Graph databases:
Where theory meets pactice

Gabor Szarnyas

Simonyi Conference 2025

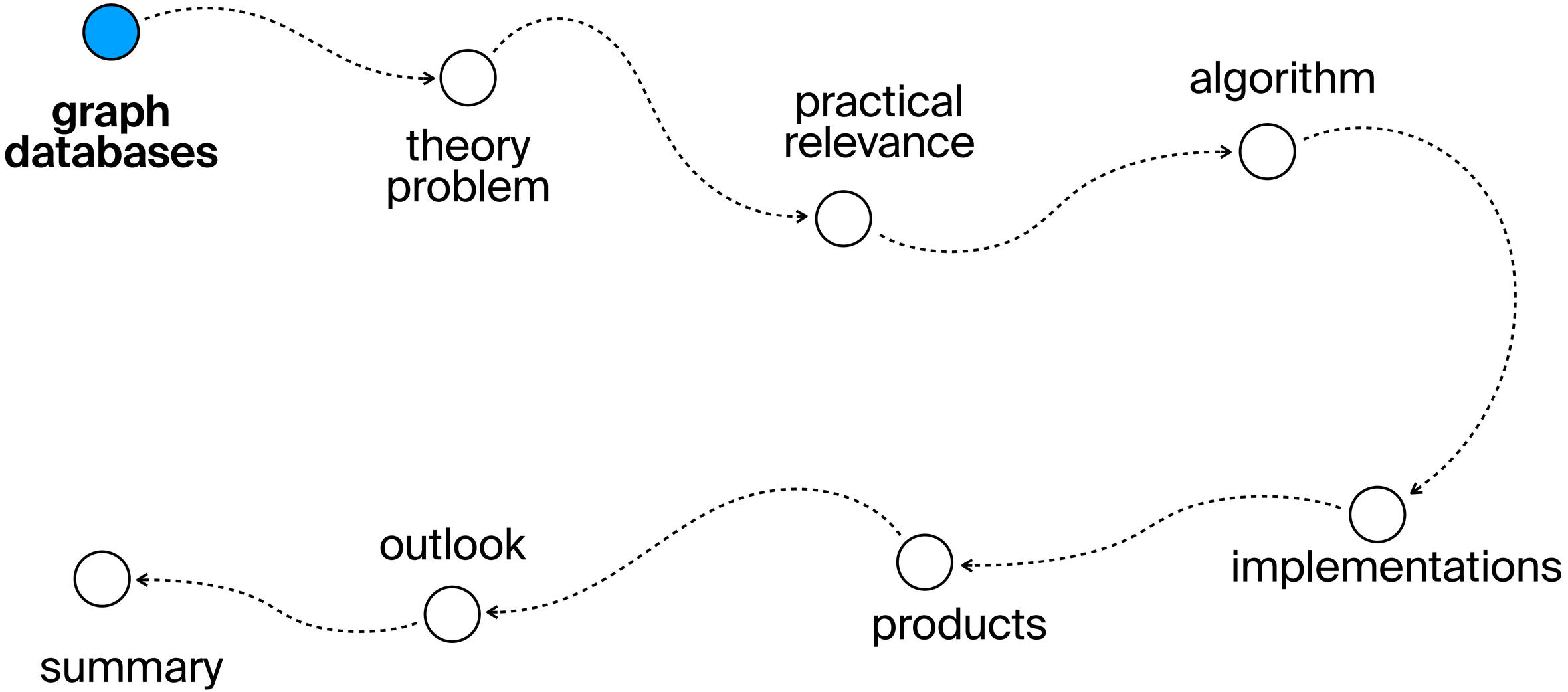
My career in databases

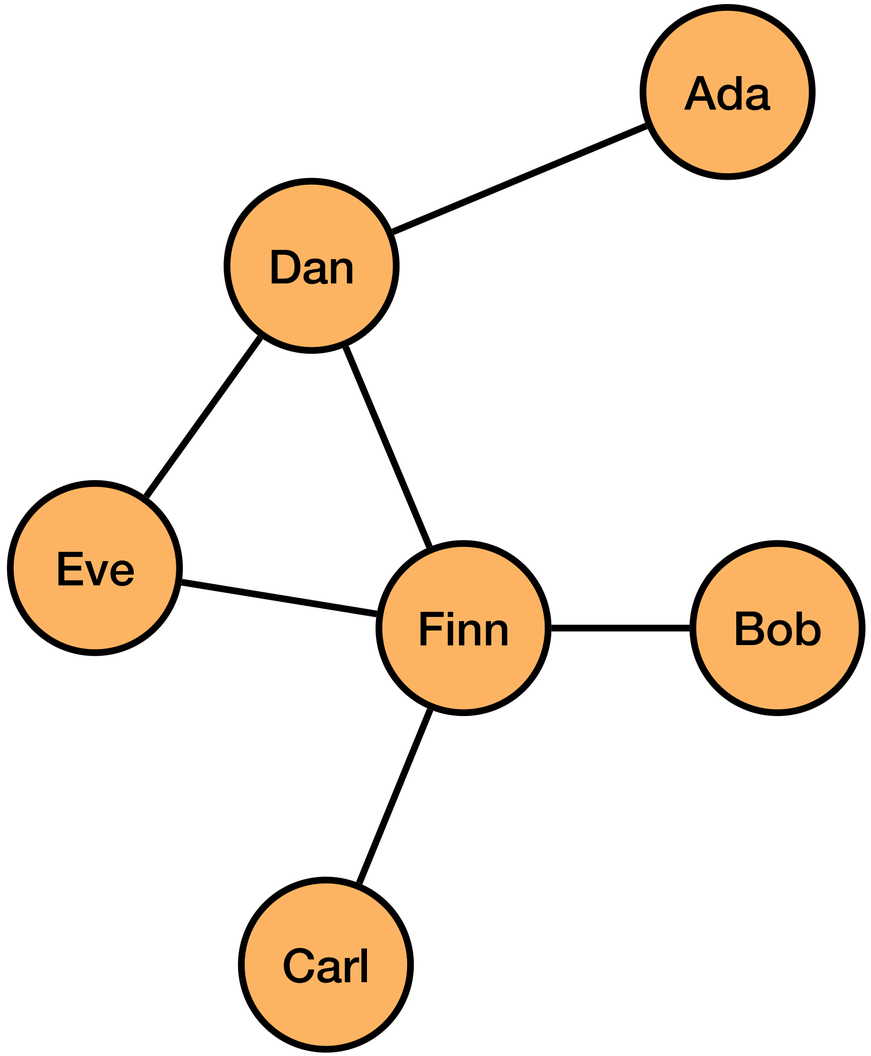
ftsrg

CWI

DuckDB Labs

2010 2013 2020 2023


graph databases
theory problem
practical relevance
algorithm
implementations
products
outlook
summary

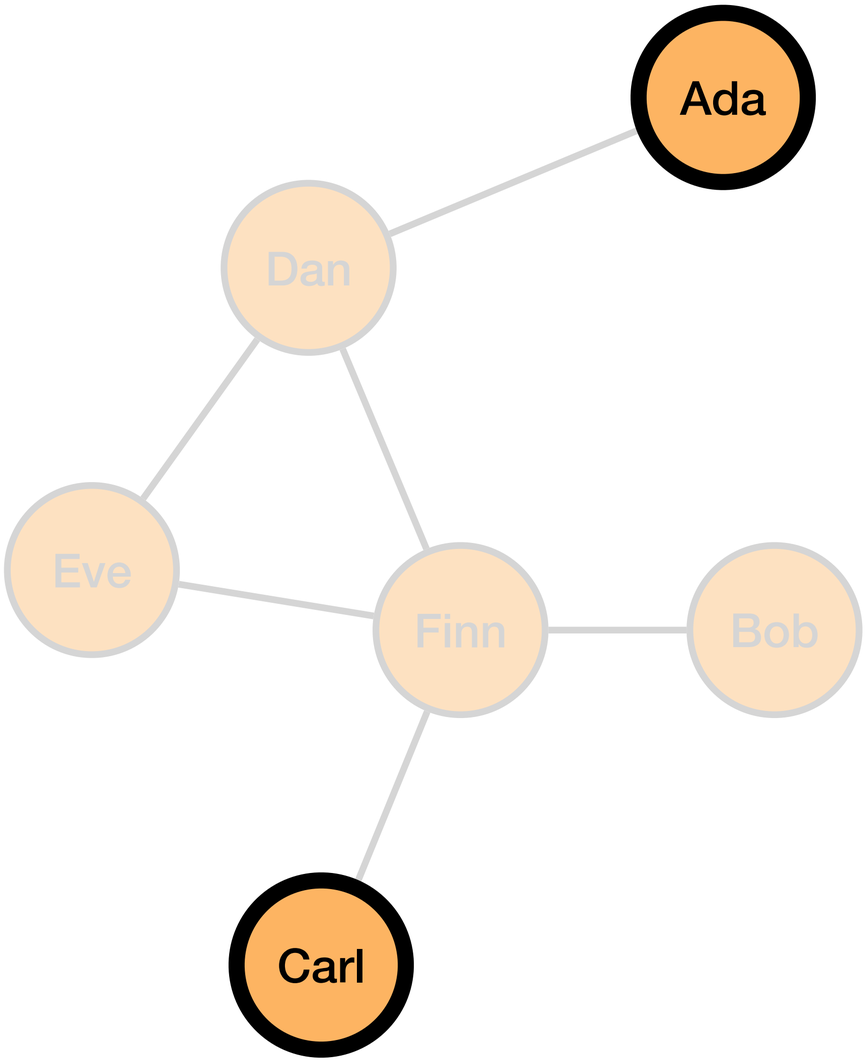


graph

$$G = (V, E)$$

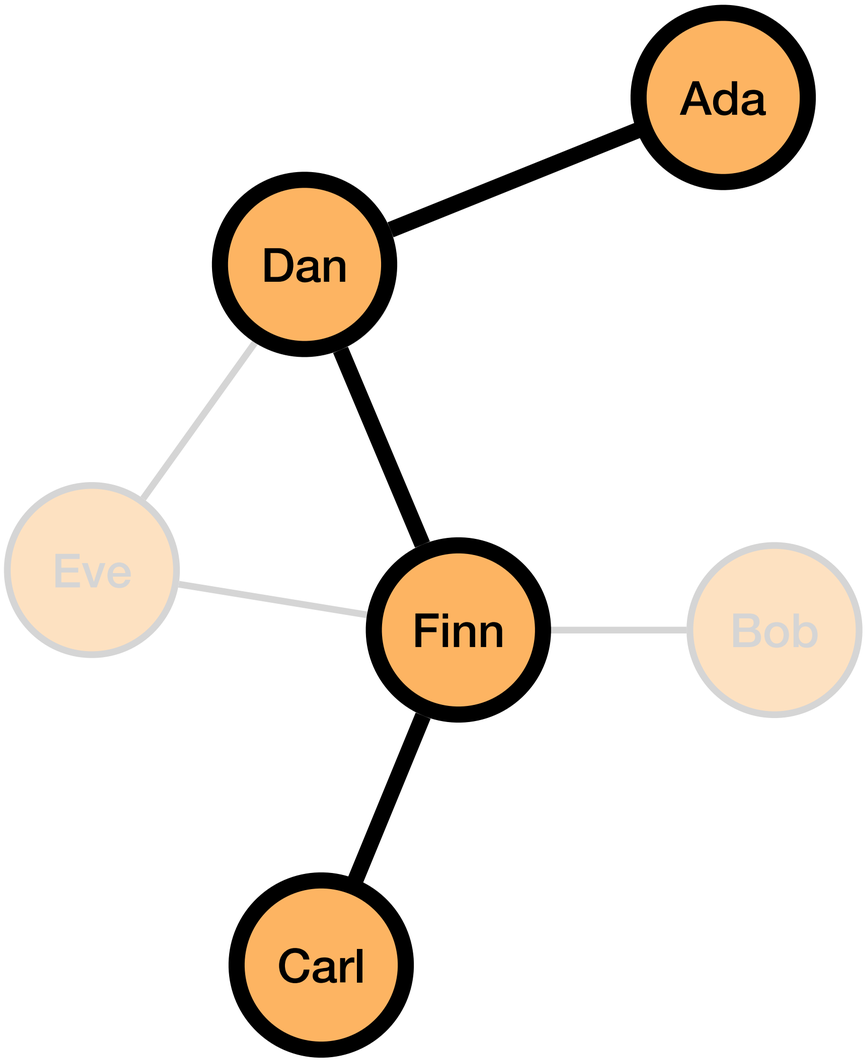
path finding

q: shortest path between two nodes
BFS (breadth-first search) Dijkstra's algorithm



path finding

q: shortest path between two nodes
BFS (breadth-first search) Dijkstra's algorithm

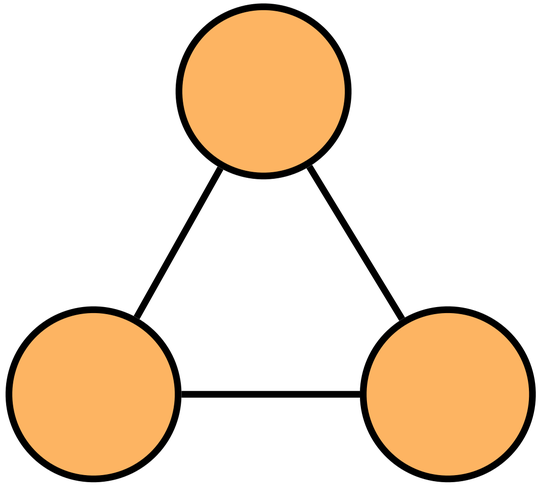


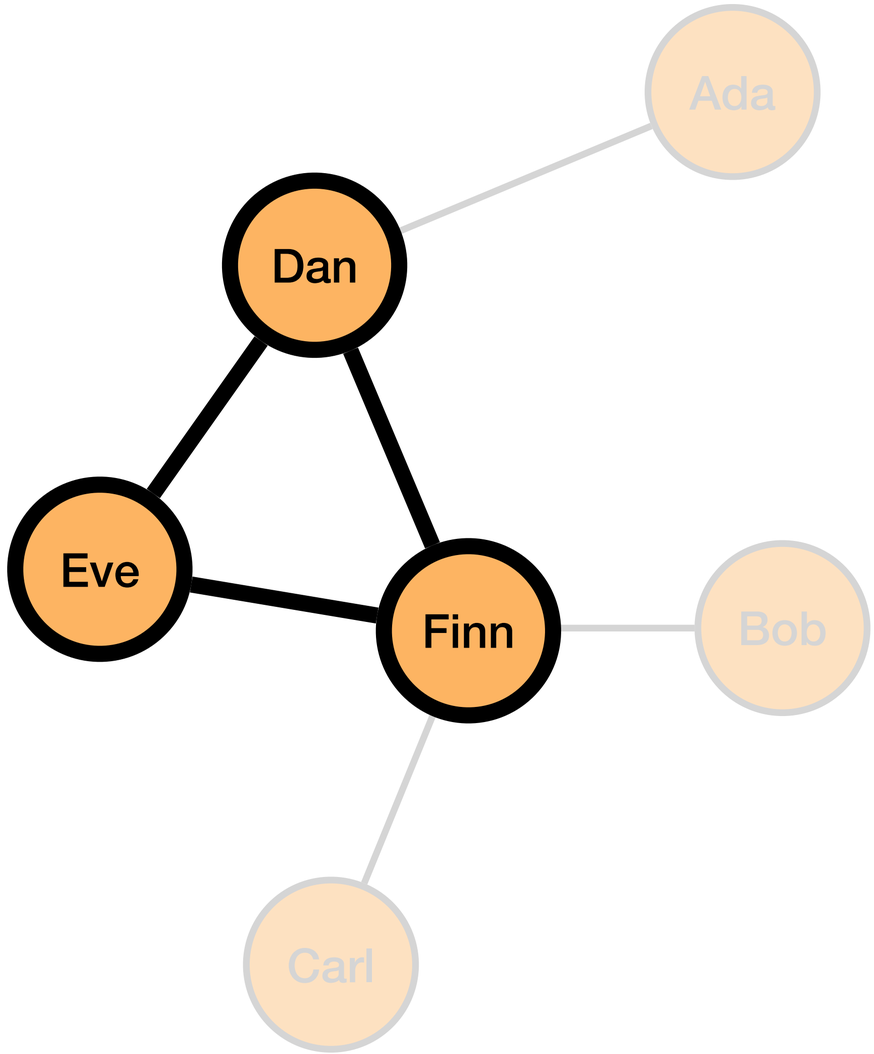
path finding

q: shortest path between two nodes
BFS (breadth-first search) Dijkstra's algorithm

graph pattern matching

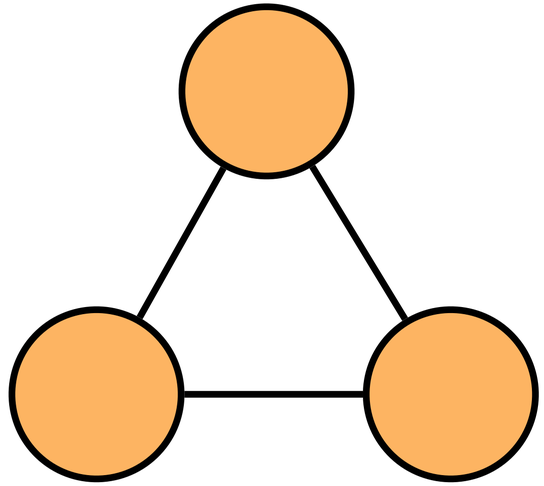
q: triangle-shaped subgraphs





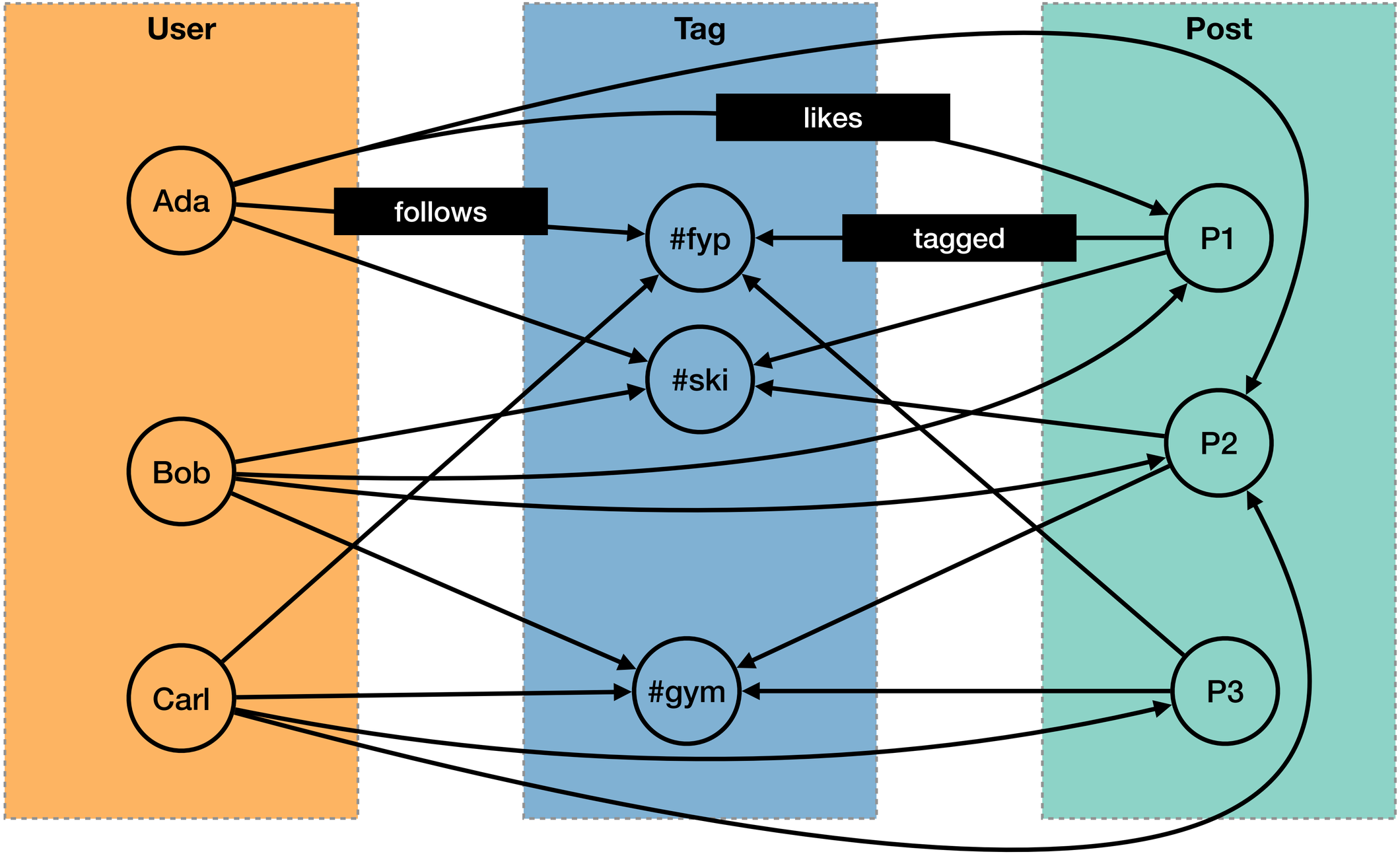
graph pattern matching

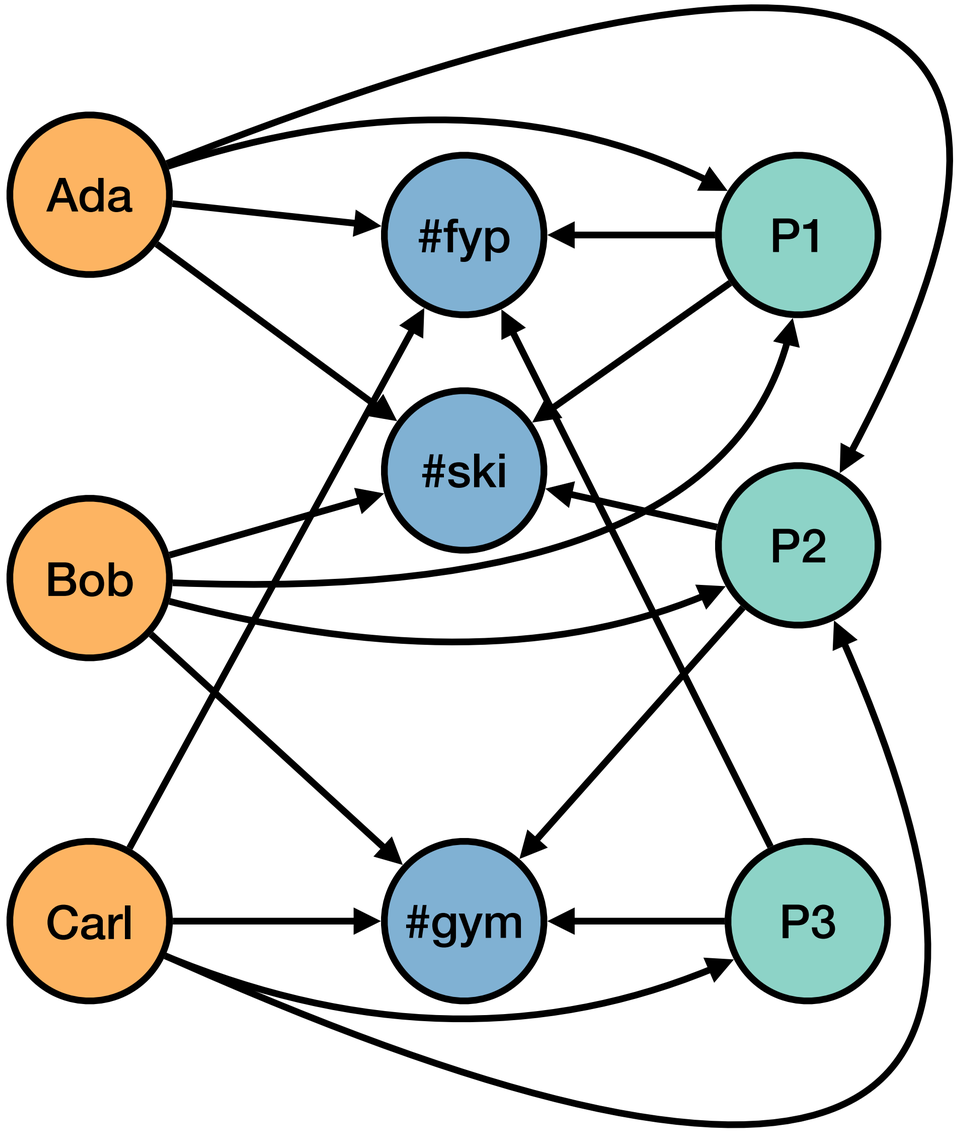
q: triangle-shaped subgraphs



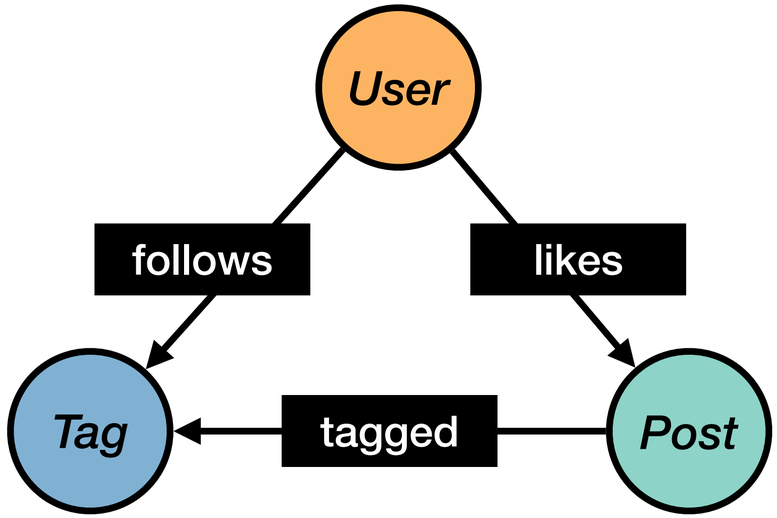
Social network

Social network

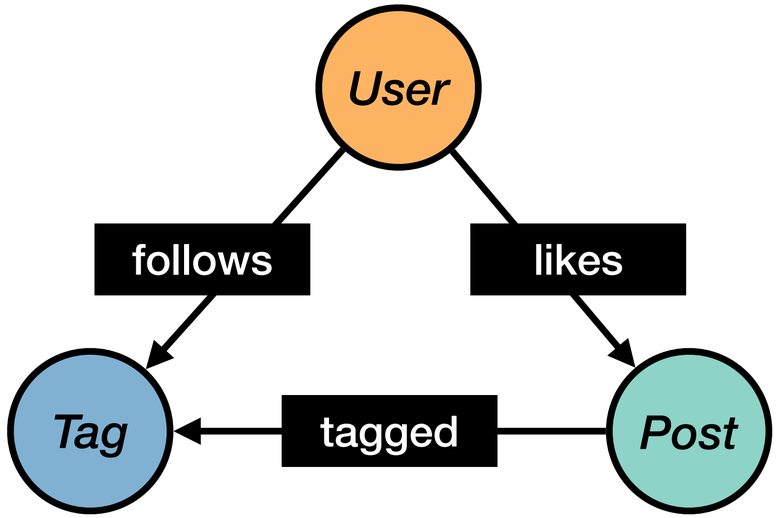

User
Tag
Post
likes
Ada
follows
#fyp
tagged
P1
#ski
Bob
P2
Carl
#gym
P3

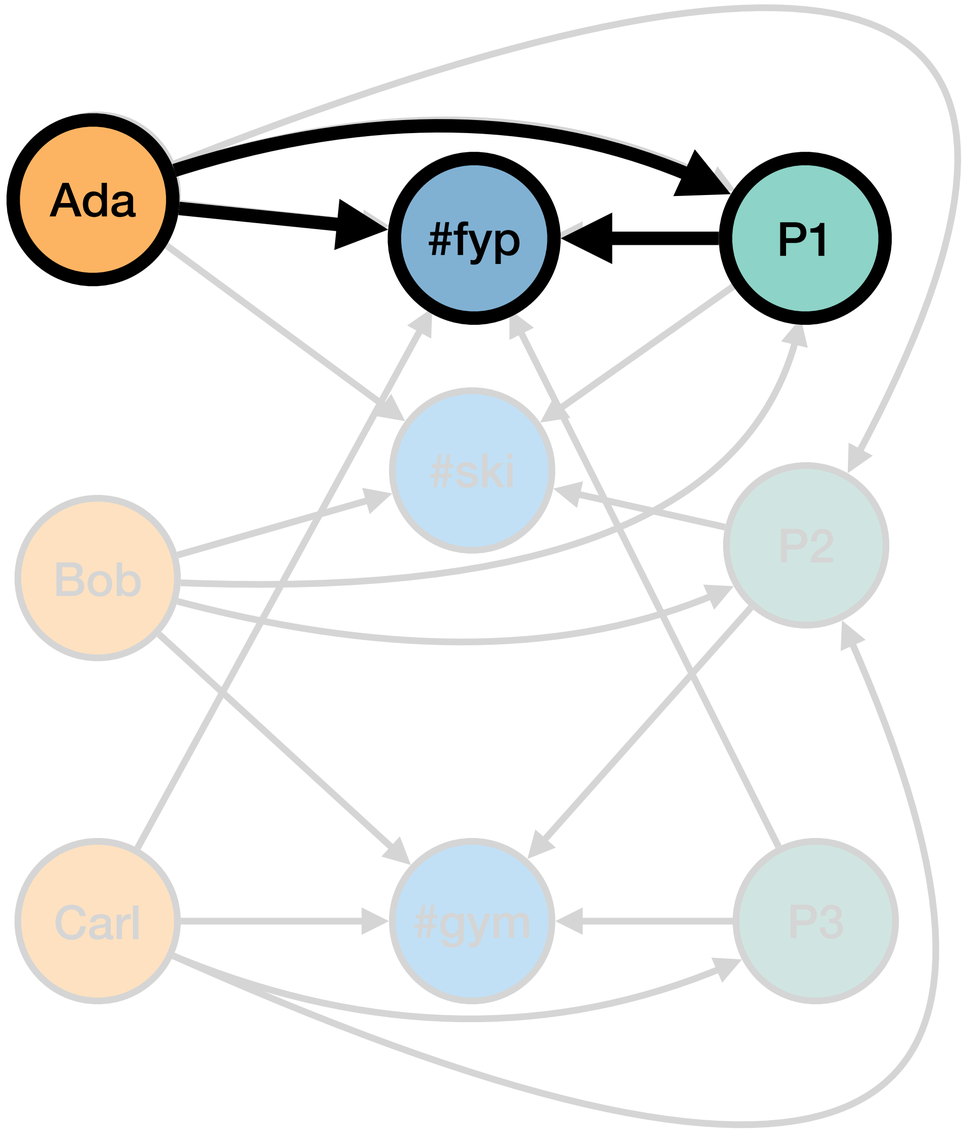


q: User-Tag-Post triangles

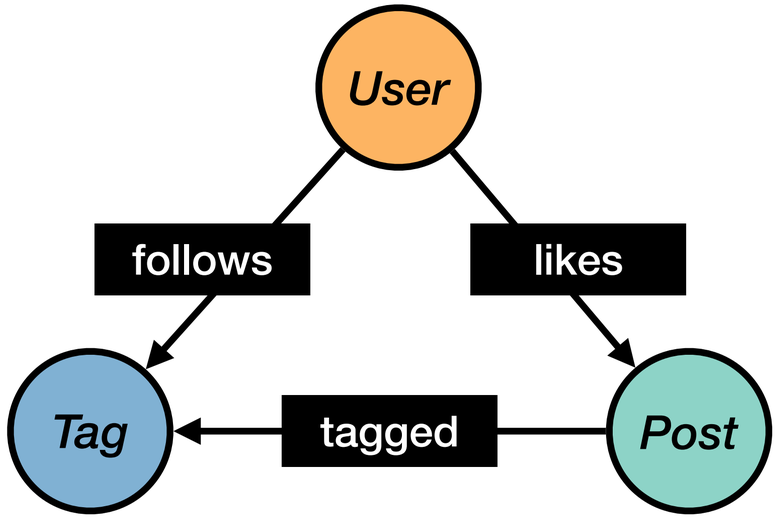


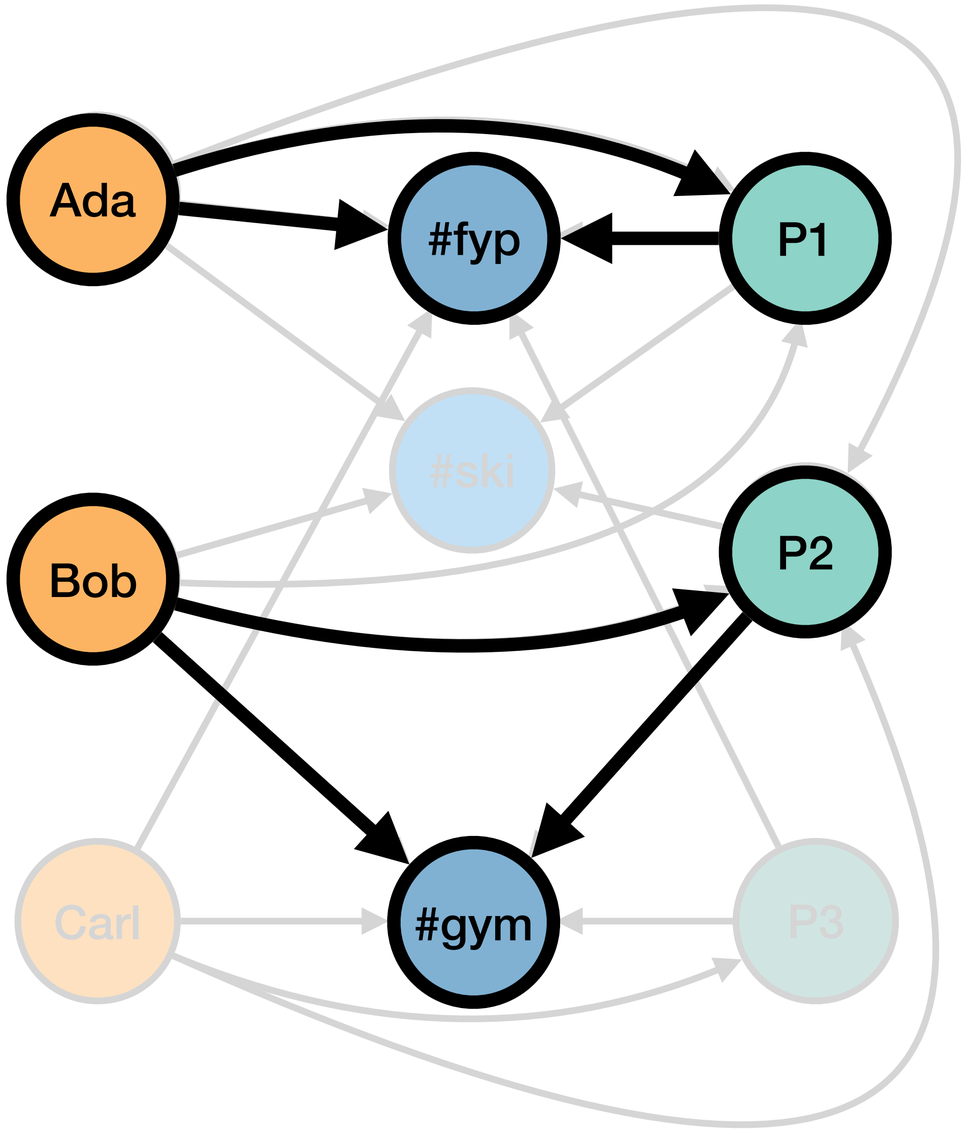
q: User-Tag-Post triangles



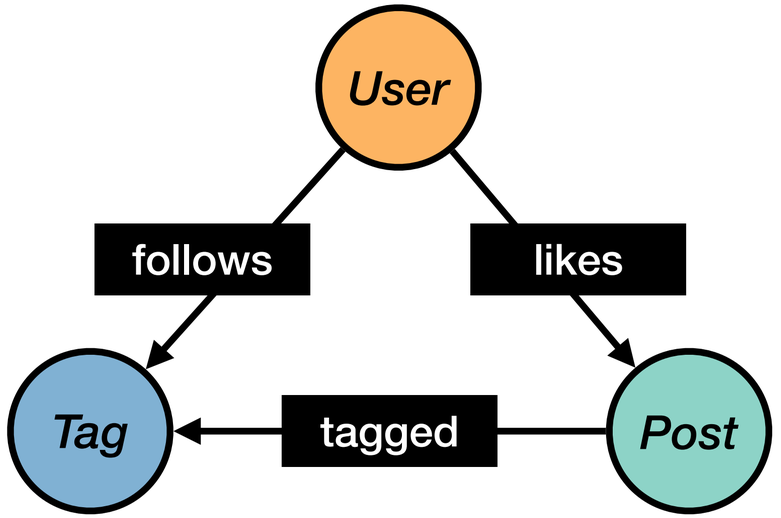


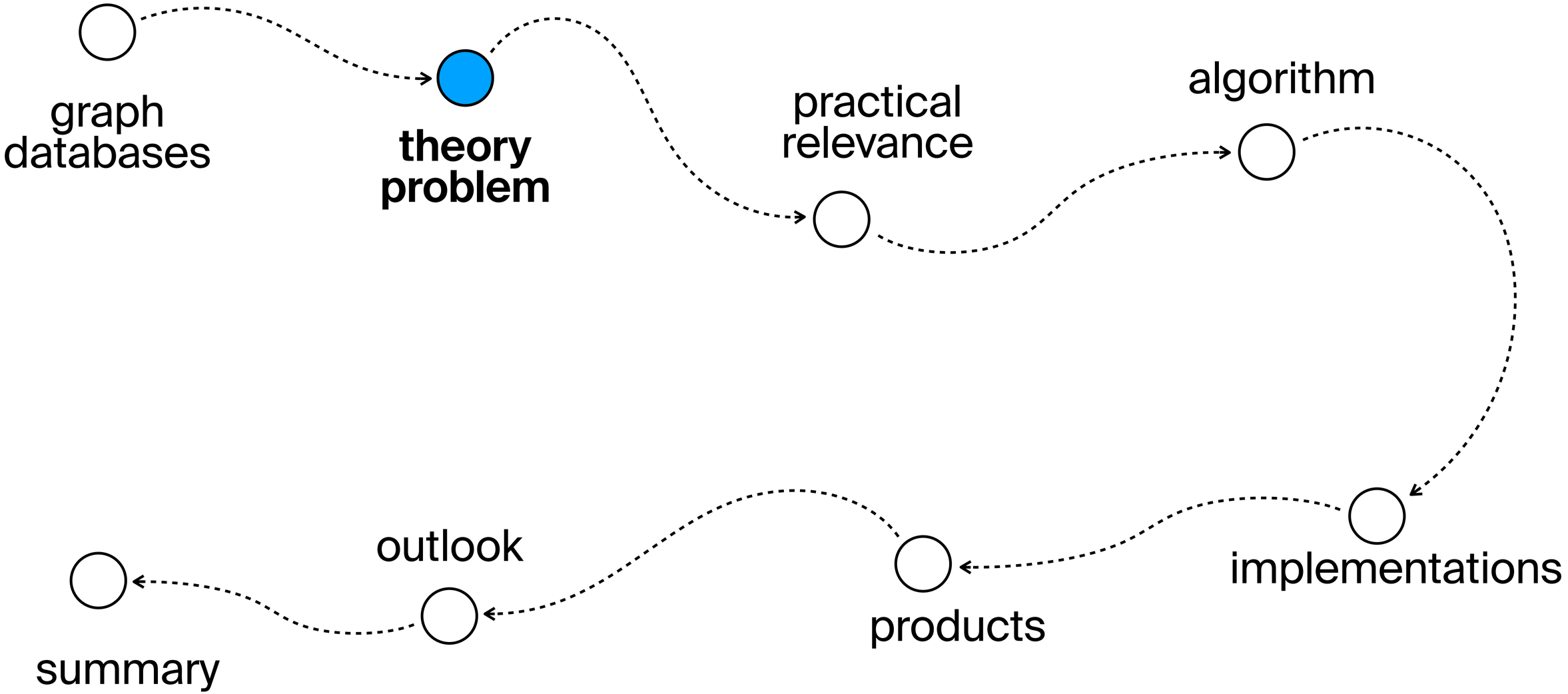
q: User-Tag-Post triangles

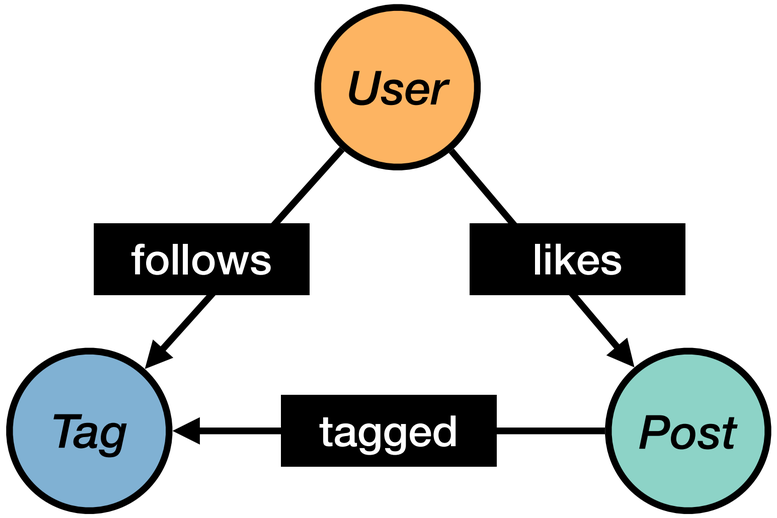




q: User-Tag-Post triangles




graph databases
theory problem
practical relevance
algorithm
implementations
products
outlook
summary



⋈: join

```
SELECT *
FROM follows
NATURAL JOIN tagged
NATURAL JOIN likes;
```

follows

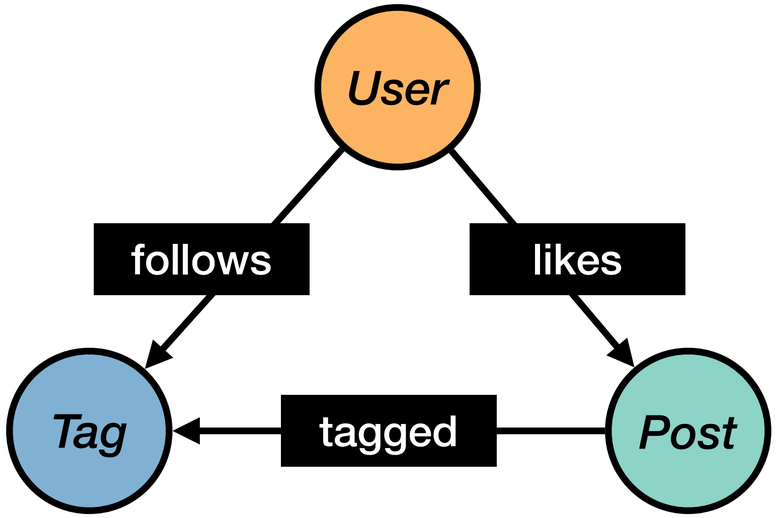
User	Tag
Ada	#fyp
Ada	#ski
Bob	#ski
Bob	#gym
Carl	#fyp
Carl	#gym

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3



⋈: join

```
SELECT *
FROM follows
NATURAL JOIN tagged
NATURAL JOIN likes;
```

follows

User	Tag
Ada	#fyp
Ada	#ski
Bob	#ski
Bob	#gym
Carl	#fyp
Carl	#gym

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

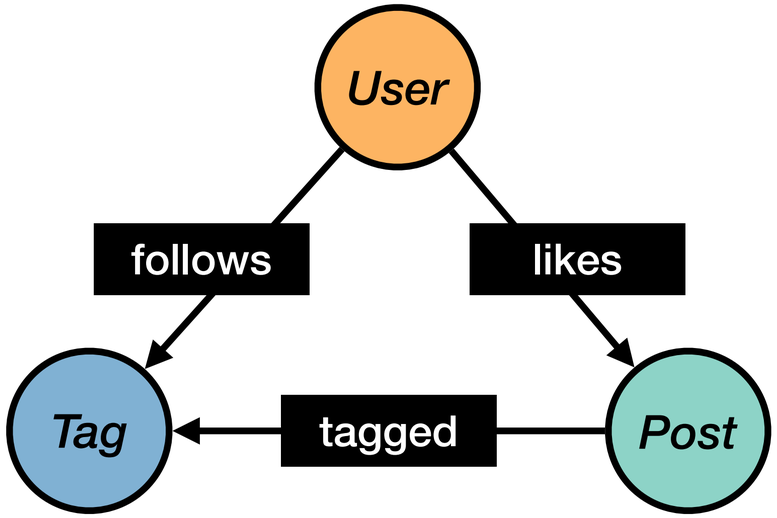
likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

follows ⋈ tagged ⋈ likes

User	Tag	Post
Ada	#ski	P1
Ada	#ski	P2
Ada	#fyp	P1
Bob	#ski	P1
Bob	#ski	P2
Bob	#gym	P2
Carl	#fyp	P3
Carl	#gym	P2
Carl	#gym	P3

9▲



⋈: join

```
SELECT *
FROM follows
NATURAL JOIN tagged
NATURAL JOIN likes;
```

follows

User	Tag
Ada	#fyp
Ada	#ski
Bob	#ski
Bob	#gym
Carl	#fyp
Carl	#gym

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

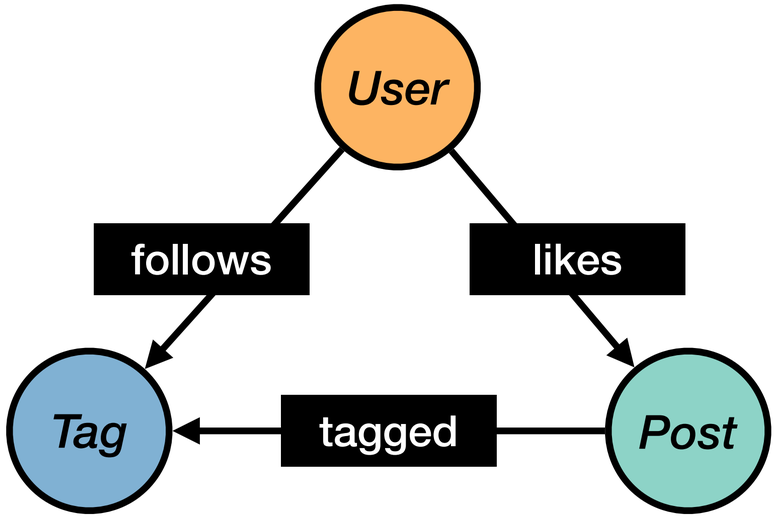
likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

follows ⋈ tagged

User	Tag	Post
Ada	#ski	P1
Ada	#ski	P2
Ada	#fyp	P1
Ada	#fyp	P3
Bob	#ski	P1
Bob	#ski	P2
Bob	#gym	P2
Bob	#gym	P3
Carl	#fyp	P1
Carl	#fyp	P3
Carl	#gym	P2
Carl	#gym	P3

12 rows



⋈: join

```
SELECT *
FROM follows
NATURAL JOIN tagged
NATURAL JOIN likes;
```

follows

User	Tag
Ada	#fyp
Ada	#ski
Bob	#ski
Bob	#gym
Carl	#fyp
Carl	#gym

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

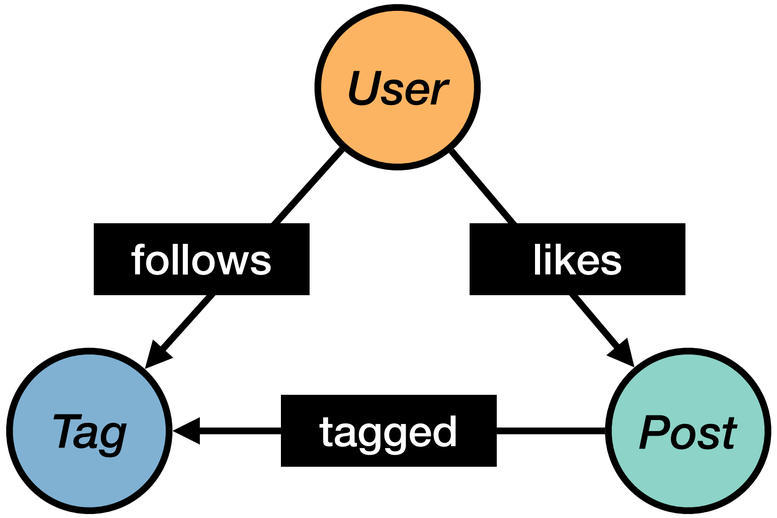
likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

tagged ⋈ likes

Post	Tag	User
P1	#ski	Ada
P1	#ski	Bob
P1	#fyp	Ada
P1	#fyp	Bob
P2	#ski	Ada
P2	#ski	Bob
P2	#ski	Carl
P2	#gym	Ada
P2	#gym	Bob
P2	#gym	Carl
P3	#fyp	Carl
P3	#gym	Carl

12 rows



⋈: join

```
SELECT *
FROM follows
NATURAL JOIN tagged
NATURAL JOIN likes;
```

follows

User	Tag
Ada	#fyp
Ada	#ski
Bob	#ski
Bob	#gym
Carl	#fyp
Carl	#gym

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

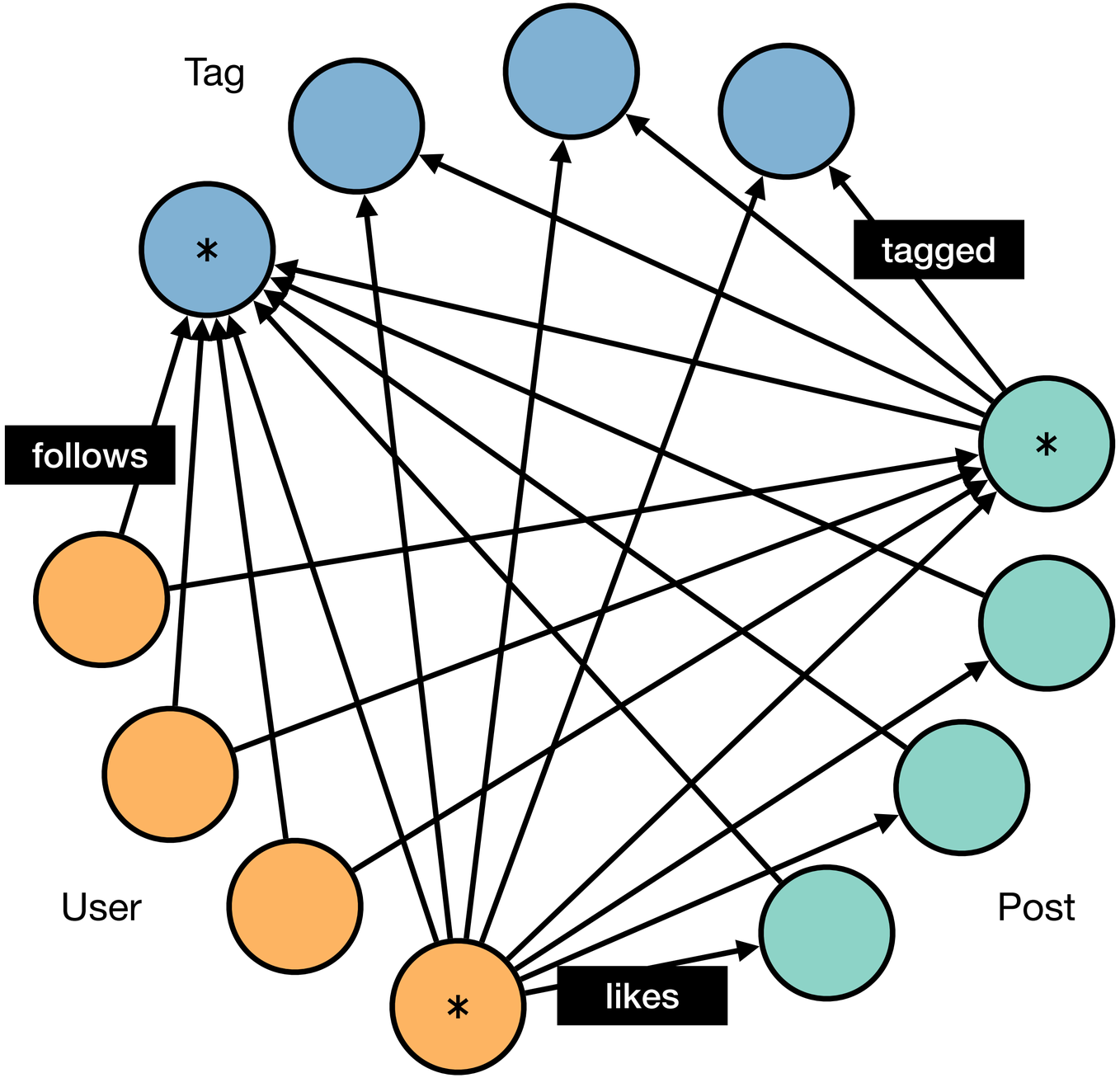
follows ⋈ likes

User	Tag	Post
Ada	#ski	P1
Ada	#ski	P2
Ada	#fyp	P1
Ada	#fyp	P2
Bob	#ski	P1
Bob	#ski	P2
Bob	#gym	P1
Bob	#gym	P2
Carl	#fyp	P2
Carl	#fyp	P3
Carl	#gym	P2
Carl	#gym	P3

12 rows

What's the complexity of the algorithm?

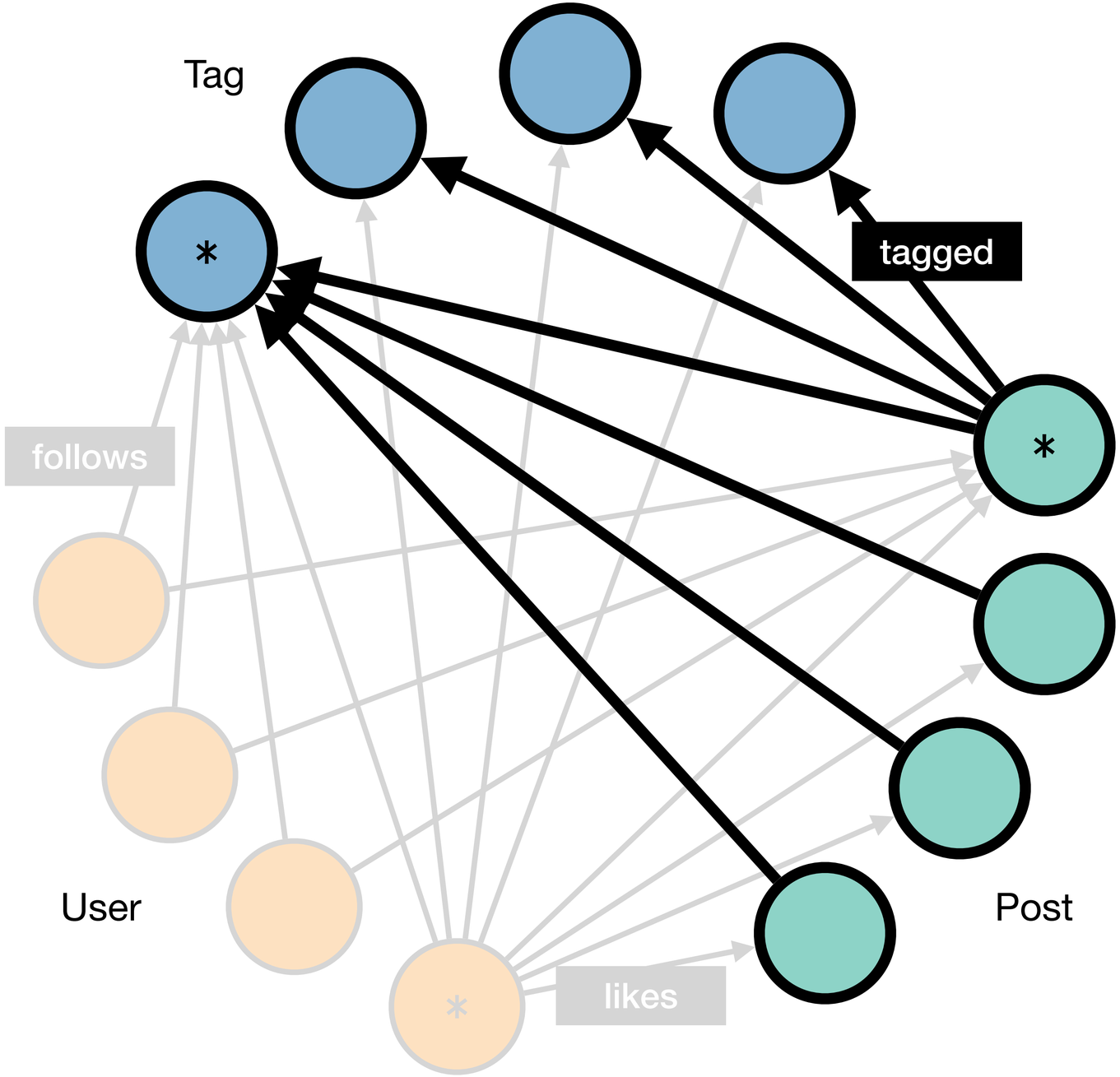
⇒ Let's look at a *worst-case* input

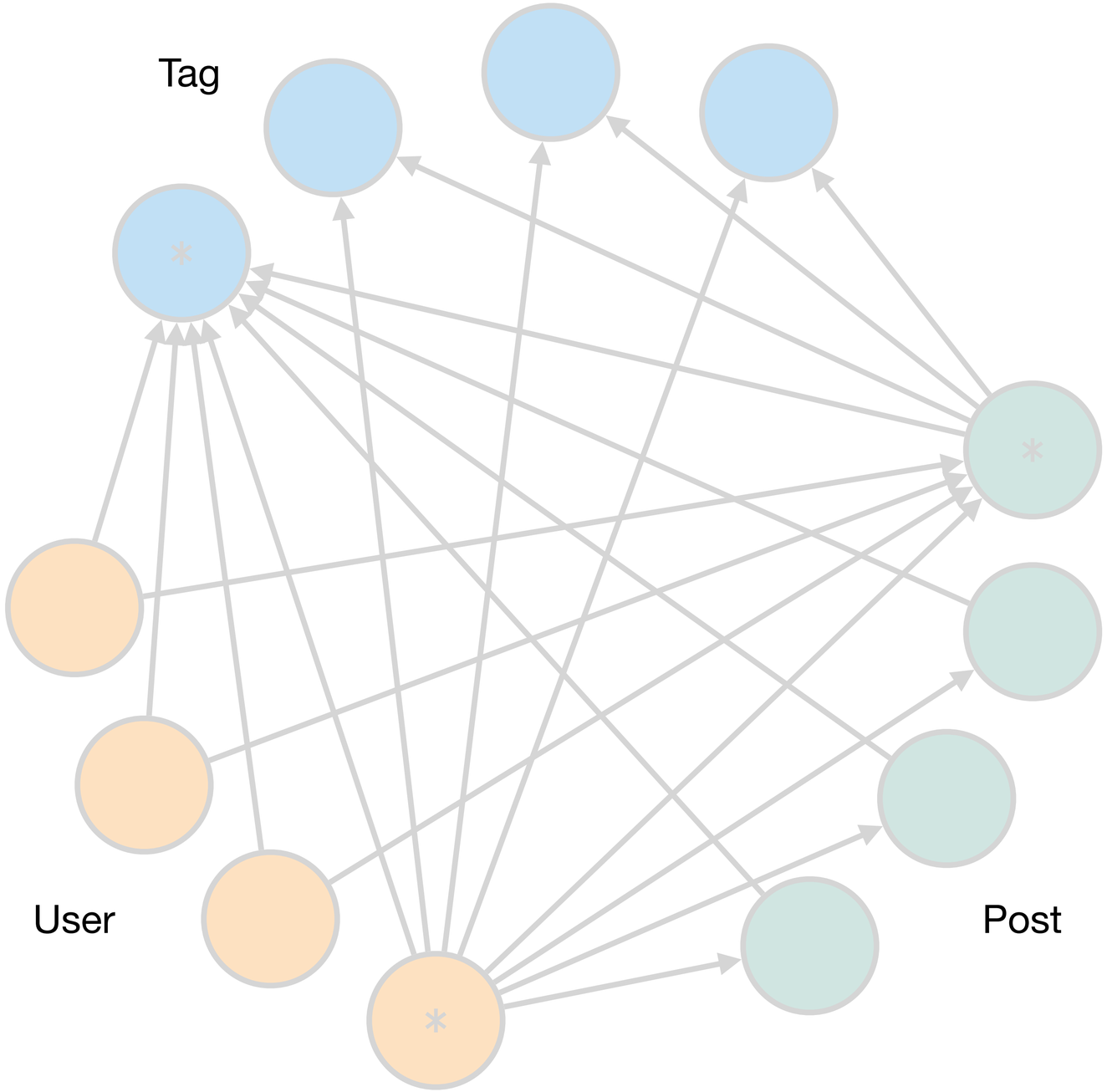


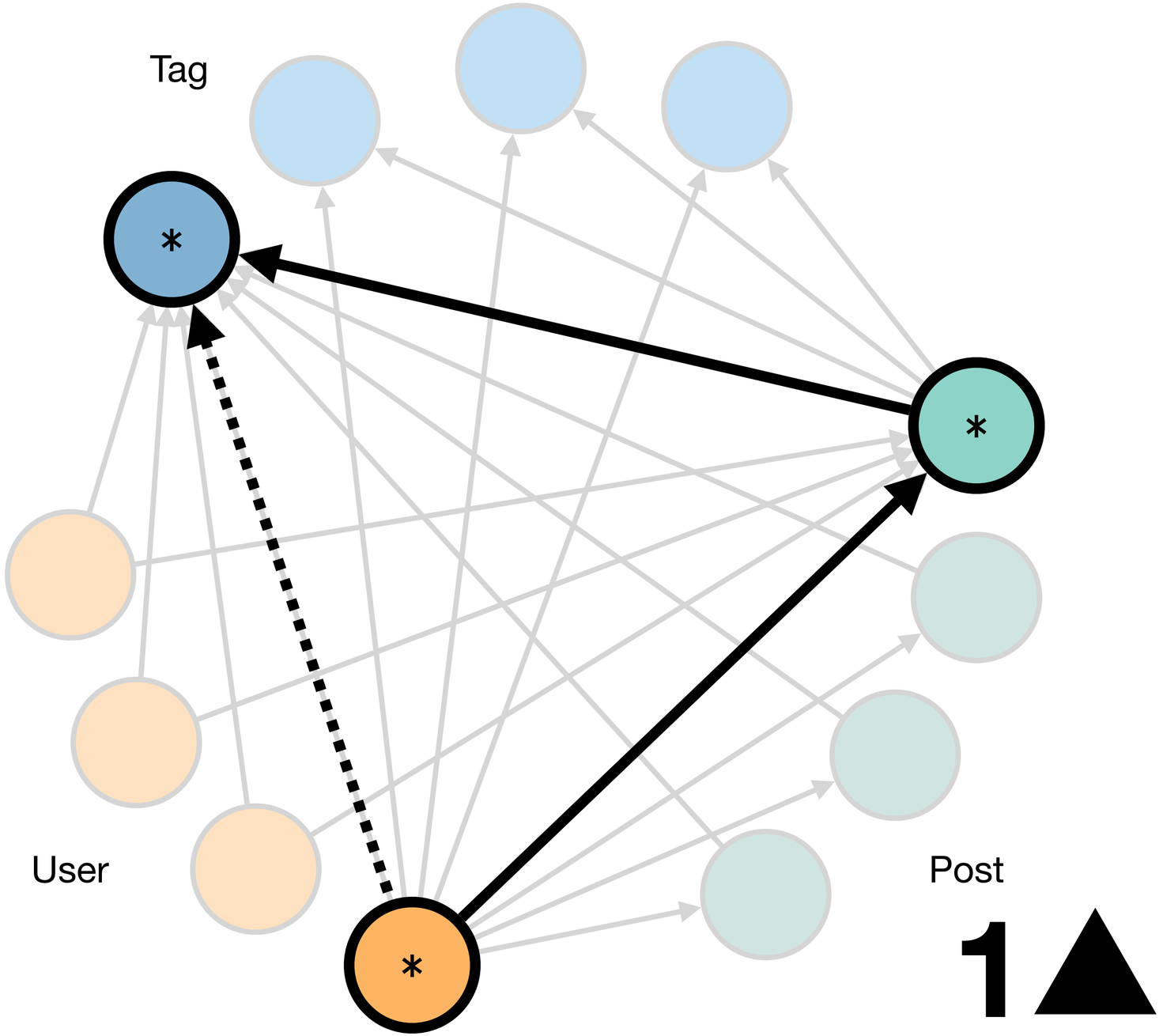
From all three node types, we have 1 distinguished ($*$) and k regular instances.

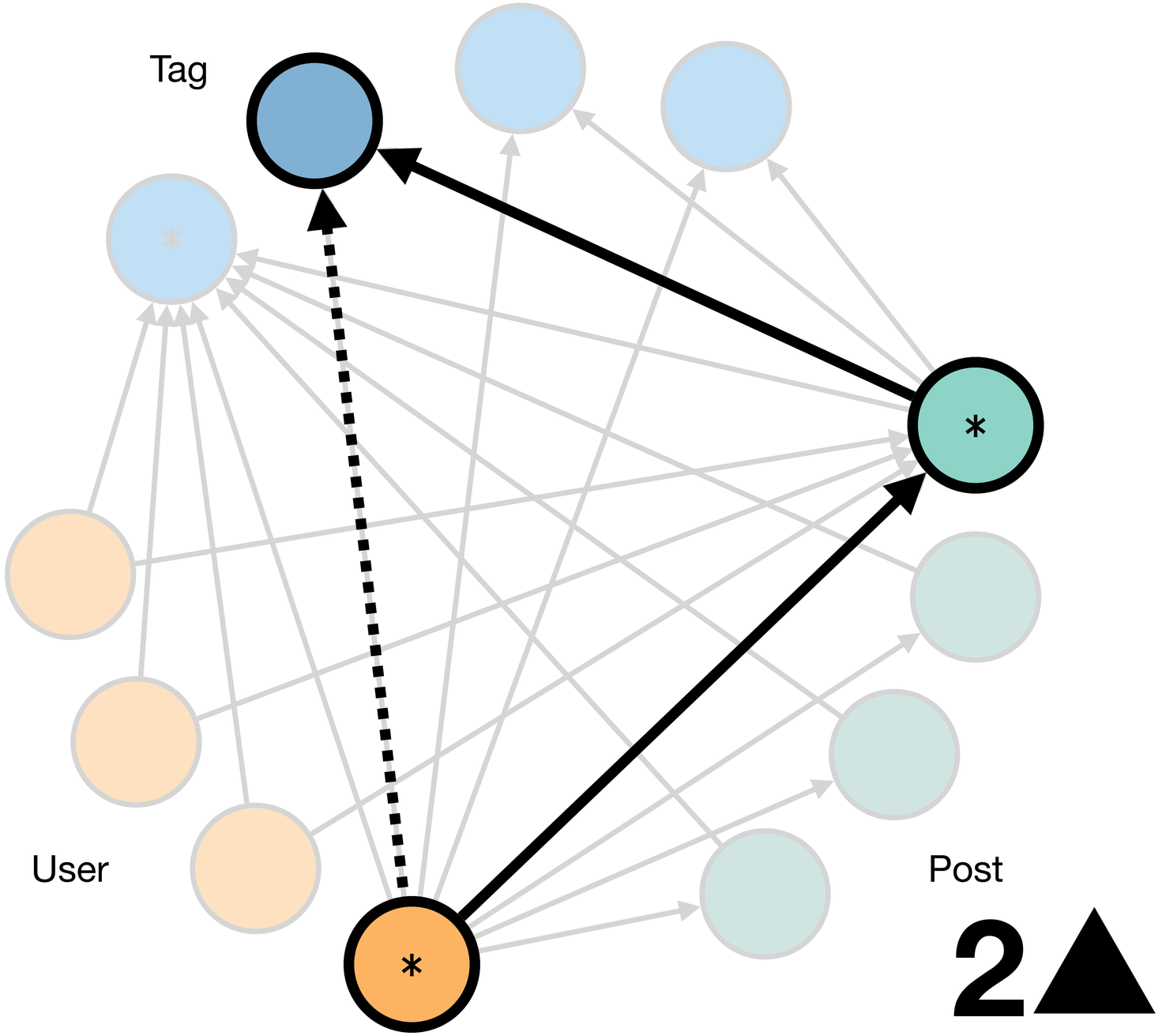
1 distinguished nodes

k regular nodes

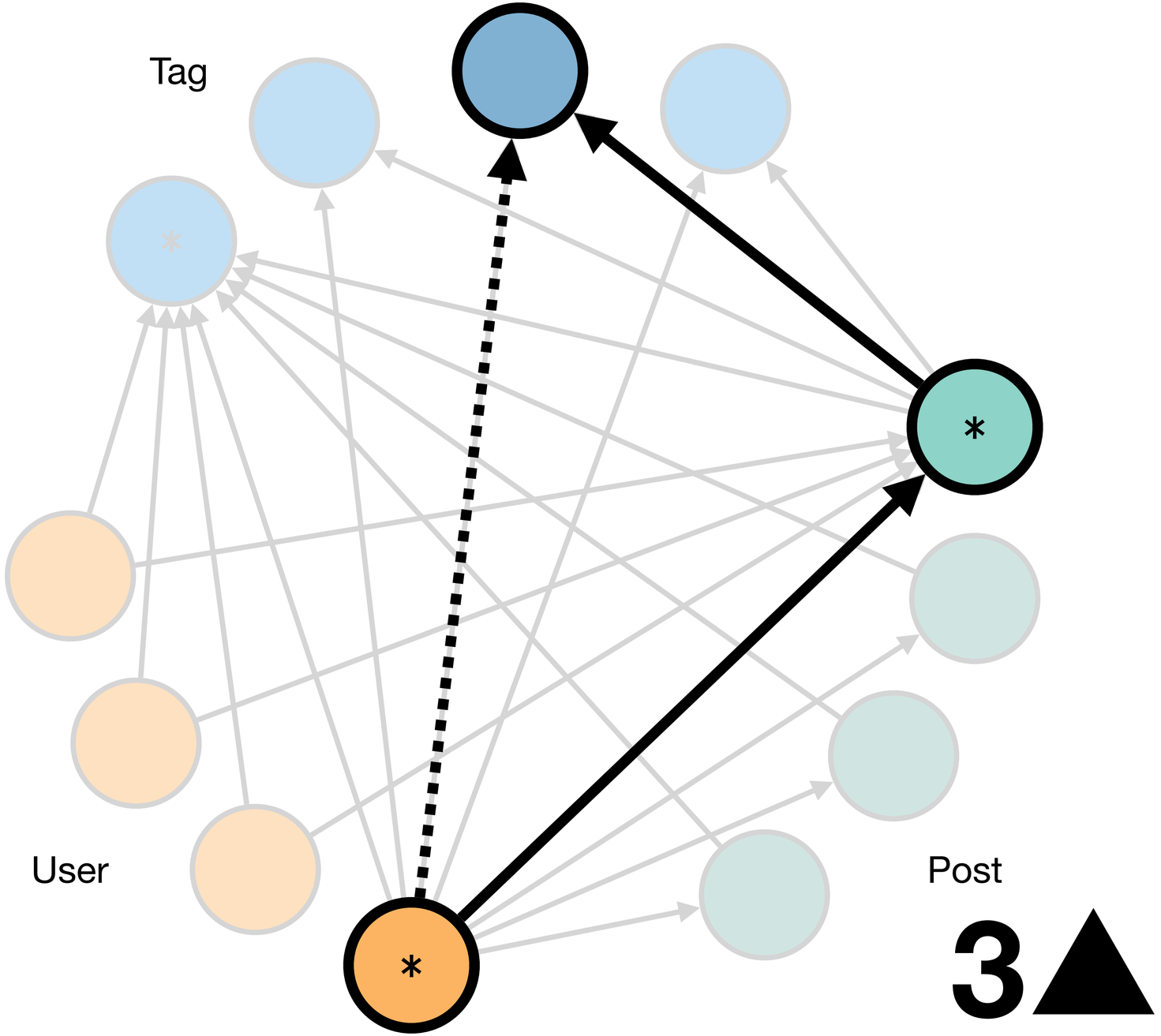

Tag
*
tagged
*
follows
User
*
likes
Post

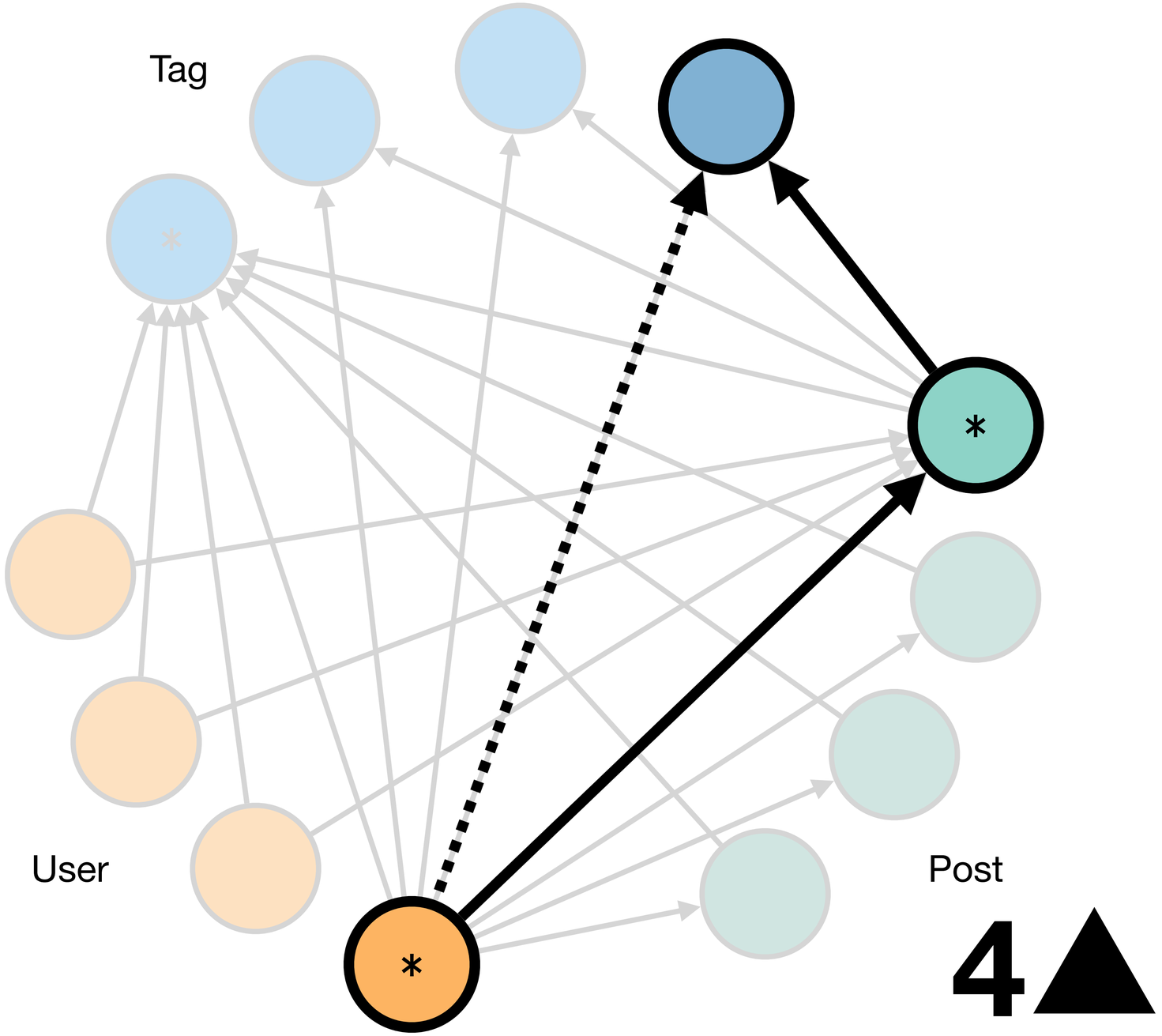

Tag
User
Post


Tag
*
*
User
*
Post
1

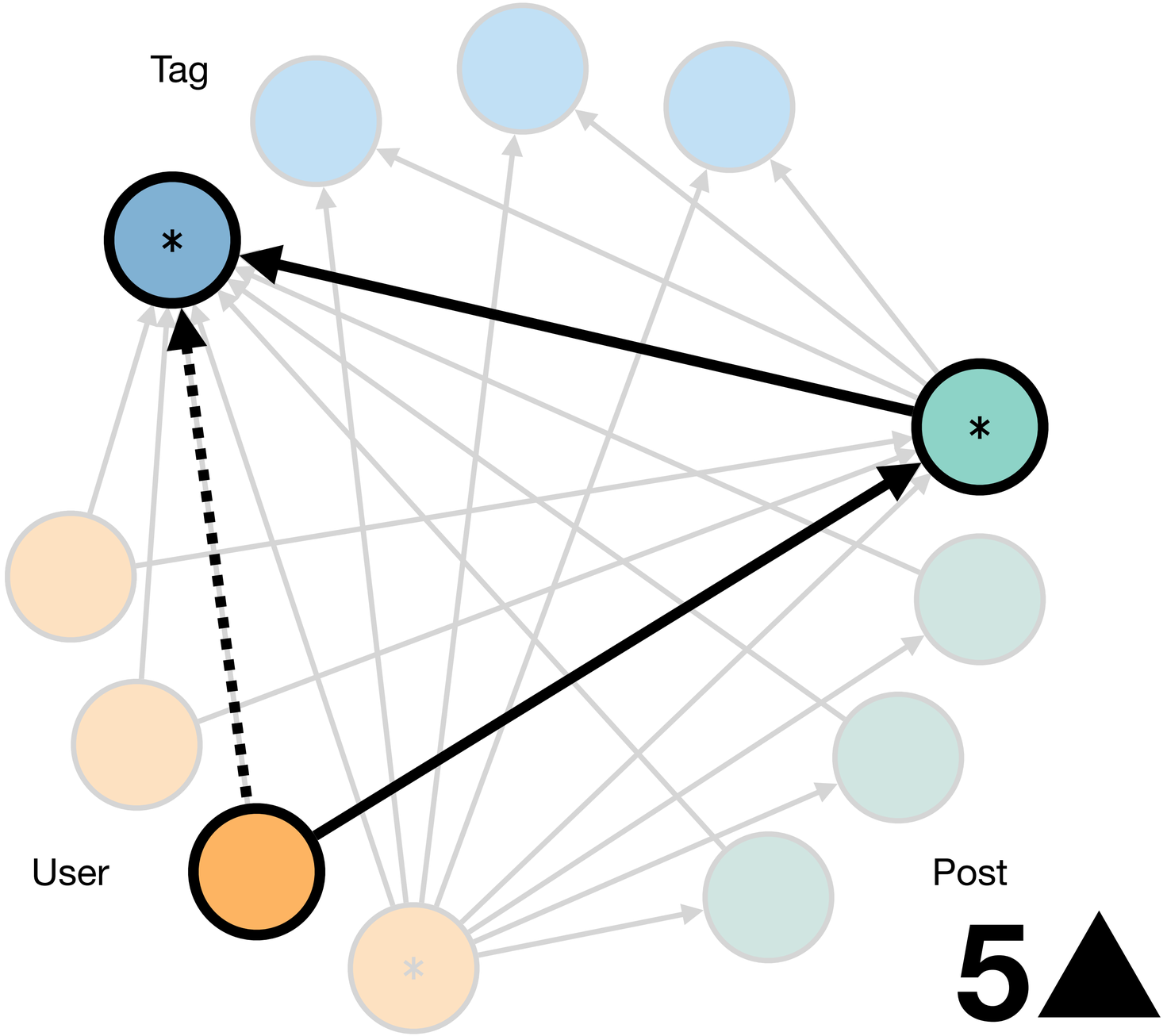


2▲

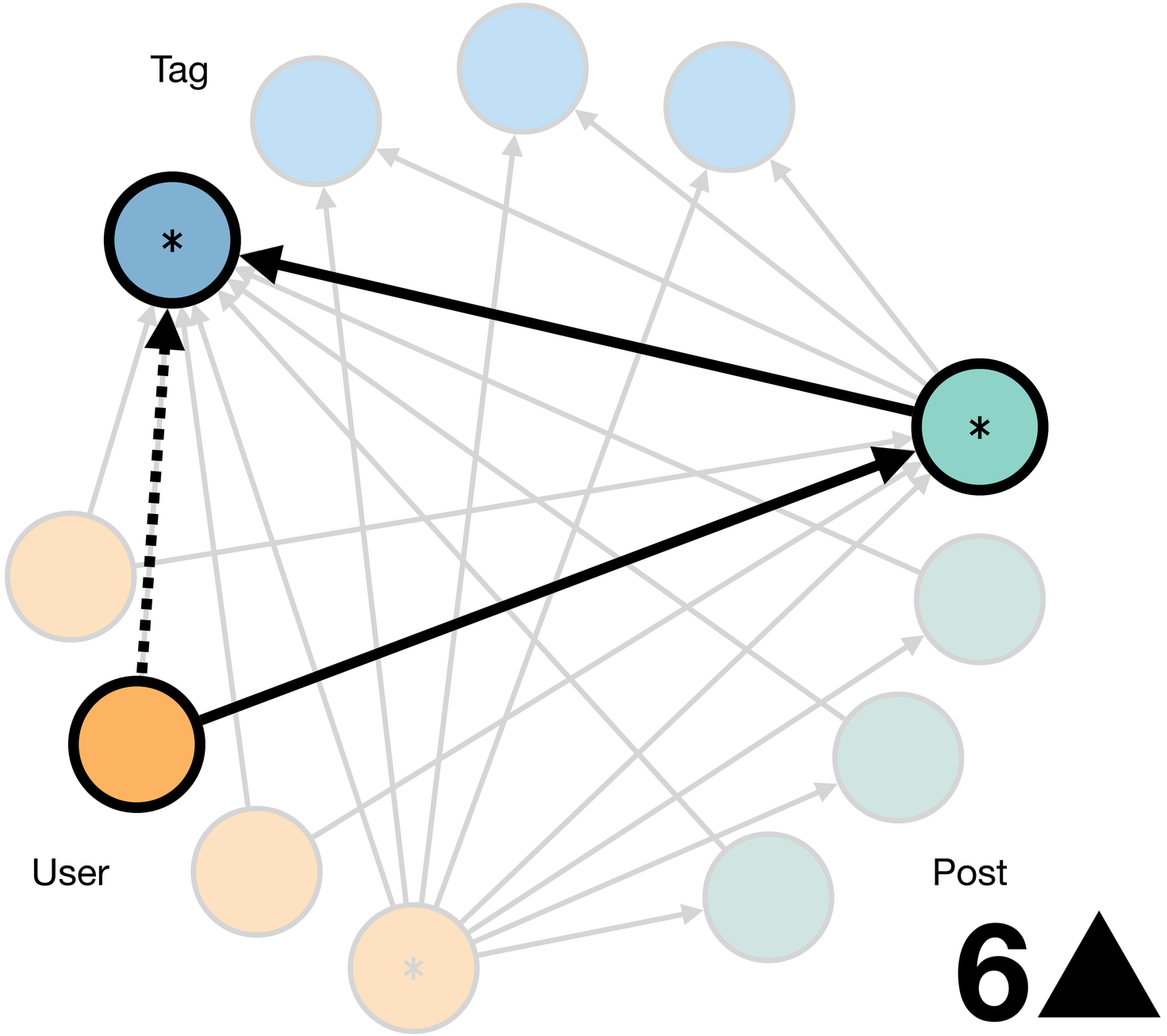

Tag
*
*
User
*
Post
3



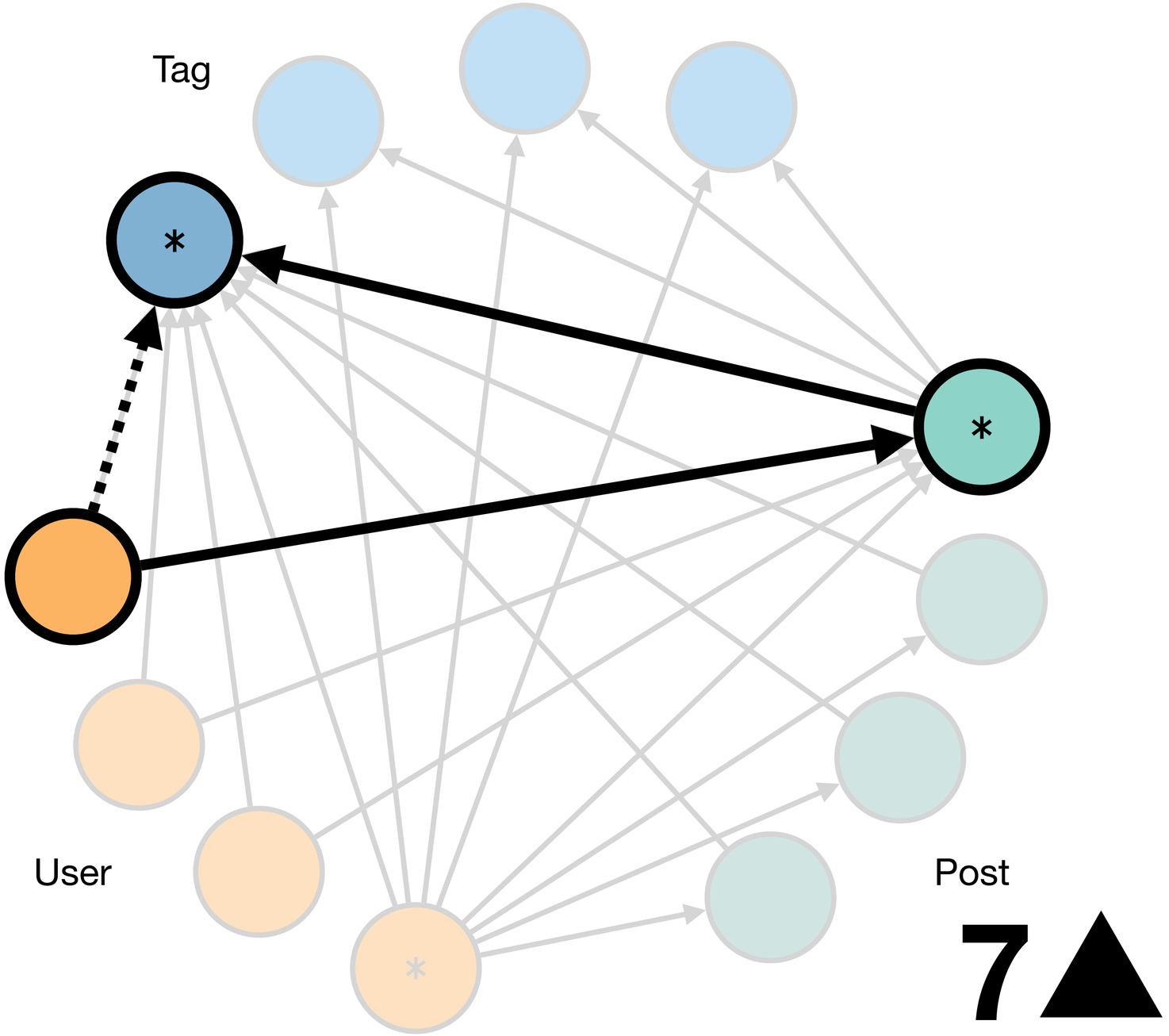
4



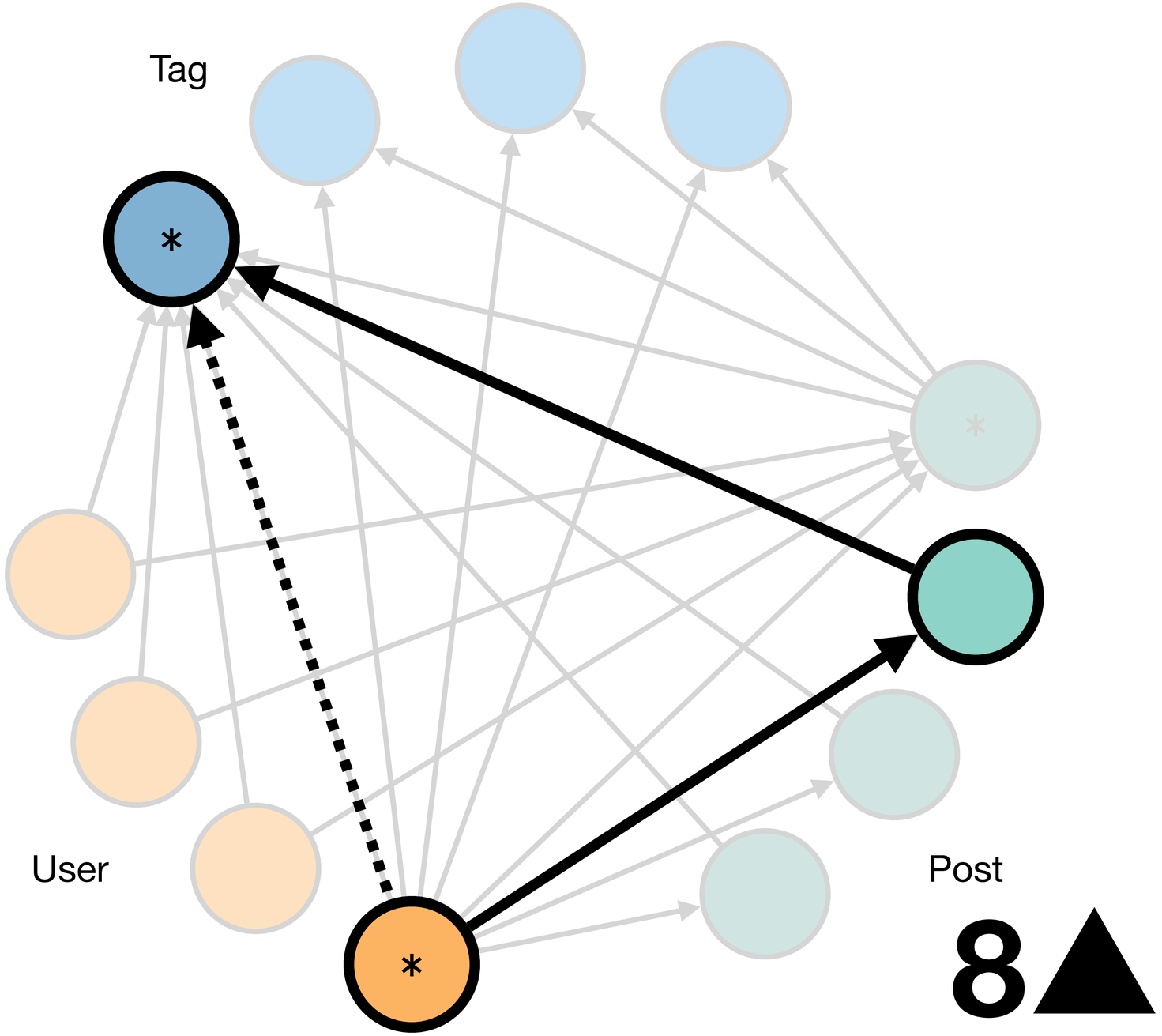
5▲



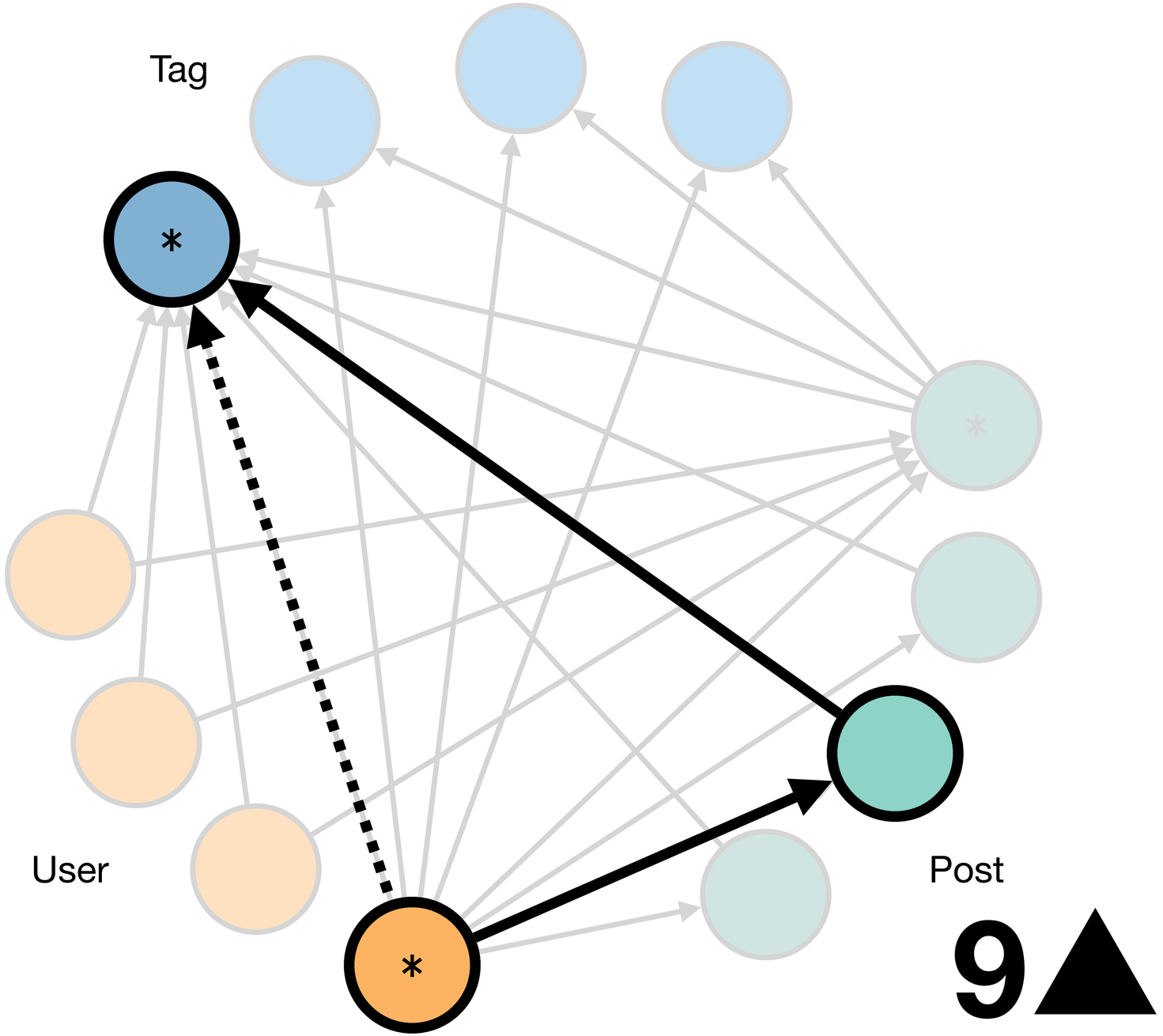
6▲

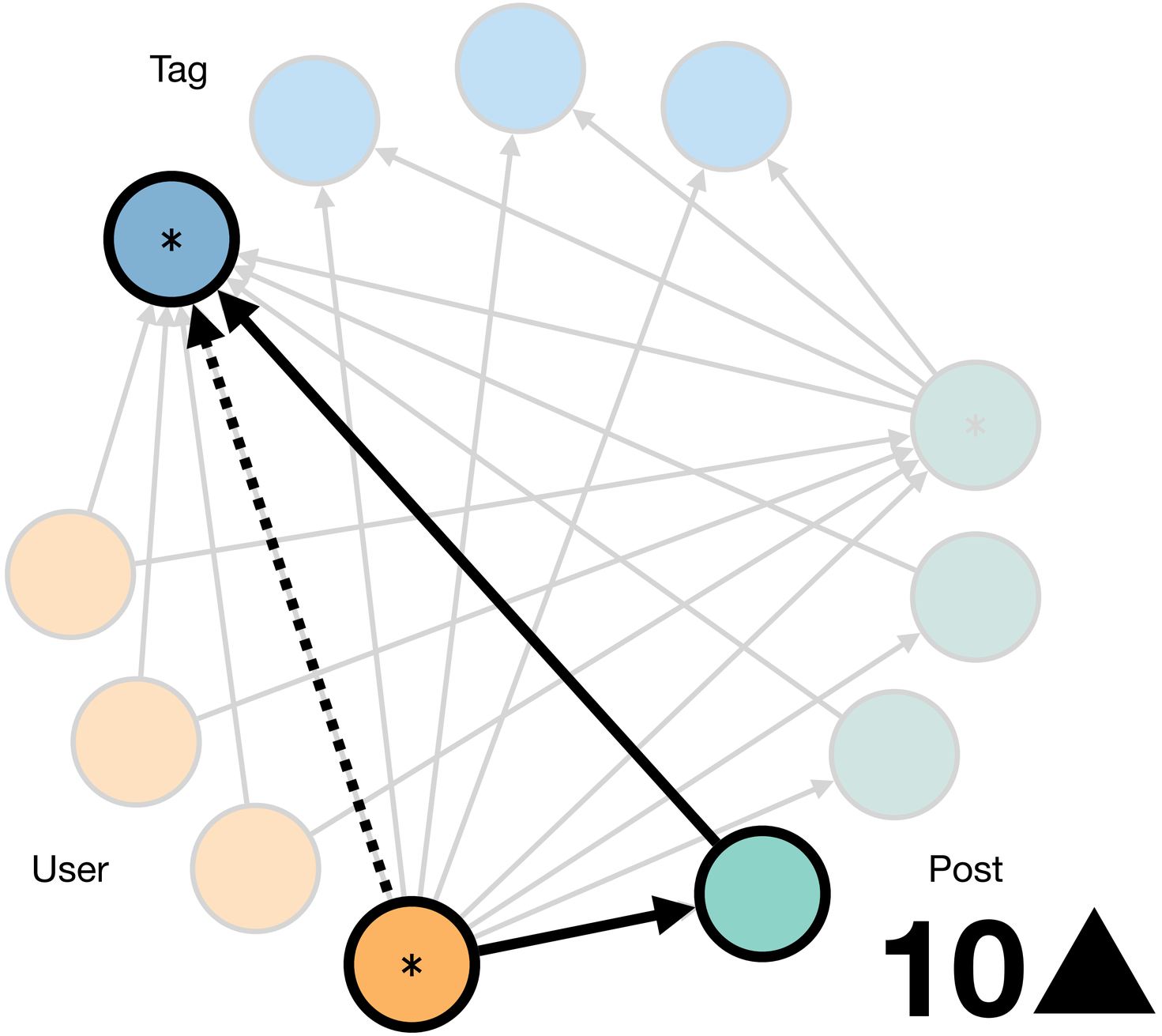


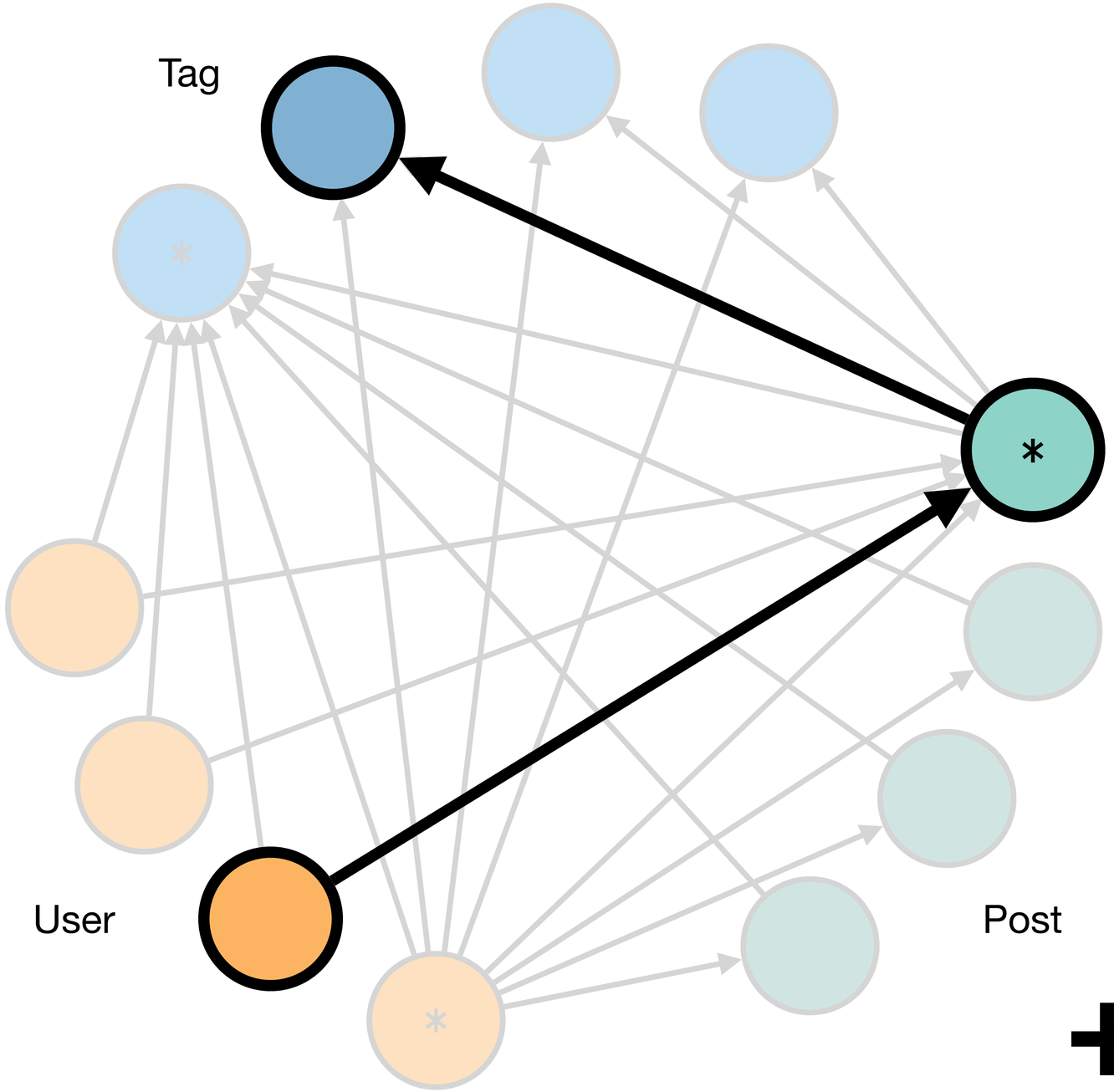
7



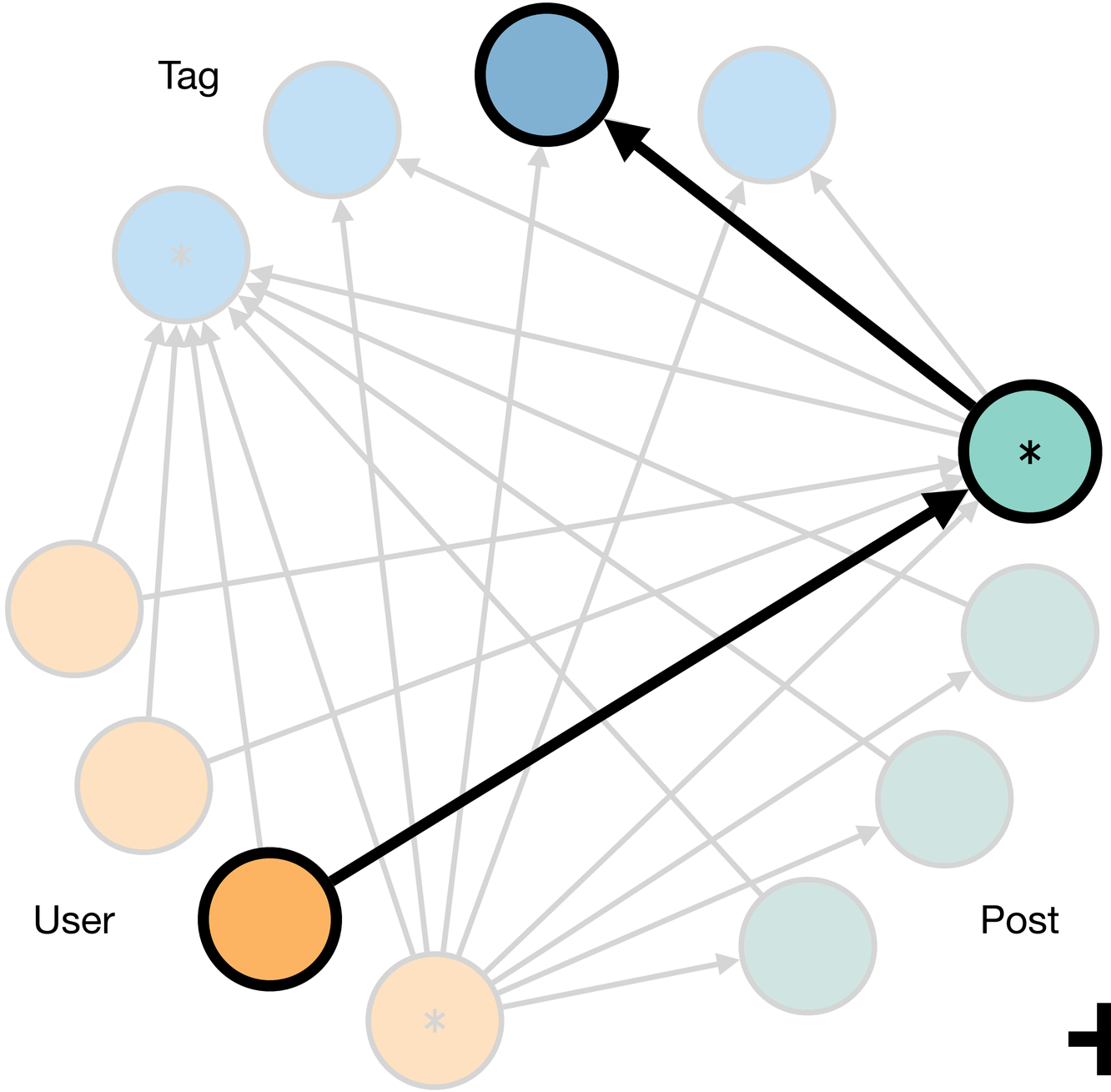
8


Tag
*
*
*
User
Post
*

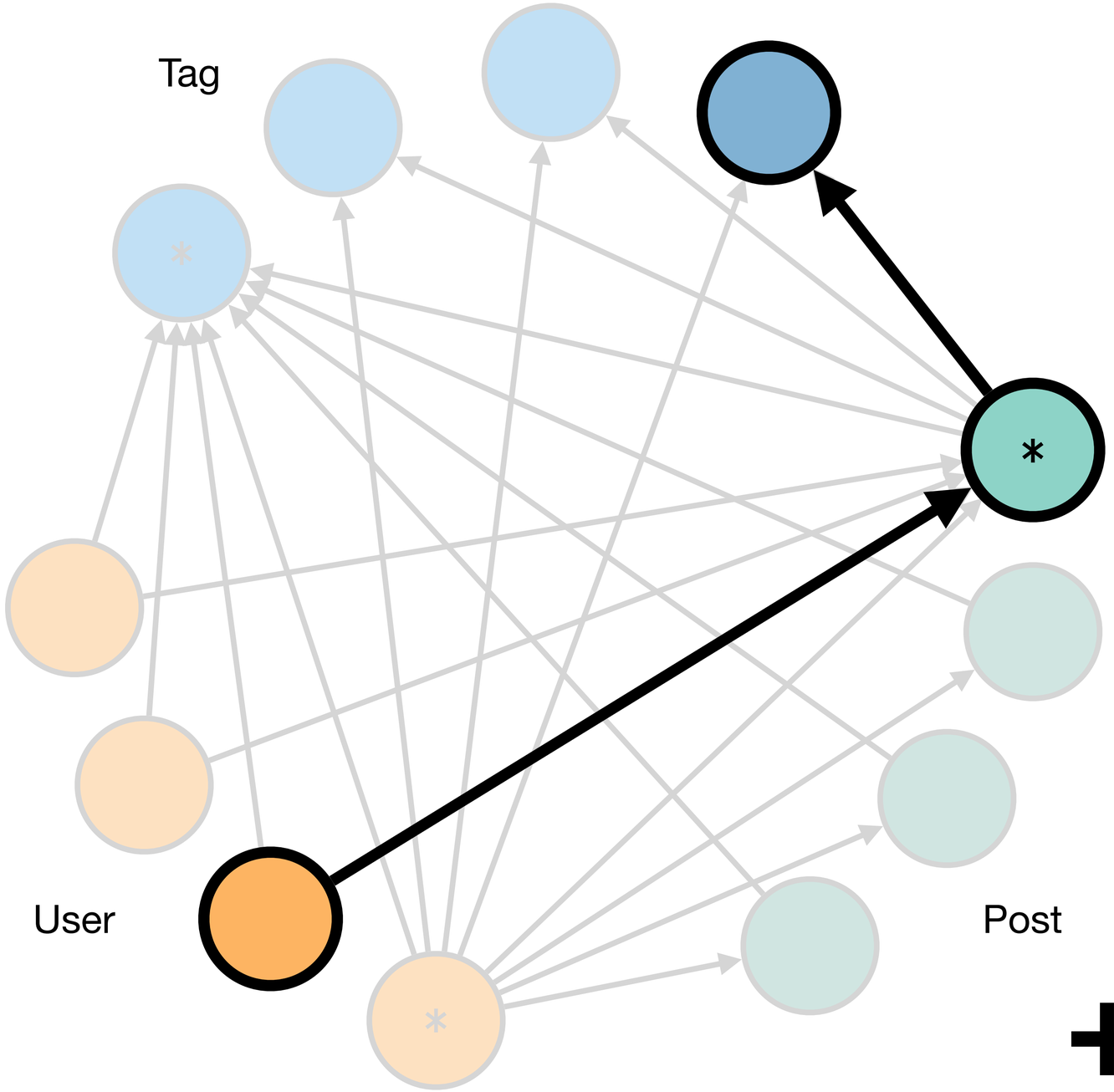

Tag
*
*
User
*
Post
10



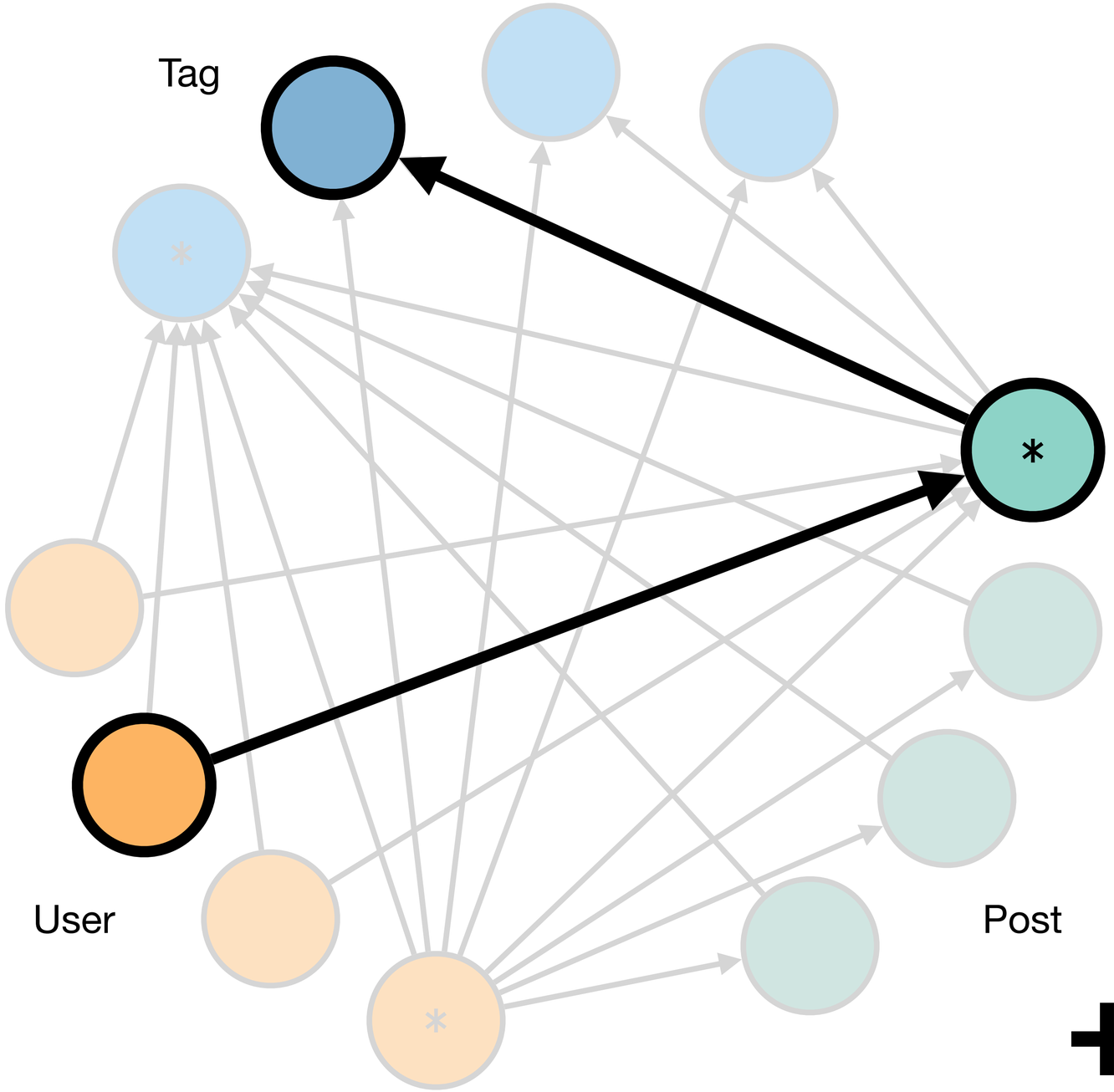
+1



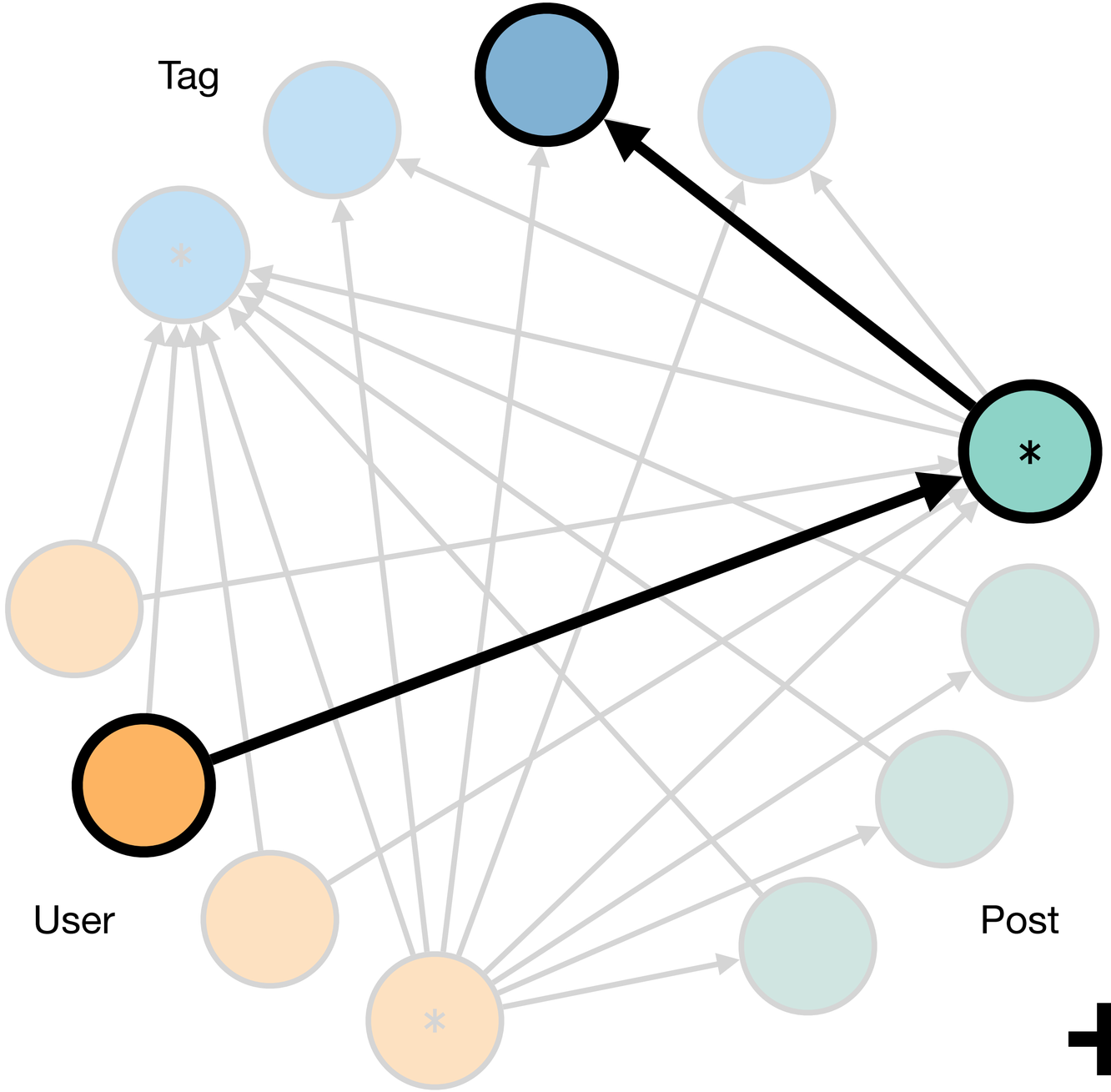
+2



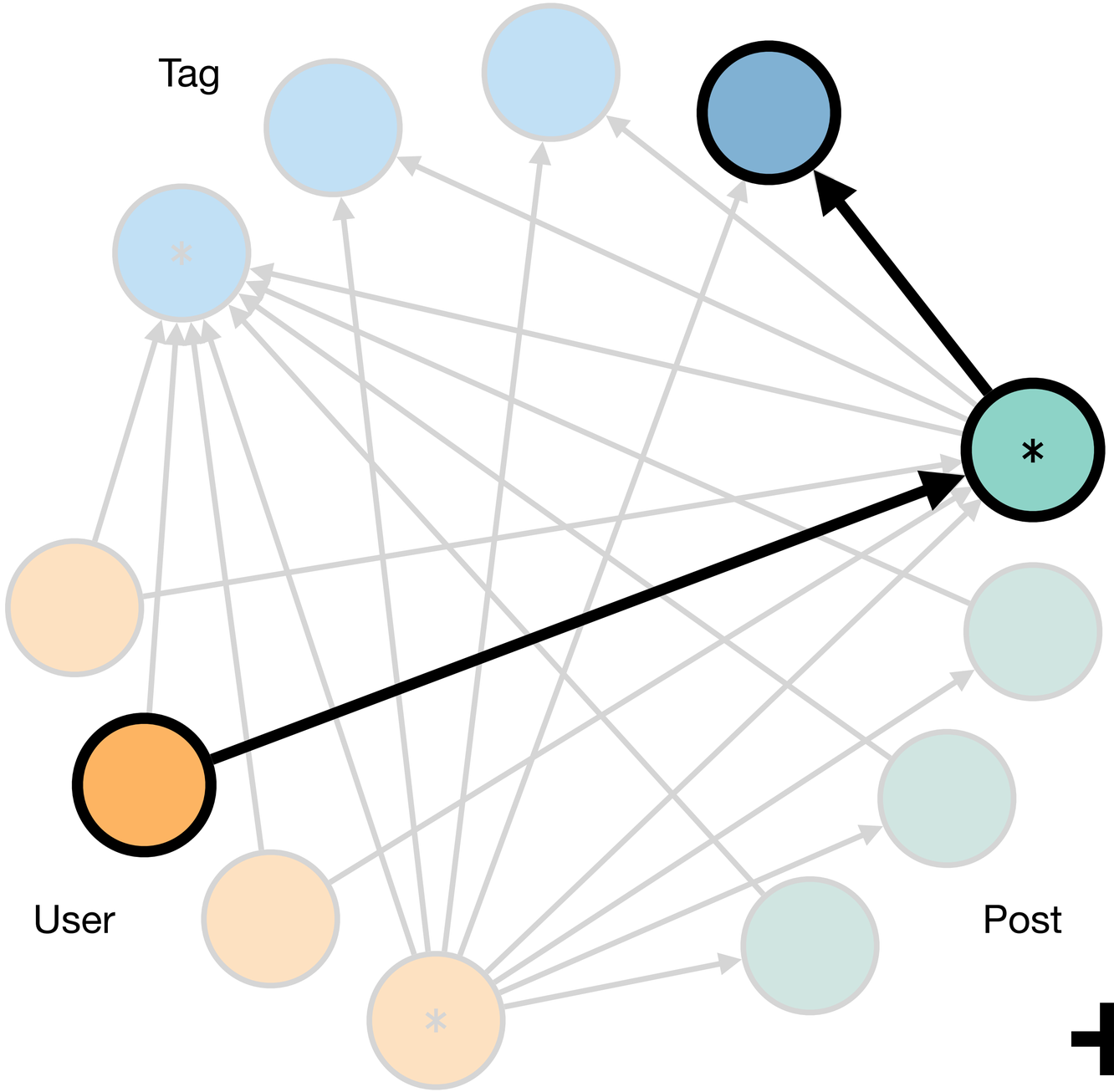
+3



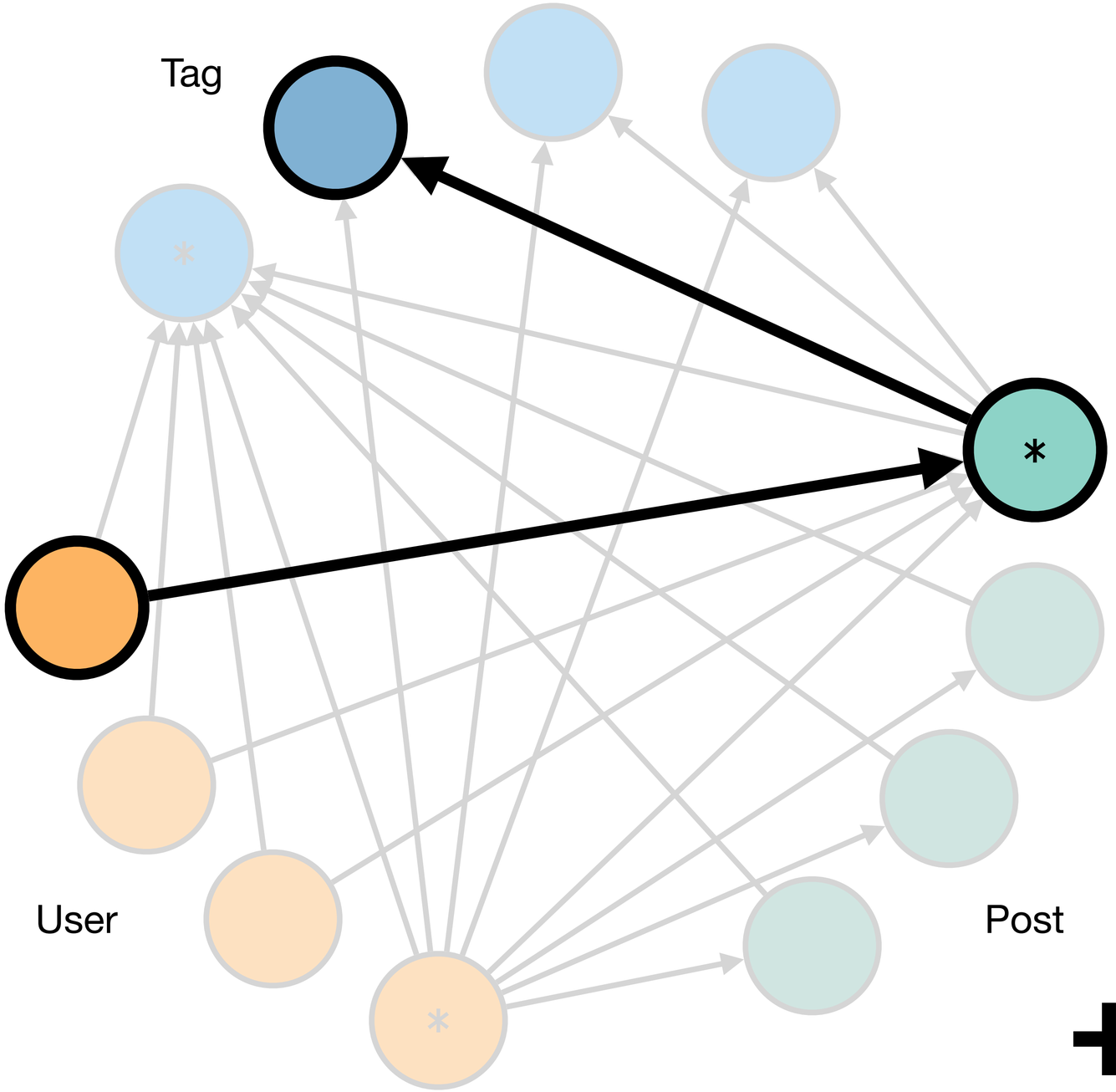
+4



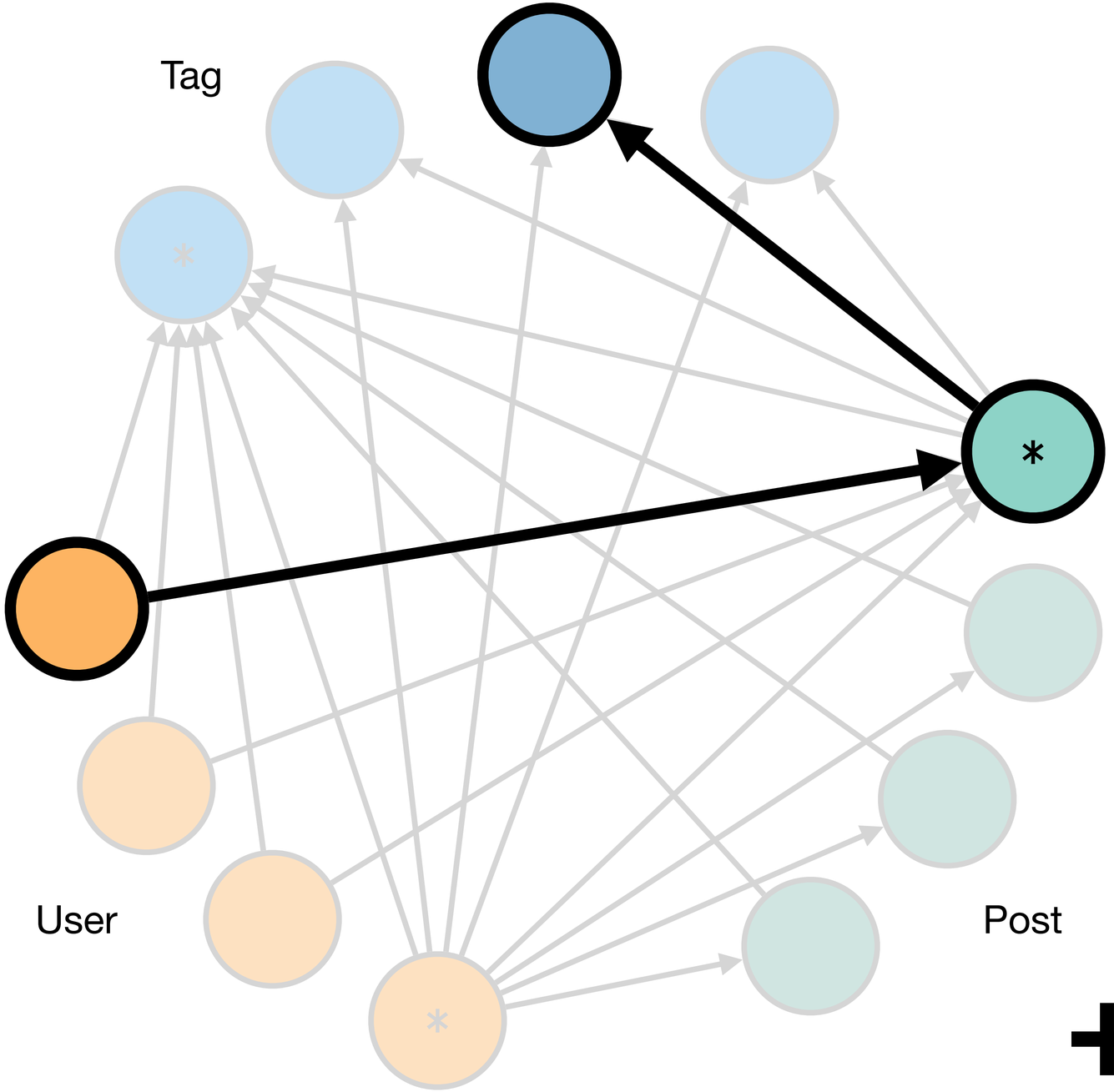
+5



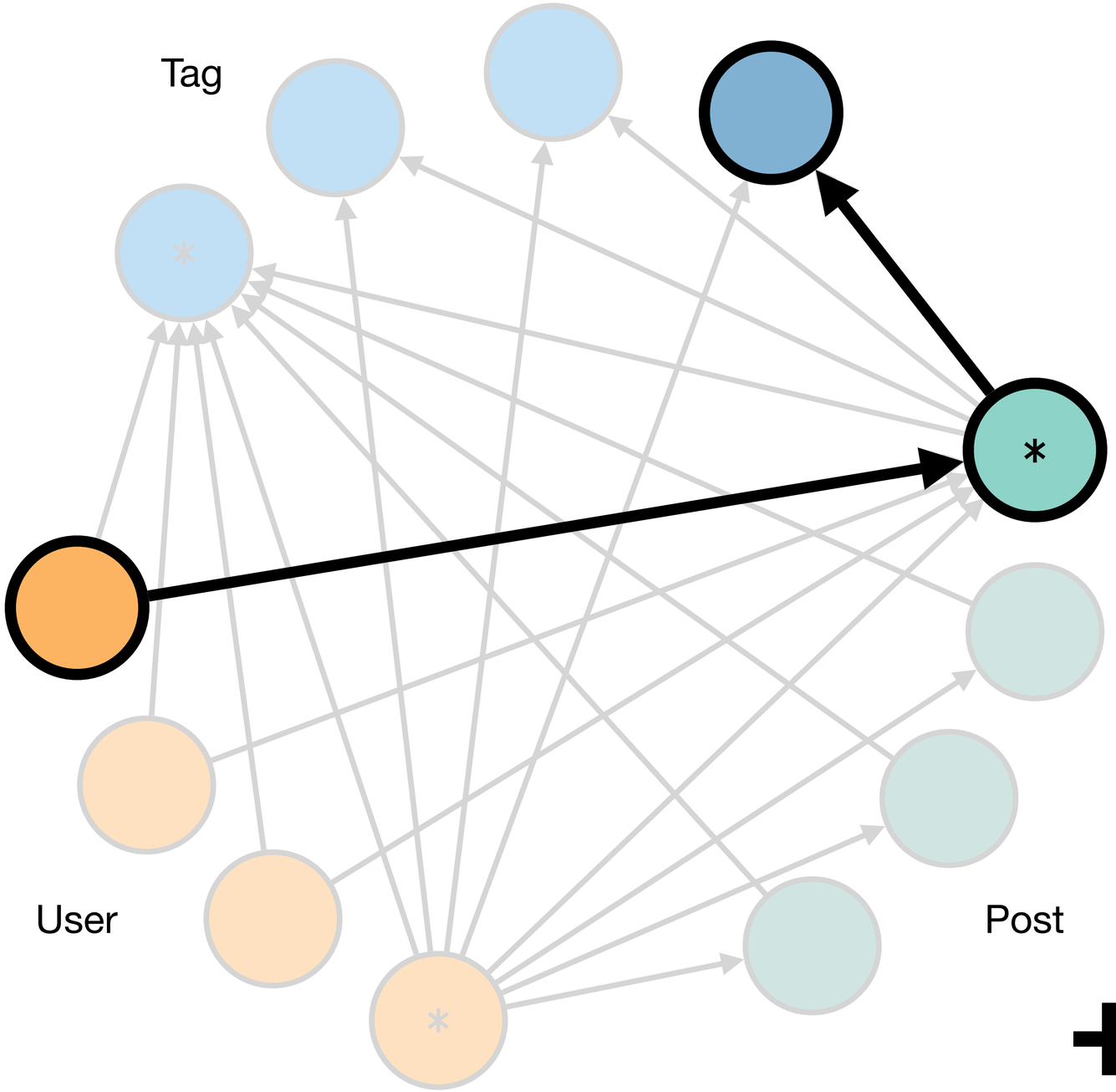
+6



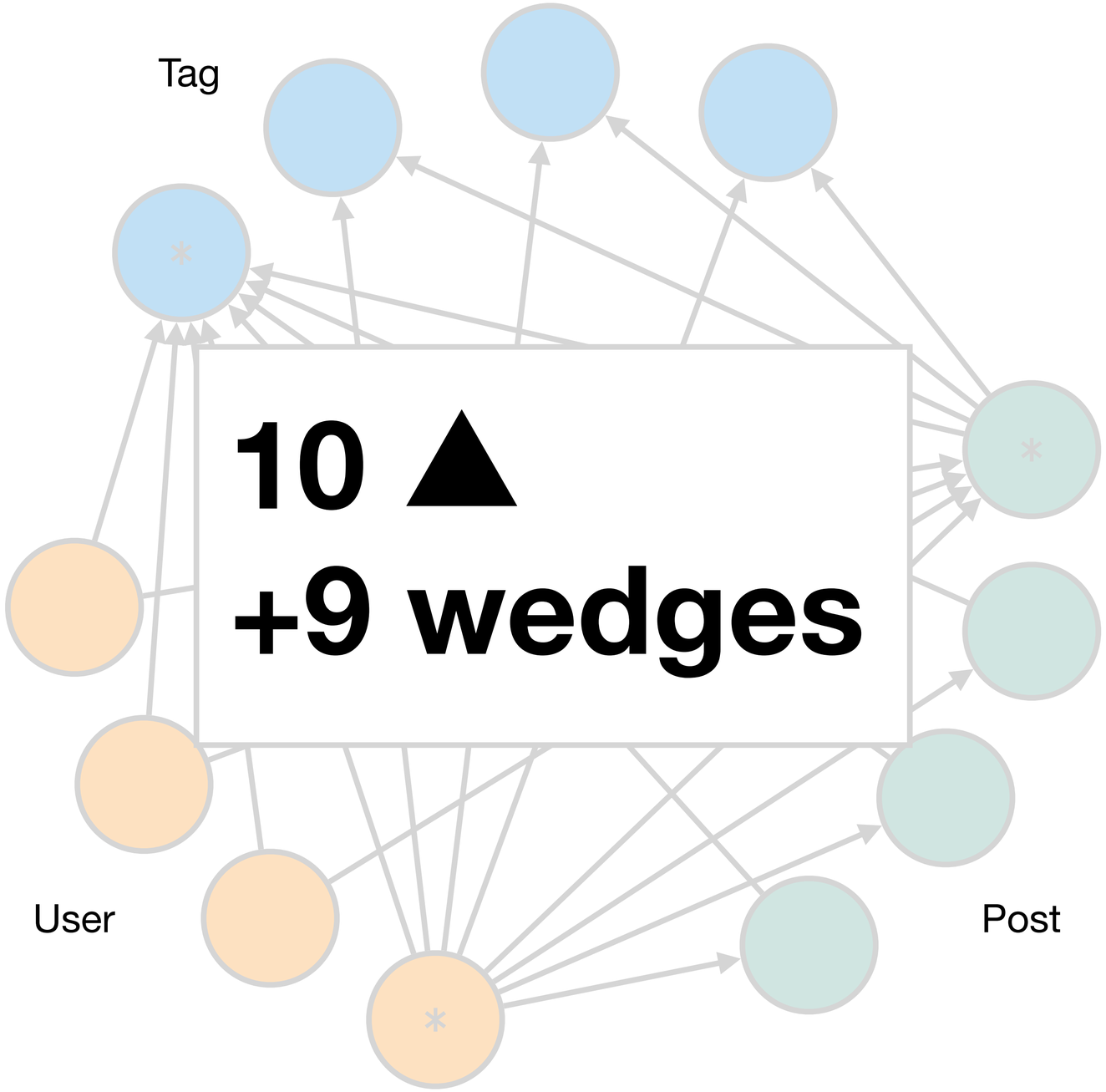
+7



+8



+9



From all three node types, we have 1 distinguished $(*)$ and k regular instances.

Size of tables:

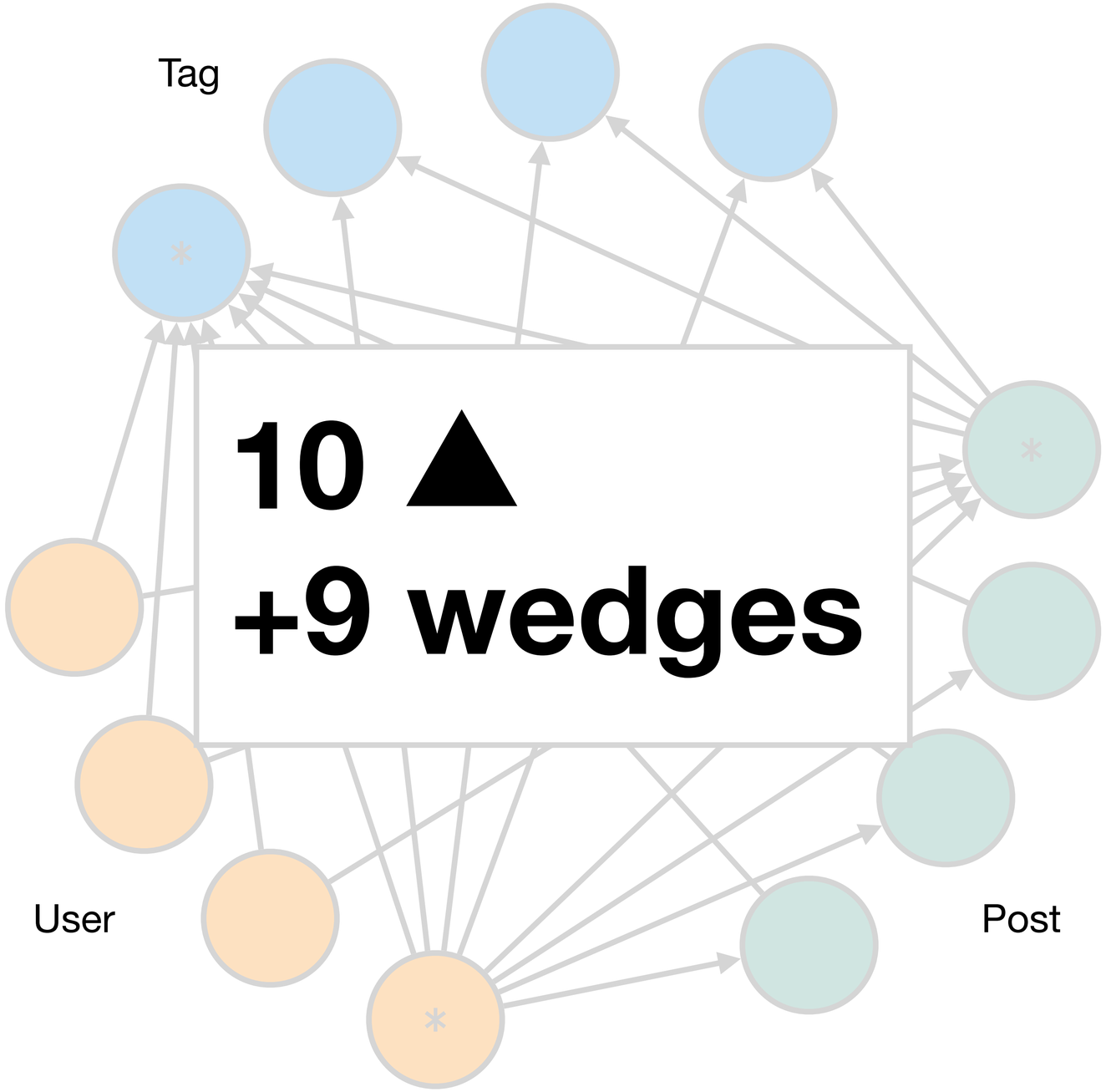
$$n = 2k + 1$$

Number of triangles:

$$q_{\triangle} = 3k + 1$$

Size of the first join:

$$q_{<} = k^2 + 3k + 1$$



From all three node types, we have 1 distinguished $(*)$ and k regular instances.

Size of tables:

$$n = 2k + 1$$

Number of triangles:

$$q_{\triangle} = 3k + 1$$

Size of the first join:

$$q_{<} = k^2 + 3k + 1$$

Foundations of Computer Science, 2007

Size Bounds and Query Plans for Relational Joins

Albert Atserias
Universitat Politècnica de Catalunya
Barcelona, Spain
atserias@lsi.upc.edu

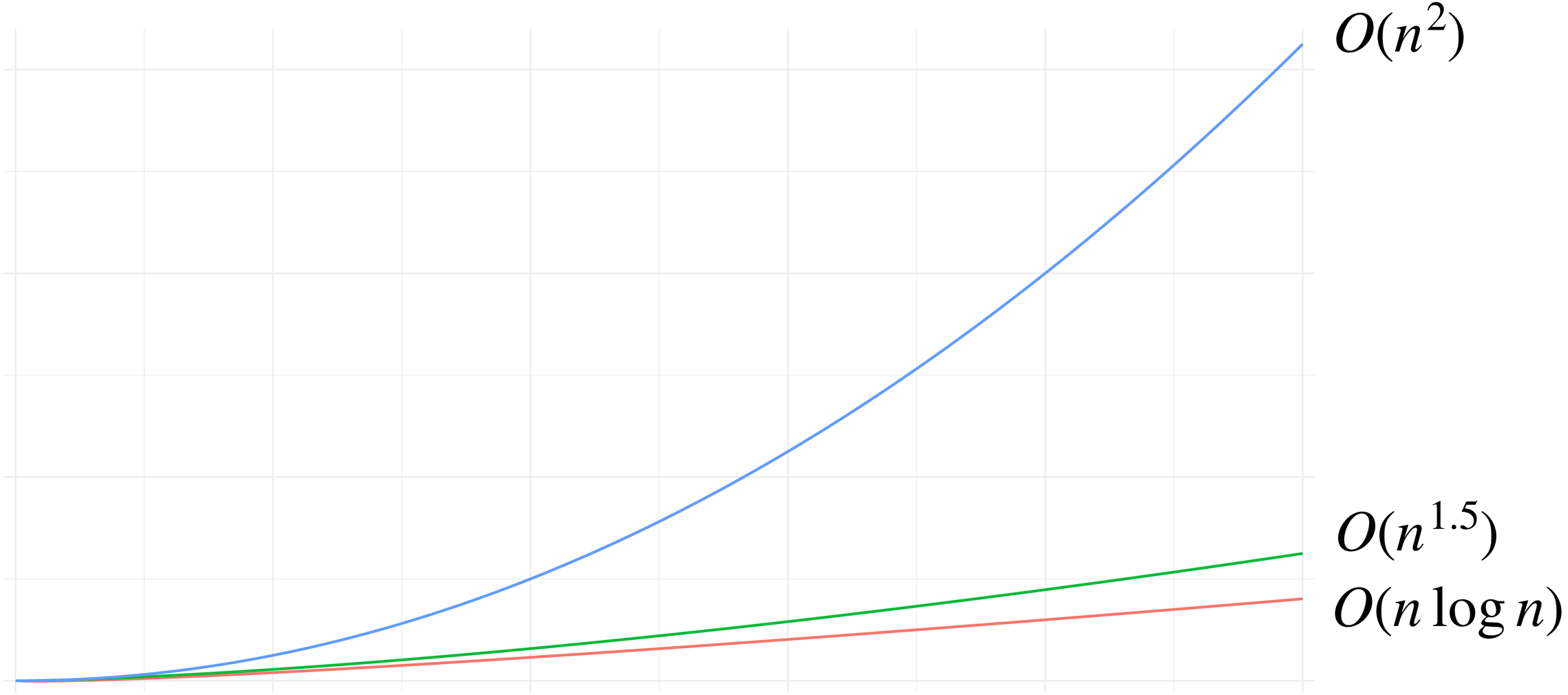
Martin Grohe
Humboldt Universität zu Berlin
Berlin, Germany
grohe@informatik.hu-berlin.de

Dániel Marx*
Budapest University of Technology and Economics
Budapest, Hungary
dmarx@cs.bme.hu

On a graph with n edges, computing $q_{\triangle}$ with joins takes $O(n^2)$

The output size of $q_{\triangle}$ is $\Theta(n^{1.5})$

Complexity: $n^{1.5}$ vs. n^2



Foundations of Computer Science, 2007

Size Bounds and Query Plans for Relational Joins

Albert Atserias
Universitat Politècnica de Catalunya
Barcelona, Spain
atserias@lsi.upc.edu

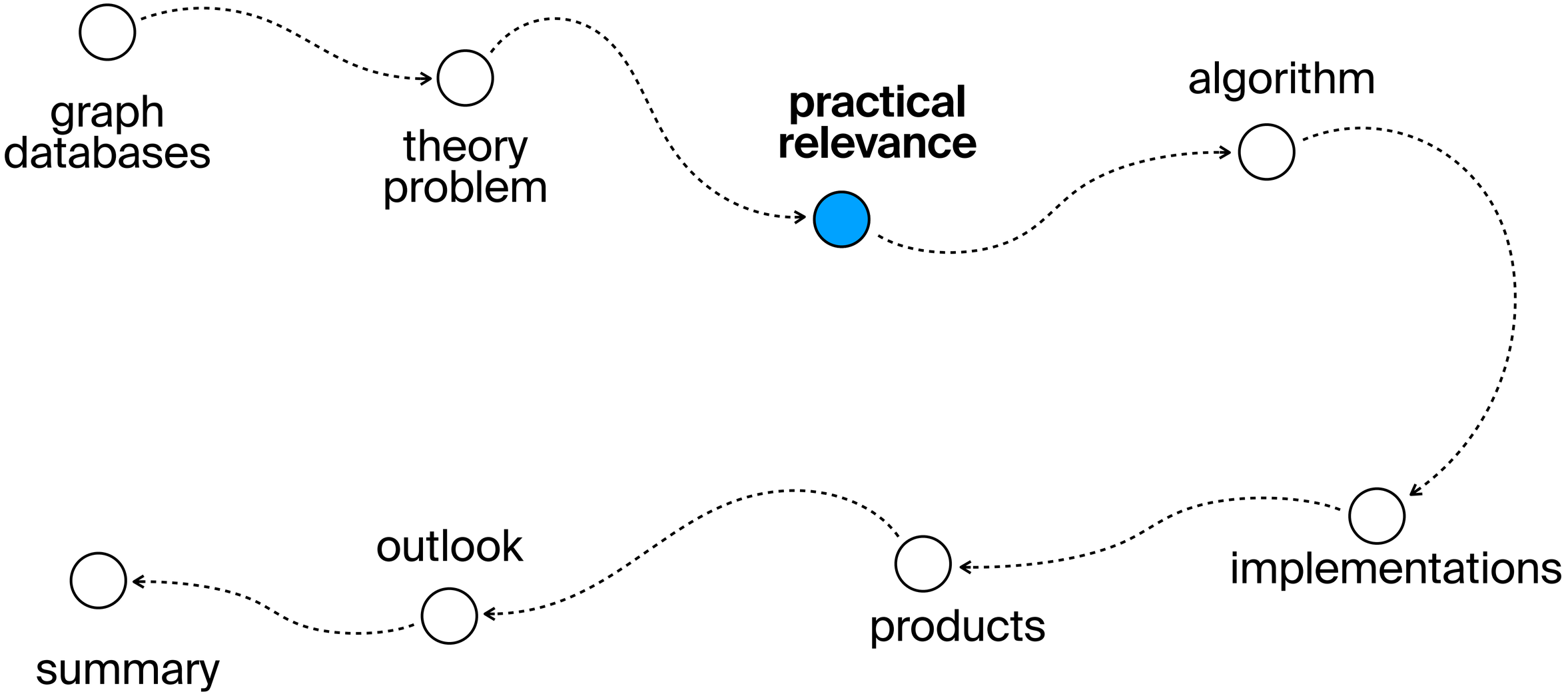
Martin Grohe
Humboldt Universität zu Berlin
Berlin, Germany
grohe@informatik.hu-berlin.de

Dániel Marx*
Budapest University of Technology and Economics
Budapest, Hungary
dmarx@cs.bme.hu

On a graph with n edges, computing $q_{\triangle}$ with joins takes $O(n^2)$

The output size of $q_{\triangle}$ is $\Theta(n^{1.5})$

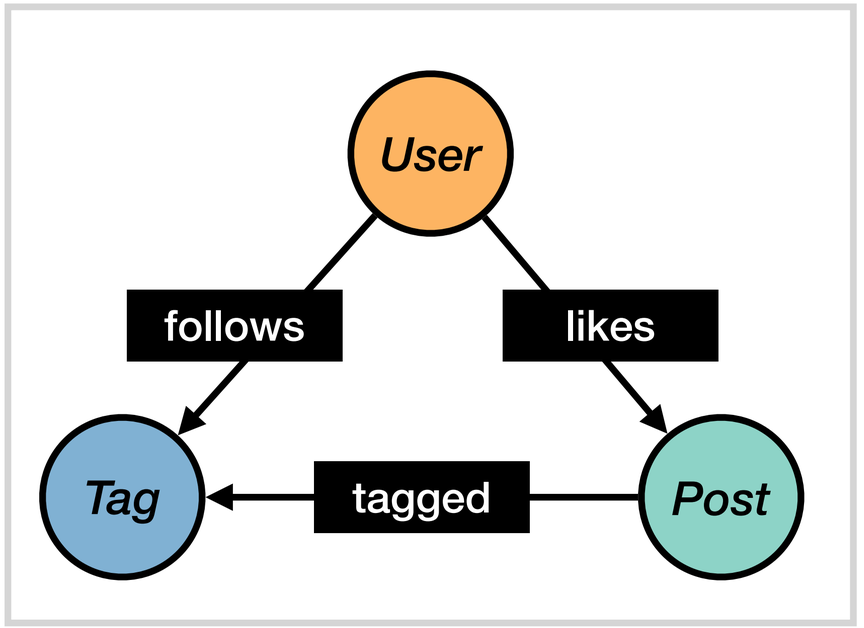
The paper doesn't give an algorithm for computing the join.


graph databases
theory problem
practical relevance
algorithm
implementations
products
outlook
summary

When does this problem occur?

Conditions:

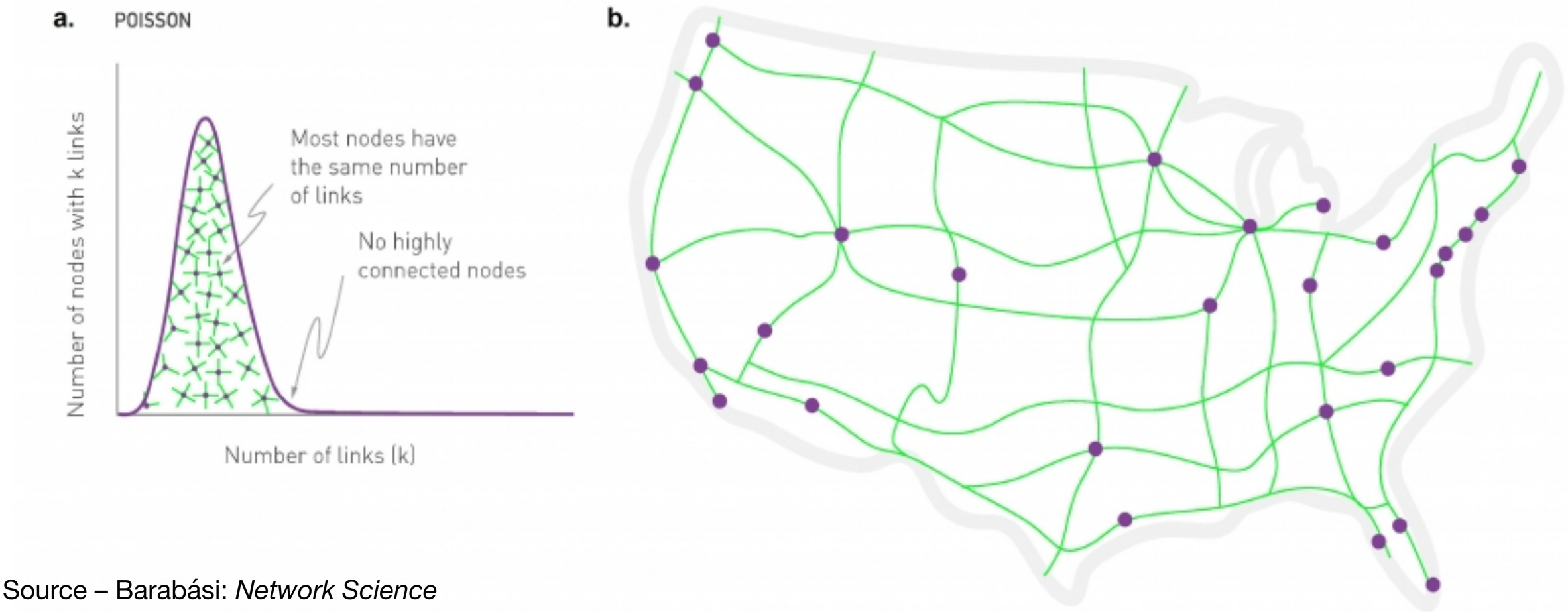
- cyclic graph pattern
- many-to-many cardinality edges
- asymmetric (skewed) degree distribution



These conditions can hold in any database but are more common in graph databases.

Degree distributions – Network science

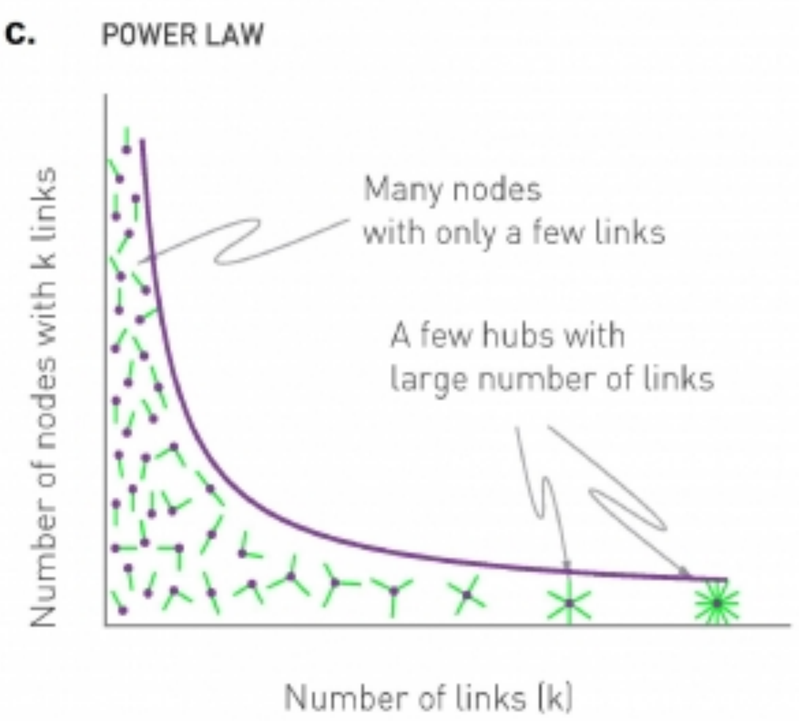
Poisson distribution



Source – Barabási: *Network Science*

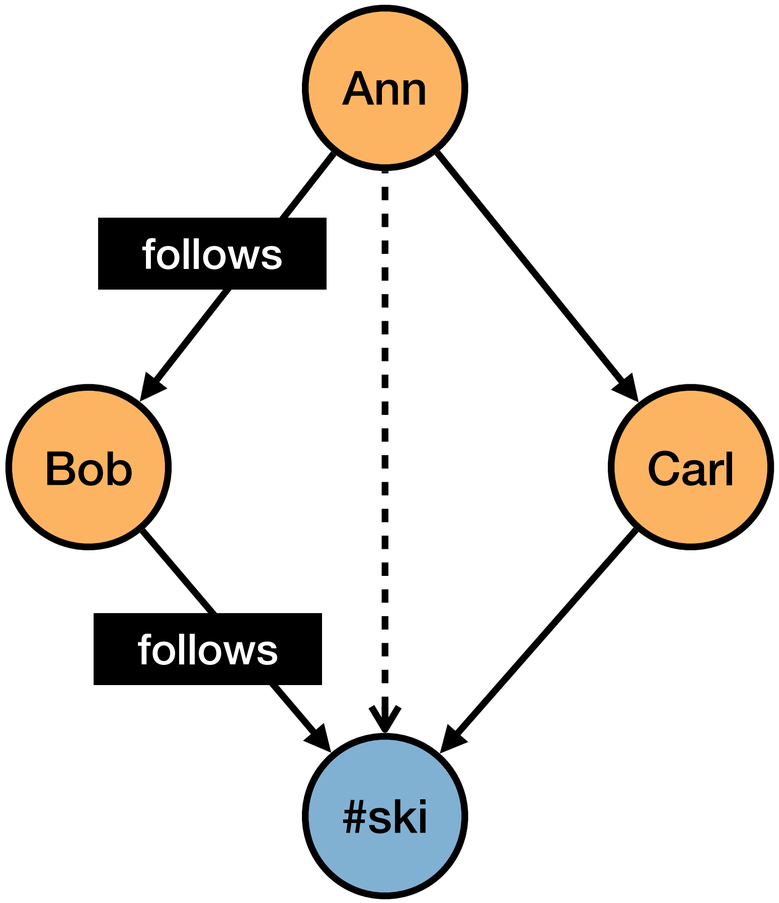
Degree distributions – Network science

Power law distribution

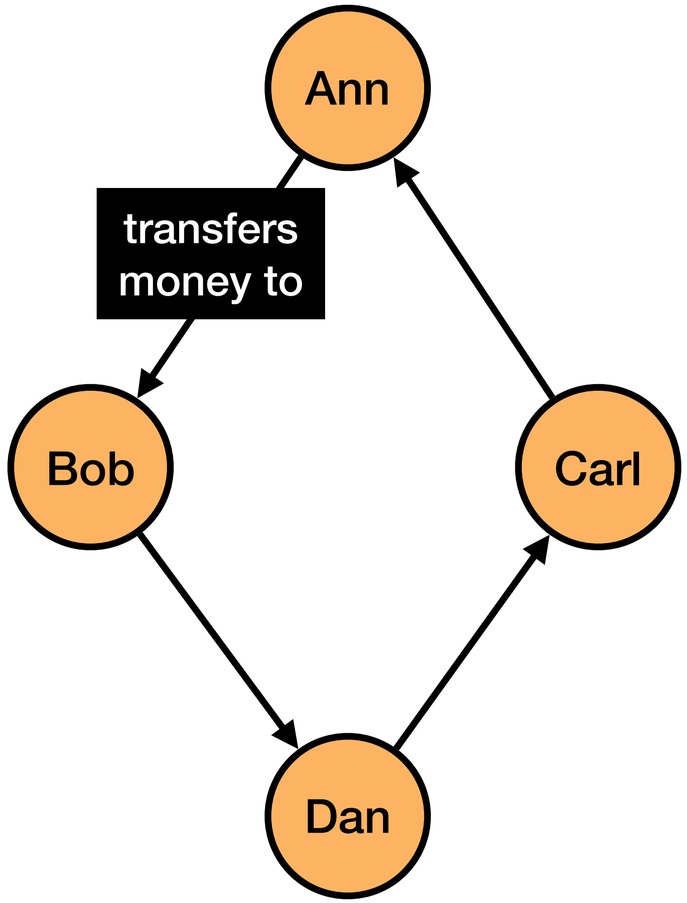


Source – Barabási: *Network Science*

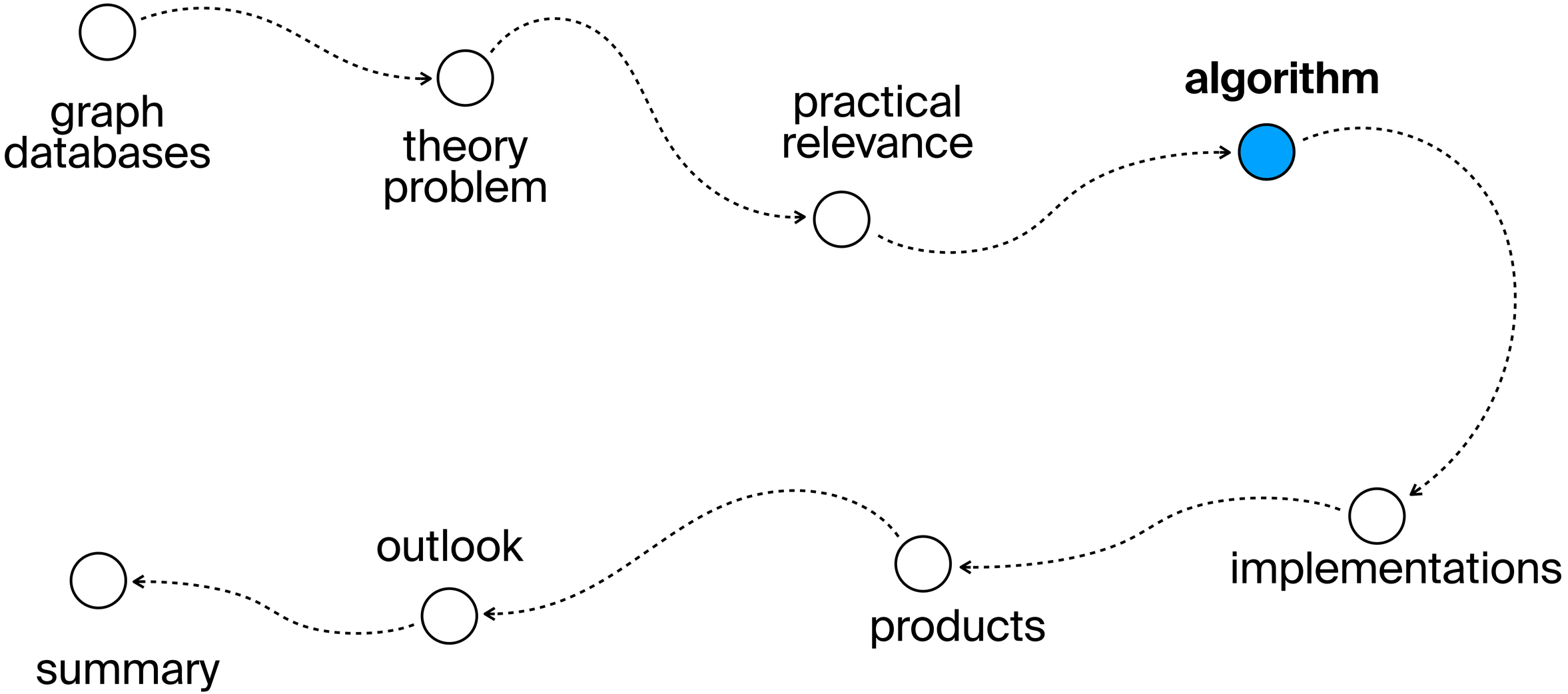
Cyclic graph queries in practice



recommendation algorithm



detecting money laundering


graph databases
theory problem
practical relevance
algorithm
implementations
products
outlook
summary

Principles of Database Systems 2012

Worst-case Optimal Join Algorithms

Hung Q. Ngo
University at Buffalo, SUNY
hungngo@buffalo.edu

Ely Porat
Bar-Ilan University
porately@cs.biu.ac.il

Christopher Ré
University of Wisconsin–Madison
chrisre@cs.wisc.edu

Atri Rudra
University at Buffalo, SUNY
atri@buffalo.edu

To understand the spirit of AGM's results, consider the following example where we have a schema with three attributes, A, B, and C, and three relations, $R(A,B)$, $S(B,C)$ and $T(A,C)$, defined over those attributes. Consider the following natural join query:

$$q = R \bowtie S \bowtie T \qquad (1)$$

Algorithm for computing $q_{\triangle}$

in $O(n^{1.5})$ steps

"Traditionally", join is a binary operator.

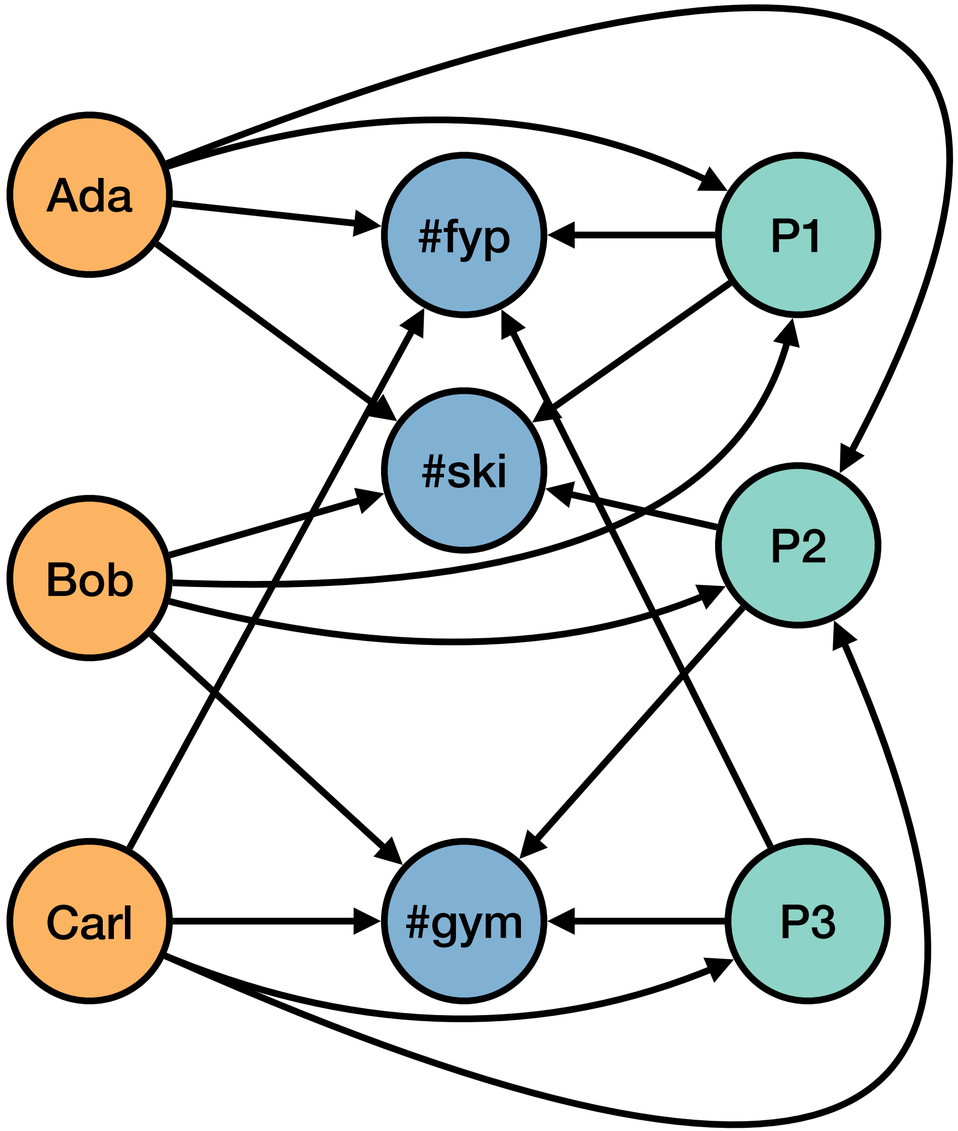
```
follows ⋈ tagged ⋈ likes
```

Idea: treat joins are n-ary operators *(multiway join)*:

```
⋈ (follows, tagged, likes)
```

The paper gives two algorithms, both using repeated intersections.

These are asymptotically optimal join algorithms, known as *worst-case optimal join algorithms.*



follows

User	Tag
Ada	#fyp
Ada	#ski
Bob	#ski
Bob	#gym
Carl	#fyp
Carl	#gym

tagged

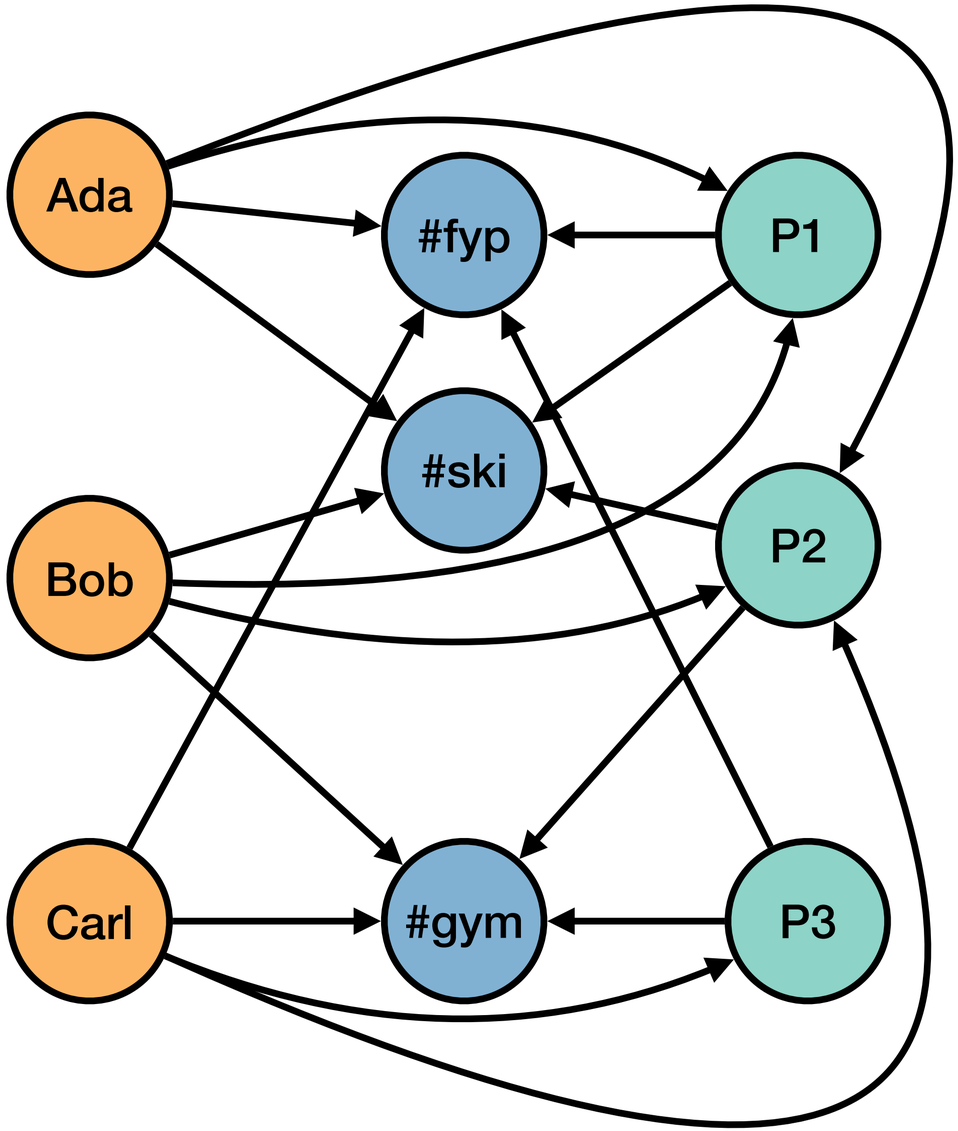
Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

∩

User
Ada
Bob
Carl



follows

User	Tag
Ada	#fyp
Ada	#ski
Bob	#ski
Bob	#gym
Carl	#fyp
Carl	#gym

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

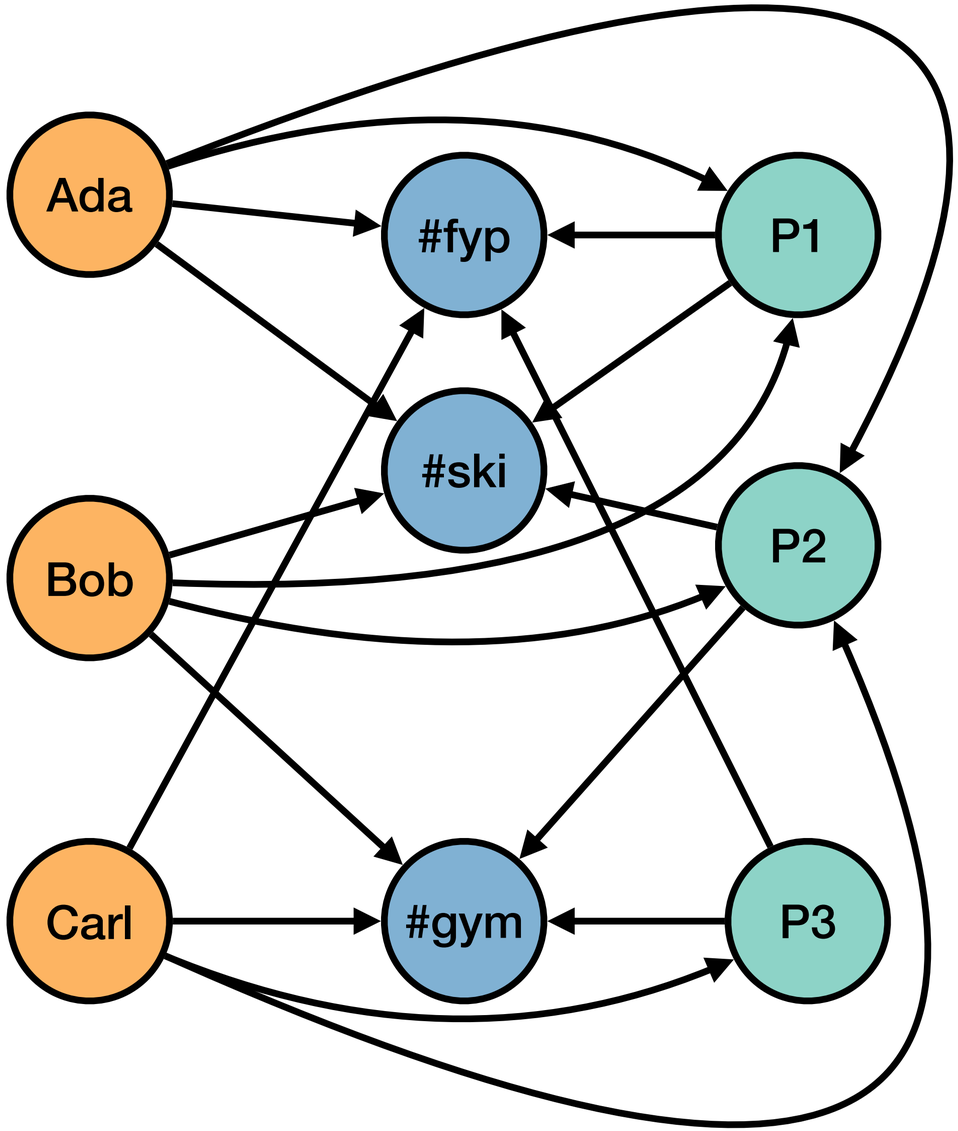
likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

∩

User
Ada
Bob
Carl

User	Tag
Ada	#fyp
Ada	#ski



follows

User	Tag
Ada	#fyp
Ada	#ski
Bob	#ski
Bob	#gym
Carl	#fyp
Carl	#gym

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

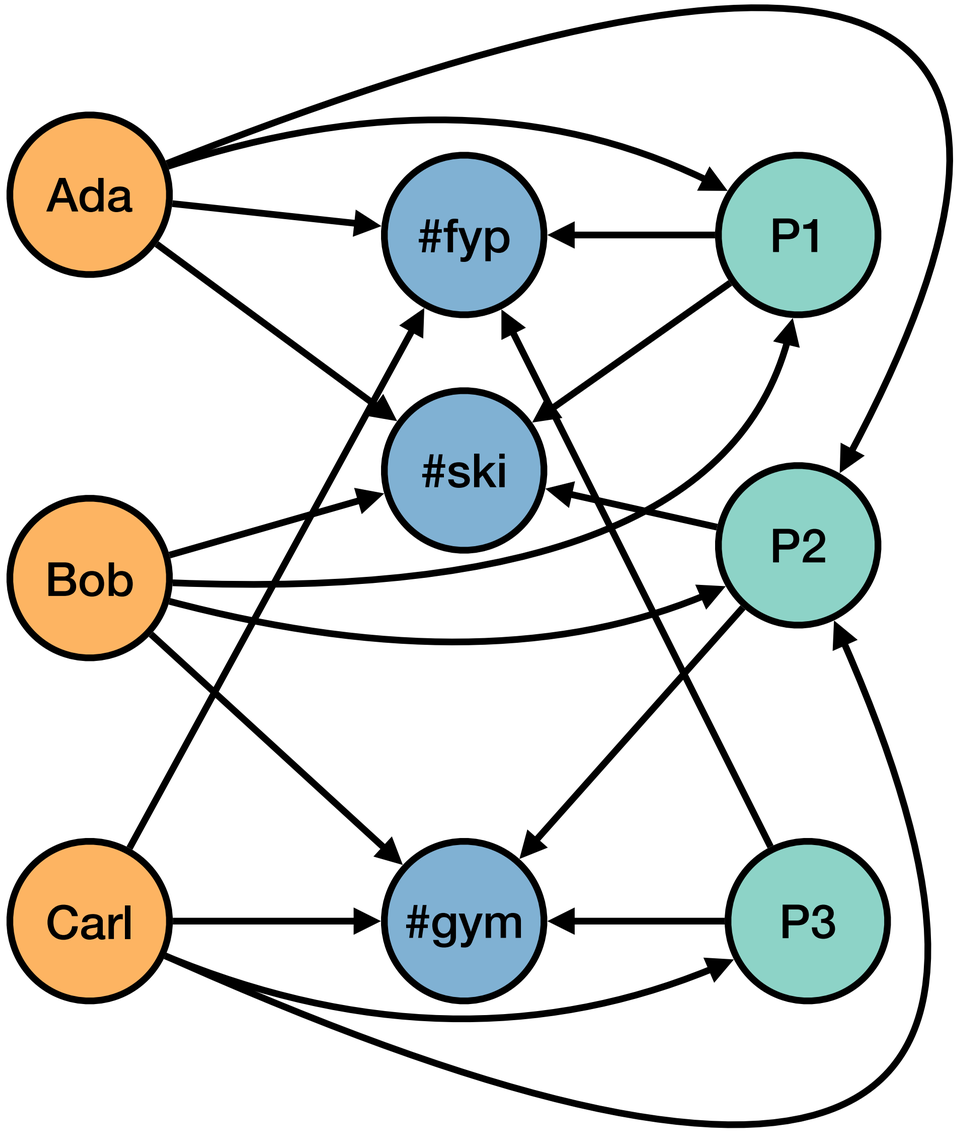
likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

∩

User
Ada
Bob
Carl

User	Tag
Ada	#fyp
Ada	#ski
Bob	#gym
Bob	#ski



follows

User	Tag
Ada	#fyp
Ada	#ski
Bob	#ski
Bob	#gym
Carl	#fyp
Carl	#gym

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

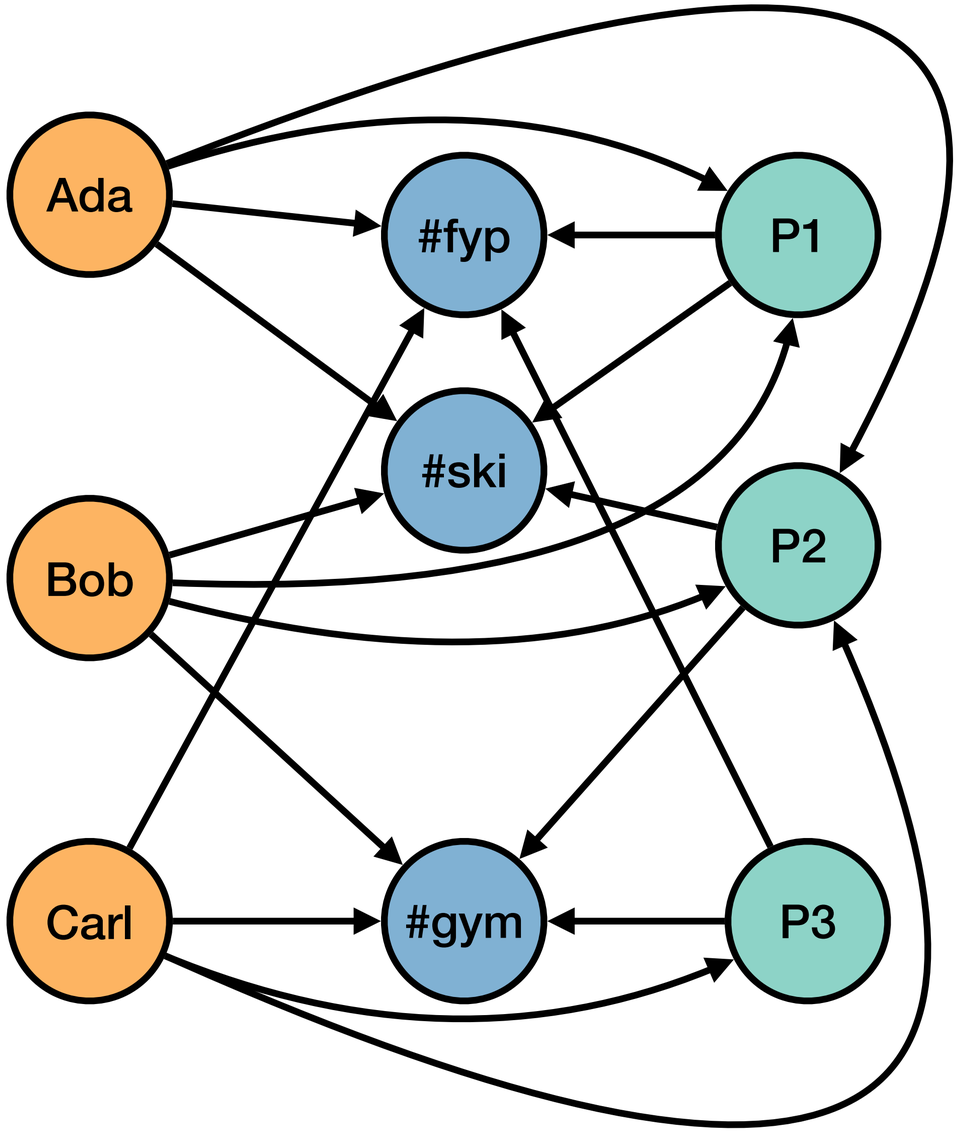
likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

∩

User
Ada
Bob
Carl

User	Tag
Ada	#fyp
Ada	#ski
Bob	#gym
Bob	#ski
Carl	#fyp
Carl	#gym



follows

User	Tag
Ada	#fyp
Ada	#ski
Bob	#ski
Bob	#gym
Carl	#fyp
Carl	#gym

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

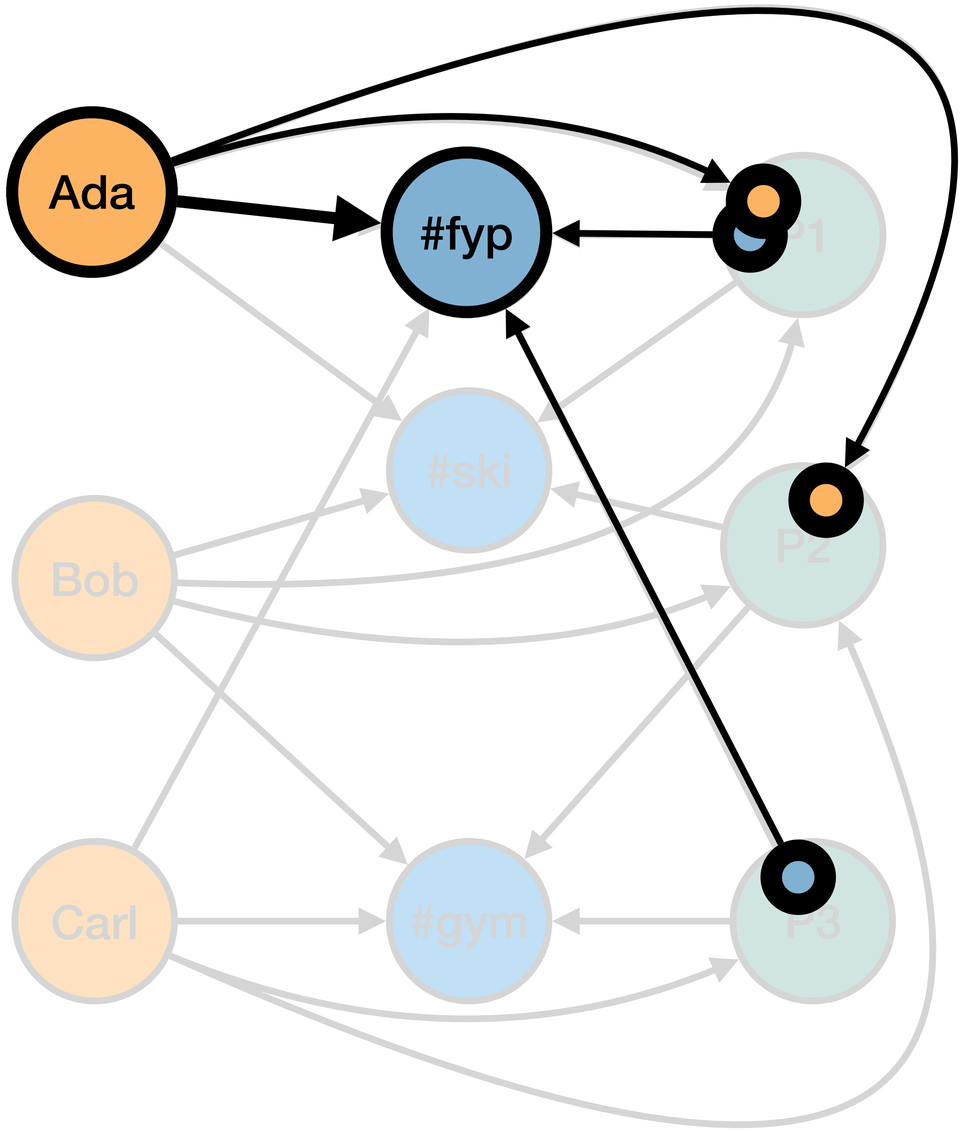
likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

∩

User	Tag
Ada	#fyp
Ada	#ski
Bob	#gym
Bob	#ski
Carl	#fyp
Carl	#gym

User	Tag	Post
Ada	#fyp	P1



follows

User	Tag
Ada	#fyp
Ada	#ski
Bob	#ski
Bob	#gym
Carl	#fyp
Carl	#gym

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

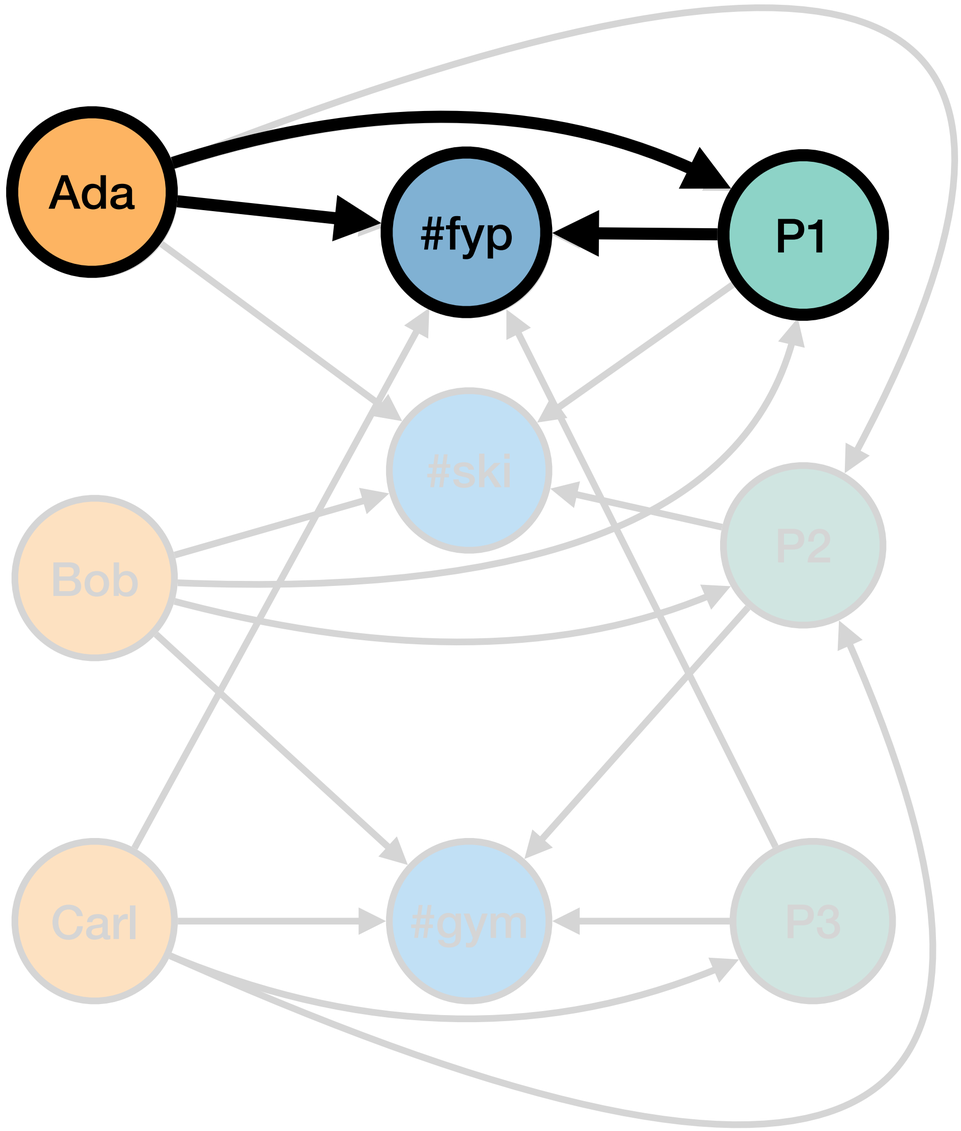
likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

∩

User	Tag
Ada	#fyp
Ada	#ski
Bob	#gym
Bob	#ski
Carl	#fyp
Carl	#gym

User	Tag	Post
Ada	#fyp	P1



follows

User	Tag
Ada	#fyp
Ada	#ski
Bob	#ski
Bob	#gym
Carl	#fyp
Carl	#gym

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

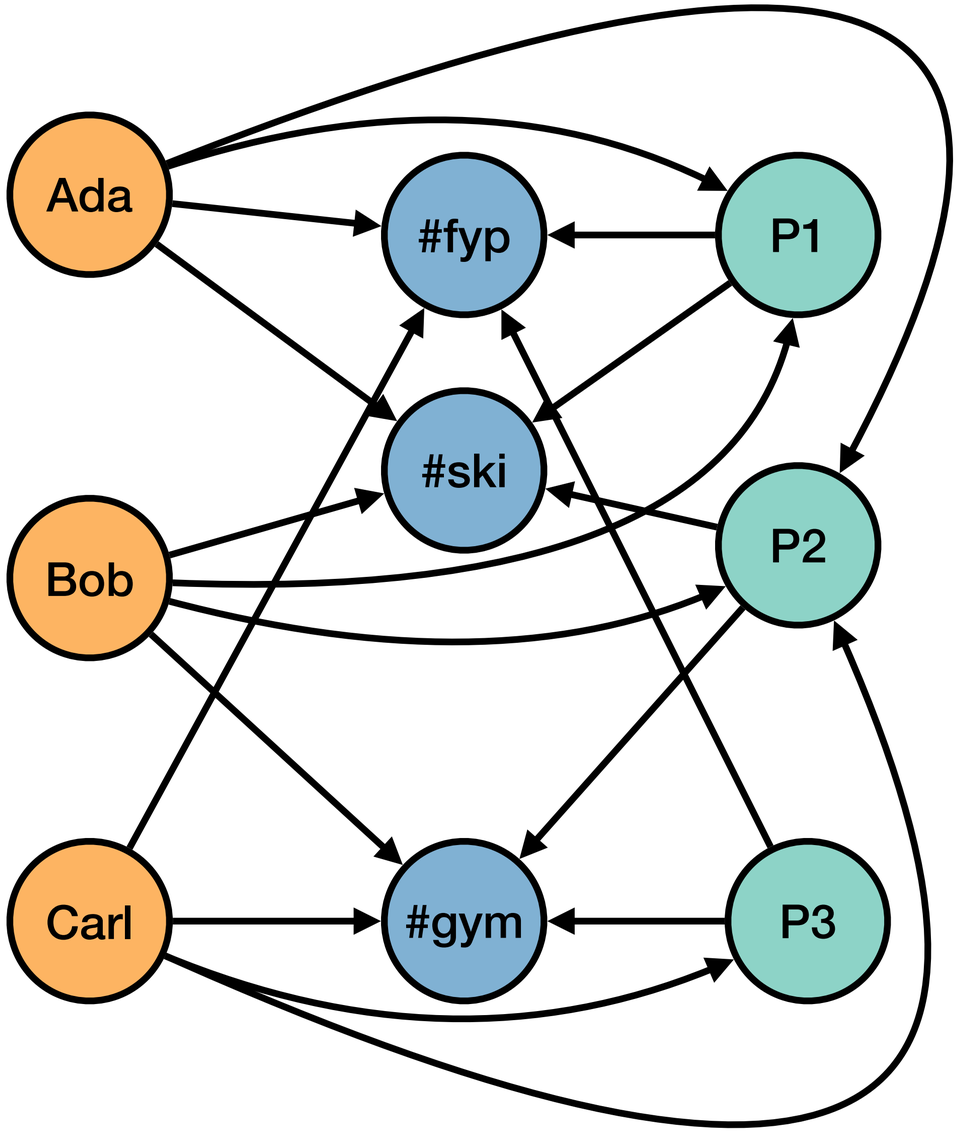
likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

∩

User	Tag
Ada	#fyp
Ada	#ski
Bob	#gym
Bob	#ski
Carl	#fyp
Carl	#gym

User	Tag	Post
Ada	#fyp	P1



follows

User	Tag
Ada	#fyp
Ada	#ski
Bob	#ski
Bob	#gym
Carl	#fyp
Carl	#gym

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

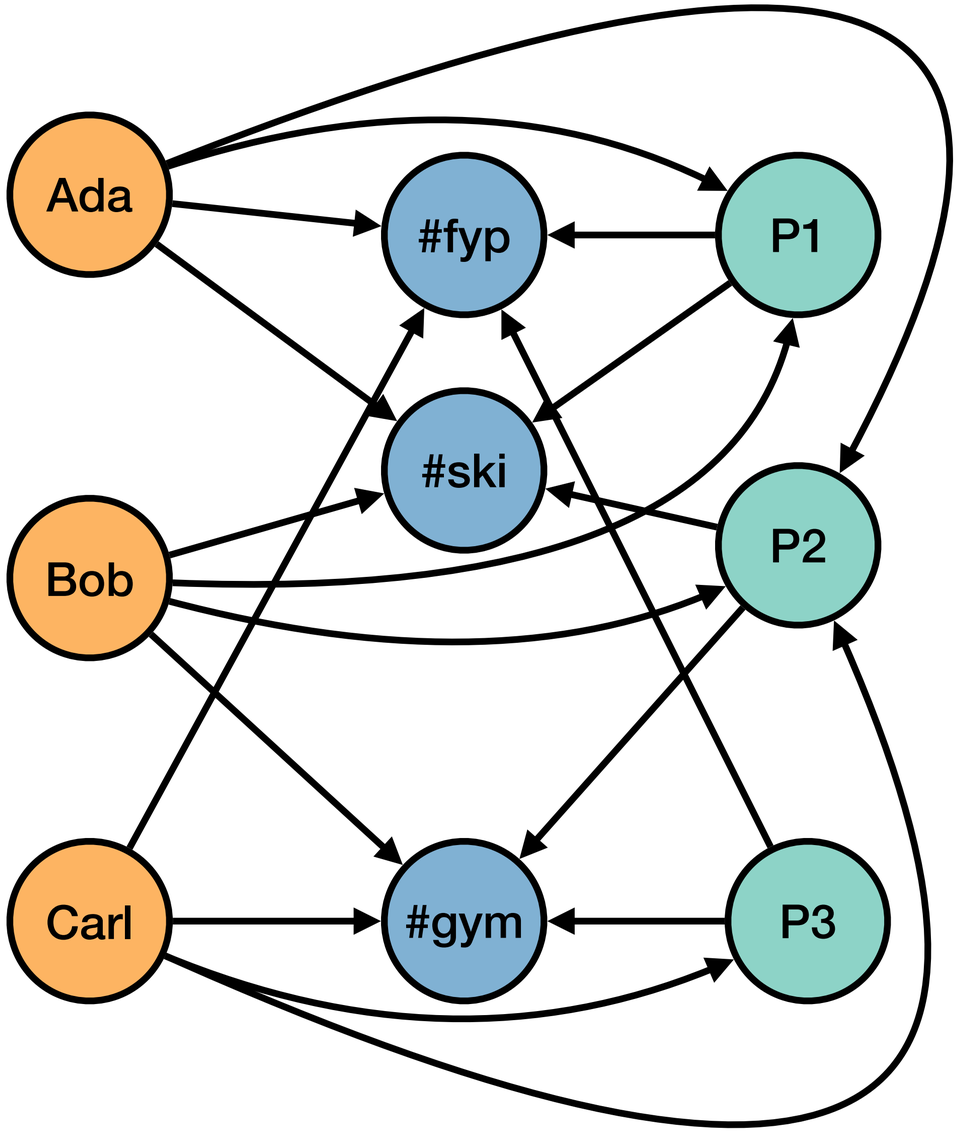
likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

∩

User	Tag
Ada	#fyp
Ada	#ski
Bob	#gym
Bob	#ski
Carl	#fyp
Carl	#gym

User	Tag	Post
Ada	#fyp	P1
Ada	#ski	P1
Ada	#ski	P2



follows

User	Tag
Ada	#fyp
Ada	#ski
Bob	#ski
Bob	#gym
Carl	#fyp
Carl	#gym

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

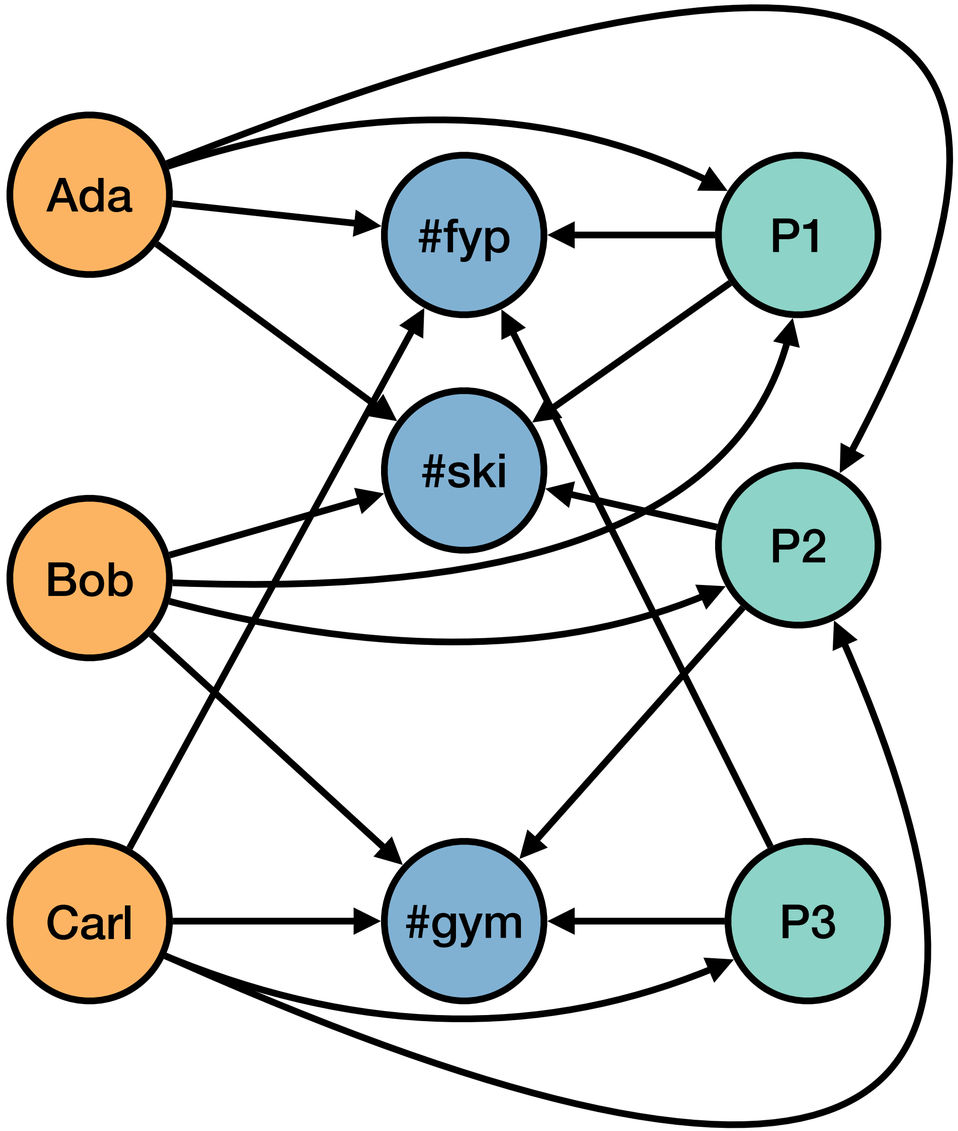
likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

∩

User	Tag
Ada	#fyp
Ada	#ski
Bob	#gym
Bob	#ski
Carl	#fyp
Carl	#gym

User	Tag	Post
Ada	#fyp	P1
Ada	#ski	P1
Ada	#ski	P2
Bob	#gym	P2



follows

User	Tag
Ada	#fyp
Ada	#ski
Bob	#ski
Bob	#gym
Carl	#fyp
Carl	#gym

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

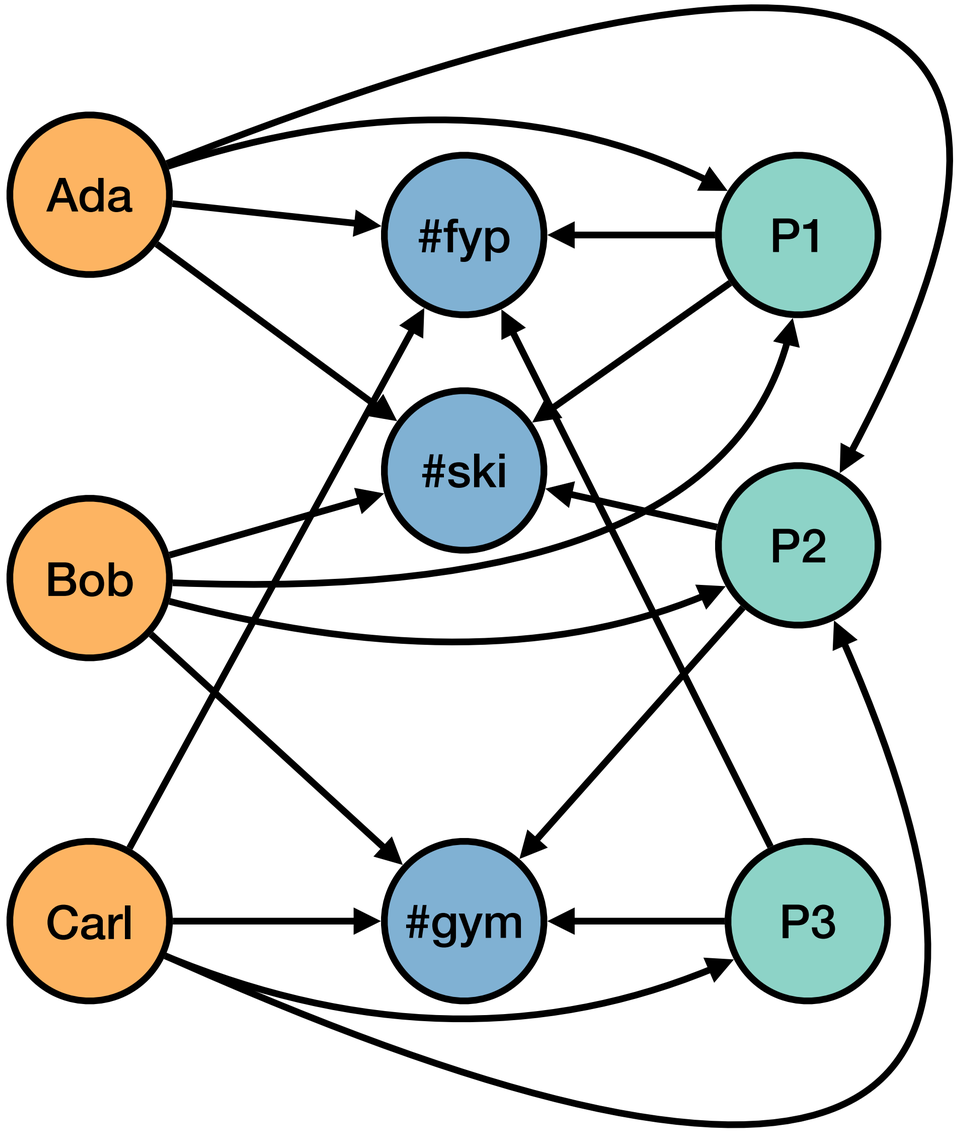
likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

∩

User	Tag
Ada	#fyp
Ada	#ski
Bob	#gym
Bob	#ski
Carl	#fyp
Carl	#gym

User	Tag	Post
Ada	#fyp	P1
Ada	#ski	P1
Ada	#ski	P2
Bob	#gym	P2
Bob	#ski	P1
Bob	#ski	P2



follows

User	Tag
Ada	#fyp
Ada	#ski
Bob	#ski
Bob	#gym
Carl	#fyp
Carl	#gym

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

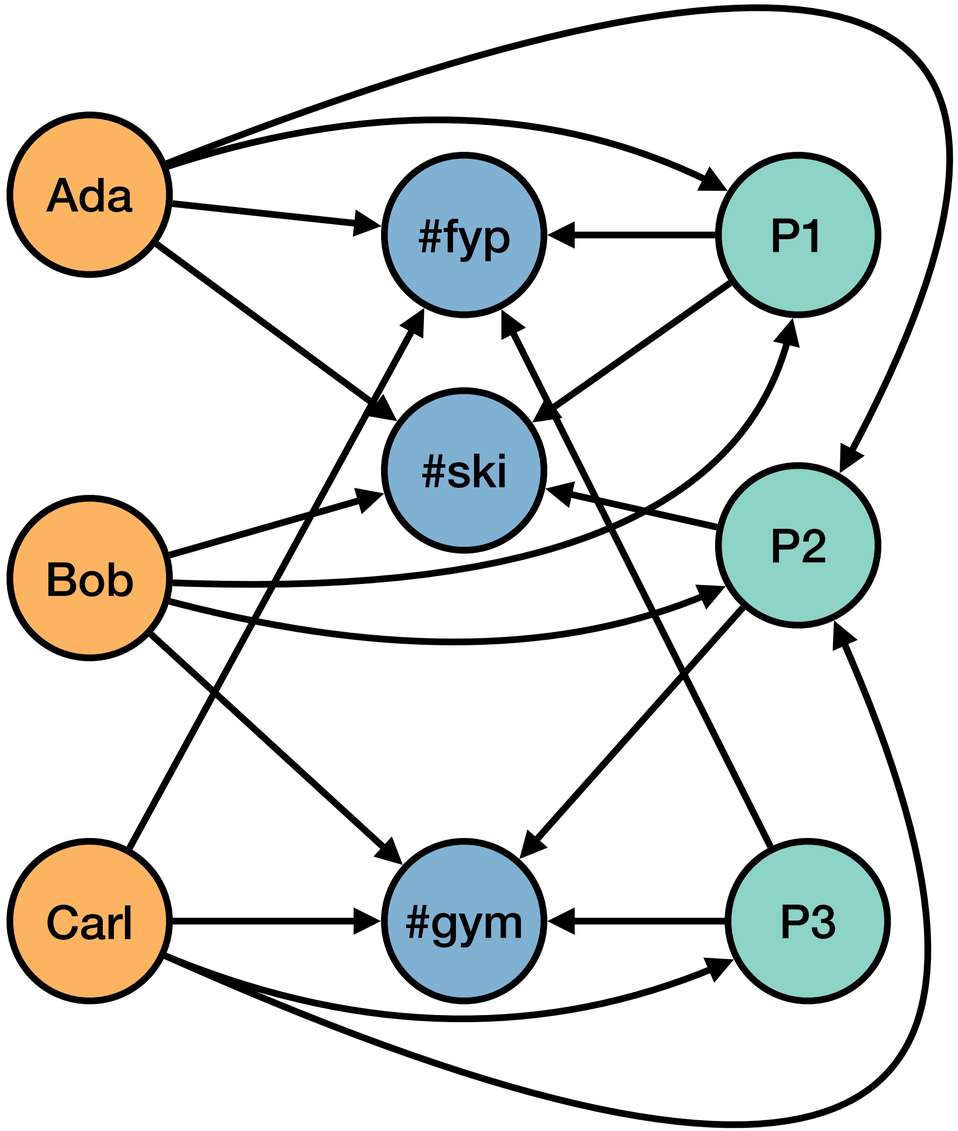
likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

∩

User	Tag
Ada	#fyp
Ada	#ski
Bob	#gym
Bob	#ski
Carl	#fyp
Carl	#gym

User	Tag	Post
Ada	#fyp	P1
Ada	#ski	P1
Ada	#ski	P2
Bob	#gym	P2
Bob	#ski	P1
Bob	#ski	P2
Carl	#fyp	P3



follows

User	Tag
Ada	#fyp
Ada	#ski
Bob	#ski
Bob	#gym
Carl	#fyp
Carl	#gym

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

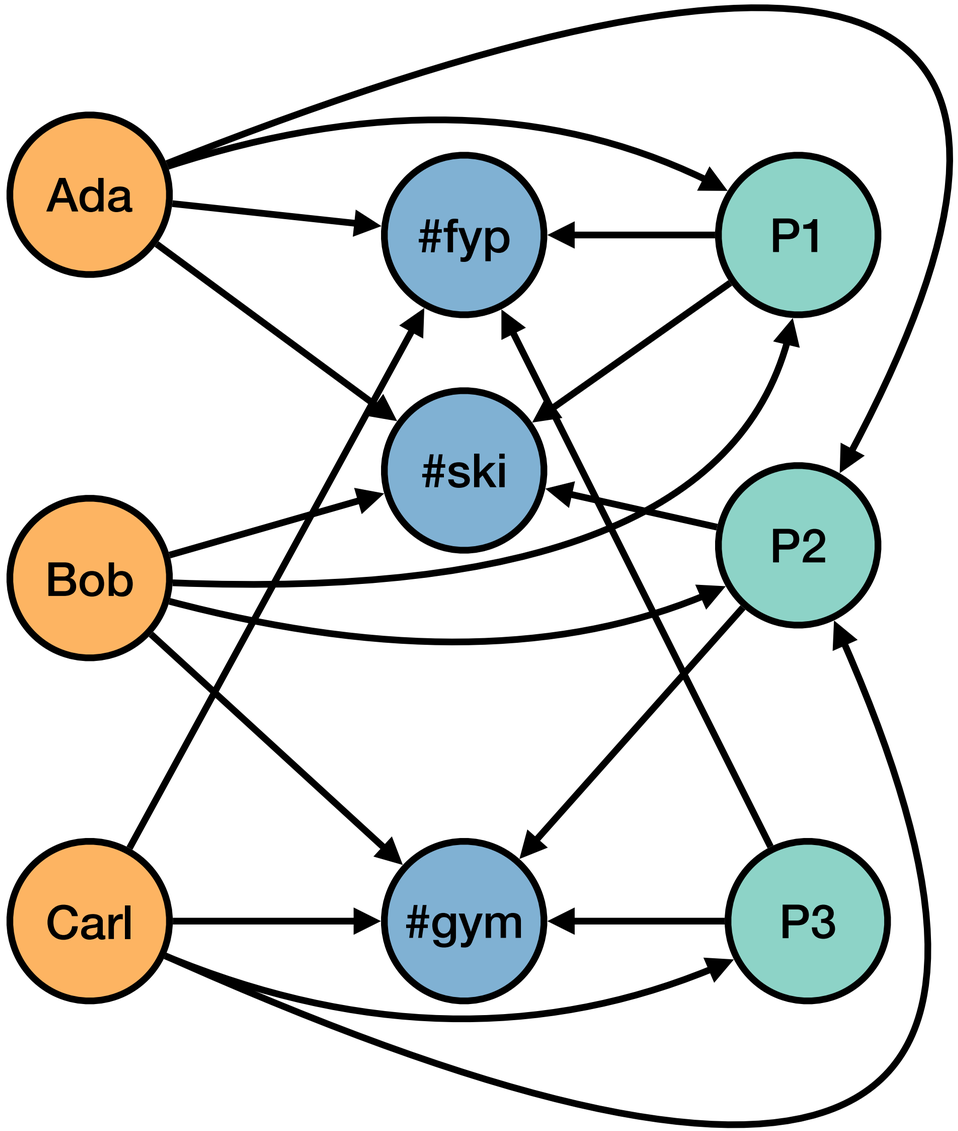
likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

∩

User	Tag
Ada	#fyp
Ada	#ski
Bob	#gym
Bob	#ski
Carl	#fyp
Carl	#gym

User	Tag	Post
Ada	#fyp	P1
Ada	#ski	P1
Ada	#ski	P2
Bob	#gym	P2
Bob	#ski	P1
Bob	#ski	P2
Carl	#fyp	P3
Carl	#gym	P2
Carl	#gym	P3



follows

User	Tag
Ada	#fyp
Ada	#ski
Bob	#ski
Bob	#gym
Carl	#fyp
Carl	#gym

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

User	Tag	Post
Ada	#fyp	P1
Ada	#ski	P1
Ada	#ski	P2
Bob	#gym	P2
Bob	#ski	P1
Bob	#ski	P2
Carl	#fyp	P3
Carl	#gym	P2
Carl	#gym	P3

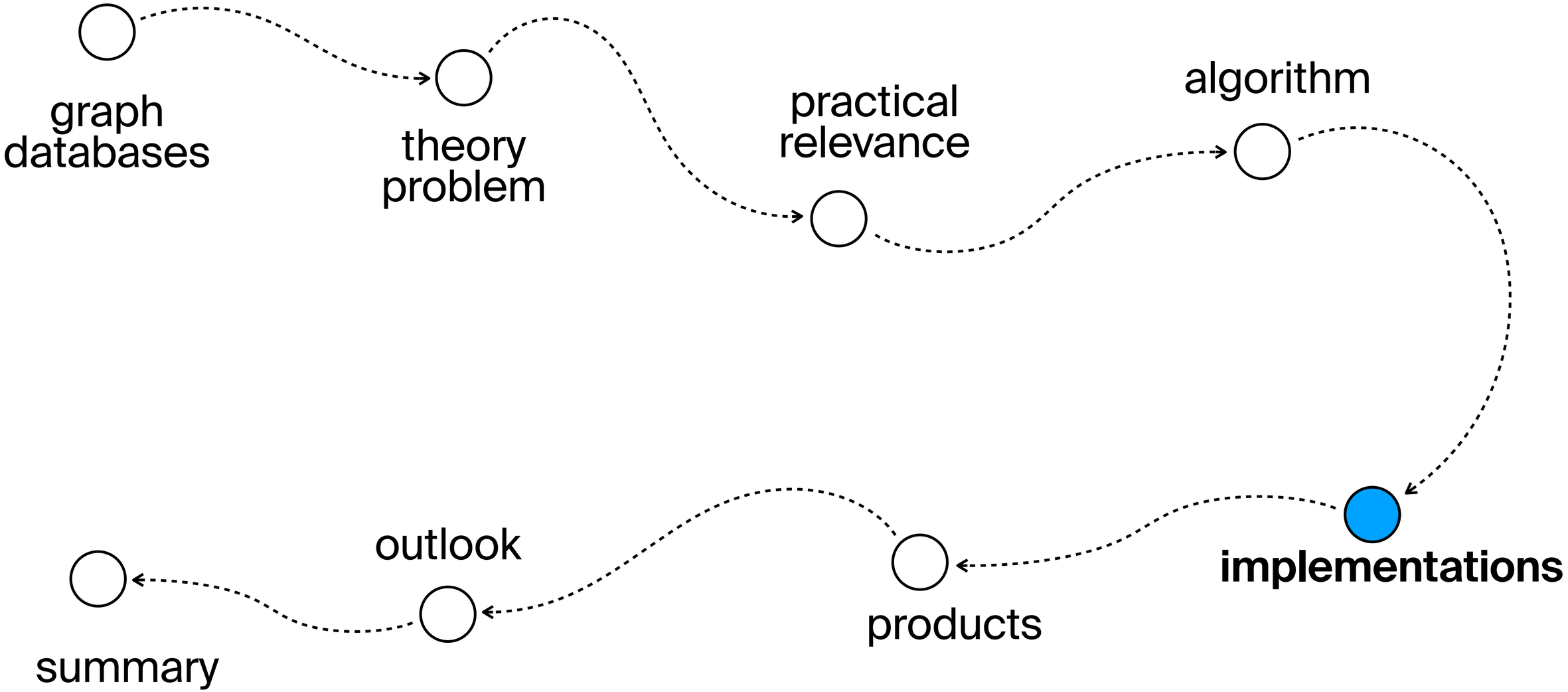
Very Large Databases 2019

Optimizing Subgraph Queries by Combining Binary and Worst-Case Optimal Joins

Amine Mhedhbi
University of Waterloo
amine.mhedhbi@uwaterloo.ca

Semih Salihoglu
University of Waterloo
semih.salihoglu@uwaterloo.ca

When to choose binary vs. multiway joins and how to **order them**?


graph
databases
theory
problem
practical
relevance
algorithm
implementations
products
outlook
summary

SIGMOD 2016

EmptyHeaded: A Relational Engine for Graph Processing

Christopher R. Aberger
Stanford University
caberger@stanford.edu

Susan Tu
Stanford University
sctu@stanford.edu

Kunle Olukotun
Stanford University
kunle@stanford.edu

Christopher Ré
Stanford University
chrismre@cs.stanford.edu

Very Large Databases 2020

Adopting Worst-Case Optimal Joins in Relational Database Systems

Michael Freitag, Maximilian Bandle, Tobias Schmidt, Alfons Kemper, Thomas Neumann
Technische Universität München
{freitagm, bandle, tobias.schmidt, kemper, neumann}@in.tum.de

Hash table-based implementation

3.2.4 *Complexity Analysis*

In the following, we present a formal investigation of the time and space complexity of the proposed hash trie join approach, proving in particular that its runtime is indeed worst-case optimal.

THEOREM 1. *The build phase of the proposed approach has time and space complexity in* $O(n \cdot \sum_{E_j \in \mathcal{E}} |R_j|)$.

PROOF. As outlined above, the same operations are performed for each input relation R_j during the build phase, hence we focus on a given R_j in the following. The initial materialization of R_j in a linked list clearly requires time and space proportional to $|R_j|$. Moving on to Algorithm 2, we note that each tuple in the input linked list L is moved to exactly one of the linked lists that are processed recursively. That is, no additional space is required for tuple storage, and the overall set of tuples that is processed in each recursive step of Algorithm 2 is some partition of R_j. As there are at most n join attributes in a relation, we obtain a total time and space complexity of $O(n \cdot |R_j|)$ for the build phase of a single relation R_j. □

3.2.4 *Complexity Analysis*

In the following, we present a formal investigation of the time and space complexity of the proposed hash trie join approach, proving in particular that its runtime is indeed worst-case optimal.

THEOREM 1. *The build phase of the proposed approach has time and space complexity in* $O(n \cdot \sum_{E_j \in \mathcal{E}} |R_j|)$.

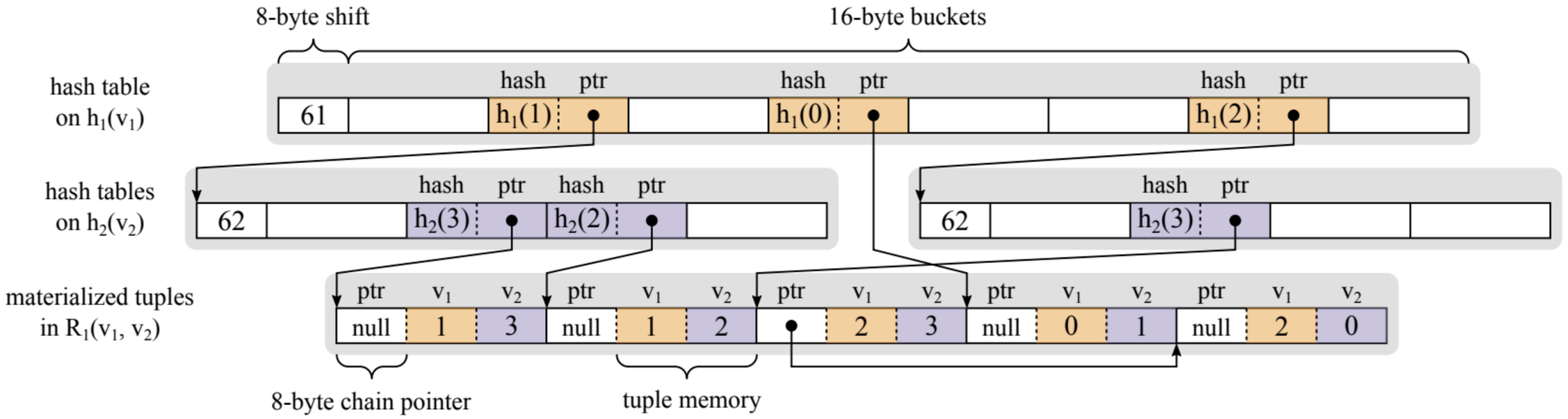
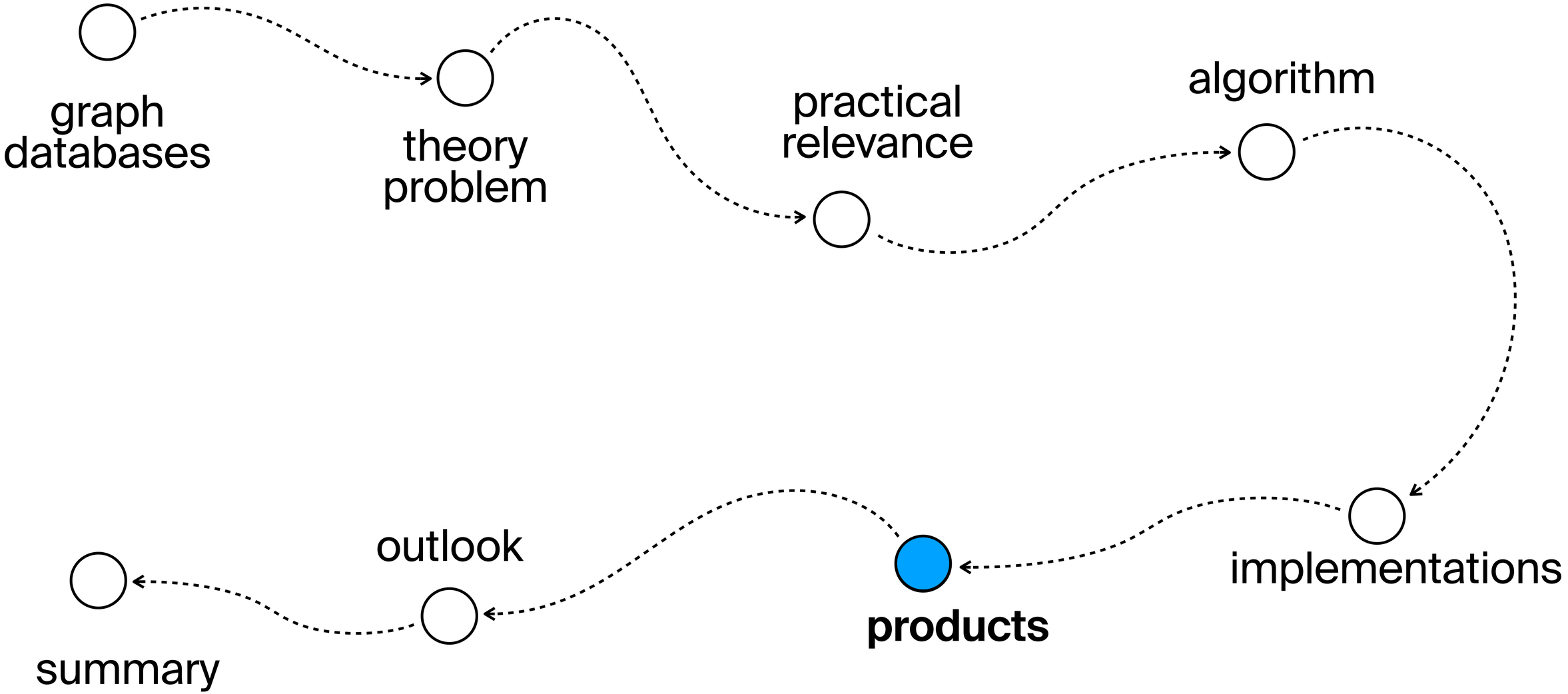
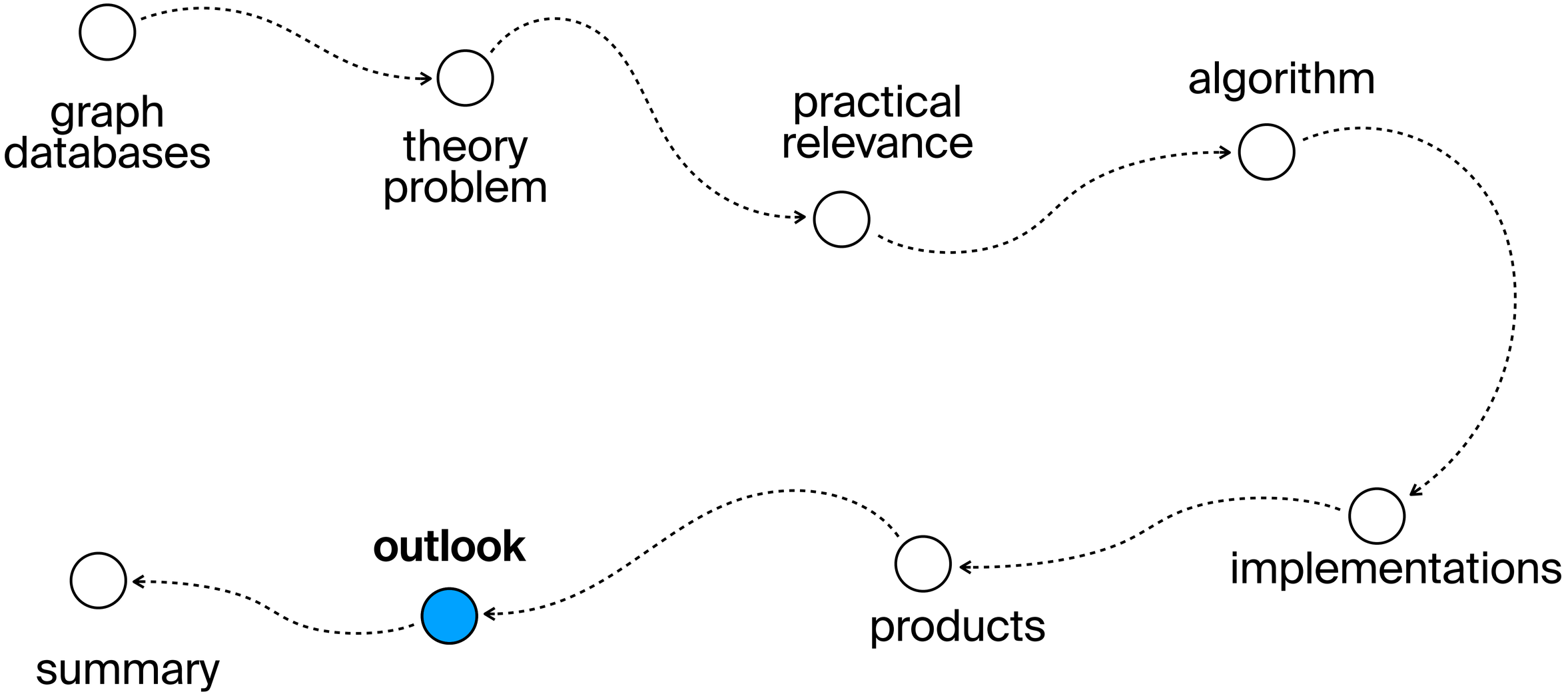
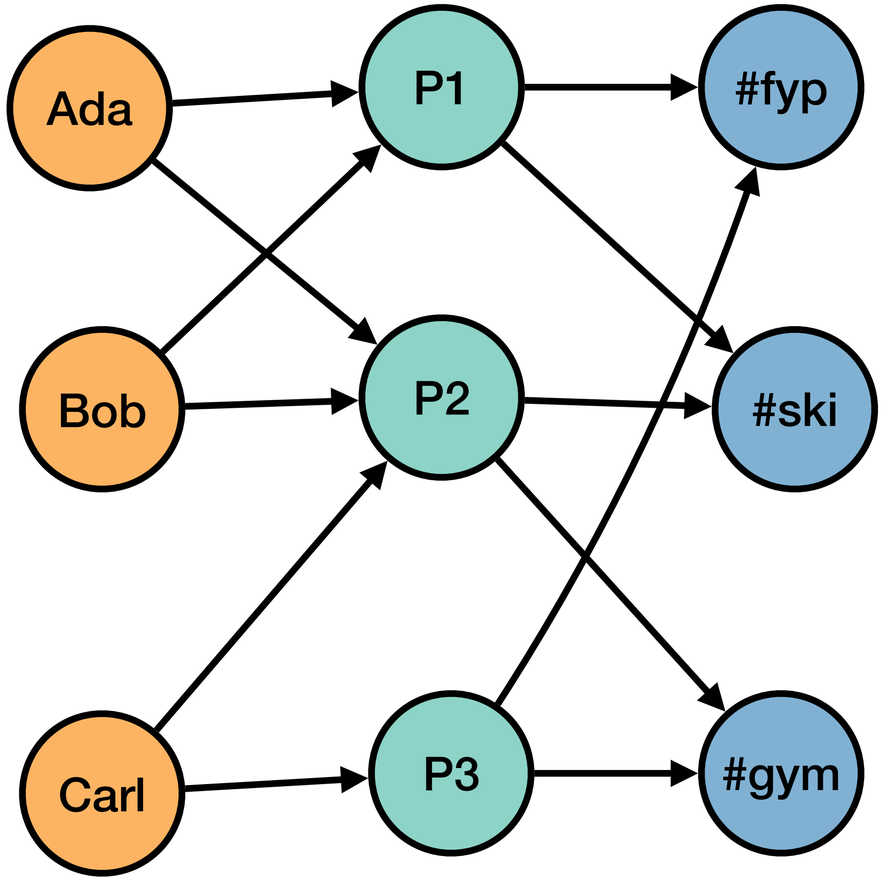


Figure 3: Memory layout of the hash trie in Figure 2. The gray boxes correspond to the individual hash tables and materialized input tuples. No nested hash table is built for the tuple $(0, 1)$ due to singleton pruning.


graph databases
theory problem
practical relevance
algorithm
implementations
products
outlook
summary

Database systems that support multiway joins


graph databases
theory problem
practical relevance
algorithm
implementations
products
outlook
summary



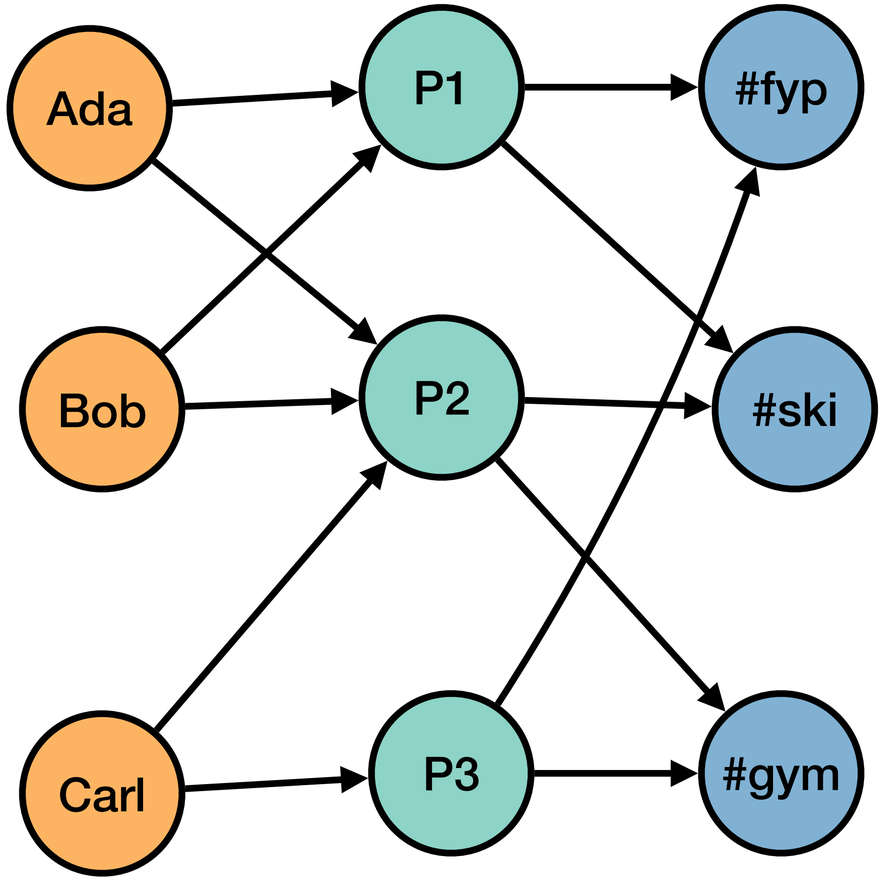

```
SELECT post
FROM likes
NATURAL JOIN tagged
GROUP BY post
HAVING count(distinct user) > count(distinct tag);
```

likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym




```
SELECT post
FROM likes
NATURAL JOIN tagged
GROUP BY post
HAVING count(distinct user) > count(distinct tag);
```

likes ⋈ tagged

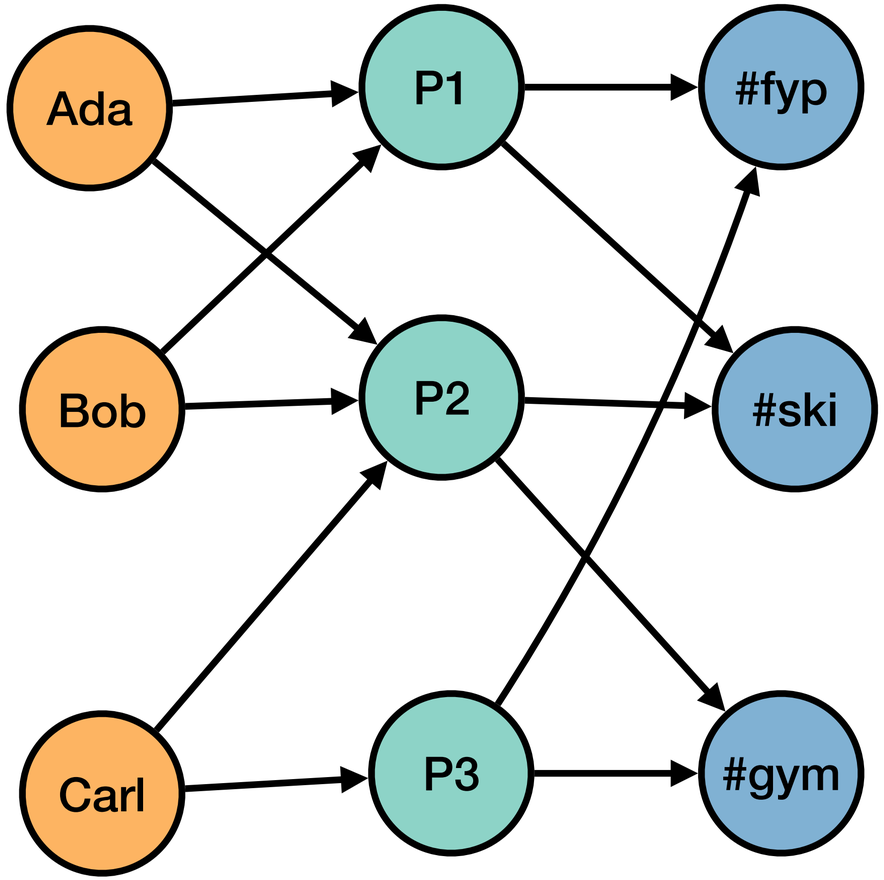
User	Post	Tag
Ada	P1	#ski
Bob	P1	#ski
Ada	P1	#fyp
Bob	P1	#fyp
Ada	P2	#gym
Bob	P2	#gym
Carl	P2	#gym
Ada	P2	#ski
Bob	P2	#ski
Carl	P2	#ski
Carl	P3	#gym
Carl	P3	#fyp

likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym




```
SELECT post
FROM likes
NATURAL JOIN tagged
GROUP BY post
HAVING count(distinct user) > count(distinct tag);
```

likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

tagged

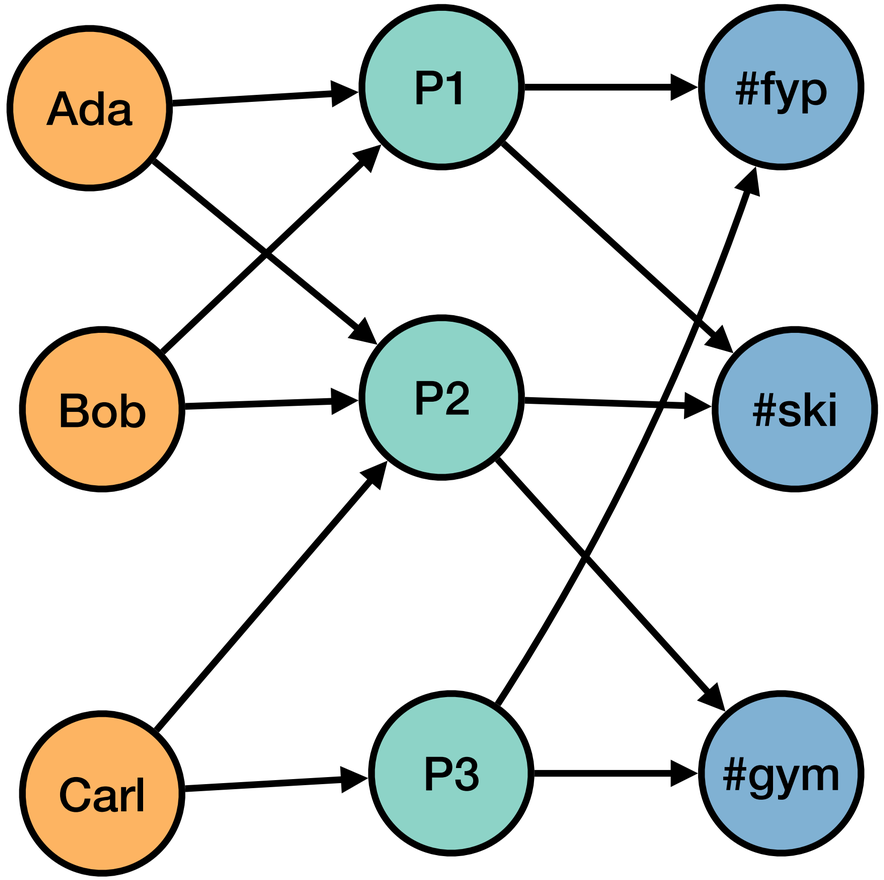
Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

likes ⋈ tagged

User	Post	Tag
Ada	P1	#ski
Bob	P1	#ski
Ada	P1	#fyp
Bob	P1	#fyp

count[user, tag] (likes ⋈ tagged)

Post	num_users	num_tags
P1	2	2
P2	3	2
P3	1	2




```
SELECT post
FROM likes
NATURAL JOIN tagged
GROUP BY post
HAVING count(distinct user) > count(distinct tag);
```

likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

likes ⋈ tagged

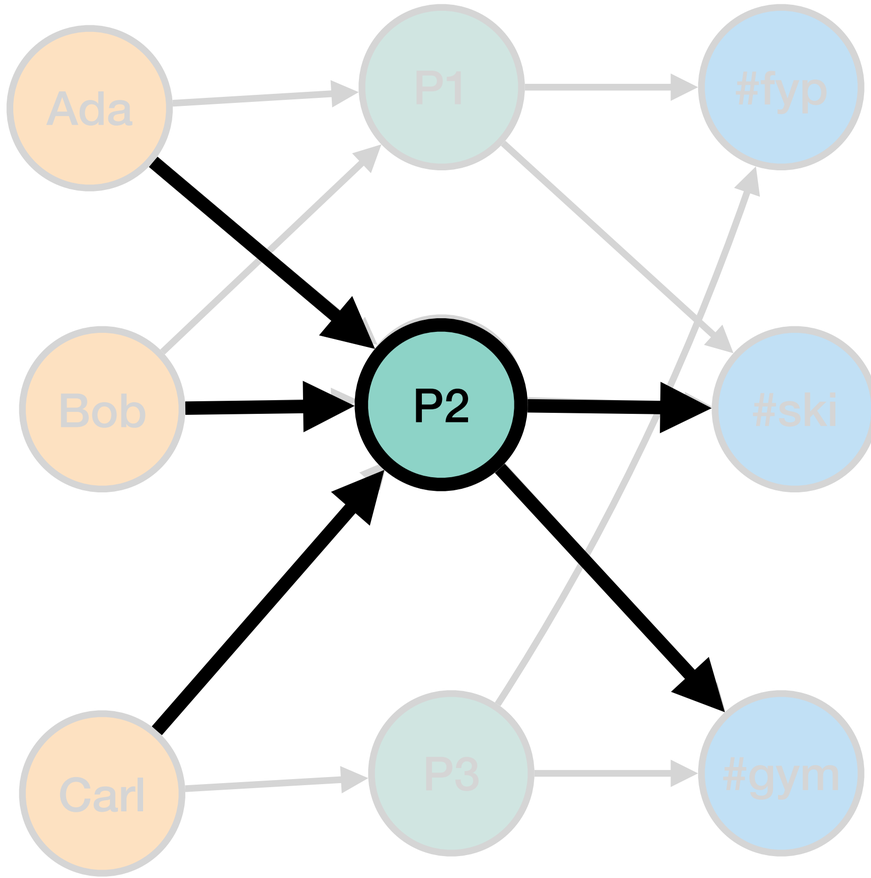
User	Post	Tag
Ada	P1	#ski
Bob	P1	#ski
Ada	P1	#fyp
Bob	P1	#fyp

count[user, tag] (likes ⋈ tagged)

Post	num_users	num_tags
P1	2	2
P2	3	2
P3	1	2

result

Post
P2




```
SELECT post
FROM likes
NATURAL JOIN tagged
GROUP BY post
HAVING count(distinct user) > count(distinct tag);
```

likes ⋈ tagged

User	Post	Tag
Ada	P1	#ski
Bob	P1	#ski
Ada	P1	#fyp
Bob	P1	#fyp

count[user, tag] (likes ⋈ tagged)

Post	num_users	num_tags
P1	2	2
P2	3	2
P3	1	2

result

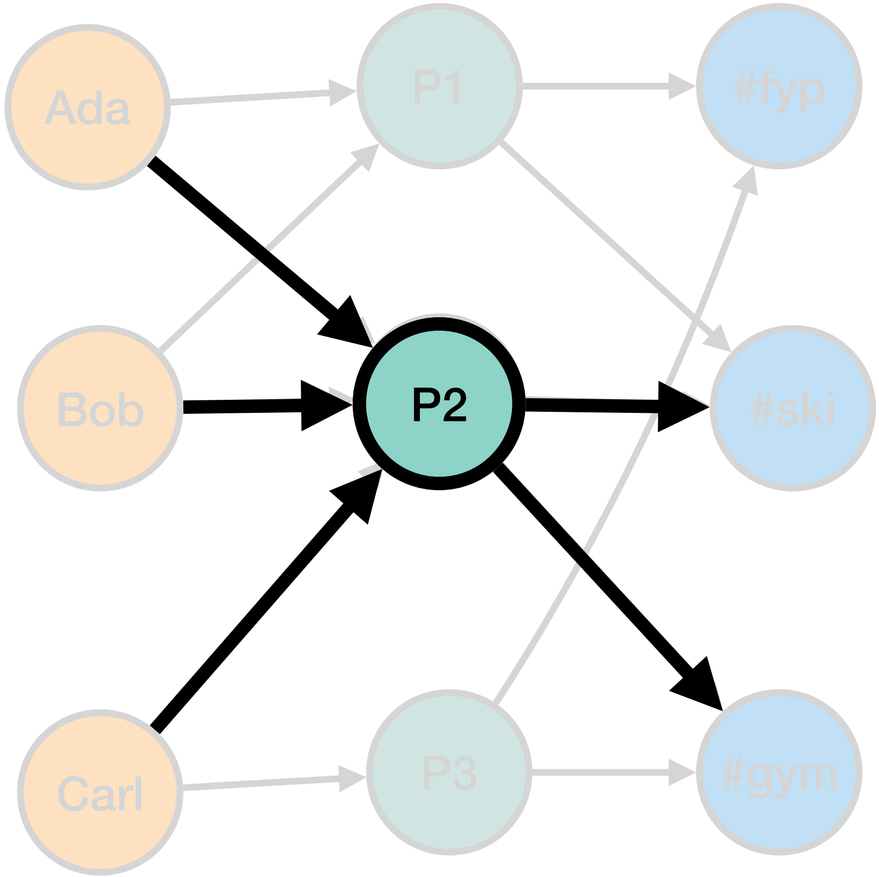
Post
P2

likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym




```
SELECT post
FROM likes
NATURAL JOIN tagged
GROUP BY post
HAVING count(distinct user) > count(distinct tag);
```

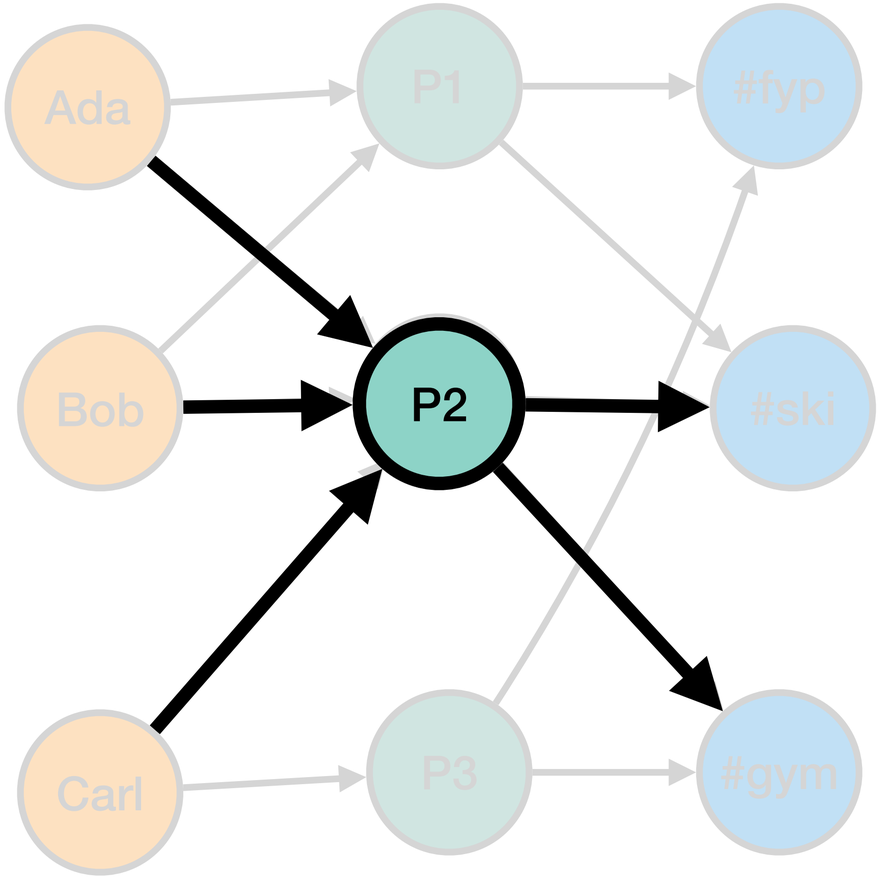
factorization: lossless compression

likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym




```
SELECT post
FROM likes
NATURAL JOIN tagged
GROUP BY post
HAVING count(distinct user) > count(distinct tag);
```

factorization: lossless compression

likes

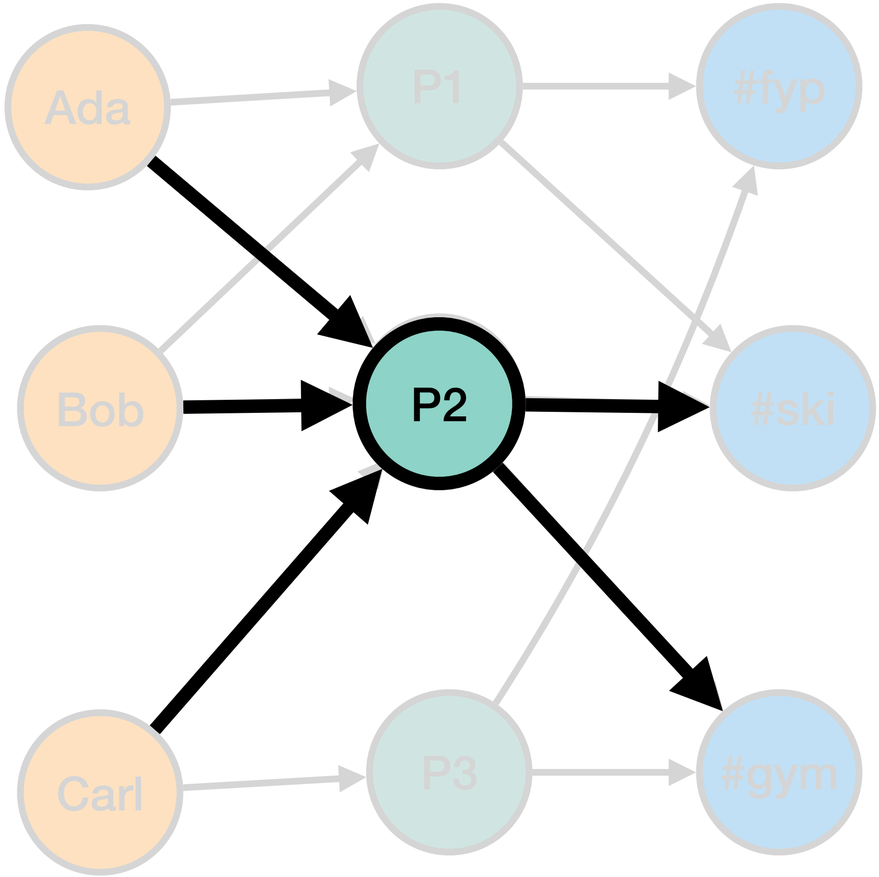
User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

tagged

Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

likes ⋈ tagged

User	Post	Tag
{Ada, Bob} {Ada, Bob, Carl} {Carl}	P1 P2 P3	{#ski, #fyp} {#gym, #ski} {#fyp, #gym}




```
SELECT post
FROM likes
NATURAL JOIN tagged
GROUP BY post
HAVING count(distinct user) > count(distinct tag);
```

factorization: lossless compression

likes

User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

tagged

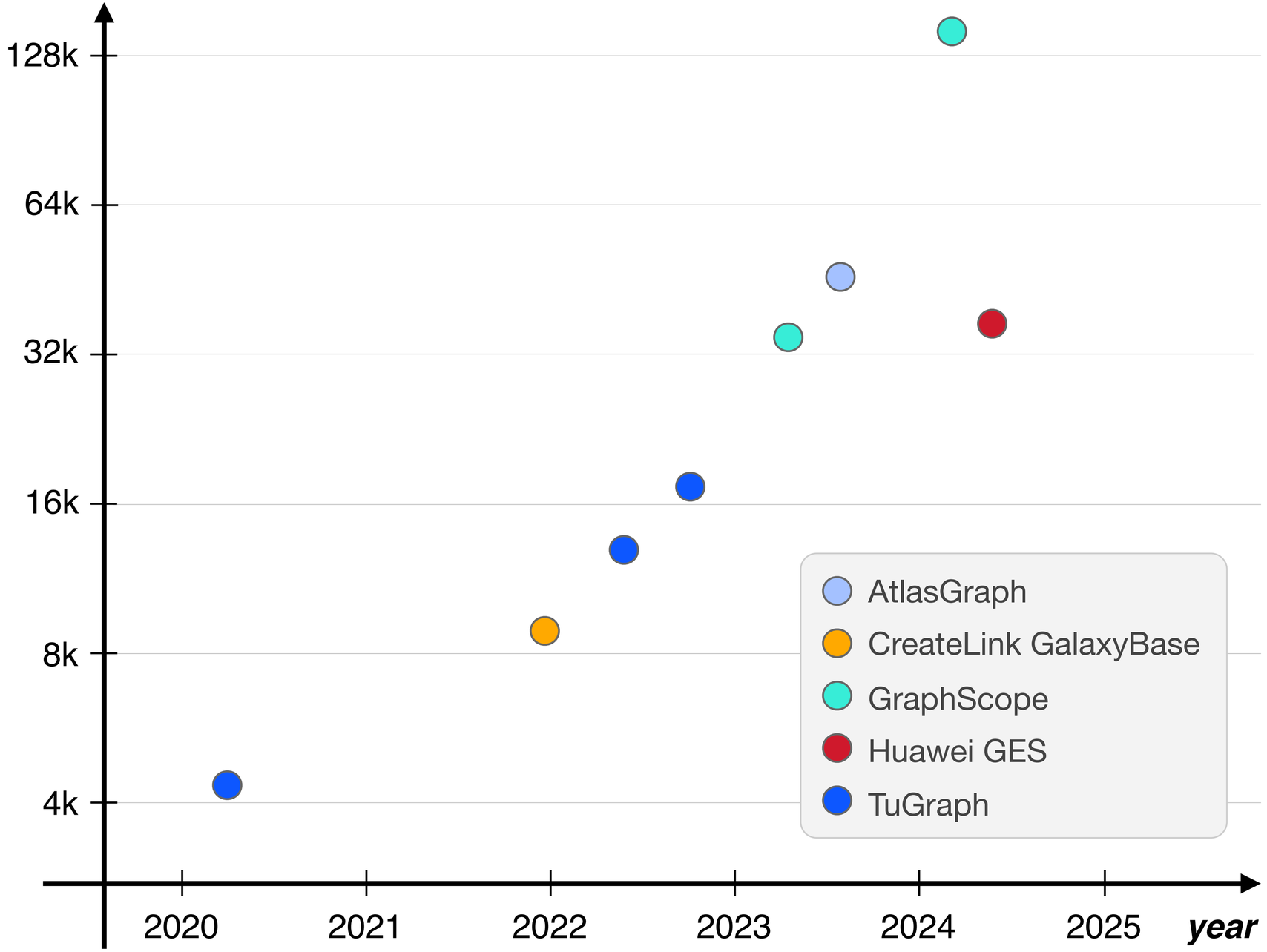
Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

likes ⋈ tagged

User	Post	Tag
{Ada, Bob}	P1	{#ski, #fyp}
{Ada, Bob, Carl}	P2	{#gym, #ski}
{Carl}	P3	{#fyp, #gym}

Industry

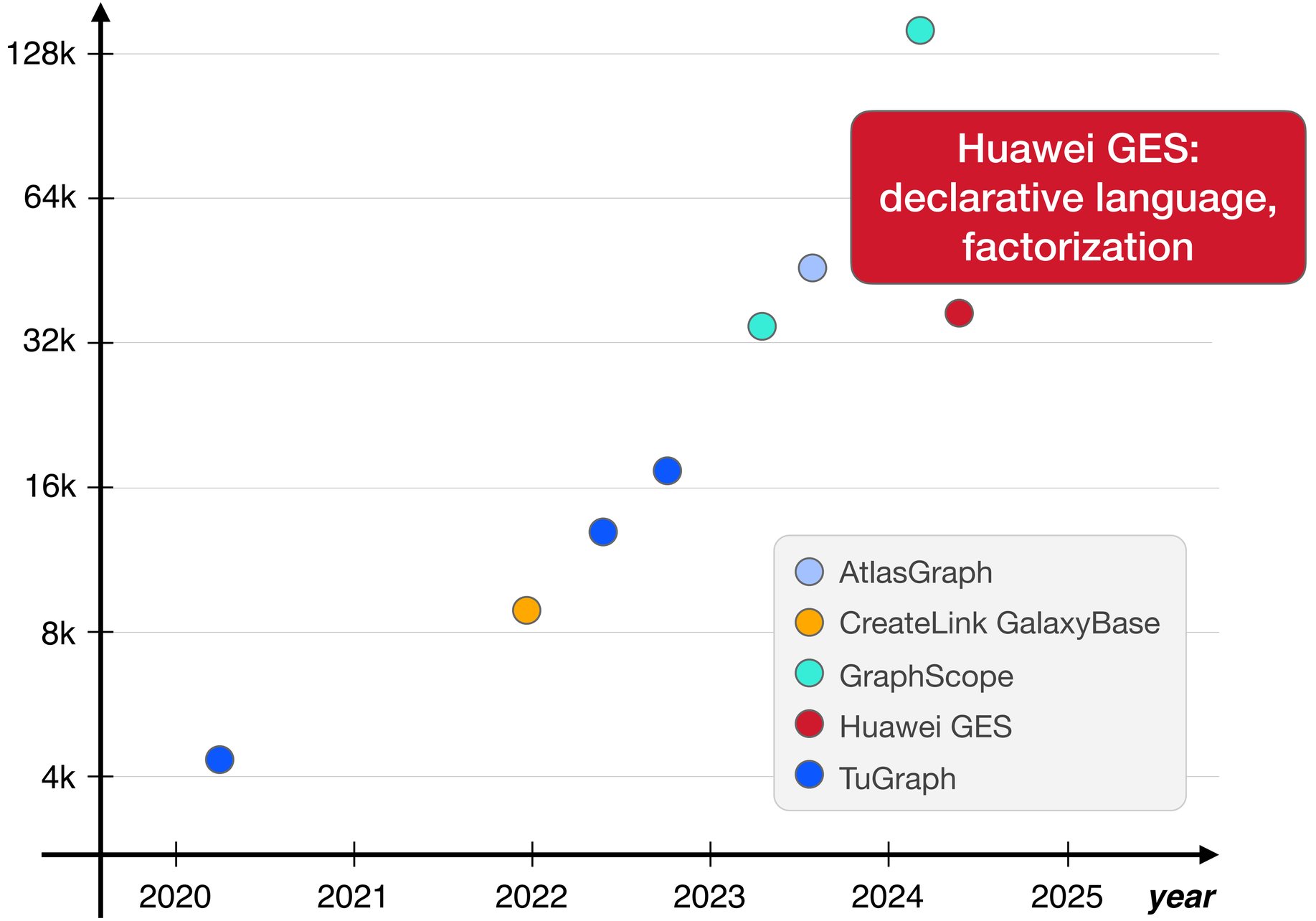
Competition is spurred by standardized benchmarks



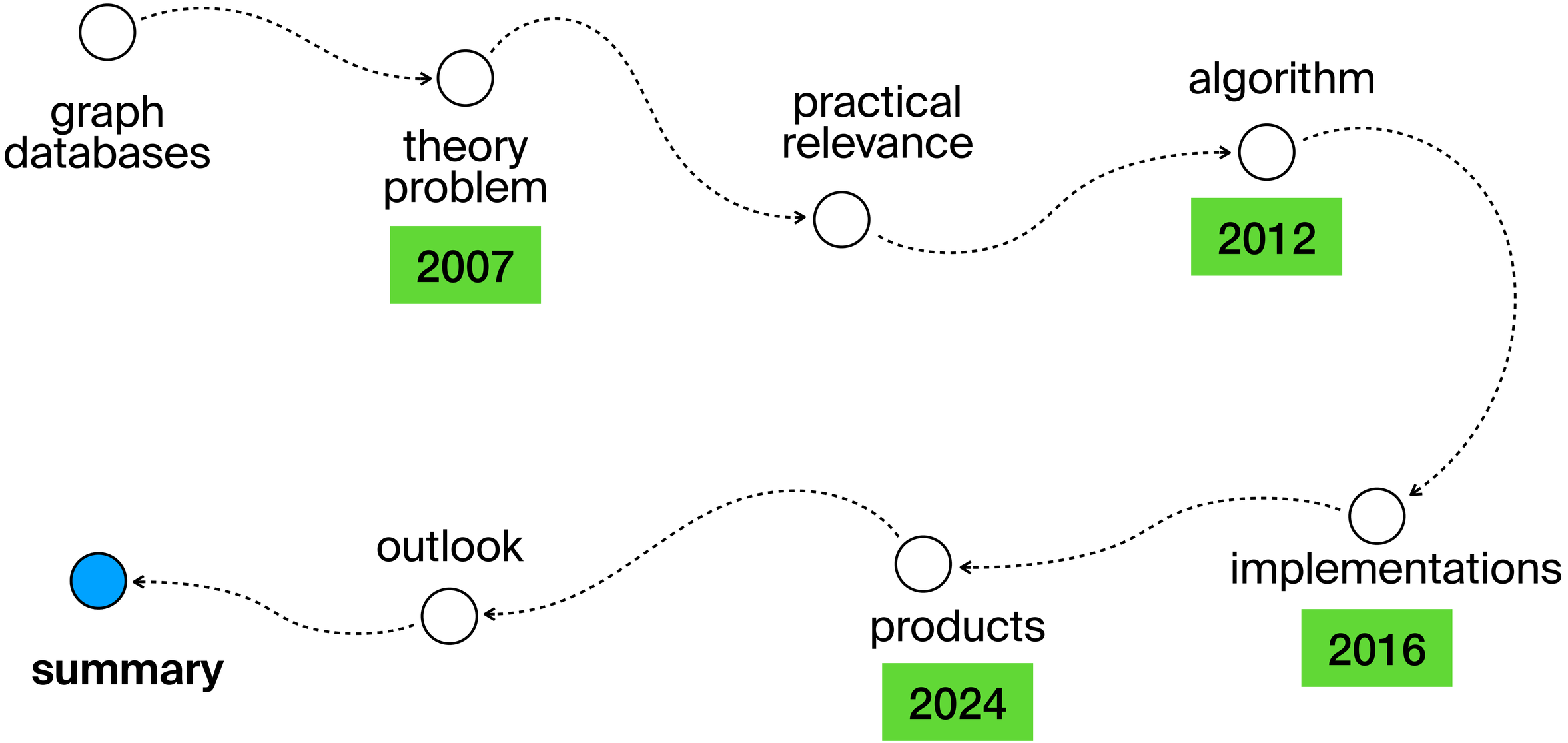
LDBC

Industry

Competition is spurred by standardized benchmarks



LDBC


graph databases
theory problem
2007
practical relevance
algorithm
2012
implementations
2016
products
2024
outlook
summary