} & \text{Otherwise} \end{cases}$$

For each vertex, LCC is #triangles / #wedges.

This algorithm is very similar to triangle count.

CDLP



Community detection using LP(*iterations:* 2)

$$L_i(v) = \min\left(\arg\max_l \left[\left|\{u \in N_{\text{in}}(v) \mid L_{i-1}(u) = l\}\right| + \left|\{u \in N_{\text{out}}(v) \mid L_{i-1}(u) = l\}\right|\right]\right)$$

In each iteration, the next label of a vertex is selected as *the minimum mode value among the labels of the neighbours.*

Graphalytics algorithms

All 6 algorithms:

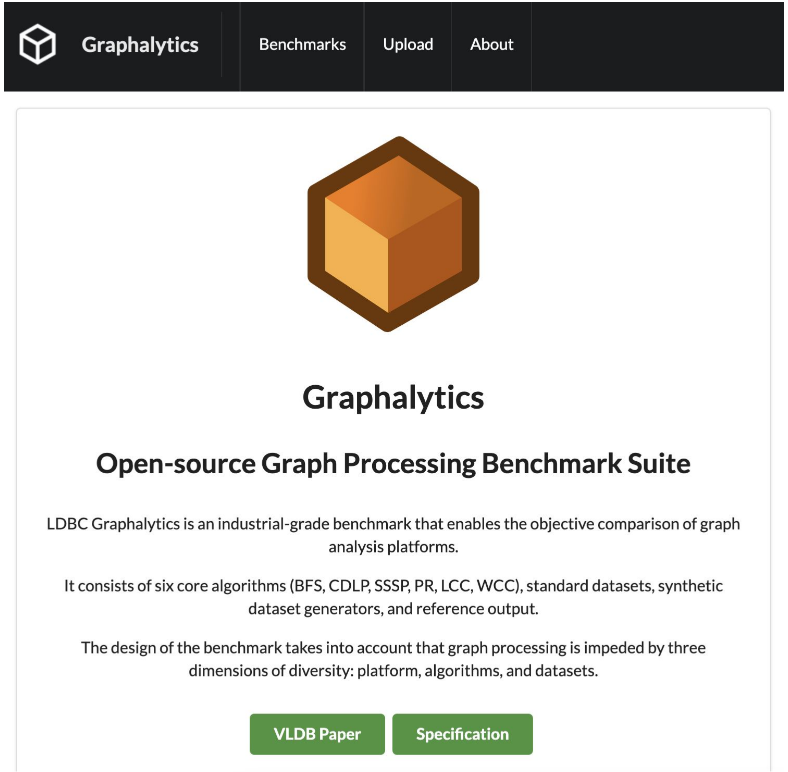
- have **directed and undirected variants**
- are **deterministic**

Validation uses different matching strategies:

- Exact match (BFS, CDLP)
- Epsilon match – relative tolerance of 0.01% (LCC, PR, SSSP)
- Equivalence match – same equivalence classes (WCC)

Competition site is now open

https://graphalytics.ldbcouncil.org/



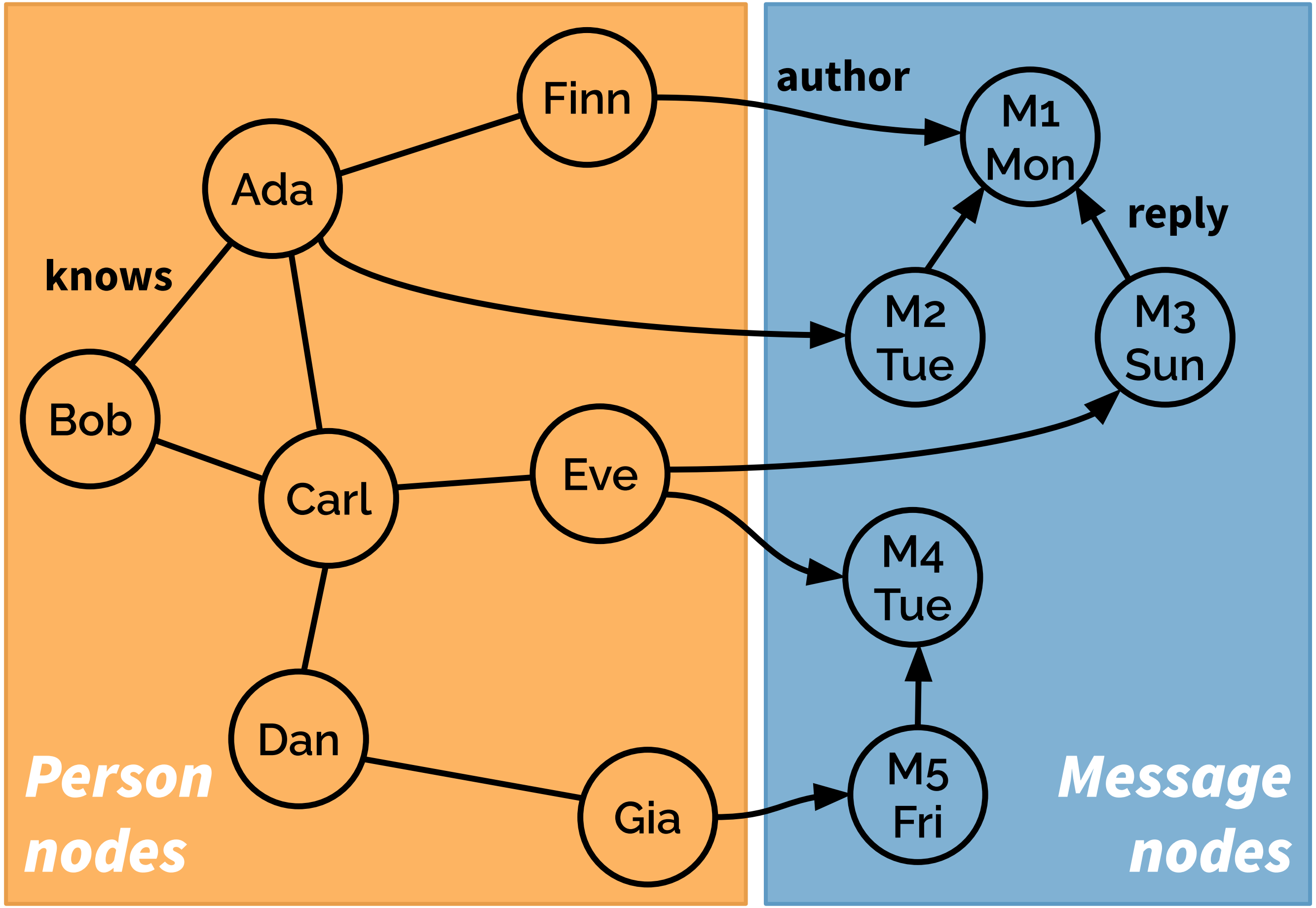
LDBC Social Network Benchmark

Data set

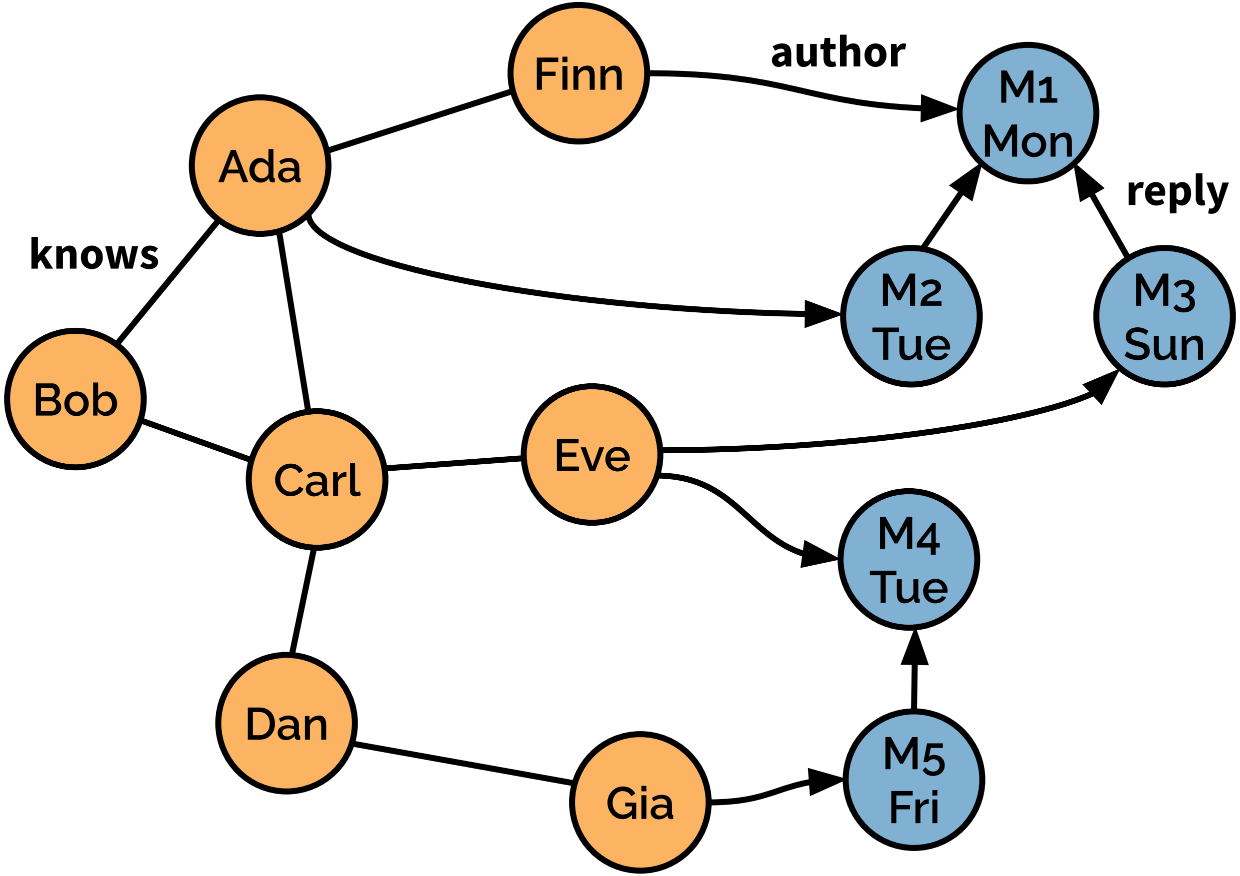
Queries

Updates

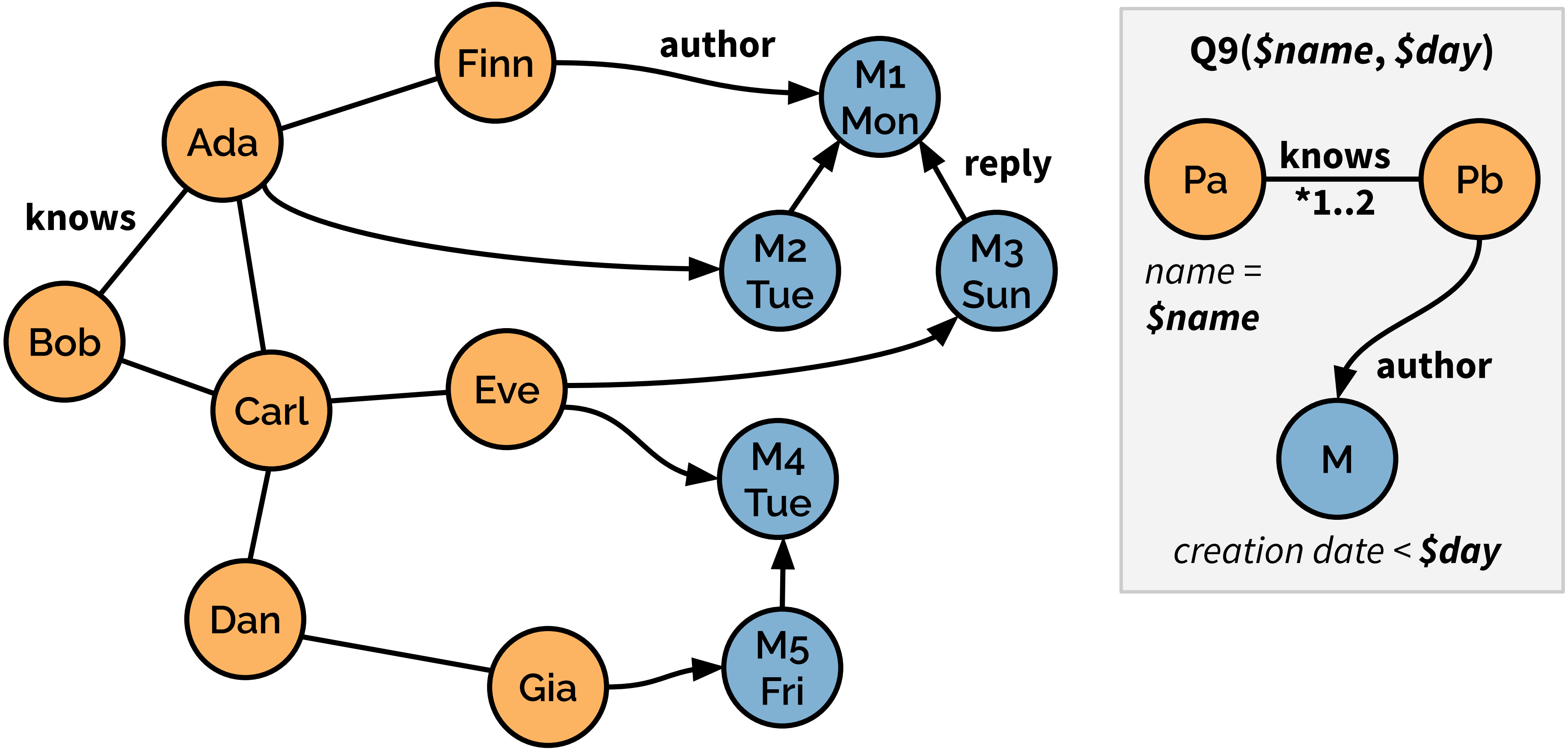
Data set | Queries | Updates



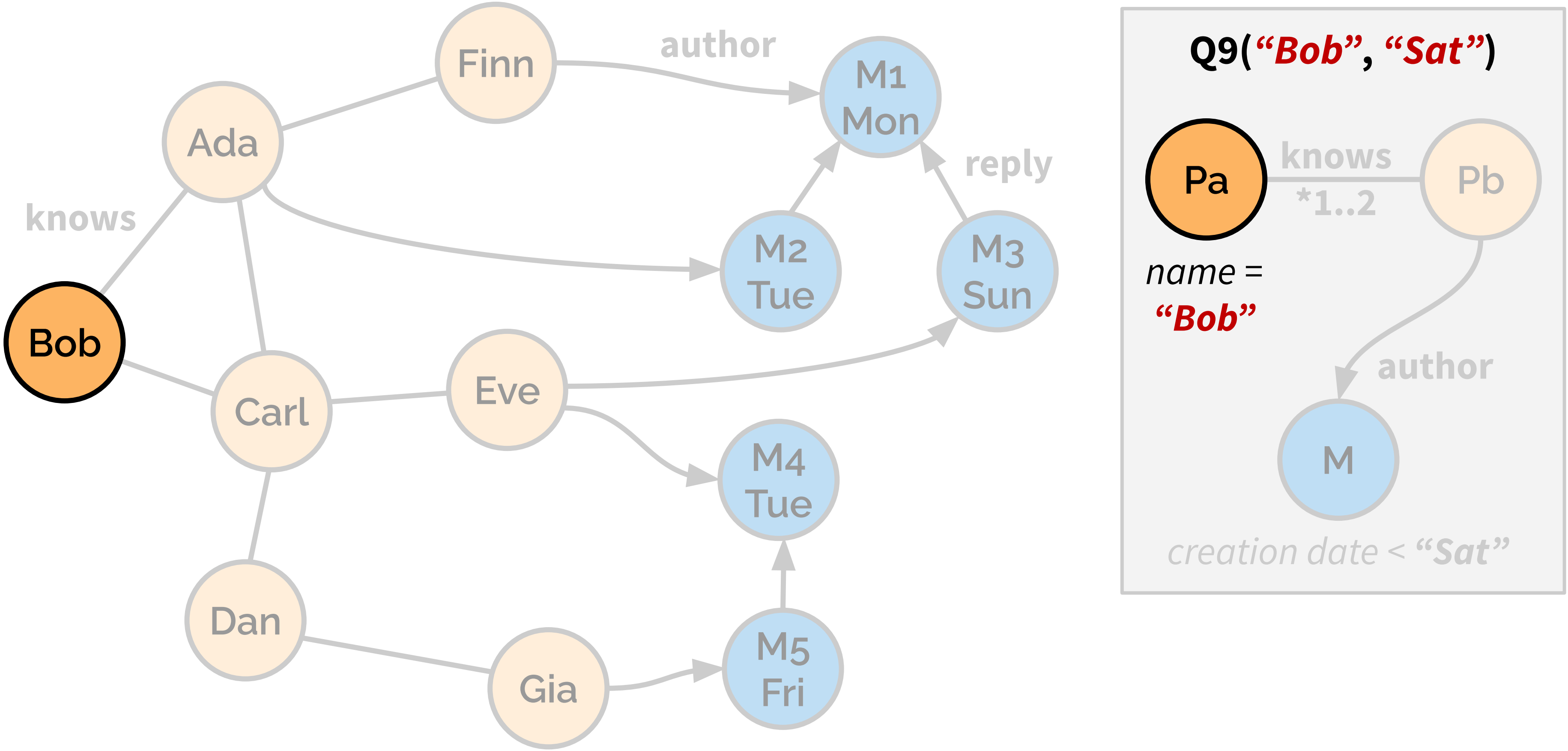
Data set | **Queries** | Updates



Data set | **Queries** | Updates



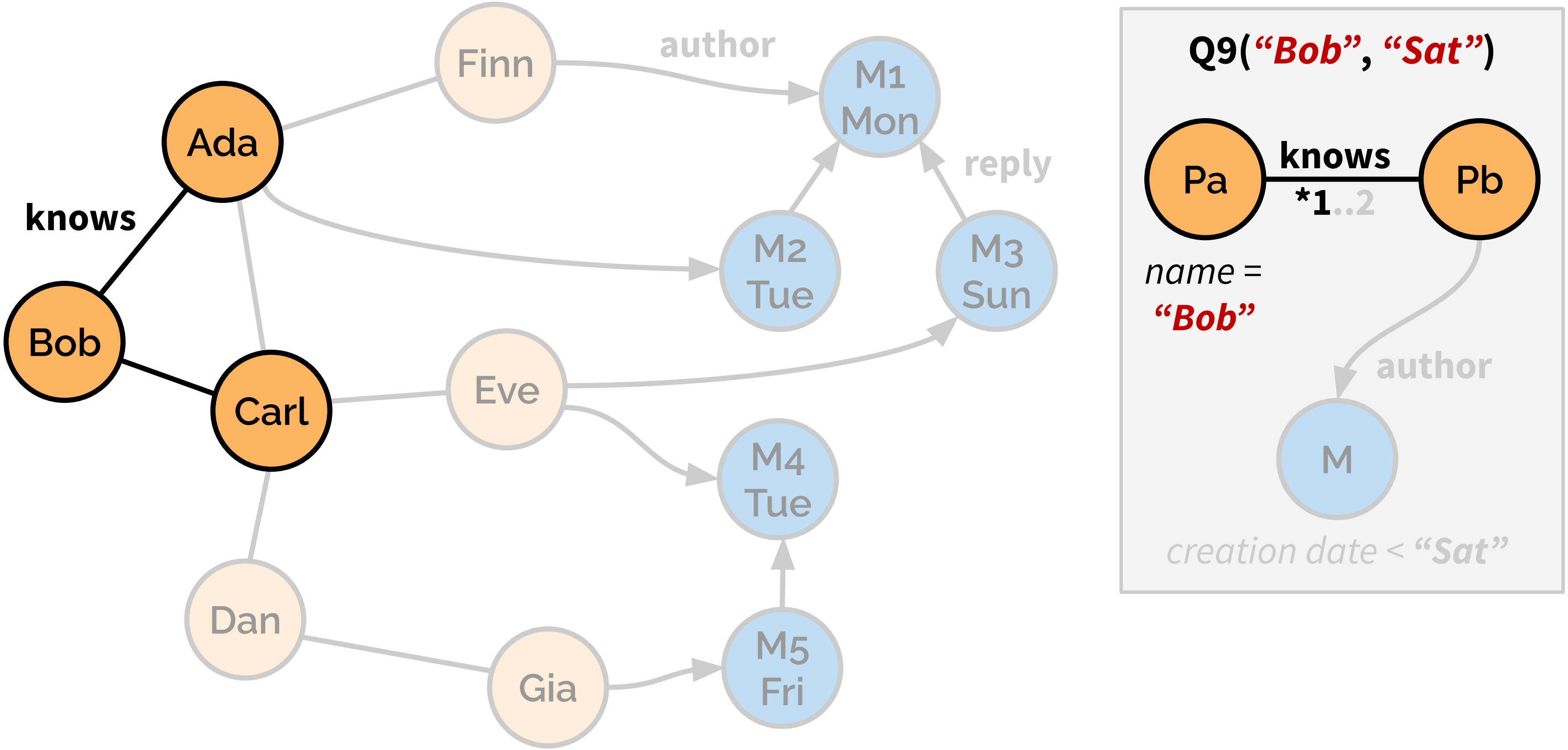
Data set | **Queries** | Updates

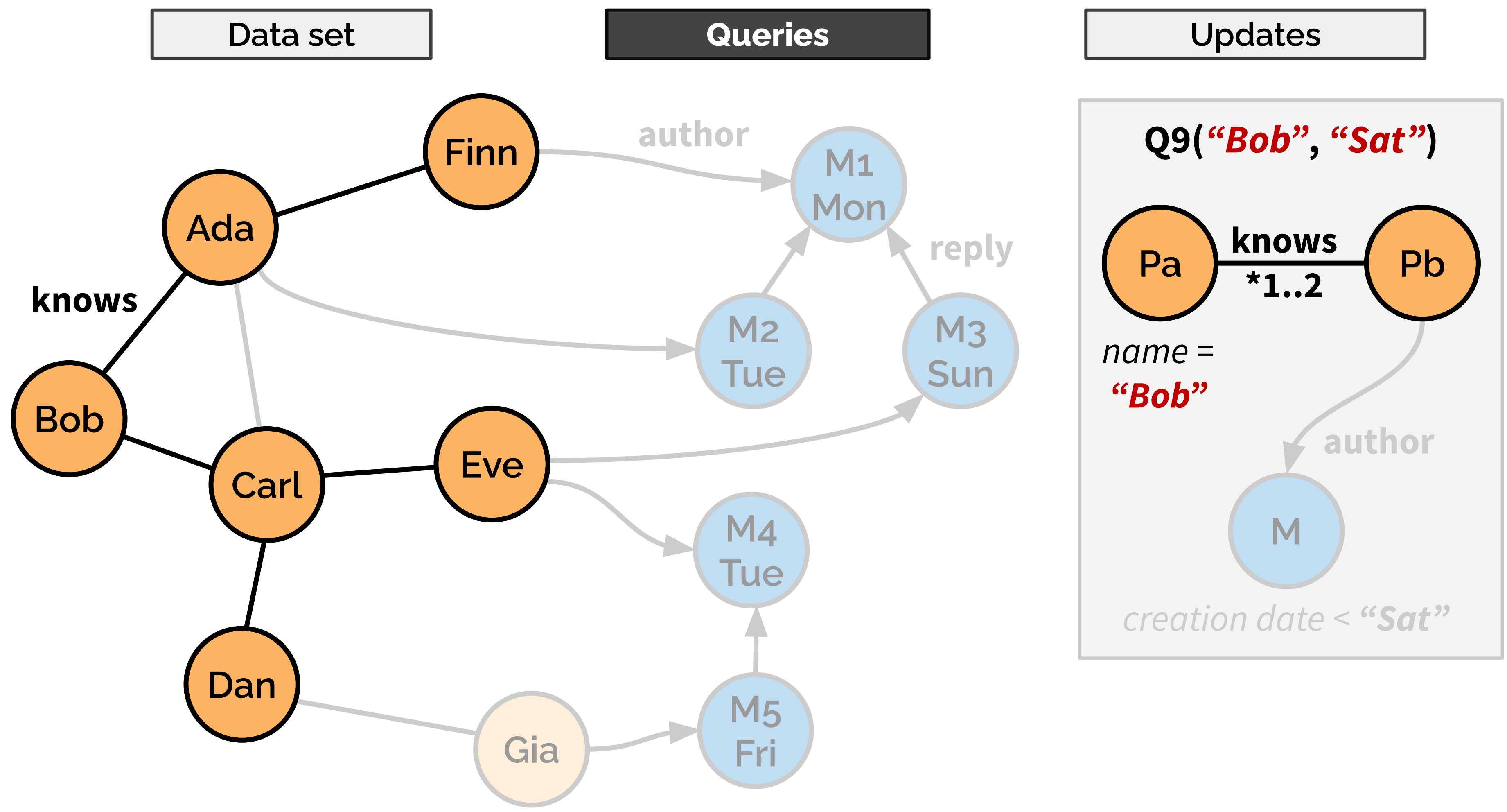


Data set

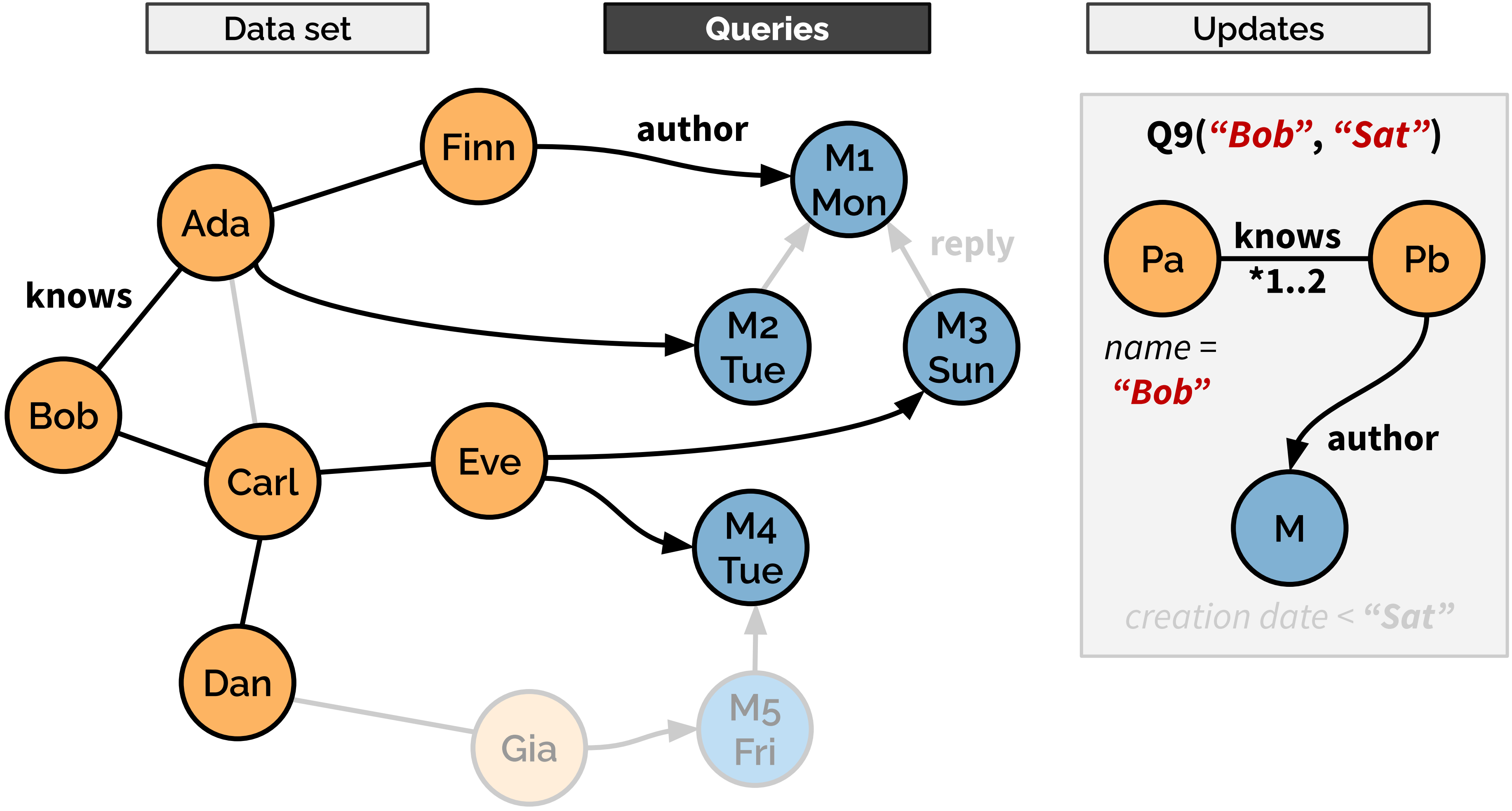
Queries

Updates

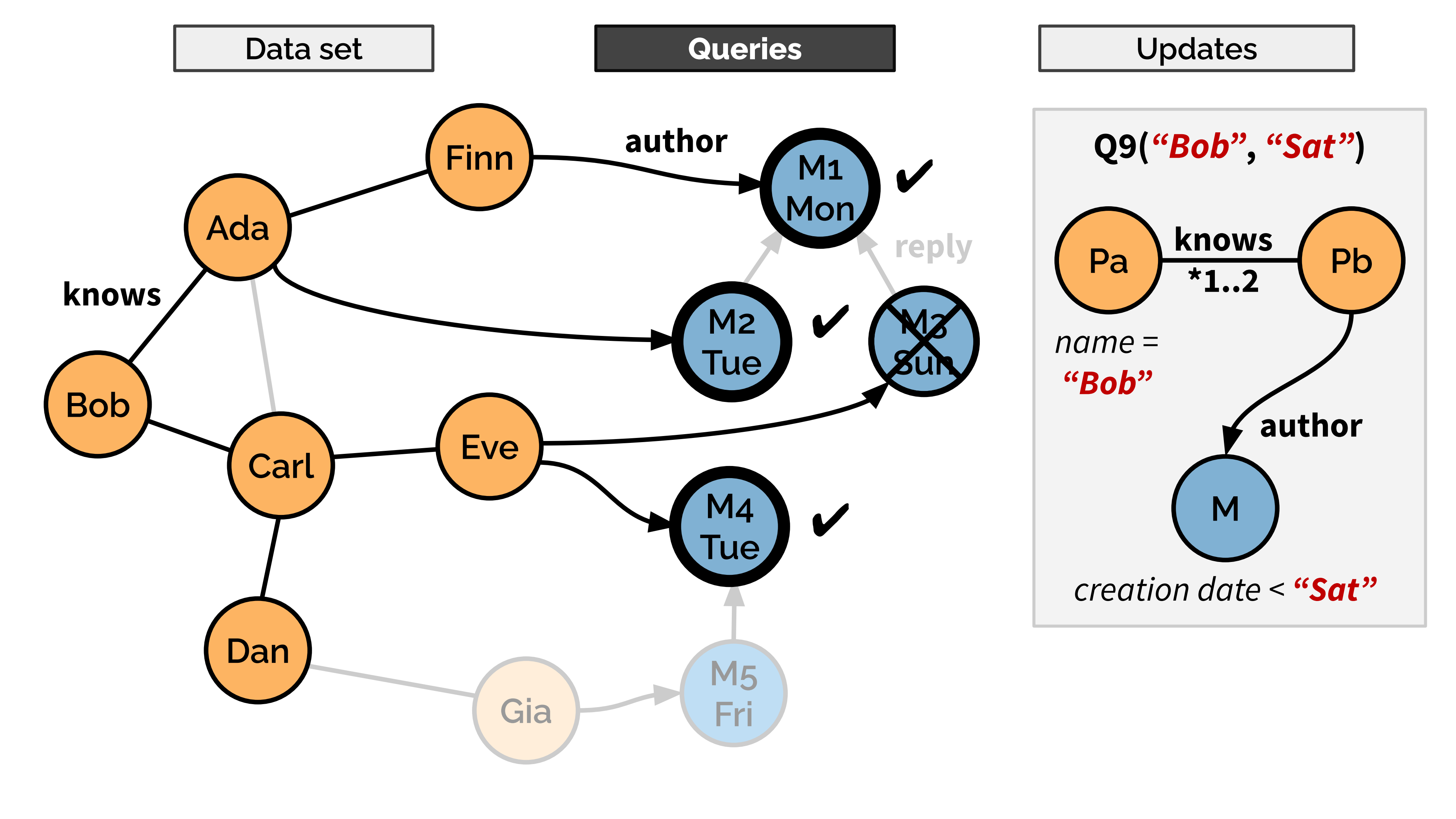


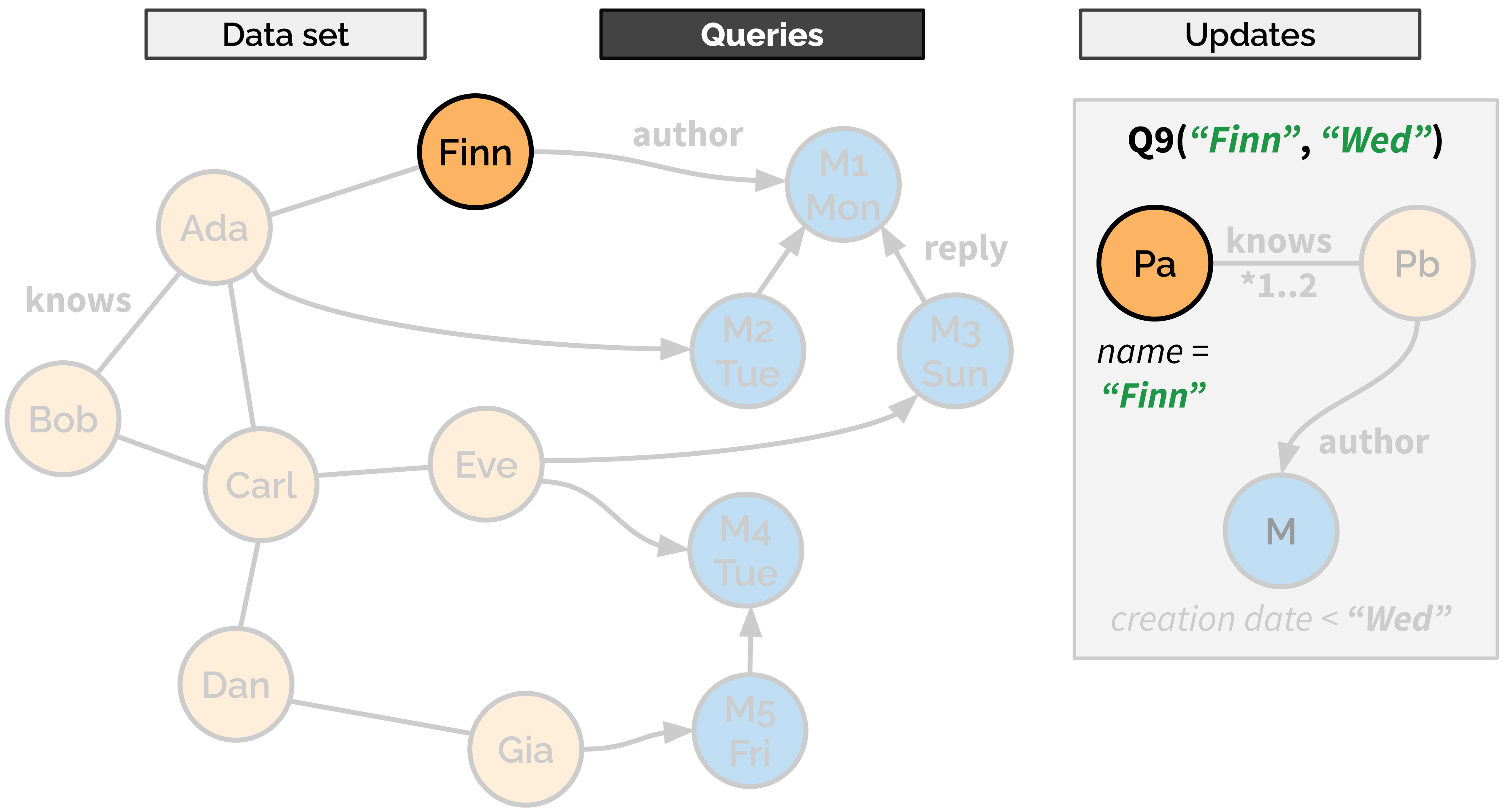


Data set
Queries
Updates
Finn
Ada
knows
Bob
Carl
Eve
Dan
Gia
author
M1
Mon
reply
M2
Tue
M3
Sun
M4
Tue
M5
Fri
Q9("Bob", "Sat")
Pa
knows
*1..2
Pb
name =
"Bob"
author
M
creation date < "Sat"

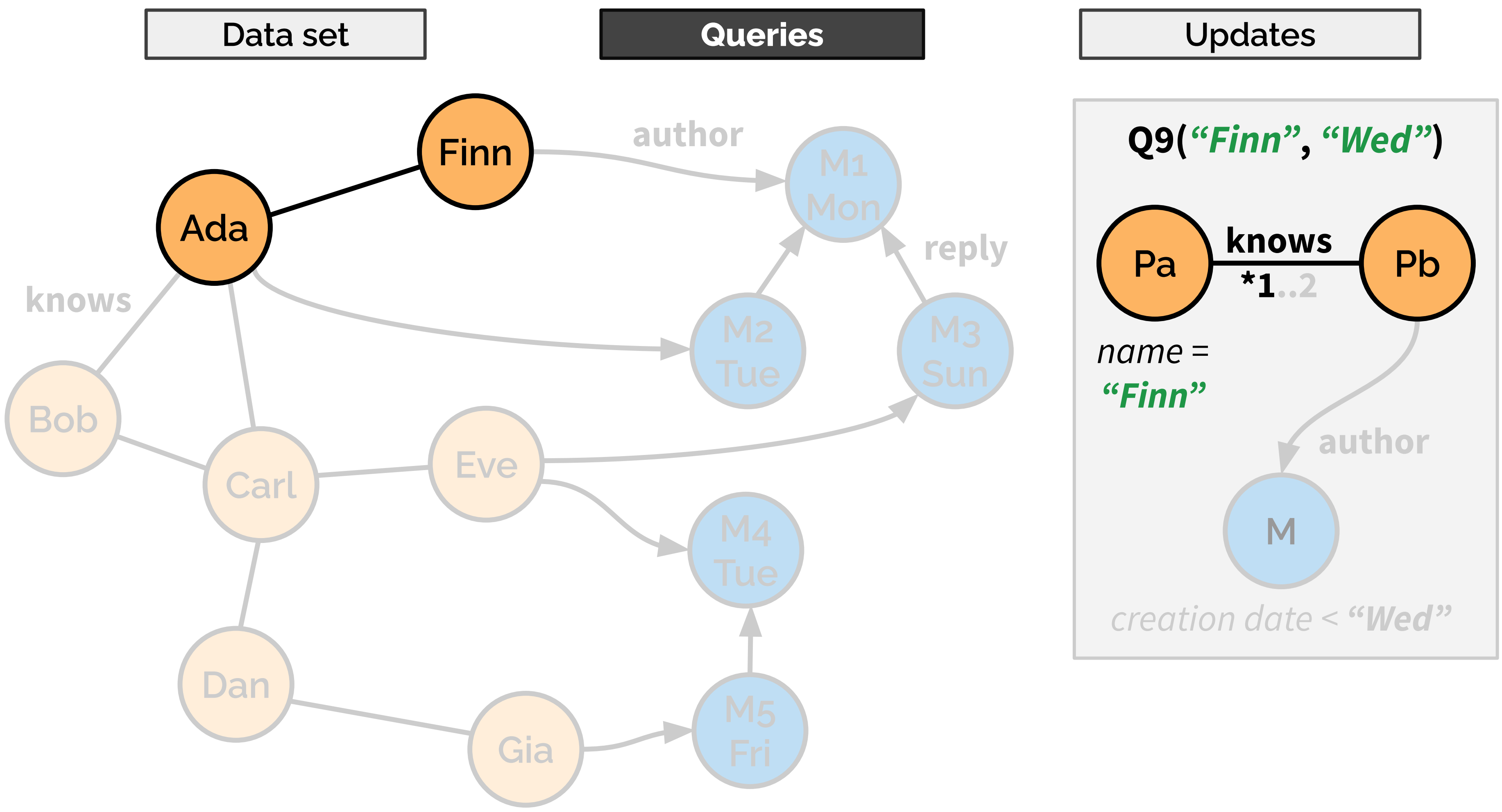


Data set
Queries
Updates
author
M1
Mon
reply
Finn
Ada
knows
M2
Tue
M3
Sun
Bob
Carl
Eve
M4
Tue
Dan
Gia
M5
Fri
Q9("Bob", "Sat")
Pa
knows
*1..2
Pb
name =
"Bob"
author
M
creation date < "Sat"


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



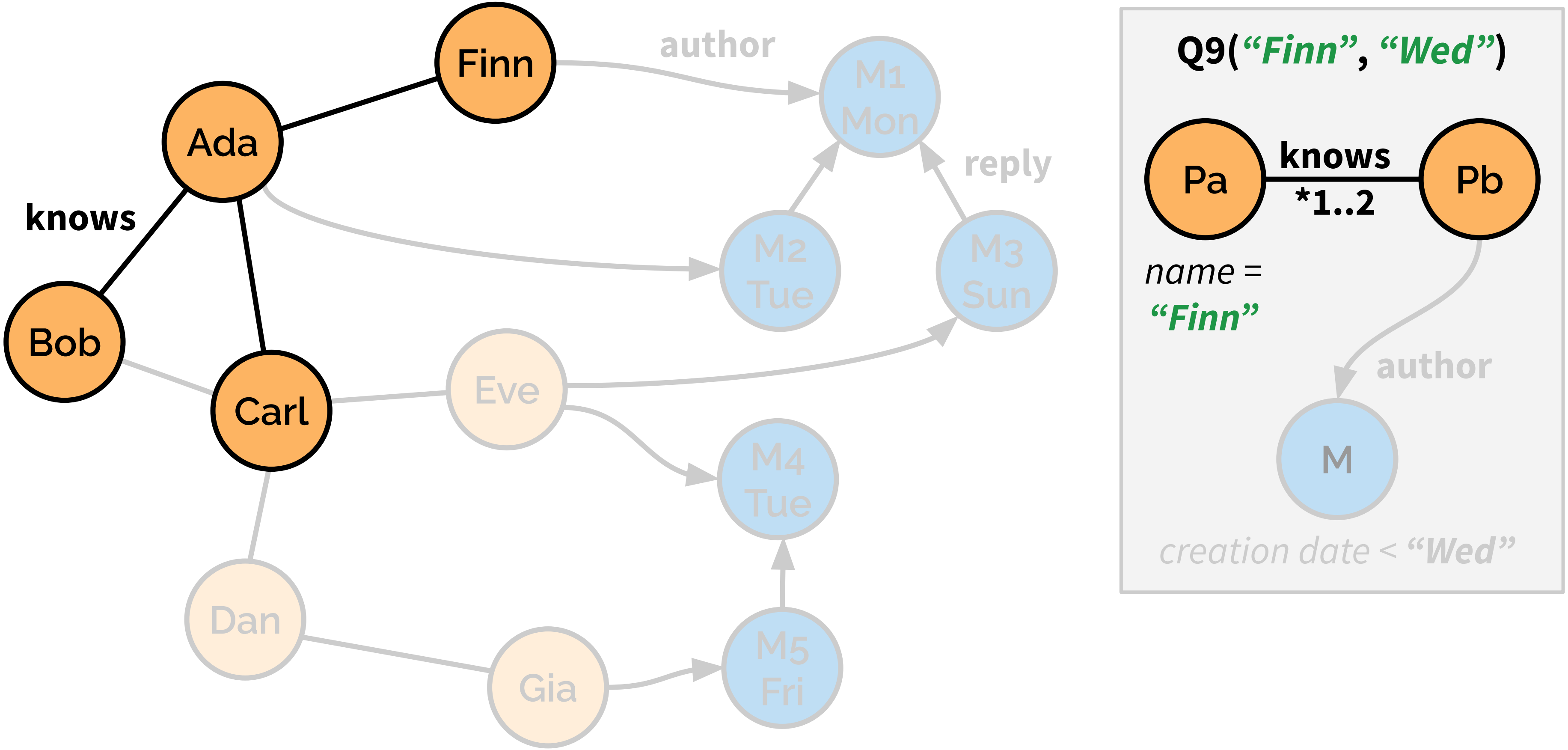
Data set
Queries
Updates
Finn
author
M1 Mon
Ada
reply
knows
M2 Tue
M3 Sun
Bob
Carl
Eve
M4 Tue
Dan
Gia
M5 Fri
Q9("Finn", "Wed")
Pa
knows *1..2
Pb
name = "Finn"
author
M
creation date < "Wed"


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

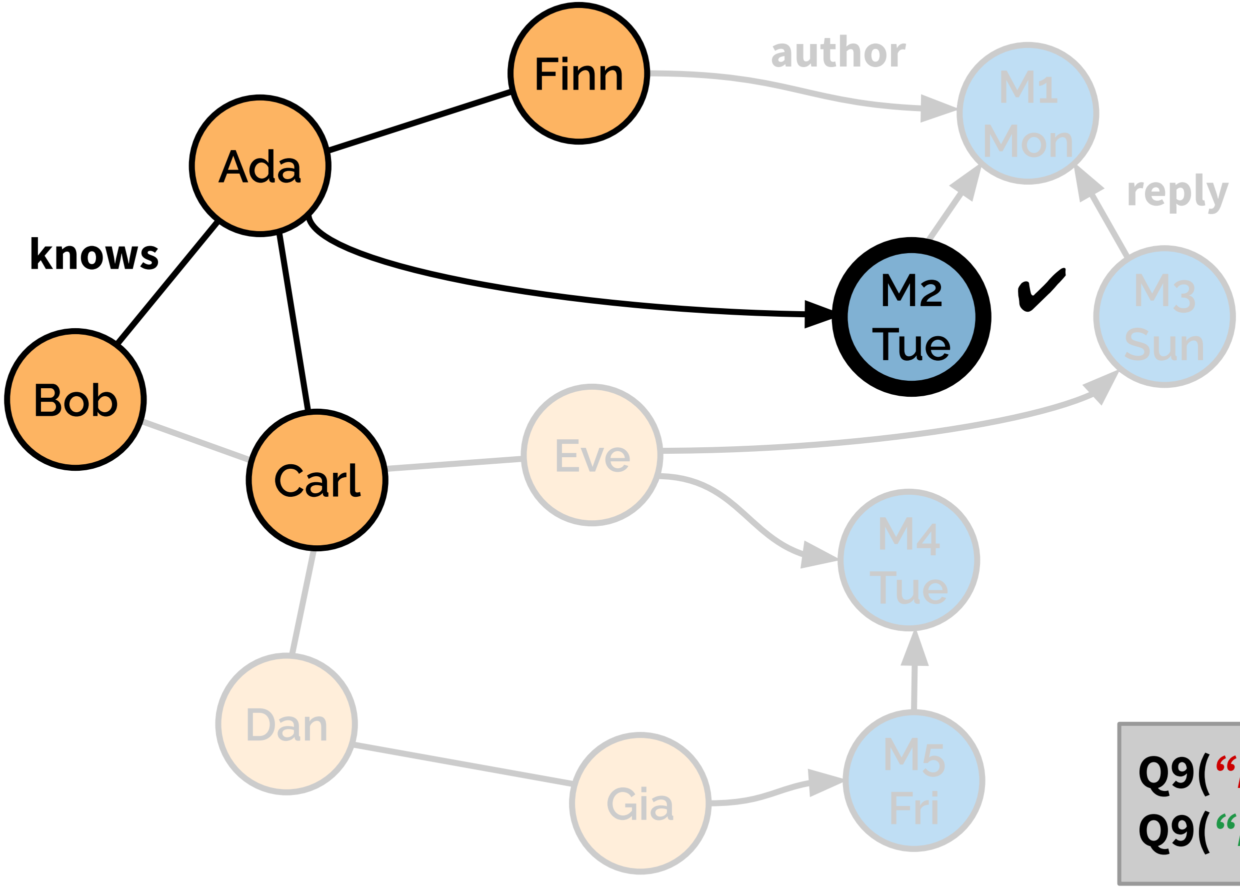
Data set

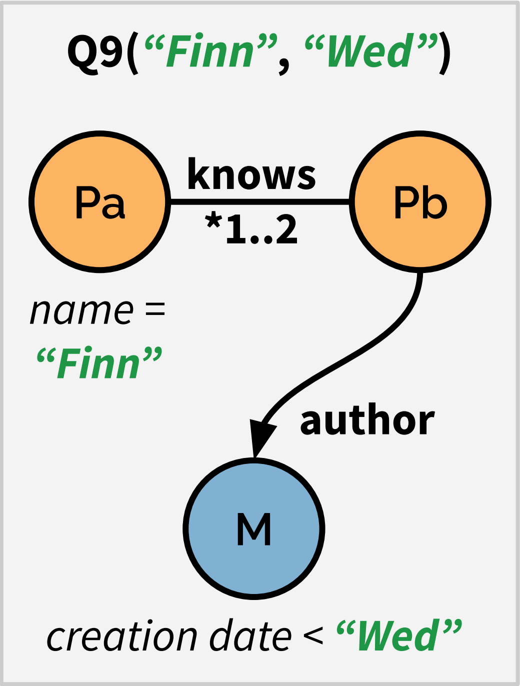
Queries

Updates

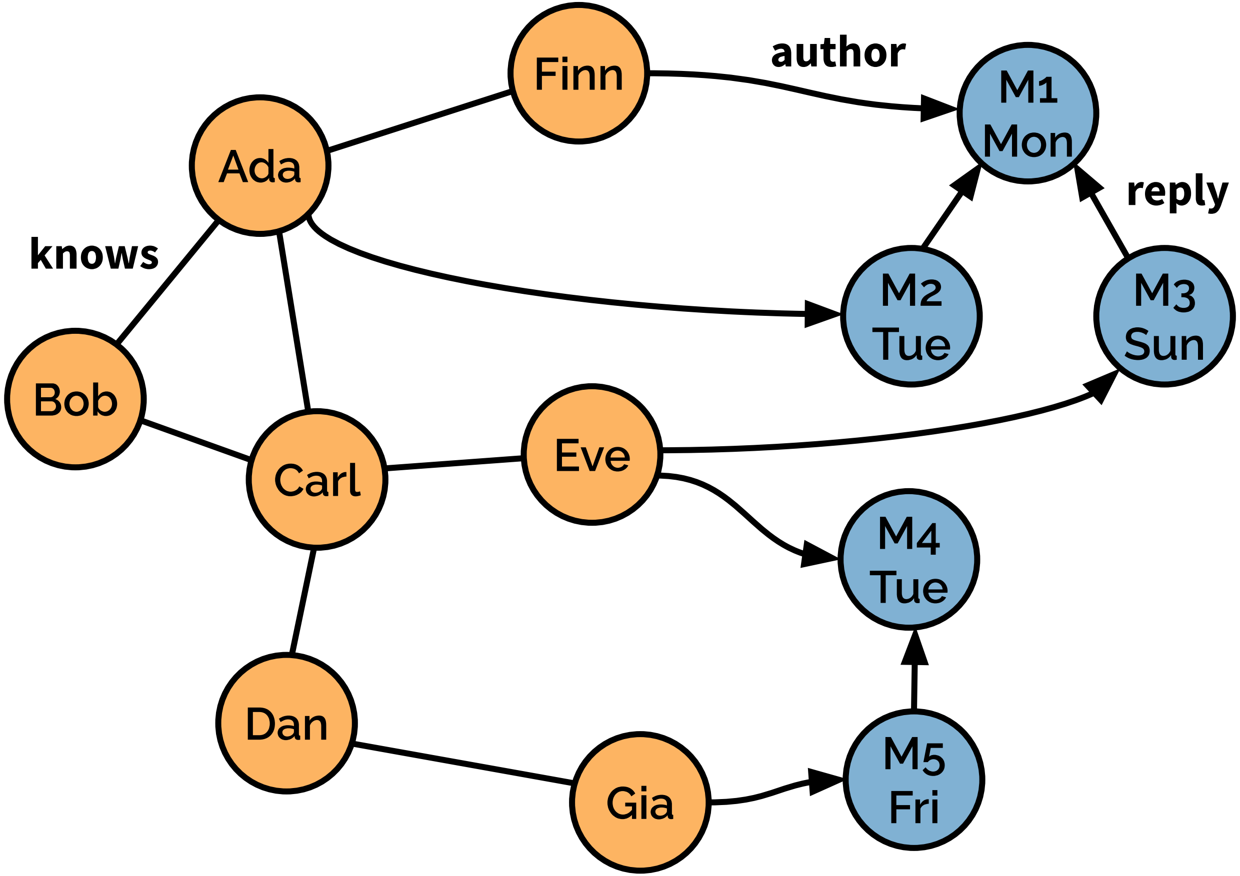


Data set | **Queries** | Updates





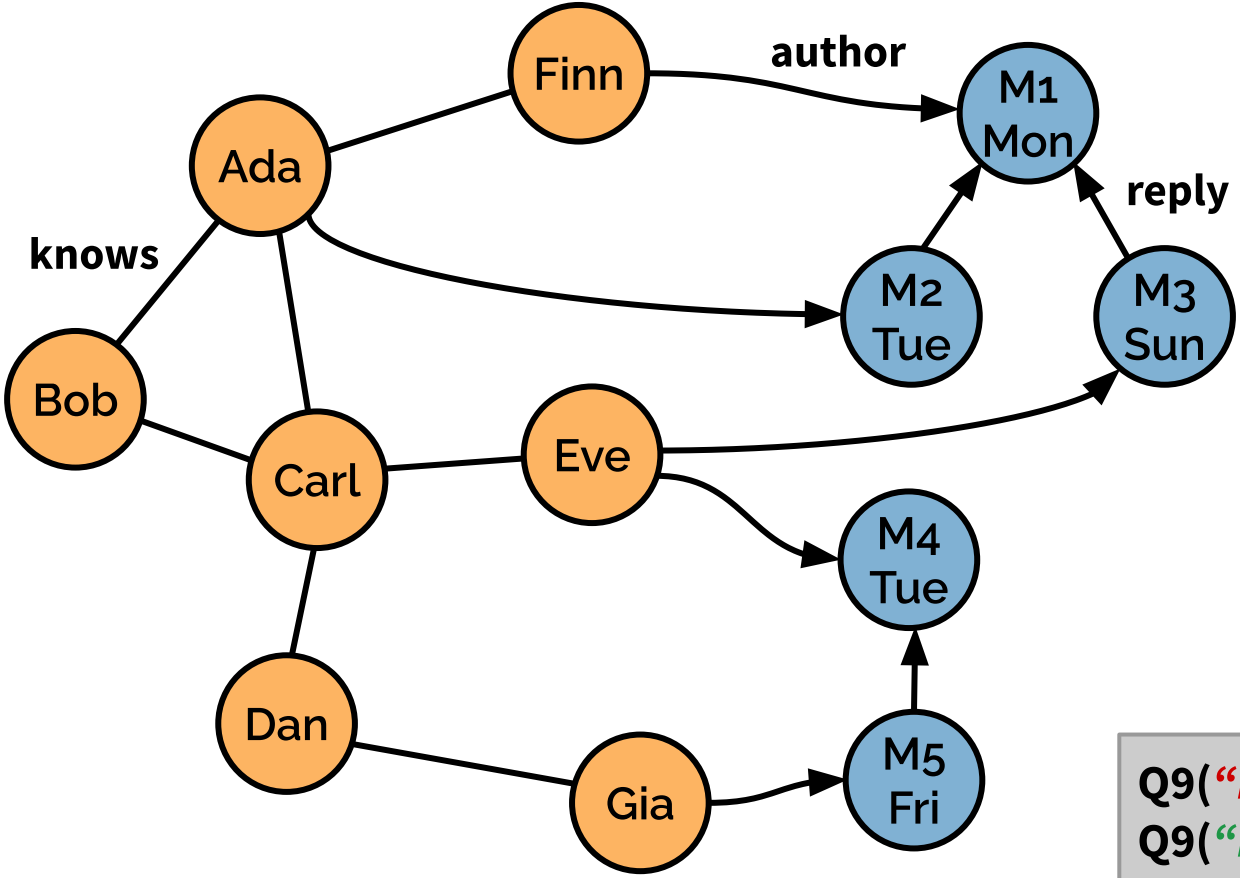
Q9(*"Bob"*, *"Sat"*): 10 nodes
Q9(*"Finn"*, *"Wed"*): 5 nodes



Factor table

name	1-hop friends	2-hop friends
Bob	**2**	**5**
Carl	4	4
Ada	3	4
Dan	2	3
Eve	1	3
Finn	**1**	**2**
Gia	1	1

Data set | **Queries** | Updates

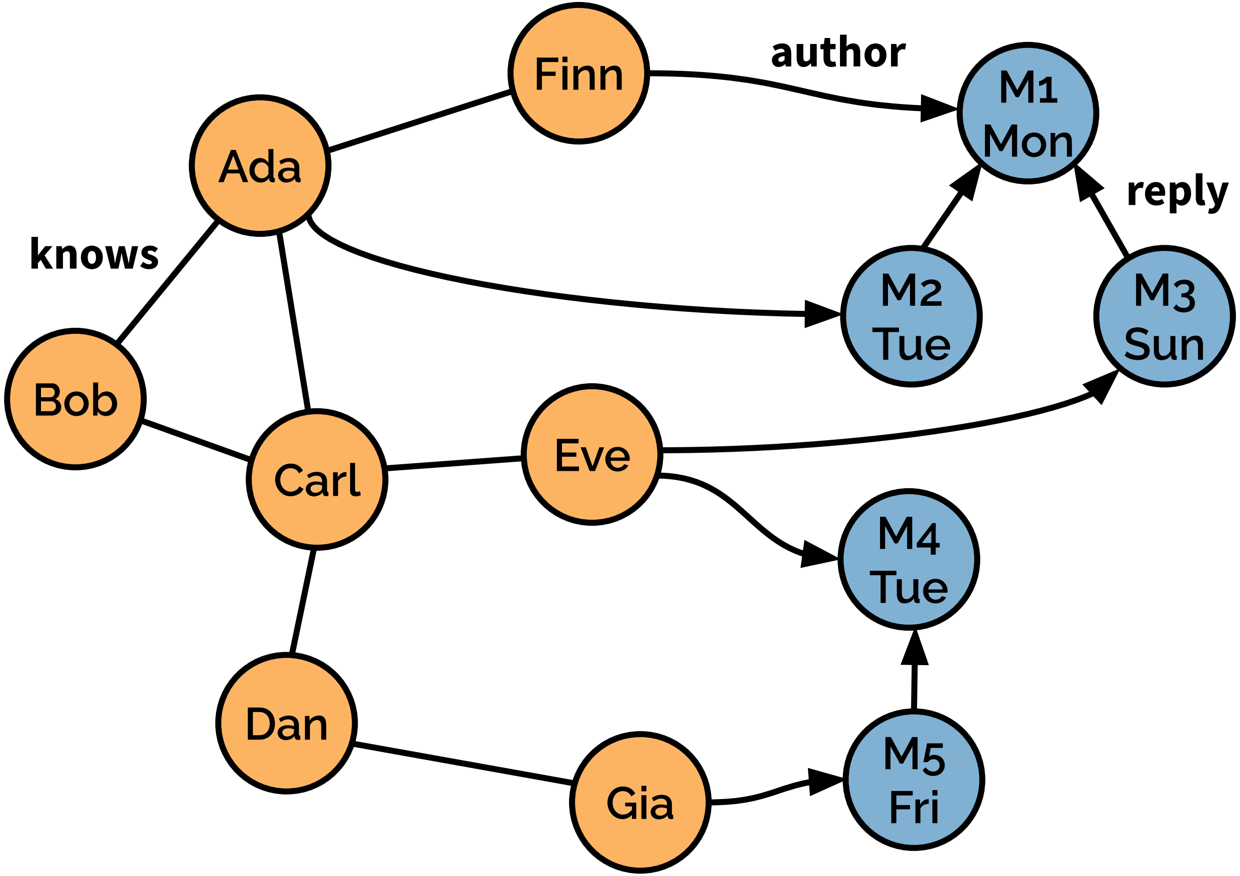


Factor table

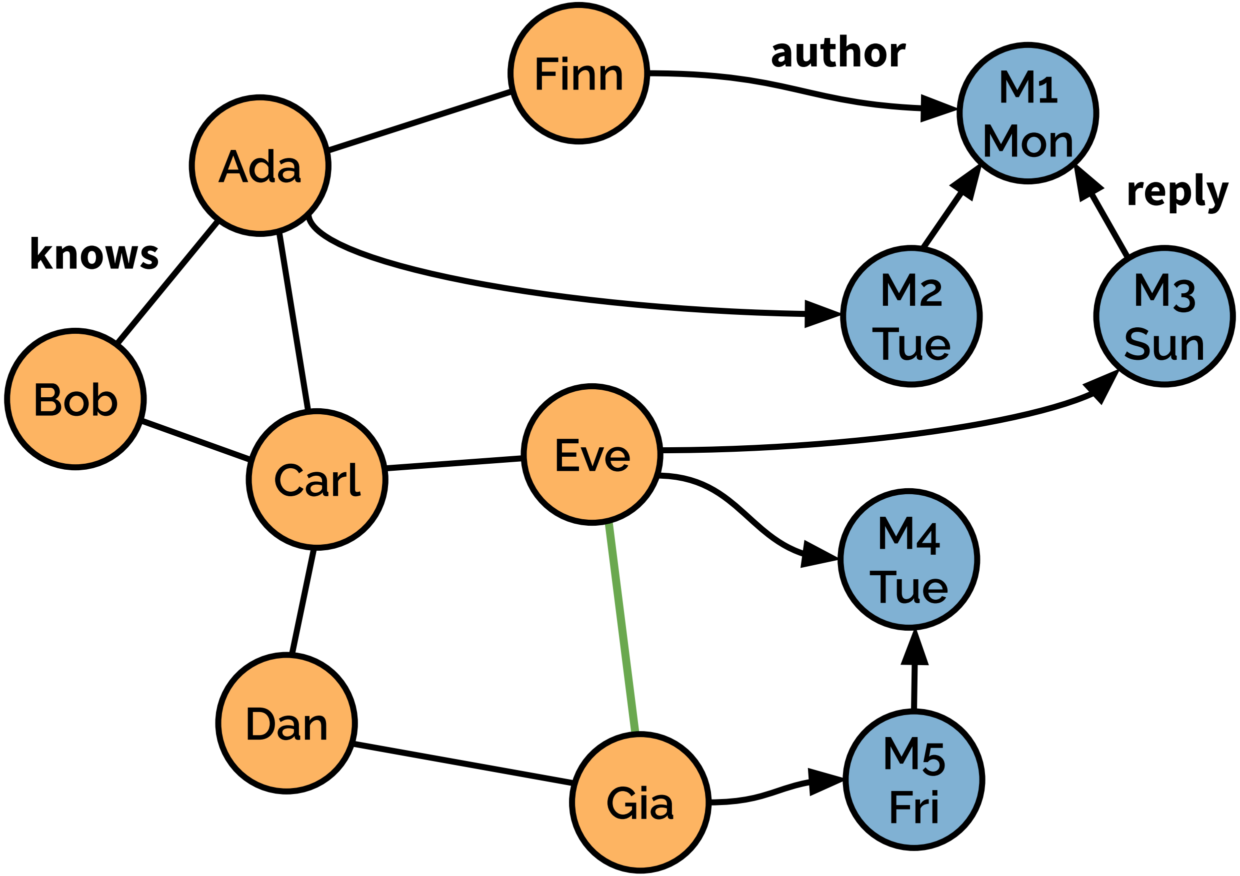
name	1-hop friends	2-hop friends
Bob	**2**	**5**
Carl	4	4
Ada	3	4
Dan	2	3
Eve	1	3
Finn	**1**	**2**
		1

Q9(“*Bob*”, “*Sat*”): 10 nodes
Q9(“*Finn*”, “*Wed*”): 5 nodes

Data set | Queries | **Updates**



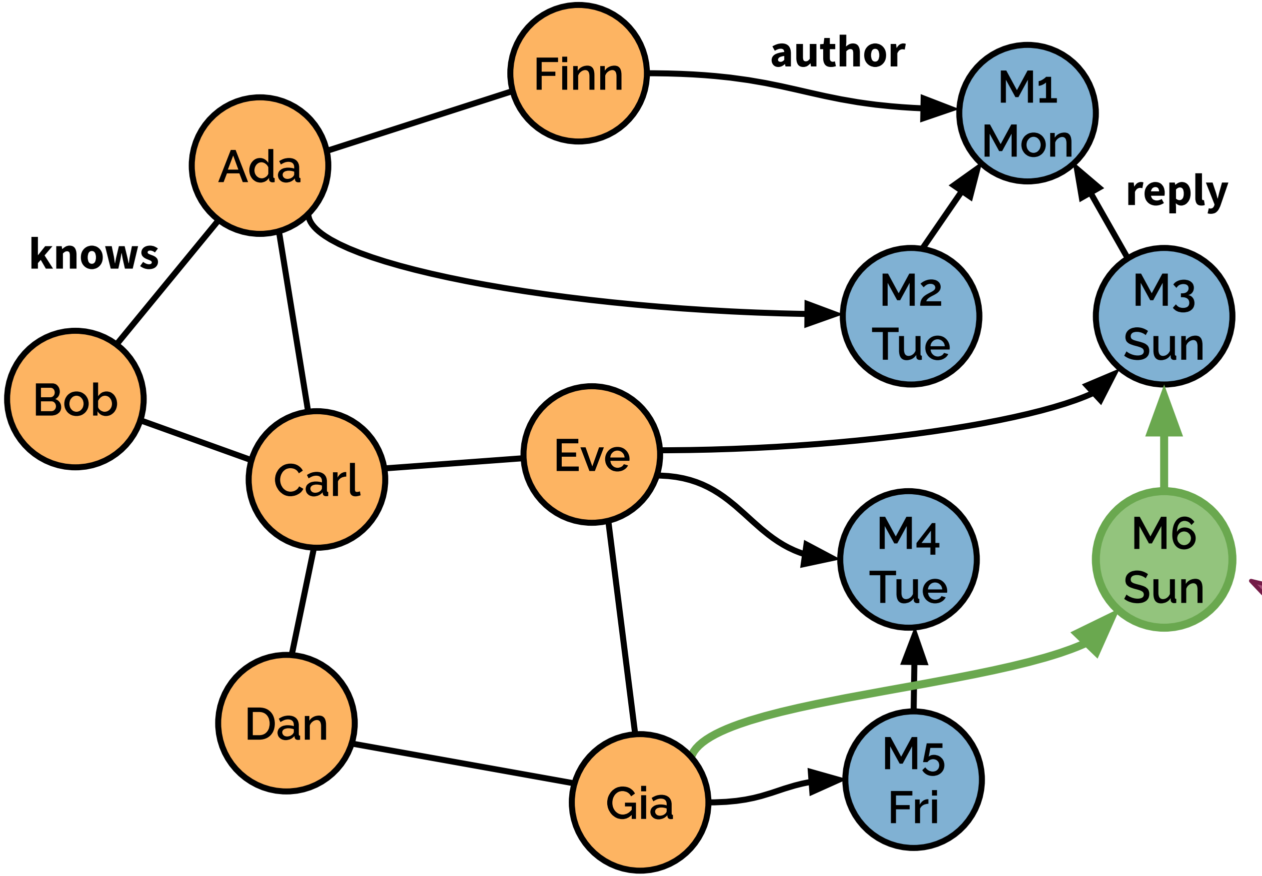
Data set | Queries | **Updates**



Updates

+ knows(“Eve”, “Gia”)

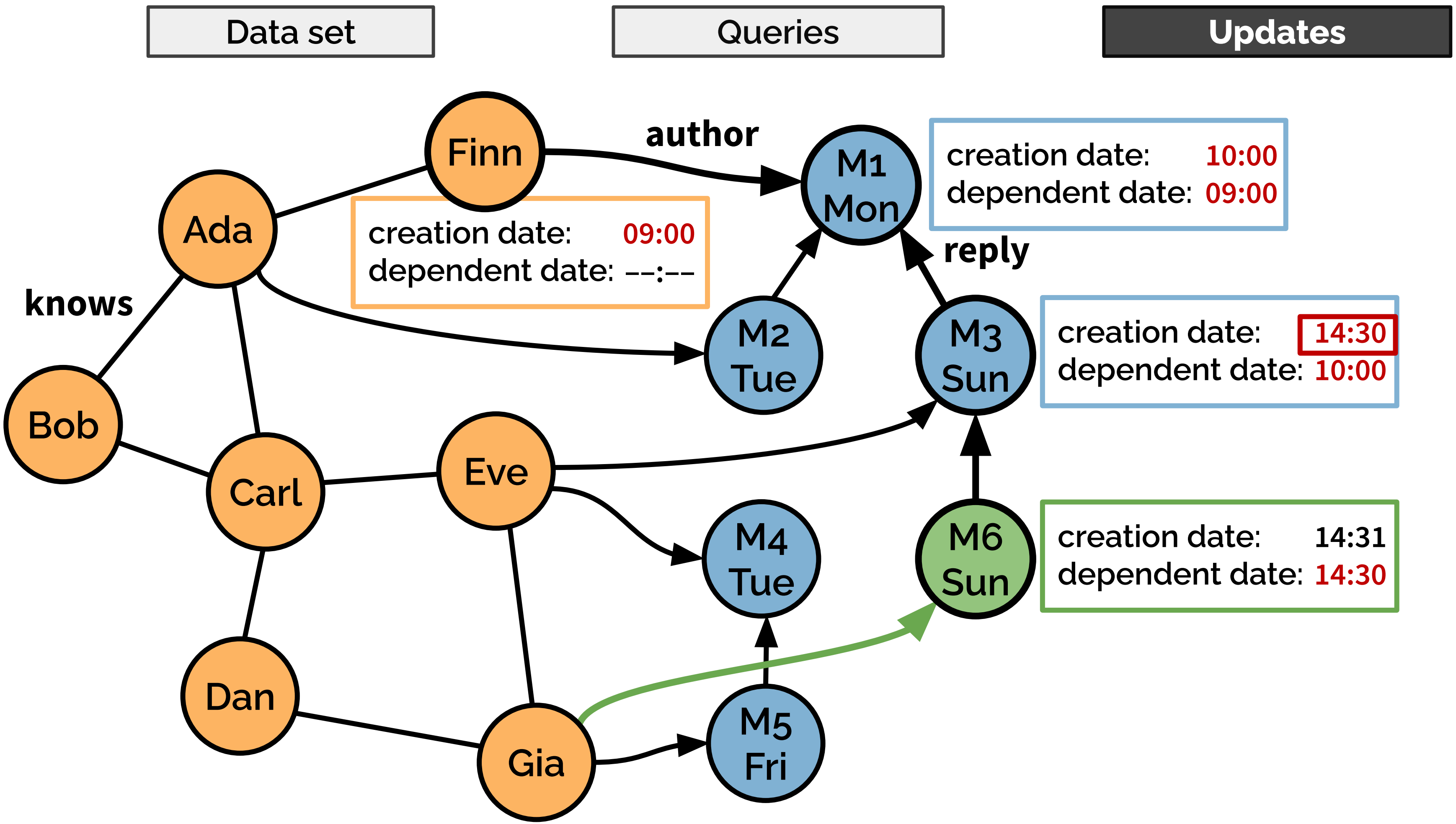
Data set | Queries | **Updates**



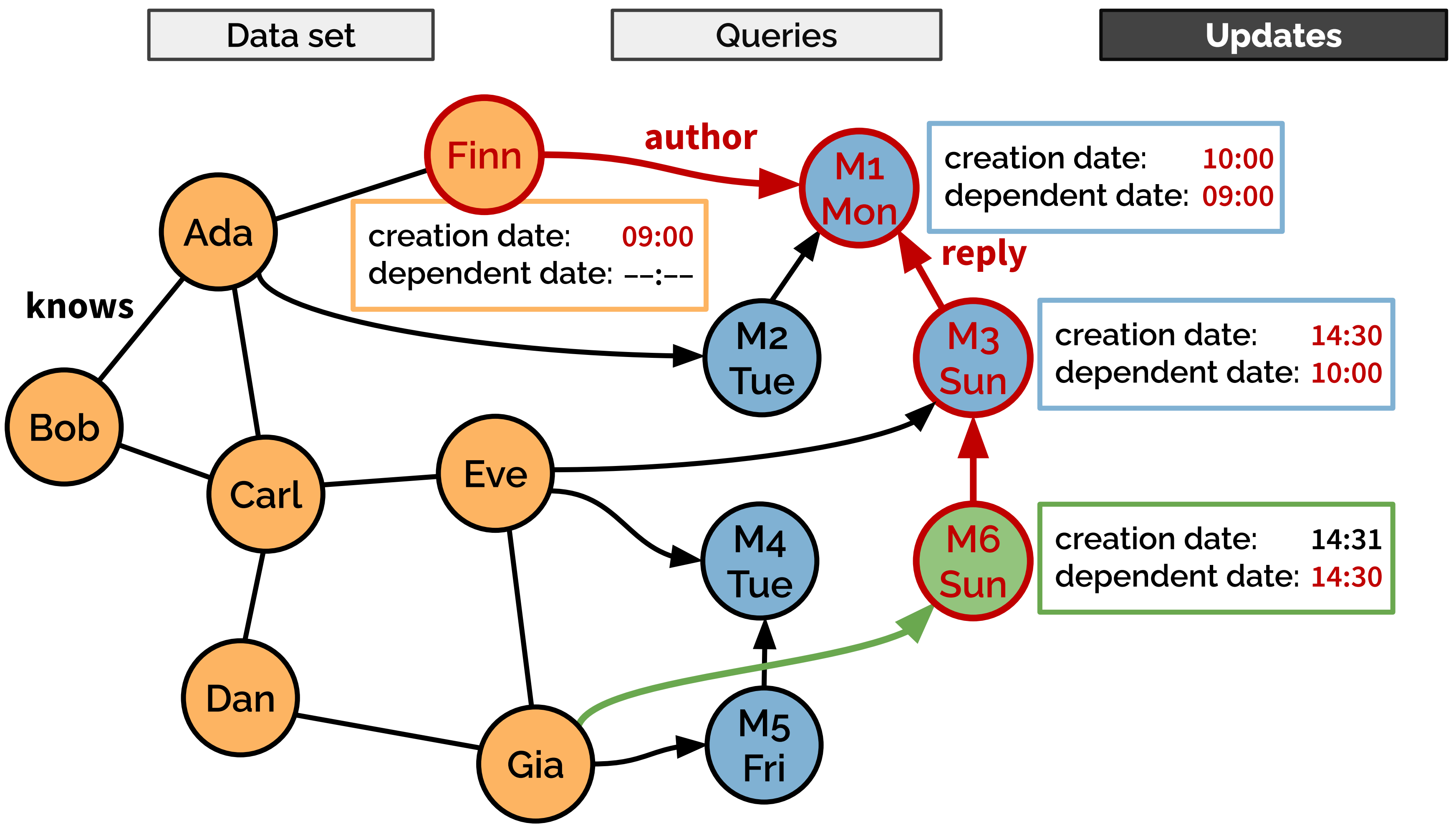
Updates

+ knows("Eve", "Gia")
+ Comment("Gia", "M3")

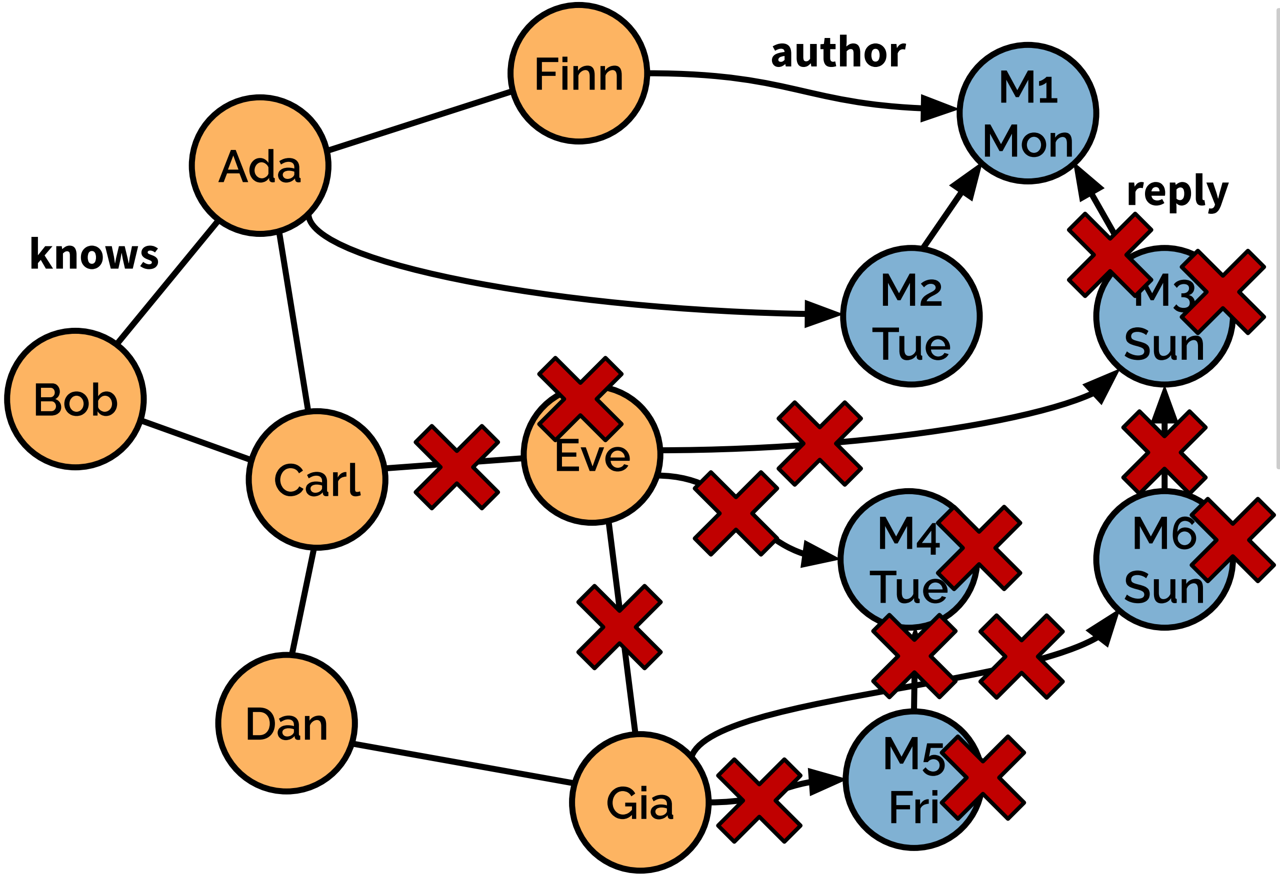
When is this operation executable?



Data set
Queries
Updates
author
knows
reply
Finn
Ada
Bob
Carl
Eve
Dan
Gia
M1 Mon
M2 Tue
M3 Sun
M4 Tue
M5 Fri
M6 Sun
creation date: 09:00
dependent date: --:--
creation date: 10:00
dependent date: 09:00
creation date: 14:30
dependent date: 10:00
creation date: 14:31
dependent date: 14:30


Data set
Queries
Updates
Finn
author
M1
Mon
creation date: 10:00
dependent date: 09:00
Ada
creation date: 09:00
dependent date: --:--
reply
knows
M2
Tue
M3
Sun
creation date: 14:30
dependent date: 10:00
Bob
Carl
Eve
M4
Tue
M6
Sun
creation date: 14:31
dependent date: 14:30
Dan
Gia
M5
Fri

Data set | Queries | **Updates**

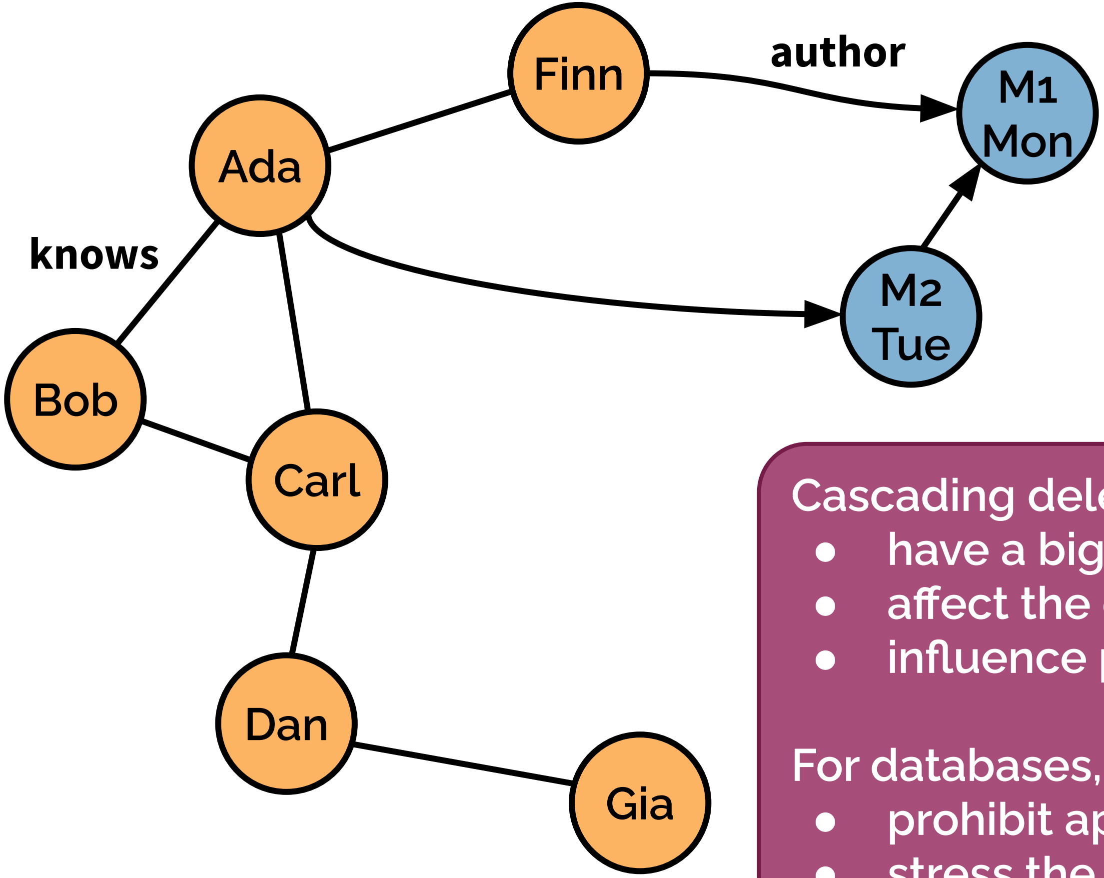


Updates

+ knows(“Eve”, “Gia”)

+ Comment(“Gia”, “M3”)

– Person(“Eve”)



Updates

+ knows(“Eve”, “Gia”)

+ Comment(“Gia”, “M3”)

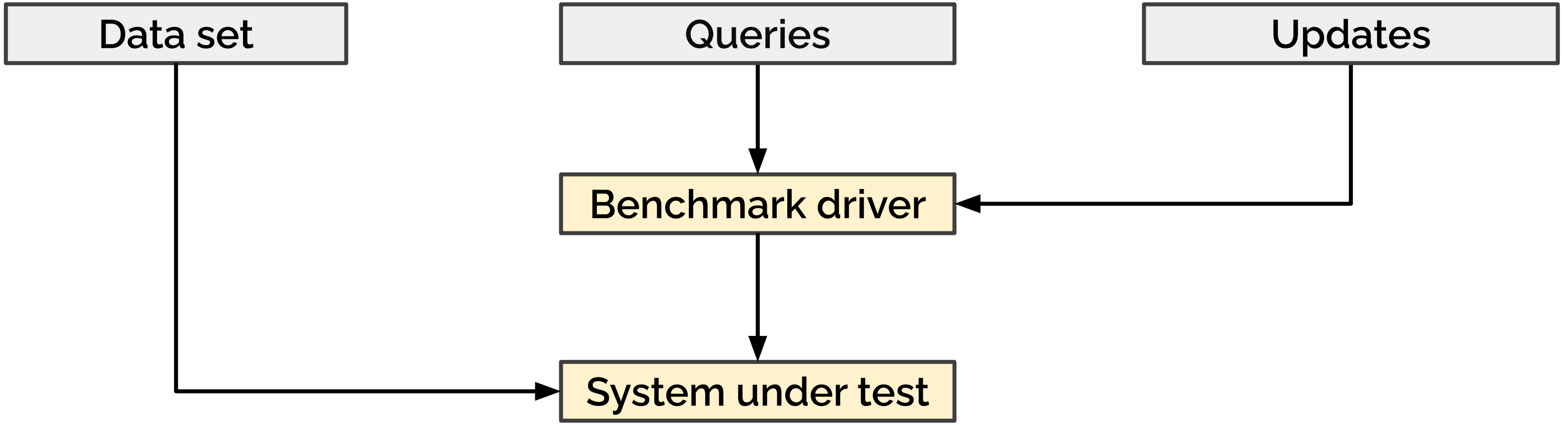
– Person(“Eve”)

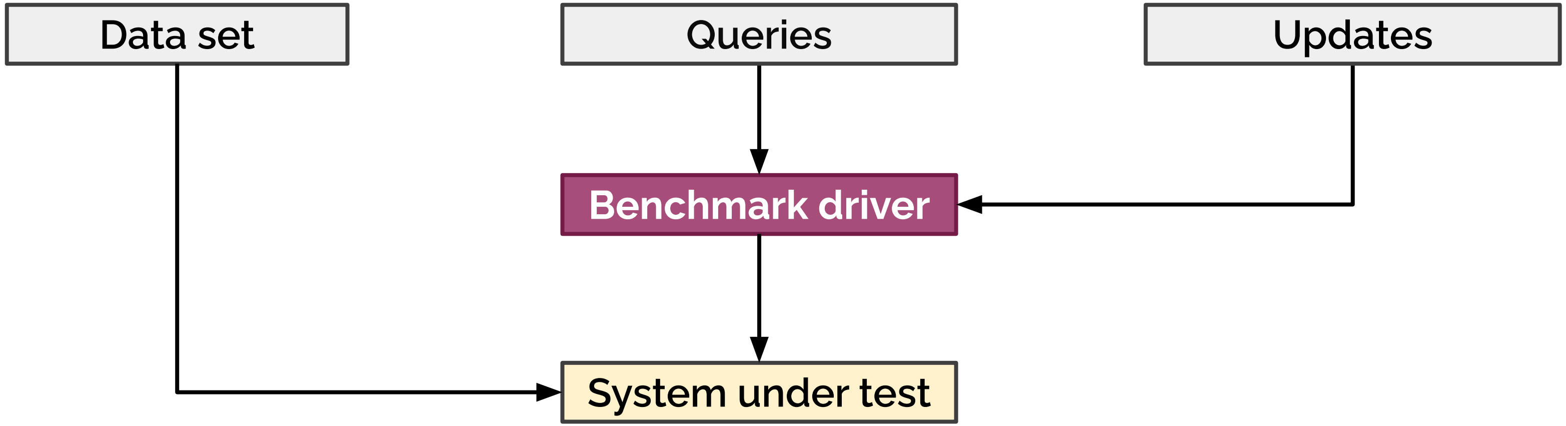
Cascading deletes remove lots of entities:

- have a big impact on the data distribution
- affect the executability of operations
- influence parameter selection

For databases, deletes:

- prohibit append-only data structures
- stress the garbage collector

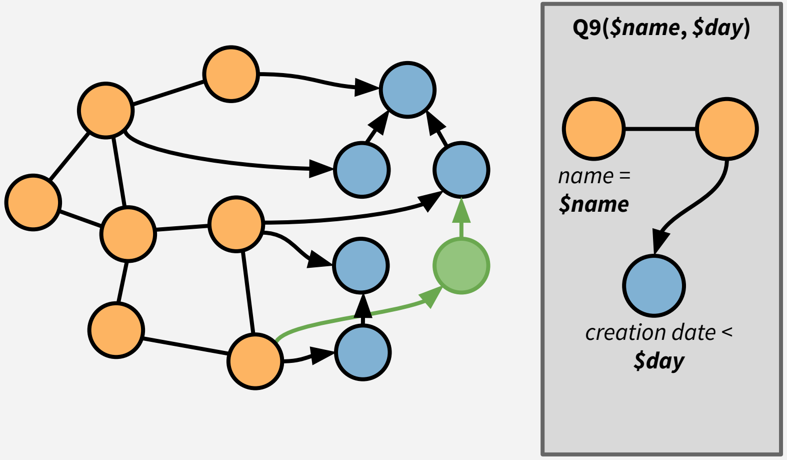

Data set
Queries
Updates
Benchmark driver
System under test



- **Schedules operations to be executable**
  (hard: needs careful parameter selection and dependency tracking)
- **Runs queries and updates concurrently**
  (hard: needs partitioned updates)
- **Collects benchmark results and performs validation**
  (very hard due to concurrent updates: we perform it sequentially)

SNB Workloads

SNB Interactive v1 (2015)

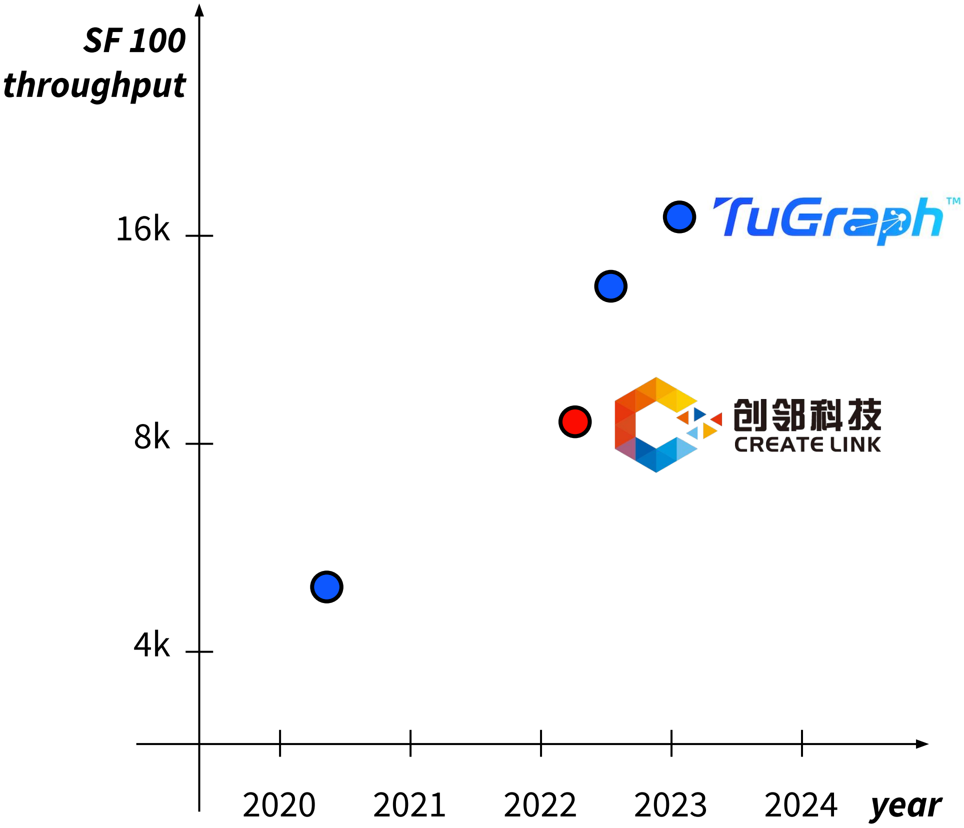


Queries start in 1–2 person nodes

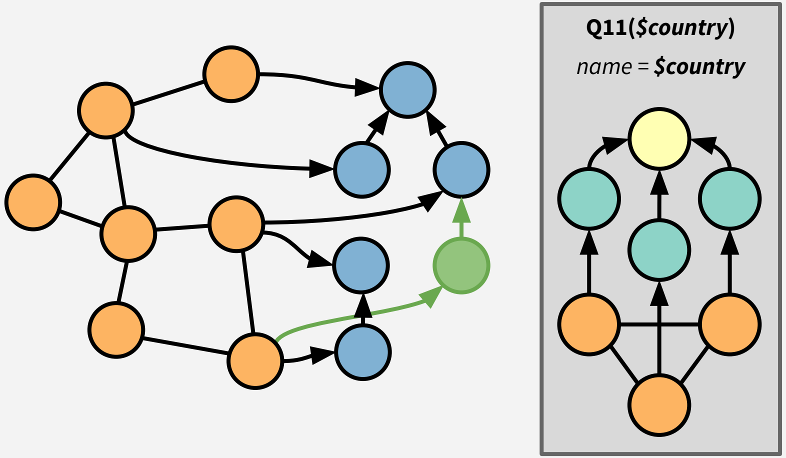
Concurrent inserts (no deletes)

Goal: high throughput (ops/s)

Results on the 100GB data set



SNB Business Intelligence (2023)



Queries touch on large portions of the data

Both bulk and concurrent updates allowed

Goal: high throughput & low query runtimes

Audited results

TigerGraph

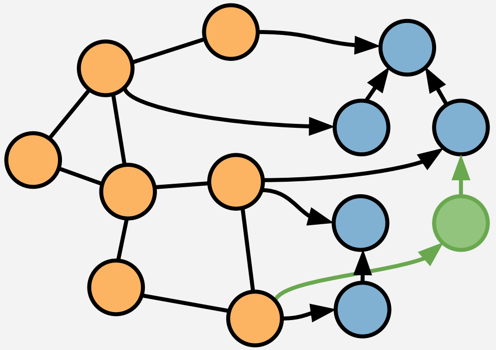
Results for 100GB, 1TB, and 10TB

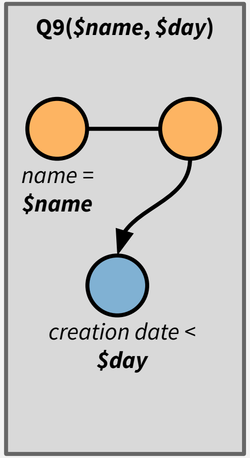
10TB:

- Power@SF: 89,444
- Throughput@SF: 30,990

More results expected in 2023

SNB Interactive v2





Queries start in 1–2 person nodes

Concurrent inserts and deletes

Goal: high throughput (ops/s)

Features backported from BI:

- delete operations
- larger scale factors up to SF30,000
- cheapest path query

New parameter generation features:

- temporal bucketing for each day
- path curation

The LDBC Social Network Benchmark Interactive Workload v2: A Transactional Graph Query Benchmark with Deep Delete Operations

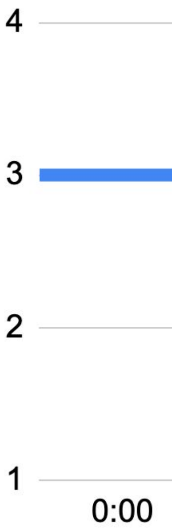
David Püroja[1], Jack Waudby[2], Peter Boncz[1], and Gábor Szárnyas[1]

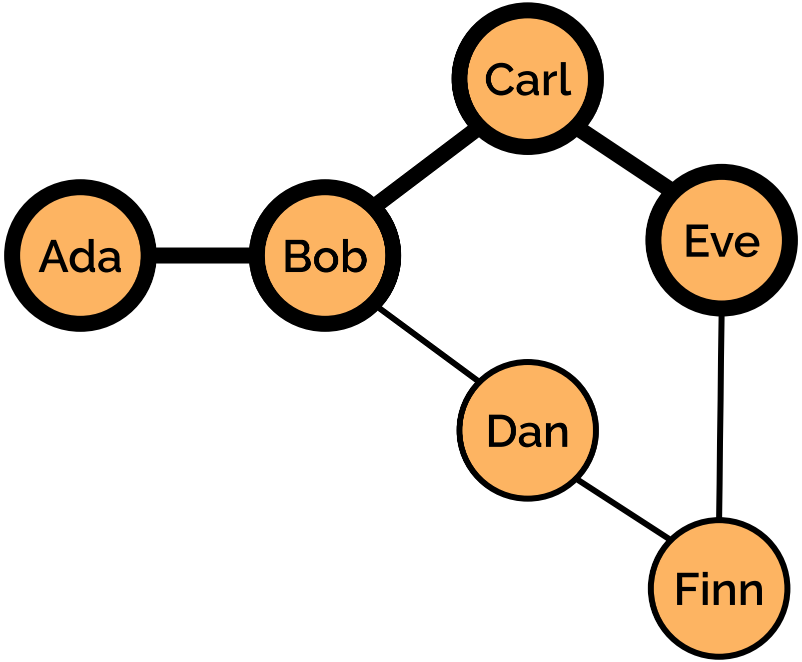
[1] CWI, the Netherlands, [2] Newcastle University, School of Computing
david.puroja@ldbcouncil.org, j.waudby2@newcastle.ac.uk, boncz@cwi.nl, gabor.szarnyas@ldbcouncil.org

Abstract. The LDBC Social Network Benchmark's Interactive workload captures an OLTP scenario operating on a correlated social network graph. It consists of complex graph queries executed concurrently with a stream of updates operation. Since its initial release in 2015, the Interactive workload has become the de facto industry standard for benchmarking transactional graph data management systems. As graph systems have matured and the community's understanding of graph processing features has evolved, we initiated the renewal of this benchmark. This paper describes the Interactive v2 workload with several new features: delete operations, a cheapest path-finding query, support for larger data sets, and a novel temporal parameter curation algorithm that ensures stable runtimes for path queries.

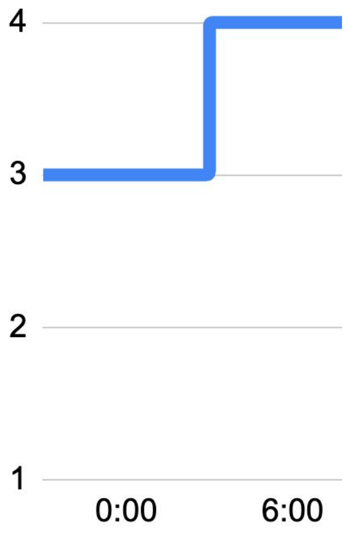
Path curation

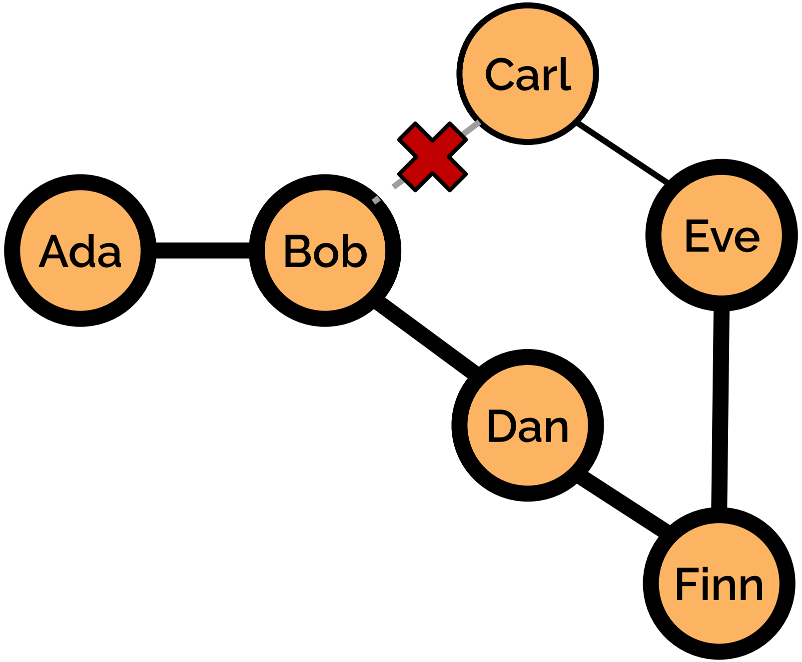
Shortest distance from “Ada” to “Eve”



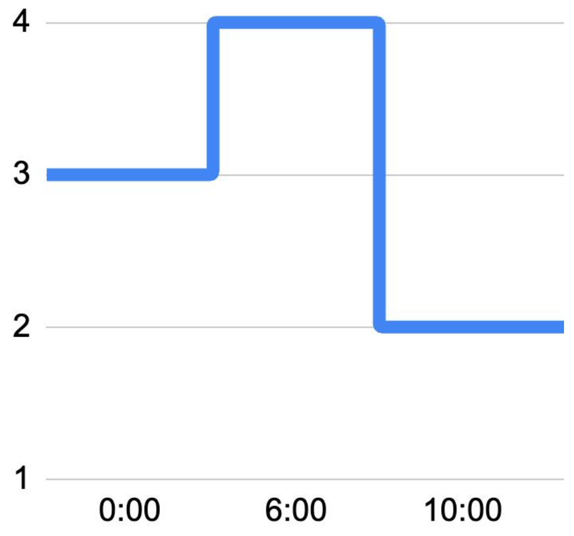


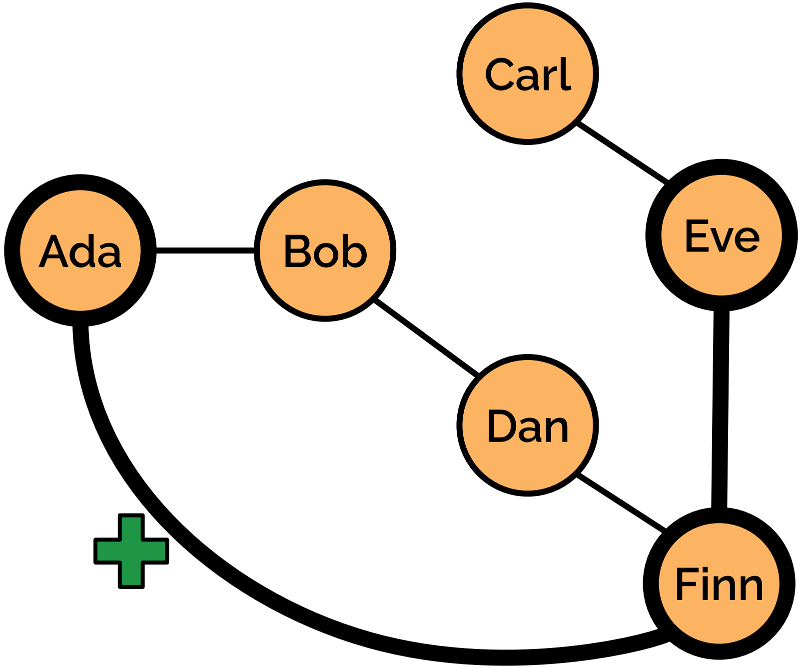
Shortest distance from “Ada” to “Eve”



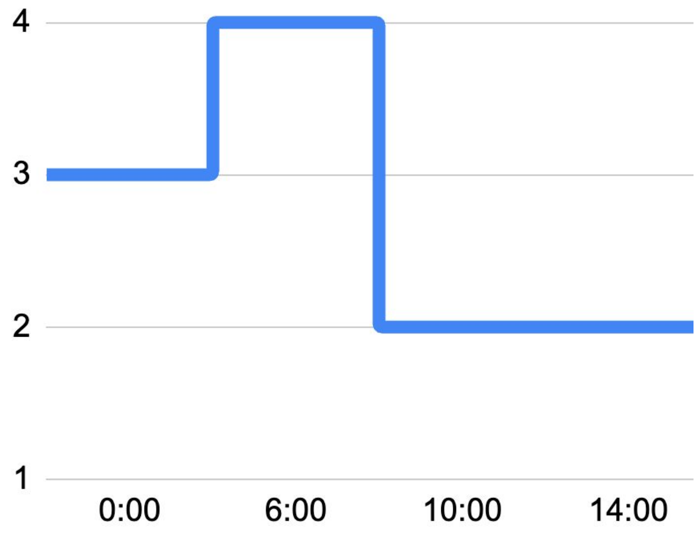


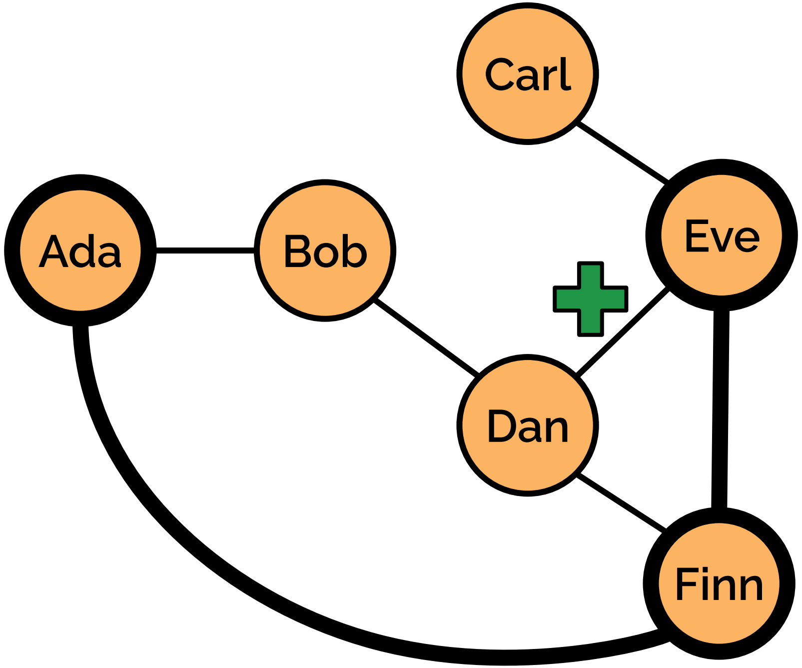
Shortest distance from “Ada” to “Eve”



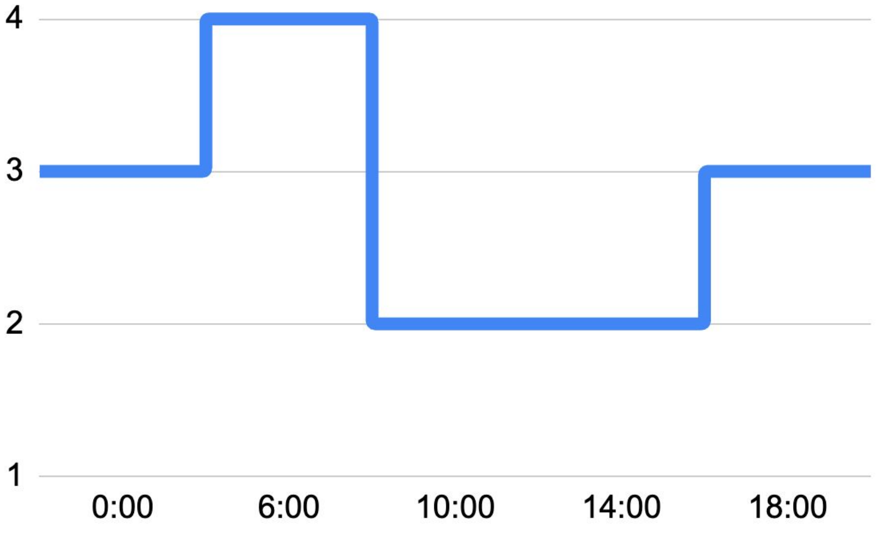


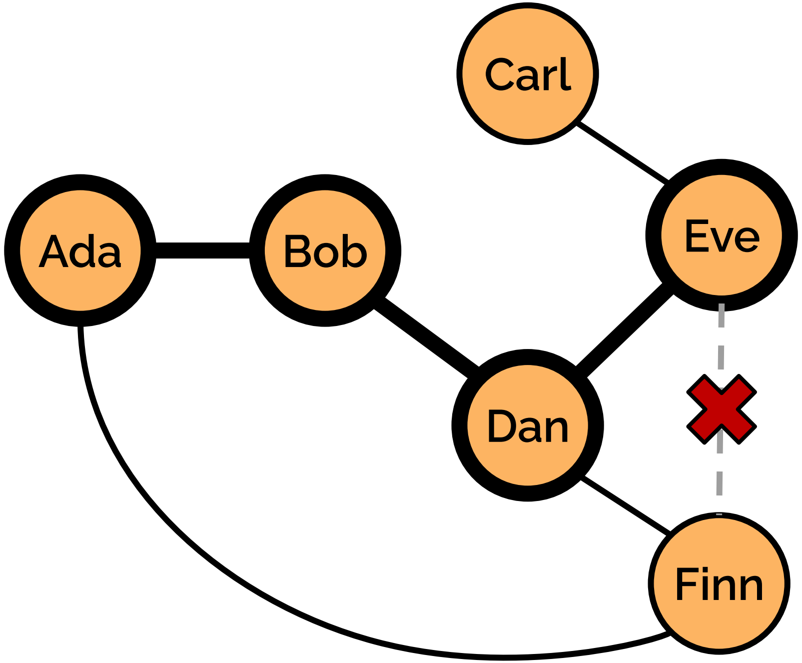
Shortest distance from “Ada” to “Eve”





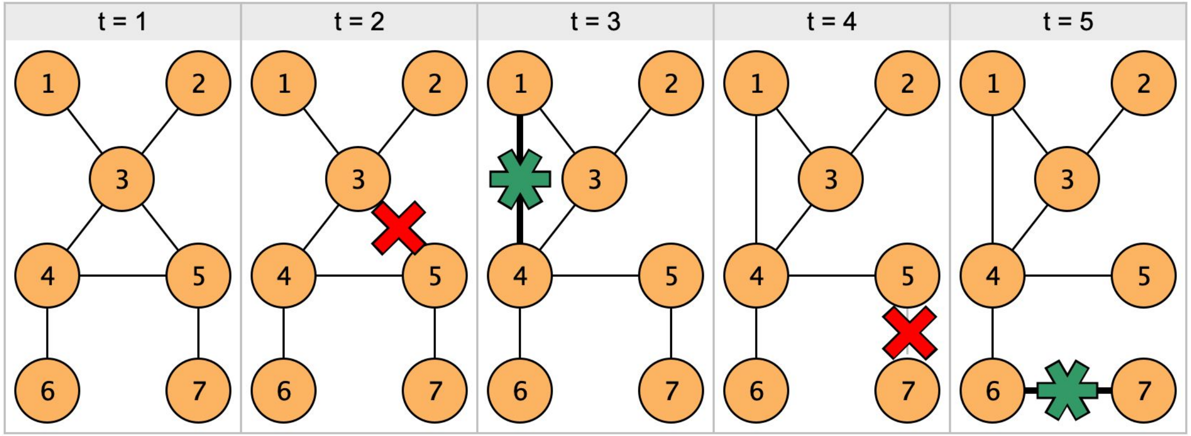
Shortest distance from “Ada” to “Eve”





The shortest path distance changes multiple times during the day.

Path curation with temporal bucketing

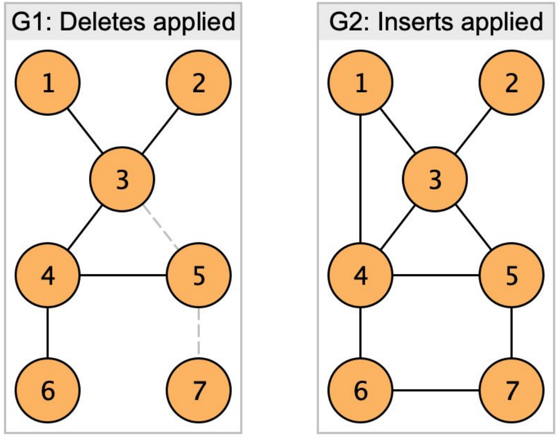


For each day, we construct:

G1 – deletes but no inserts, setting an *upper* bound

G2 – inserts but no deletes, setting a *lower* bound

lower ≤ actual length ≤ upper



Pairs of nodes yielding 3-hop paths in G1 and G2:

- ~~1 to 5~~
- ~~1 to 6~~
- ~~2 to 5~~
- 2 to 6

Is path curation sufficient?

Not yet:

- We also have to consider the degree distribution of the source–target nodes.

Is path curation sufficient?

Not yet:

- We also have to consider the degree distribution of the source–target nodes.

Actually:

- For "perfect" parameter curation, we would need to run the entire workload with many parameter candidates and only keep ones which showed a similar behaviour

Is path curation sufficient?

Not yet:

- We also have to consider the degree distribution of the source–target nodes.

Actually:

- For “perfect” parameter curation, we would need to run the entire workload with many parameter candidates and only keep ones which showed a similar behaviour

The real question:

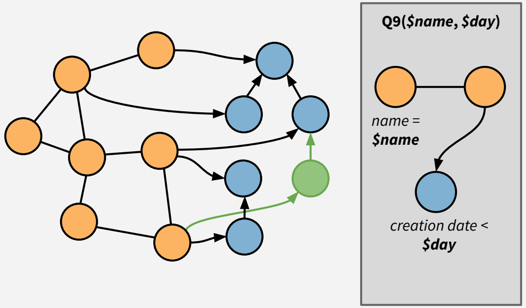
- Is it worth spending more effort on optimizing the parameter curation?

I'm leaving academia

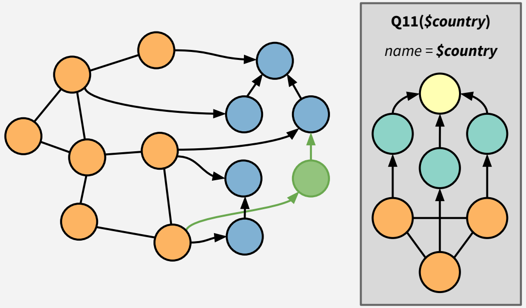
- Moving to DuckDB Labs (CWI spin-off in Amsterdam)
- Staying involved with LDBC at ~1 day / month

CWI Database Architectures → DuckDB Labs

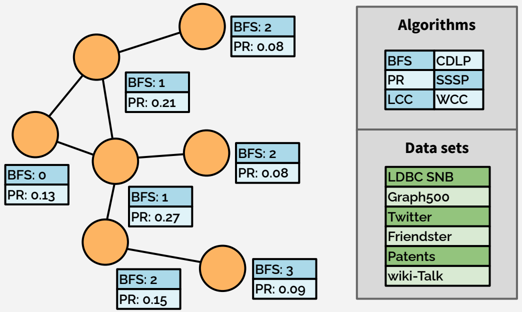
SNB Interactive



SNB Business Intelligence



Graphalytics



Semantic Publishing Benchmark

Target: **RDF/SPARQL**

Domain: **Media/publishing industry**

Inferencing & continuous updates

Financial Benchmark

Target: **Distributed systems**

Domain: **Financial fraud detection**

Strict latency bound (20 ms)

Future benchmark ideas

GNNs

Graph mining

Graph streaming

LDBC

The graph & RDF
benchmark reference