Gráfadatbázisok:
Ahol az elmélet és a gyakorlat találkozik

Dr. Szárnyas Gábor

Simonyi Konferencia 2025

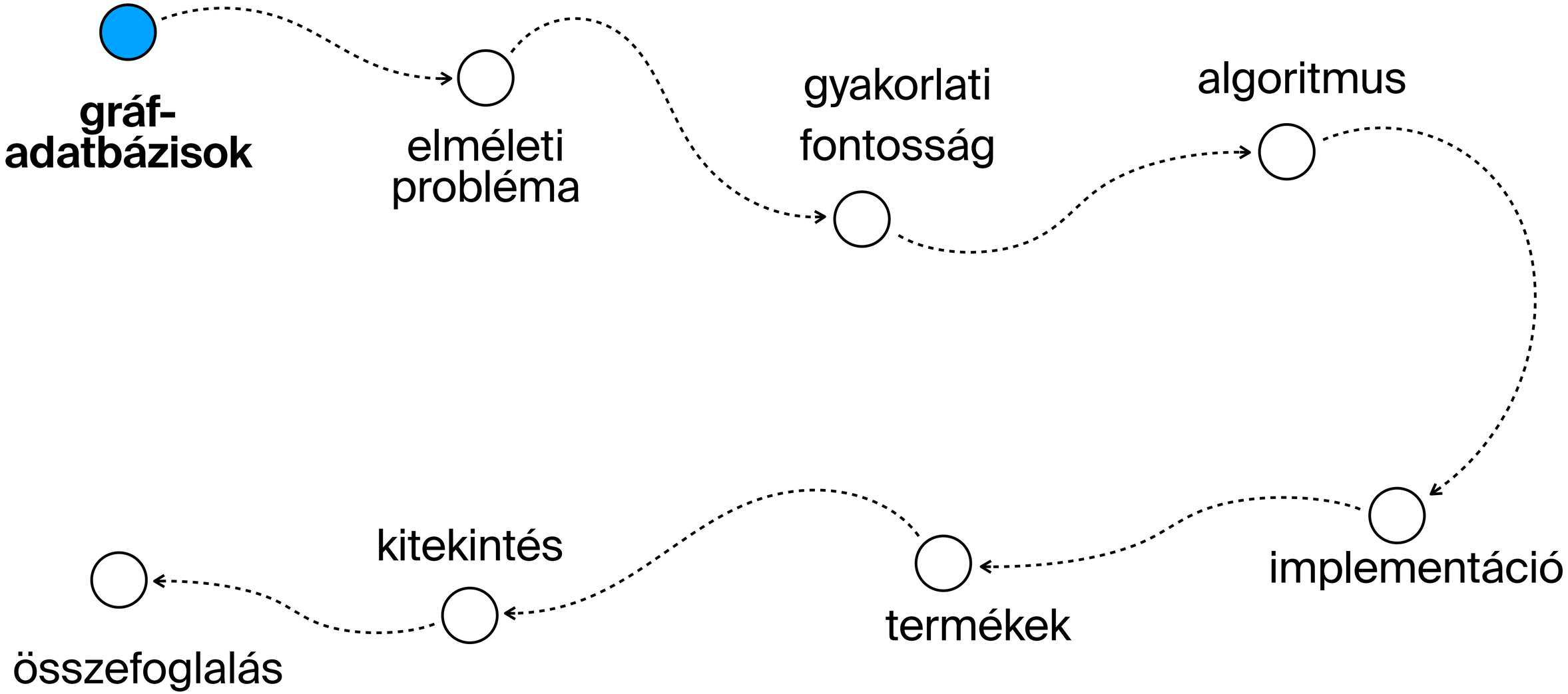
Karrierút

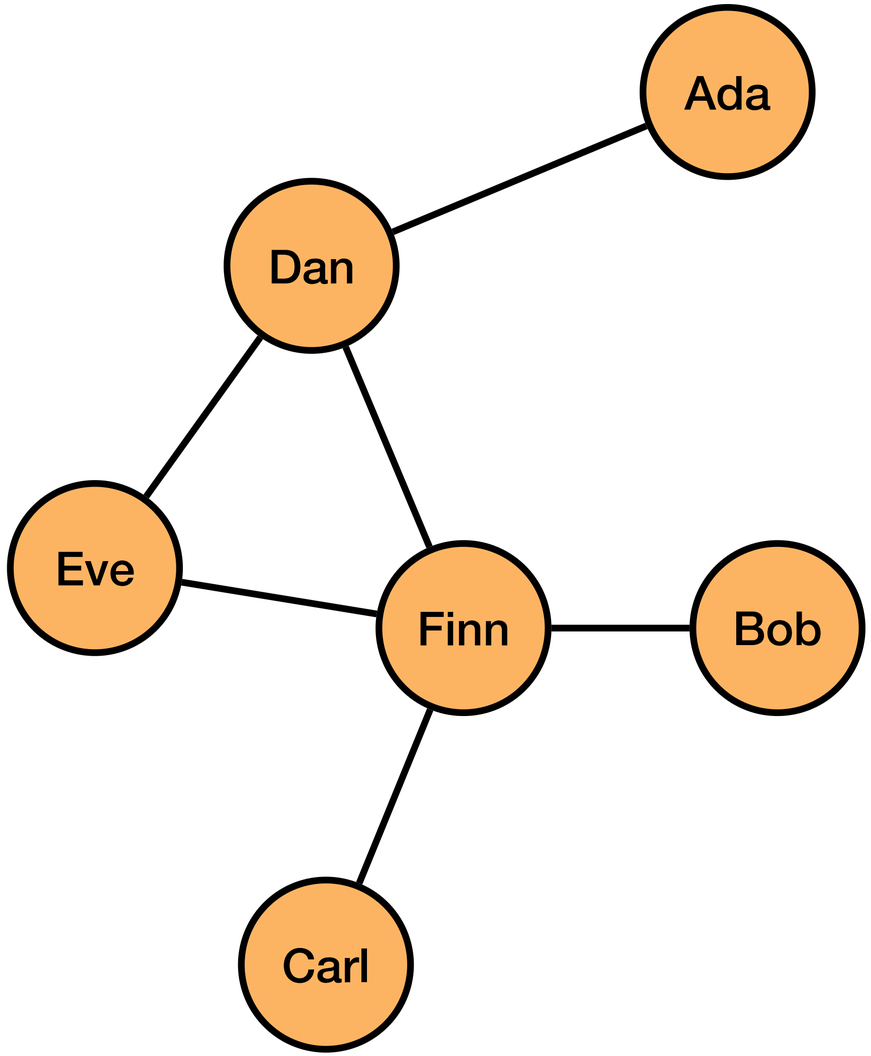
ftsrg

CWI

DuckDB Labs

2010 2013 2020 2023


gráf-
adatbázisok
elméleti
probléma
gyakorlati
fontosság
algoritmus
implementáció
termékek
kitekintés
összefoglalás



gráf

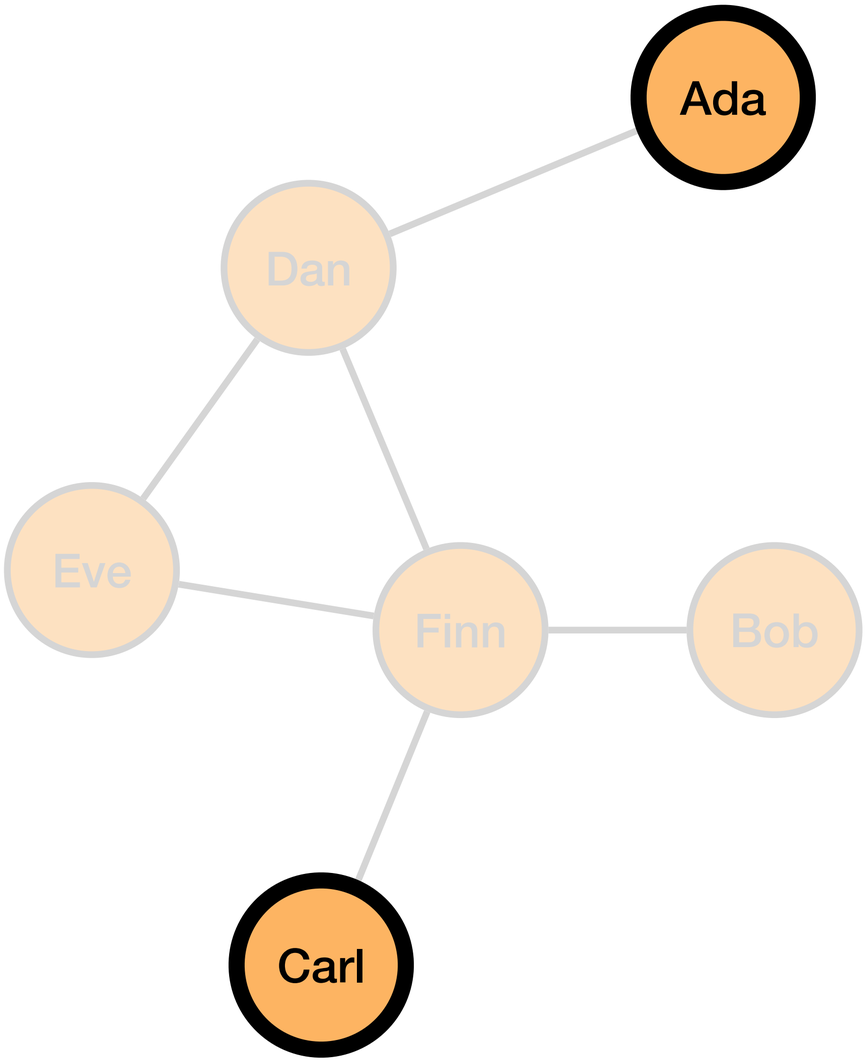
$$G = (V, E)$$

útvonalkeresés

q: két pont közötti legrövidebb út

BFS (szélességi keresés)

Dijkstra algoritmusa

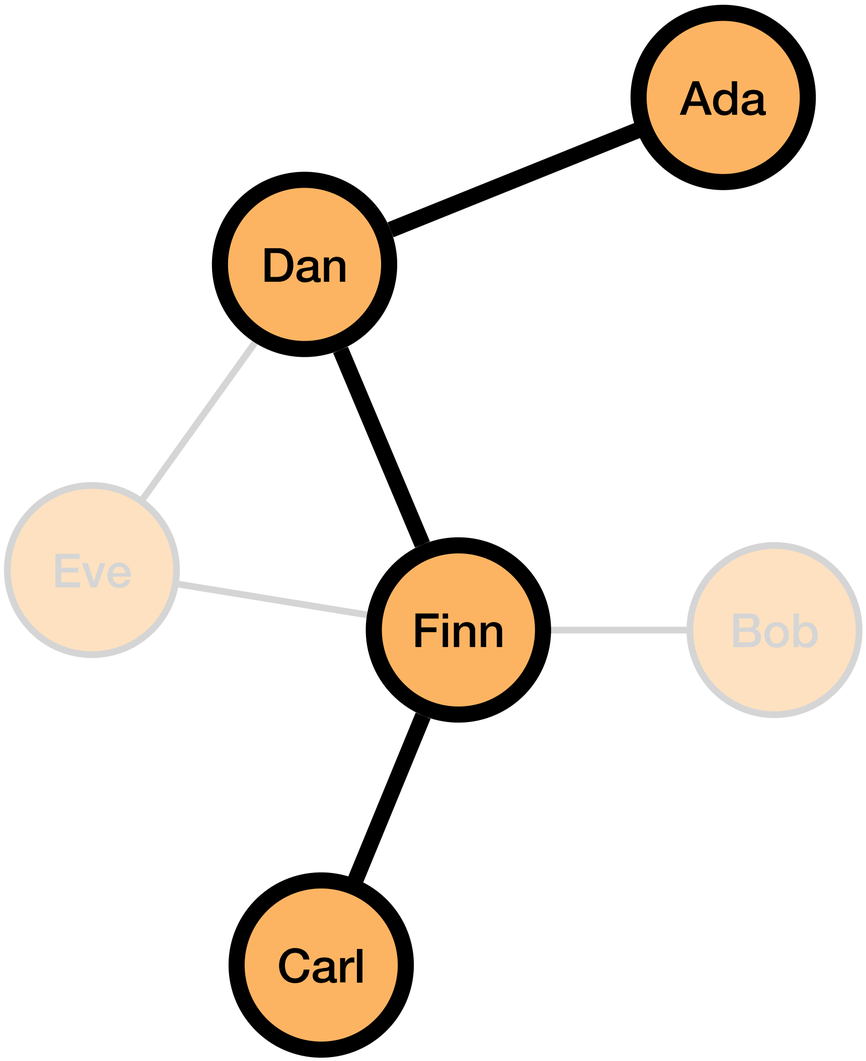


útvonalkeresés

q: két pont közötti legrövidebb út

BFS (szélességi keresés)

Dijkstra algoritmusa

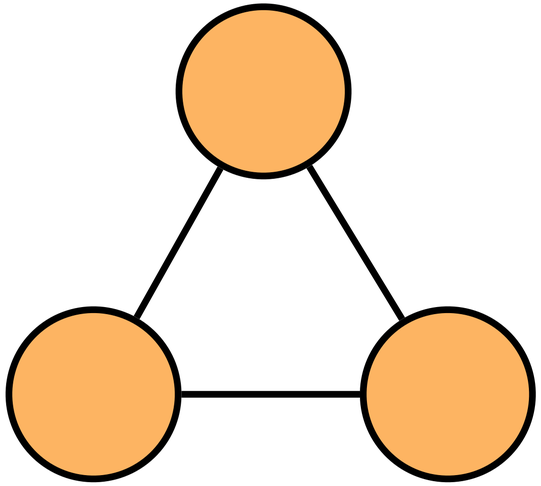


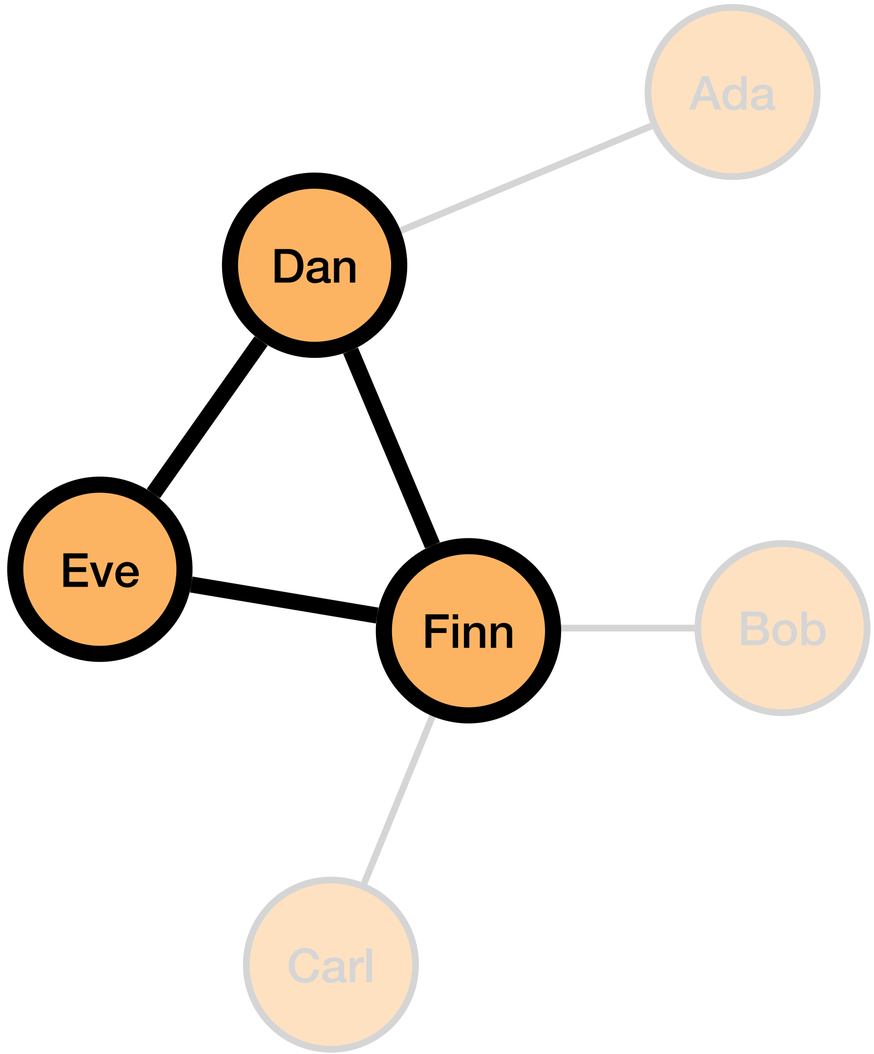
útvonalkeresés

q: két pont közötti legrövidebb út
BFS (szélességi keresés) Dijkstra algoritmusa

gráfmintaillesztés

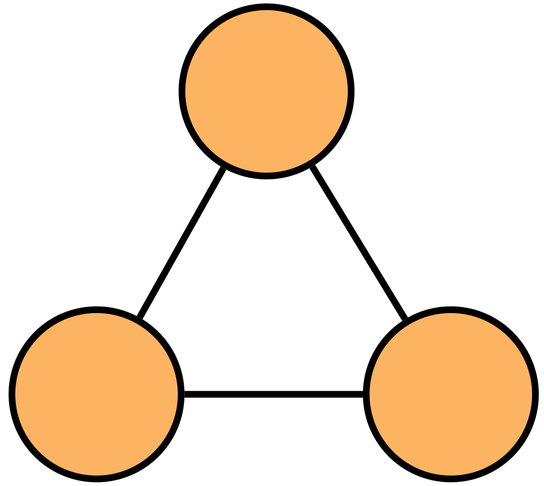
q: háromszög alakú részgráfok





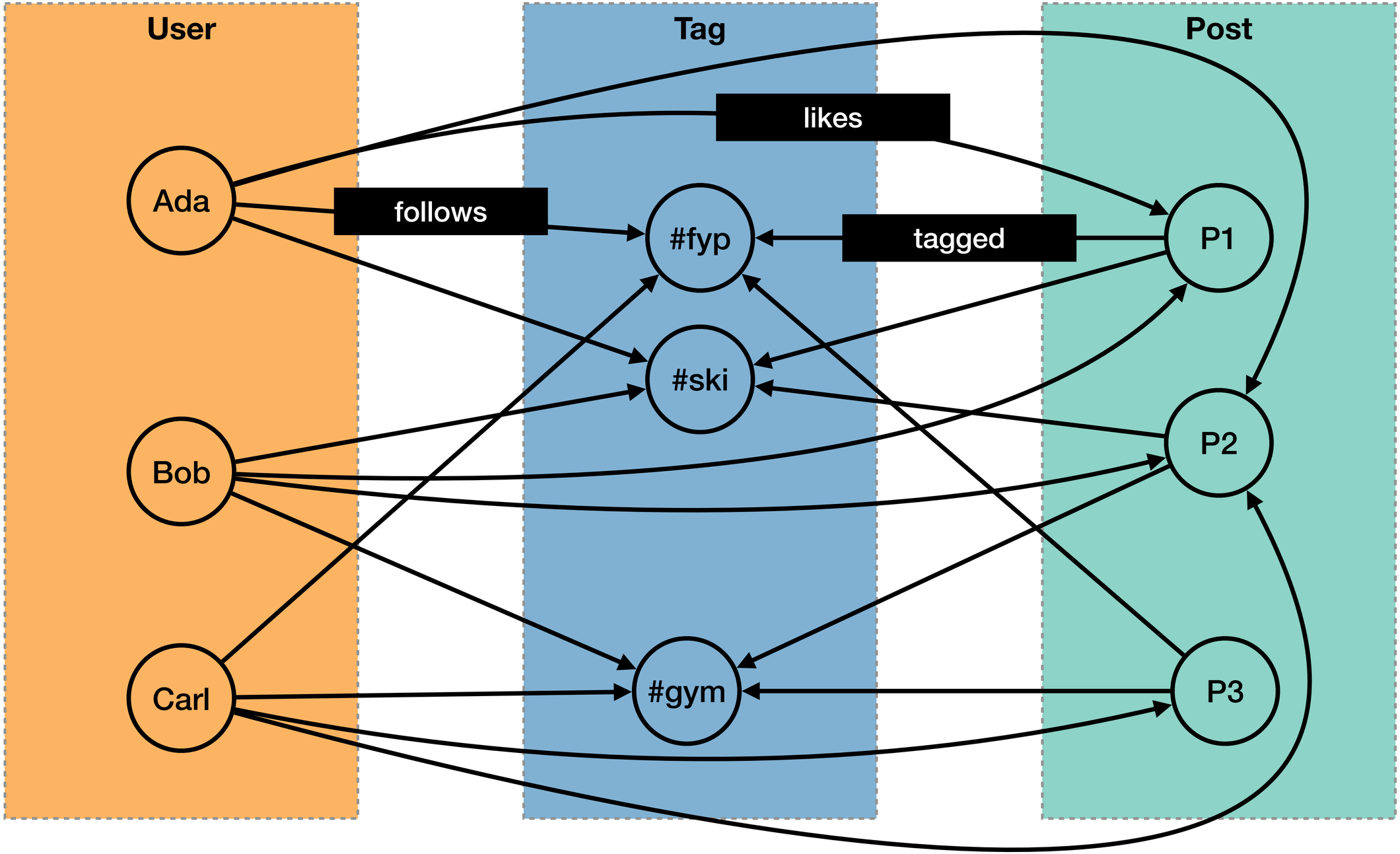
gráfmintaillesztés

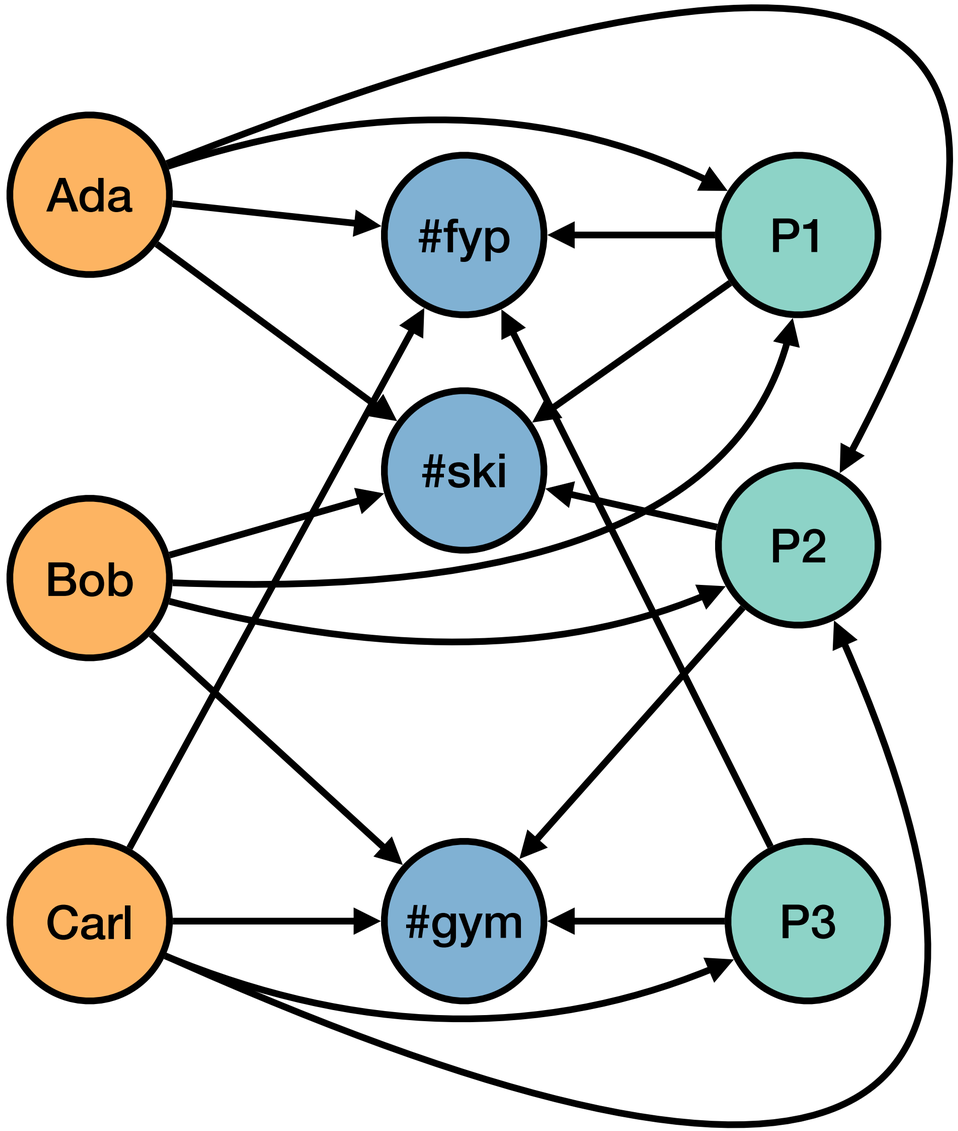
q: háromszög alakú részgráfok



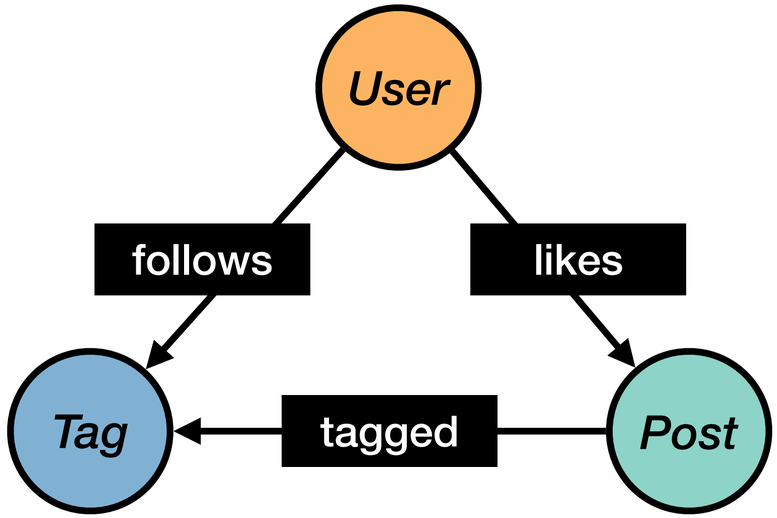
Közösségi háló

Közösségi háló

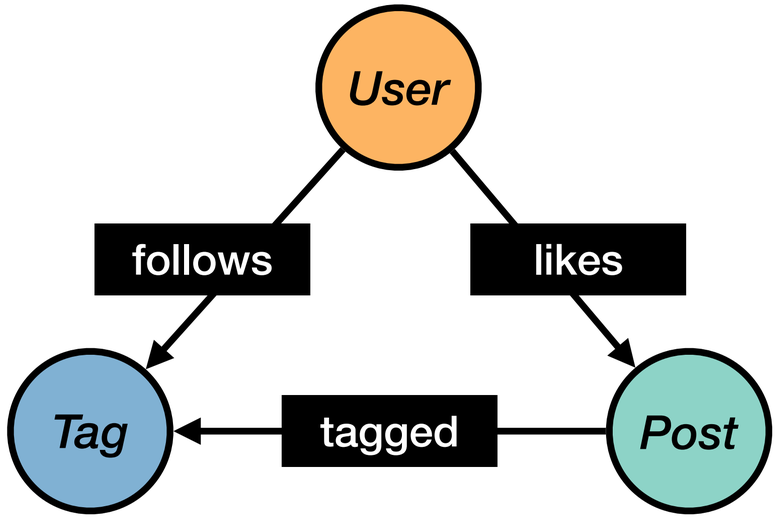

User
Tag
Post
likes
Ada
follows
#fyp
tagged
P1
#ski
Bob
P2
Carl
#gym
P3

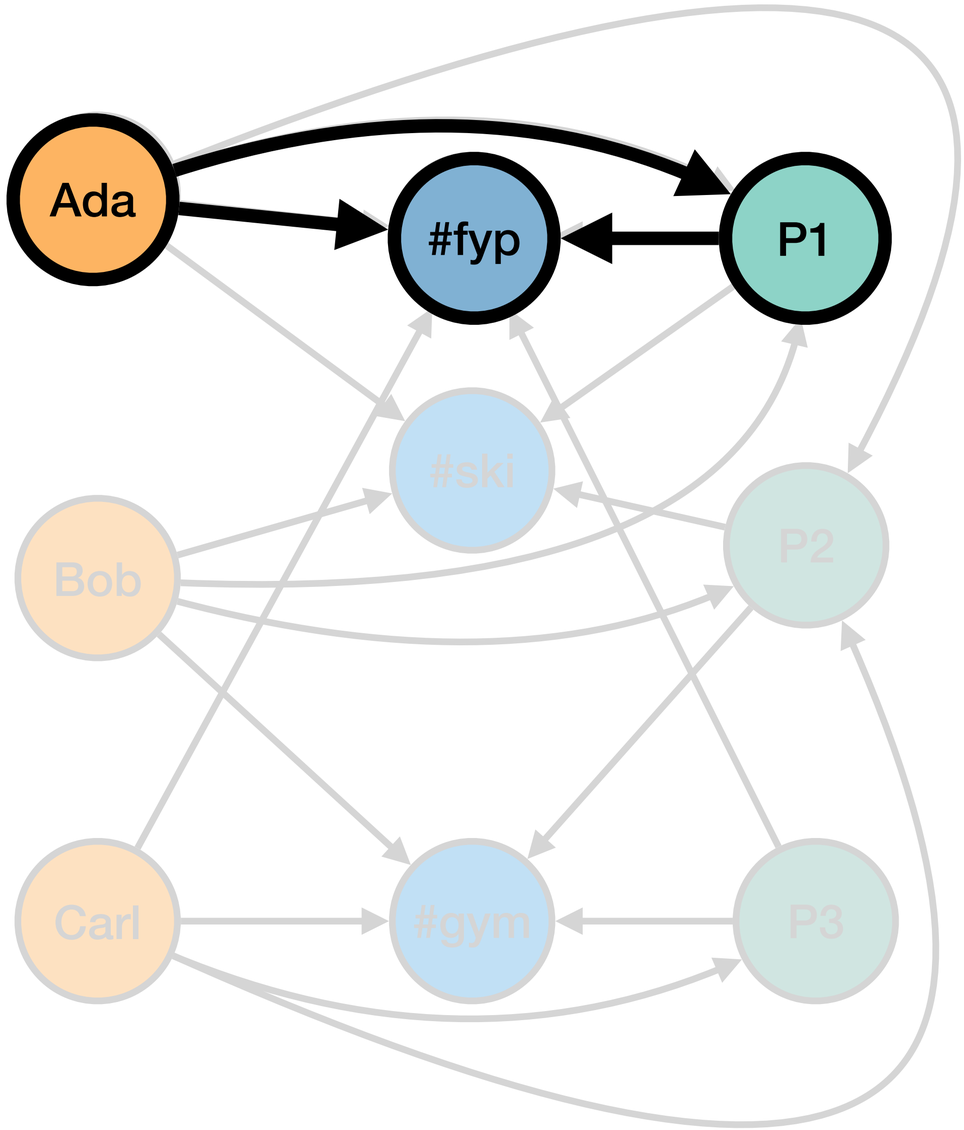


q: User-Tag-Post háromszögek

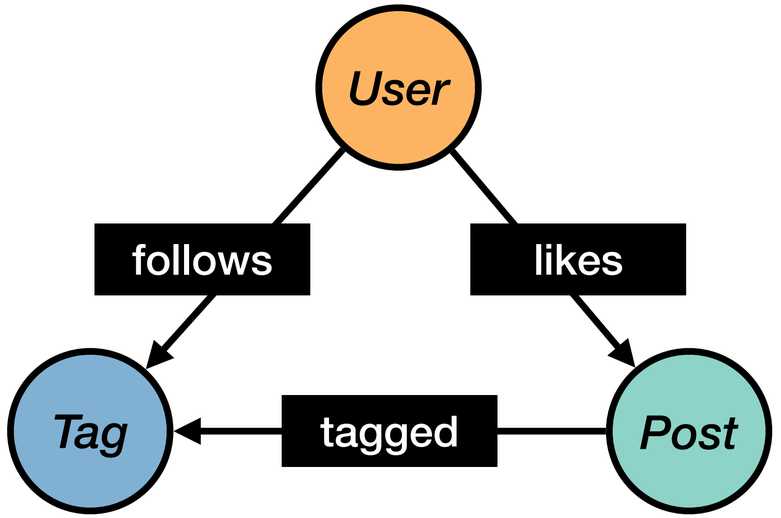


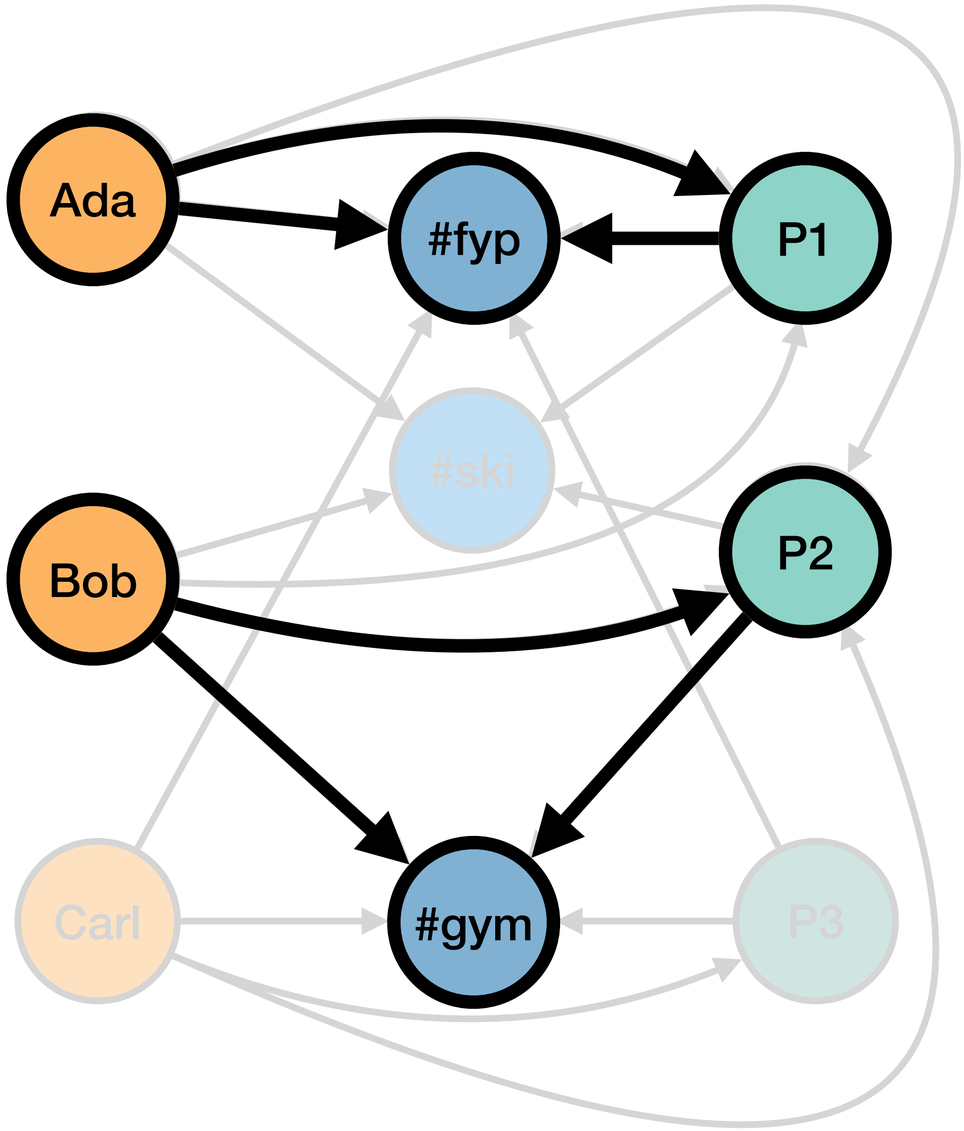
q: User-Tag-Post háromszögek



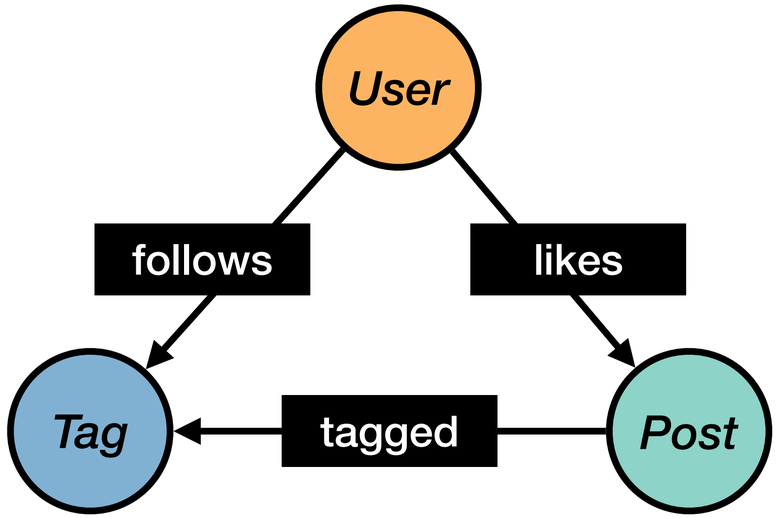


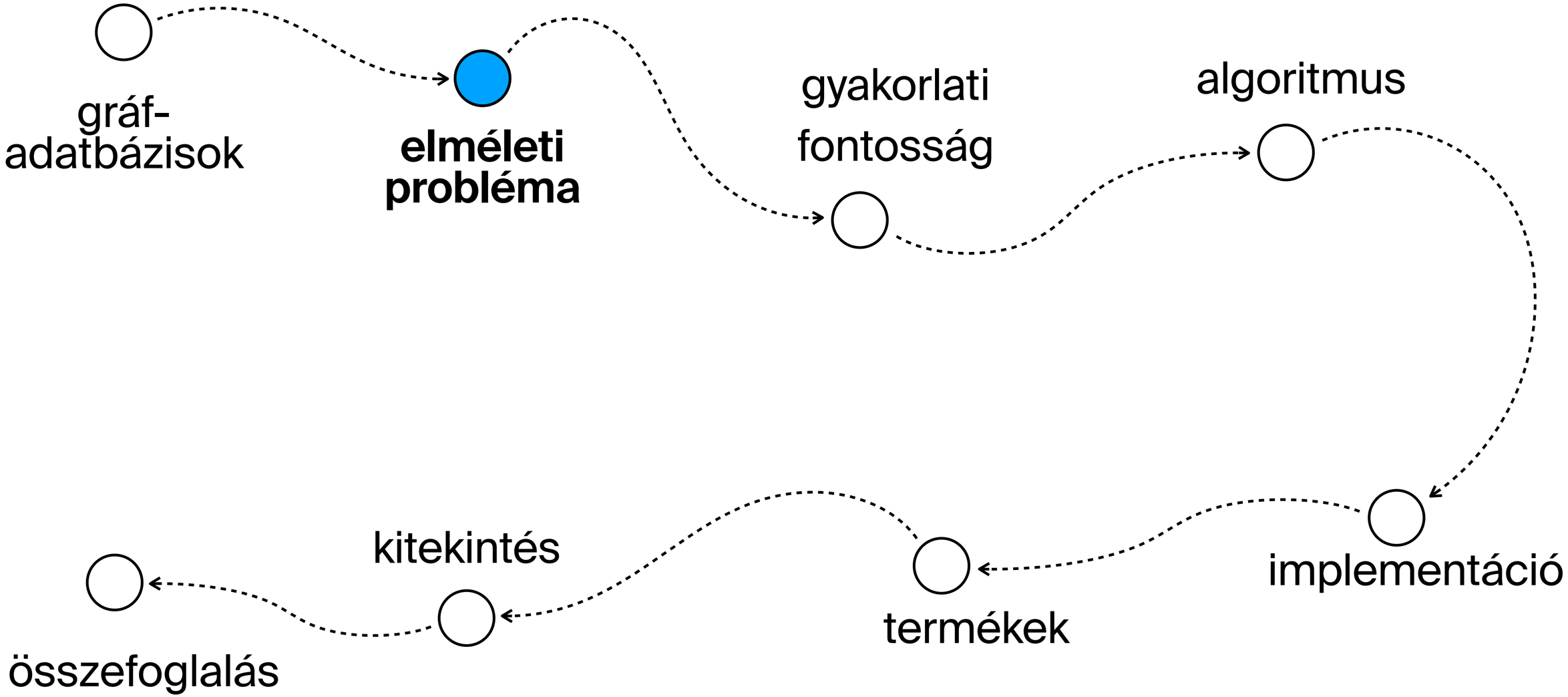
q: User-Tag-Post háromszögek

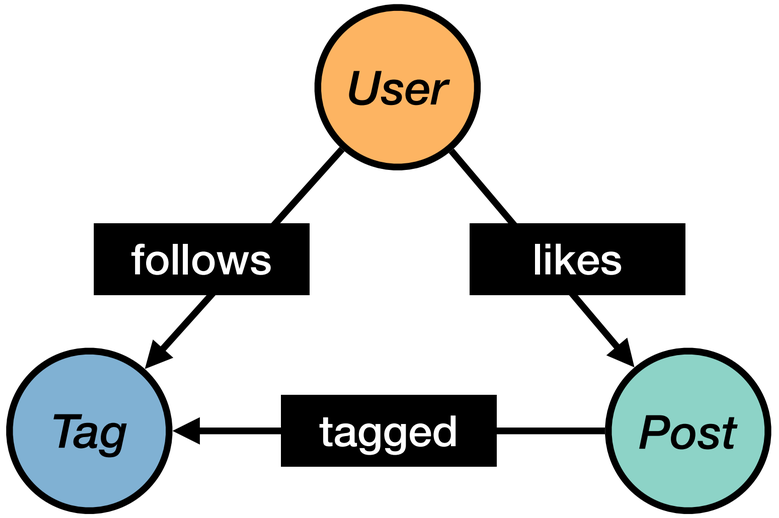




q: User-Tag-Post háromszögek




gráf-
adatbázisok
elméleti
probléma
gyakorlati
fontosság
algoritmus
implementáció
termékek
kitekintés
összefoglalás



⋈: illesztés, *join*

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

follows

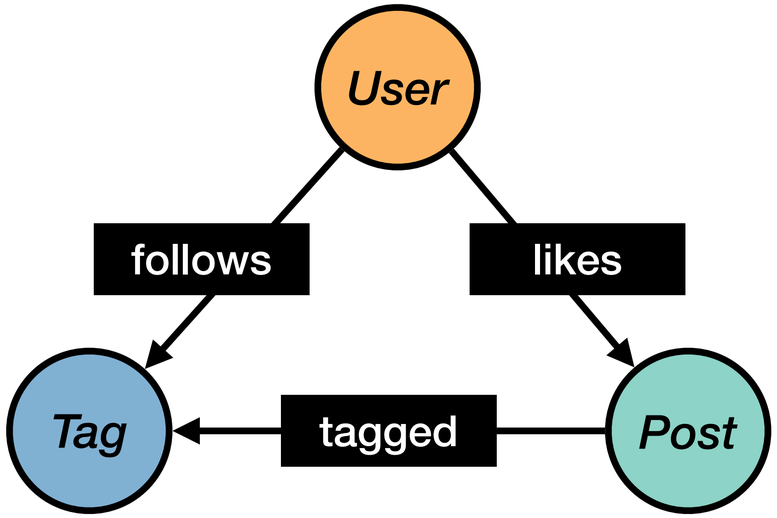
User	Tag
Ada	#fyp
Ada	#ski
Bob	#ski
Bob	#gym
Carl	#fyp
Carl	#gym

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



⋈: illesztés, *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

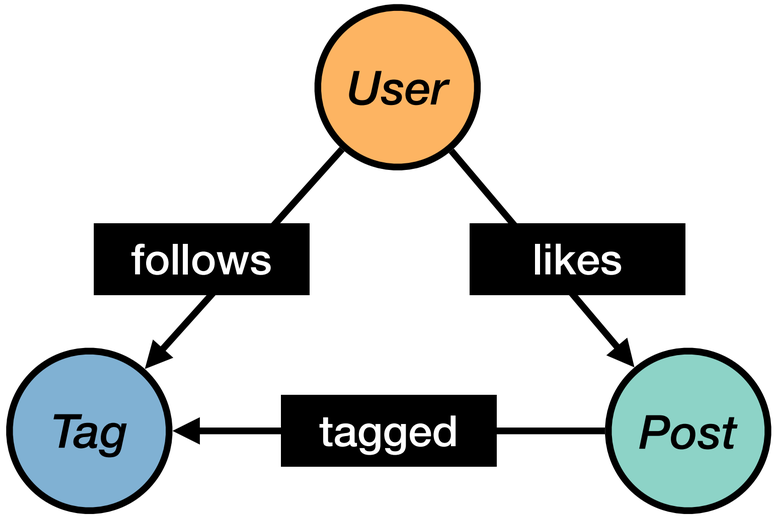
Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

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



⋈: illesztés, *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

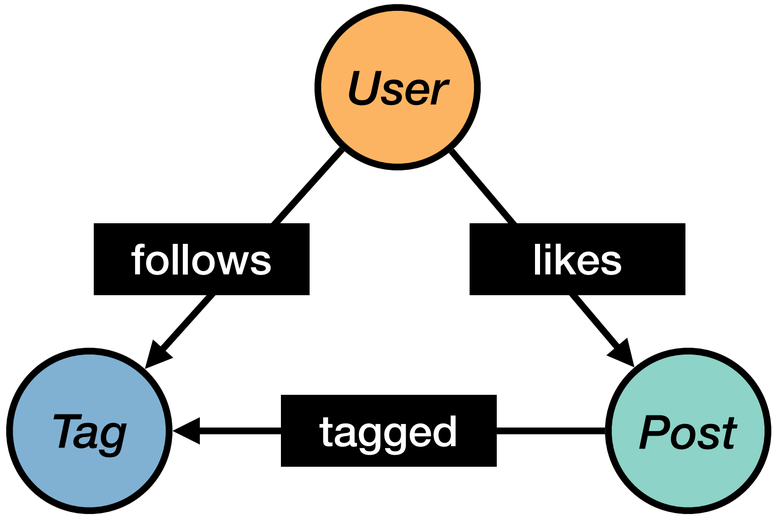
likes

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 sor



⋈: illesztés, *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

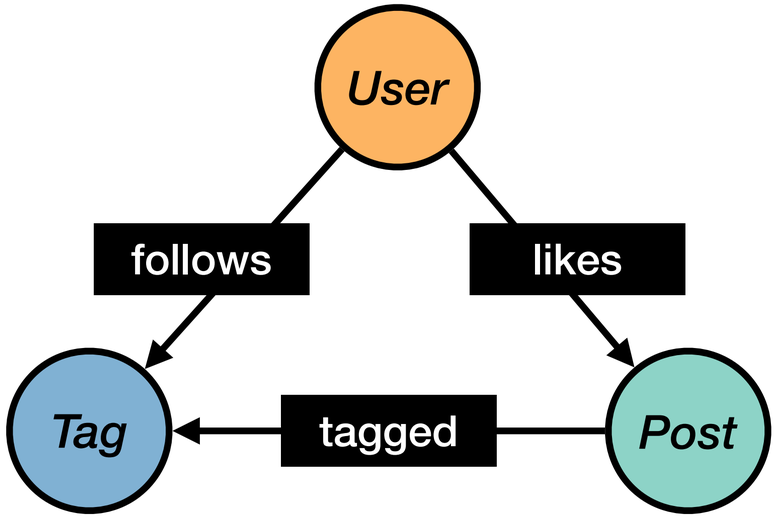
likes

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 sor



⋈: illesztés, *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

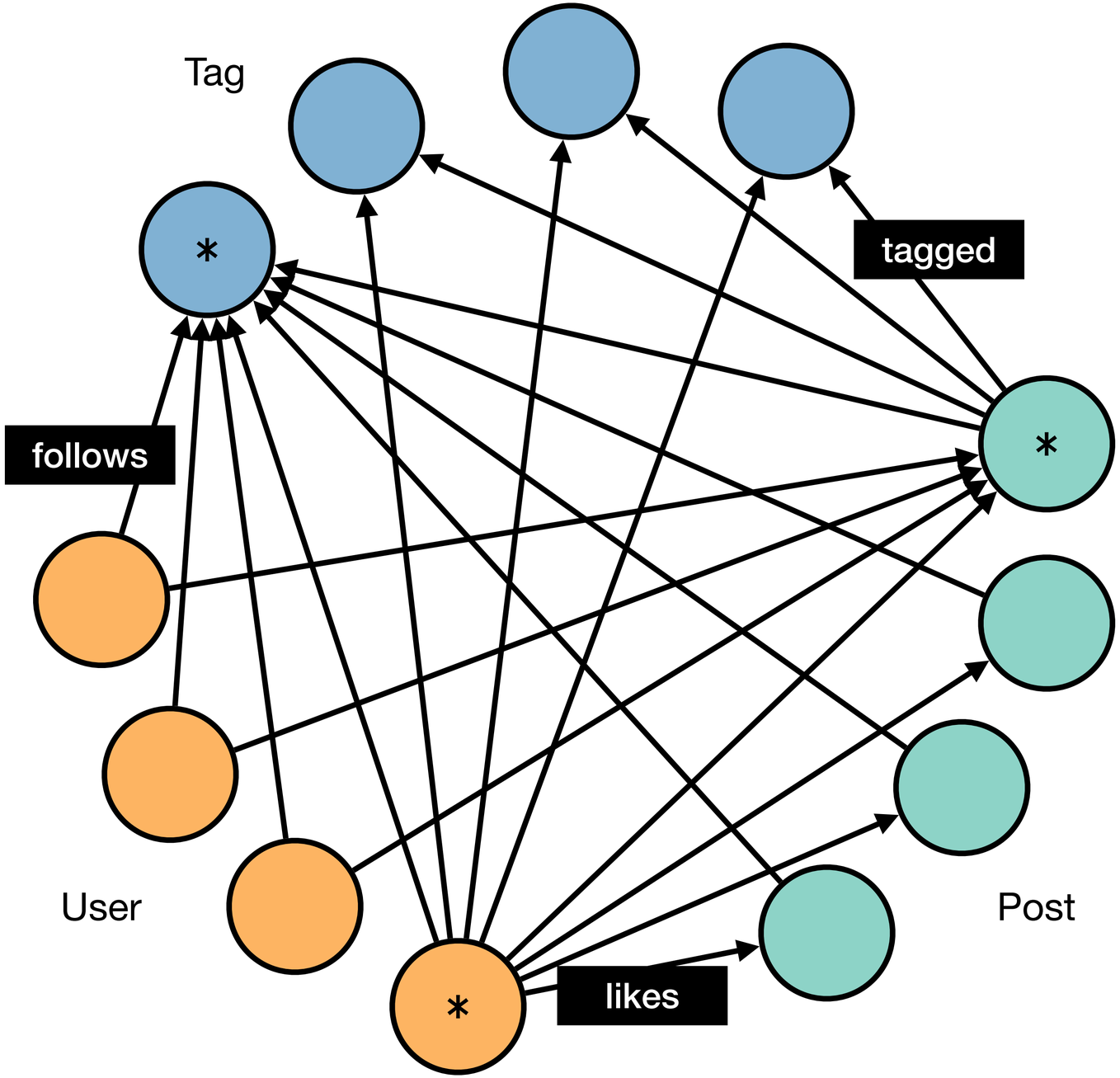
follows ⋈ likes

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 sor

Mi a komplexitása az algoritmusnak?

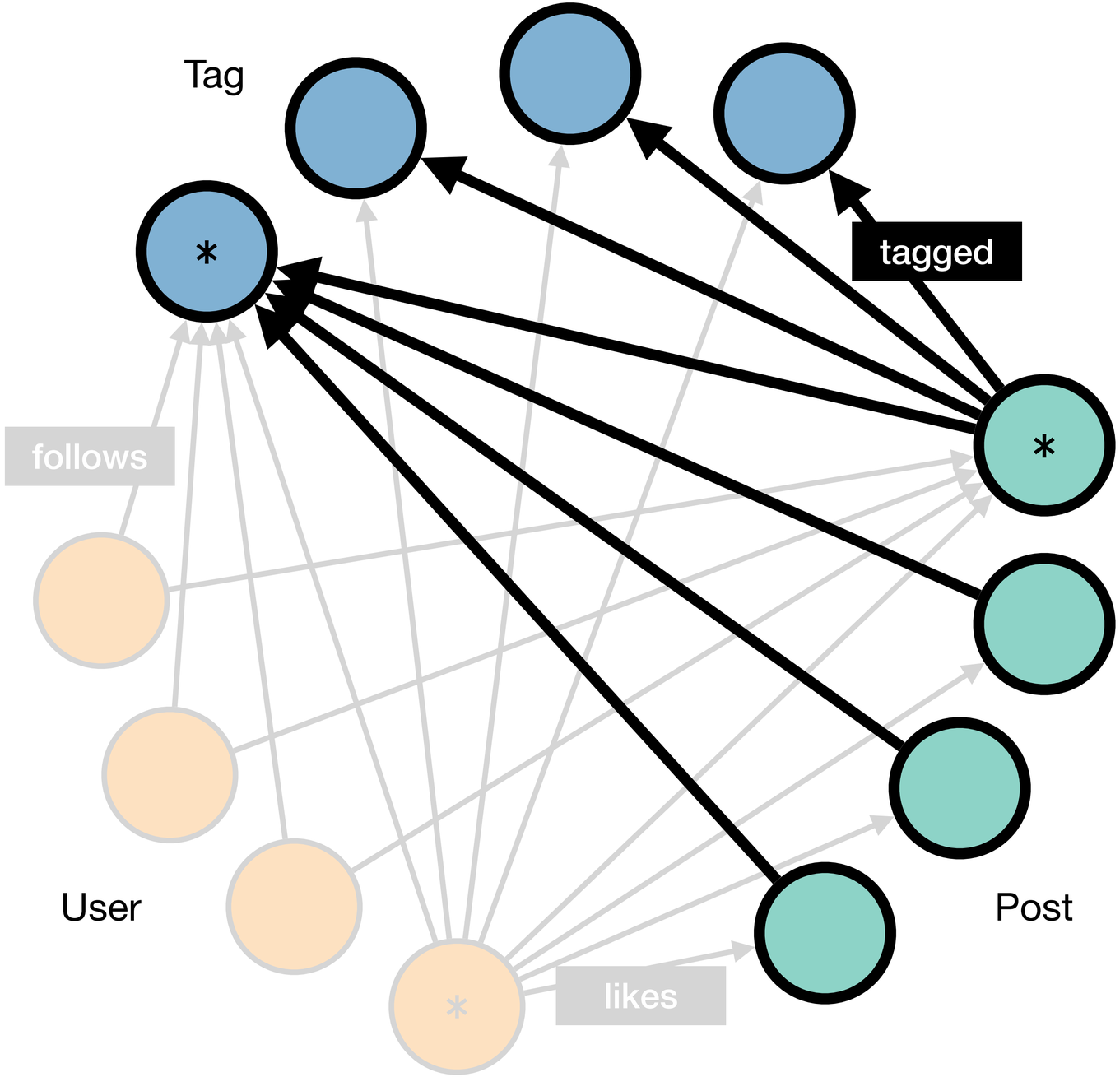
⇒ Nézzünk meg egy *worst-case* bemenetet



Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

1 kiemelt

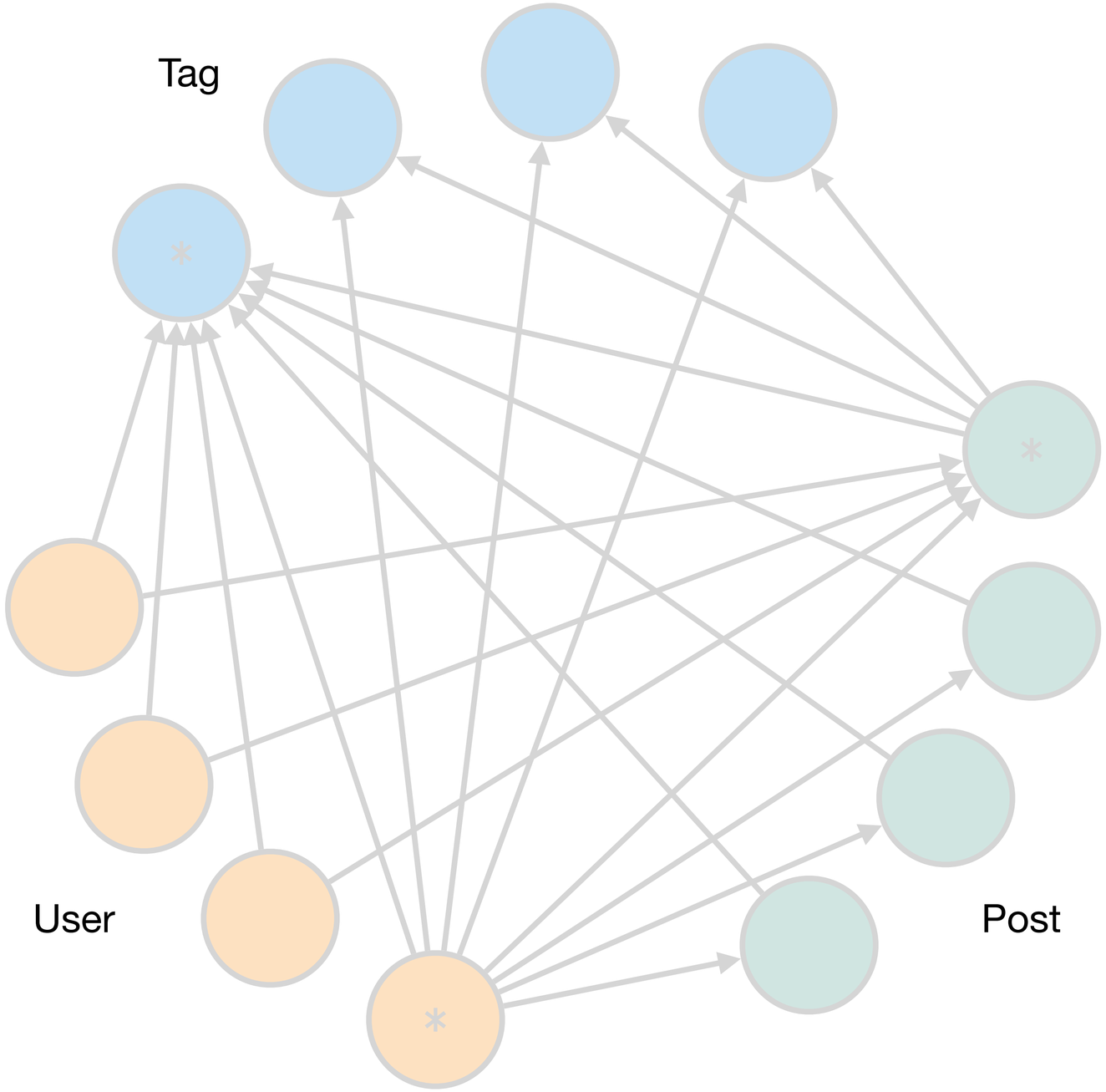
k sima



Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

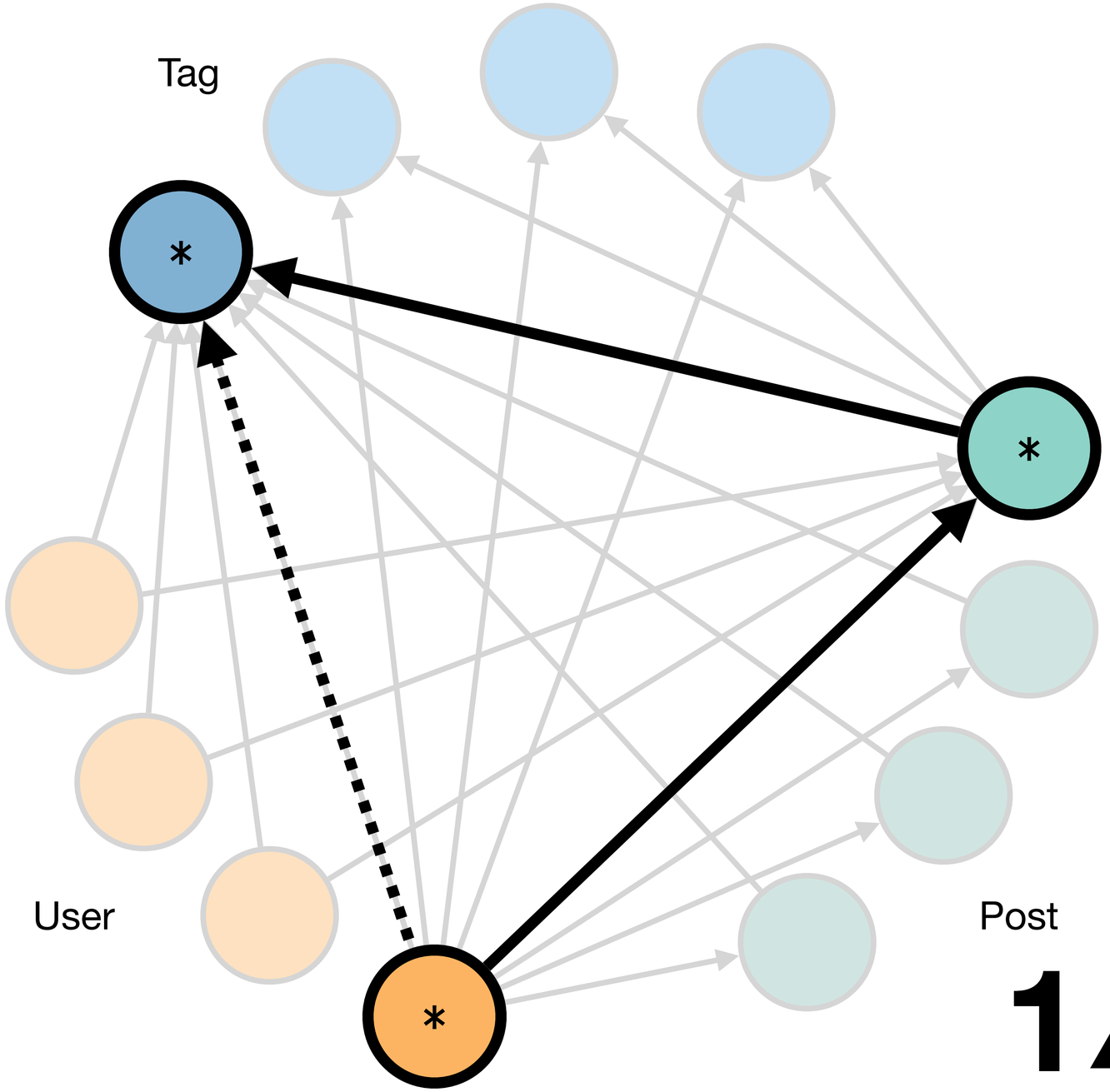
$$n = 2k + 1$$



Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

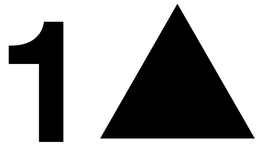
$$n = 2k + 1$$

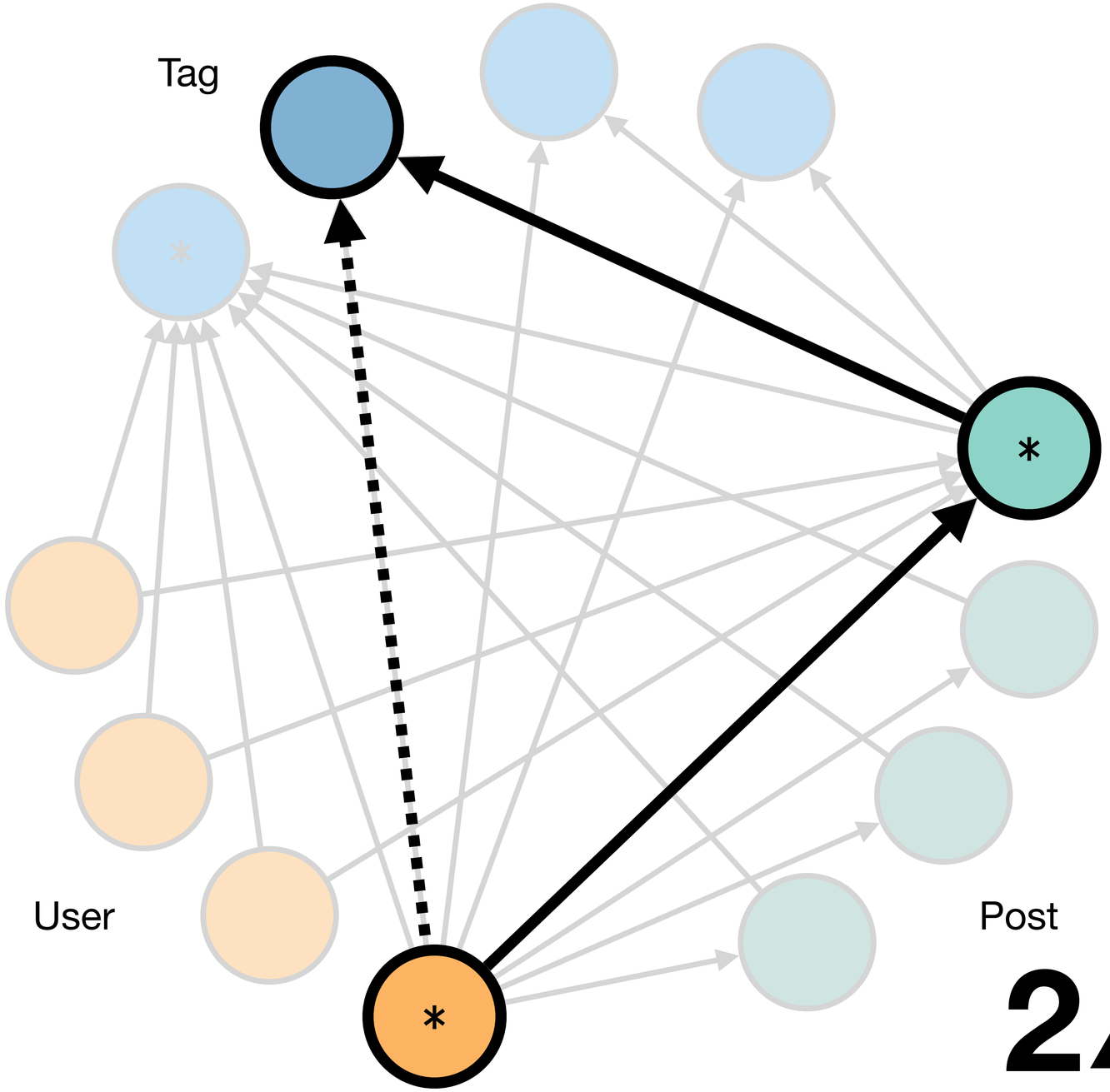


Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

$$n = 2k + 1$$

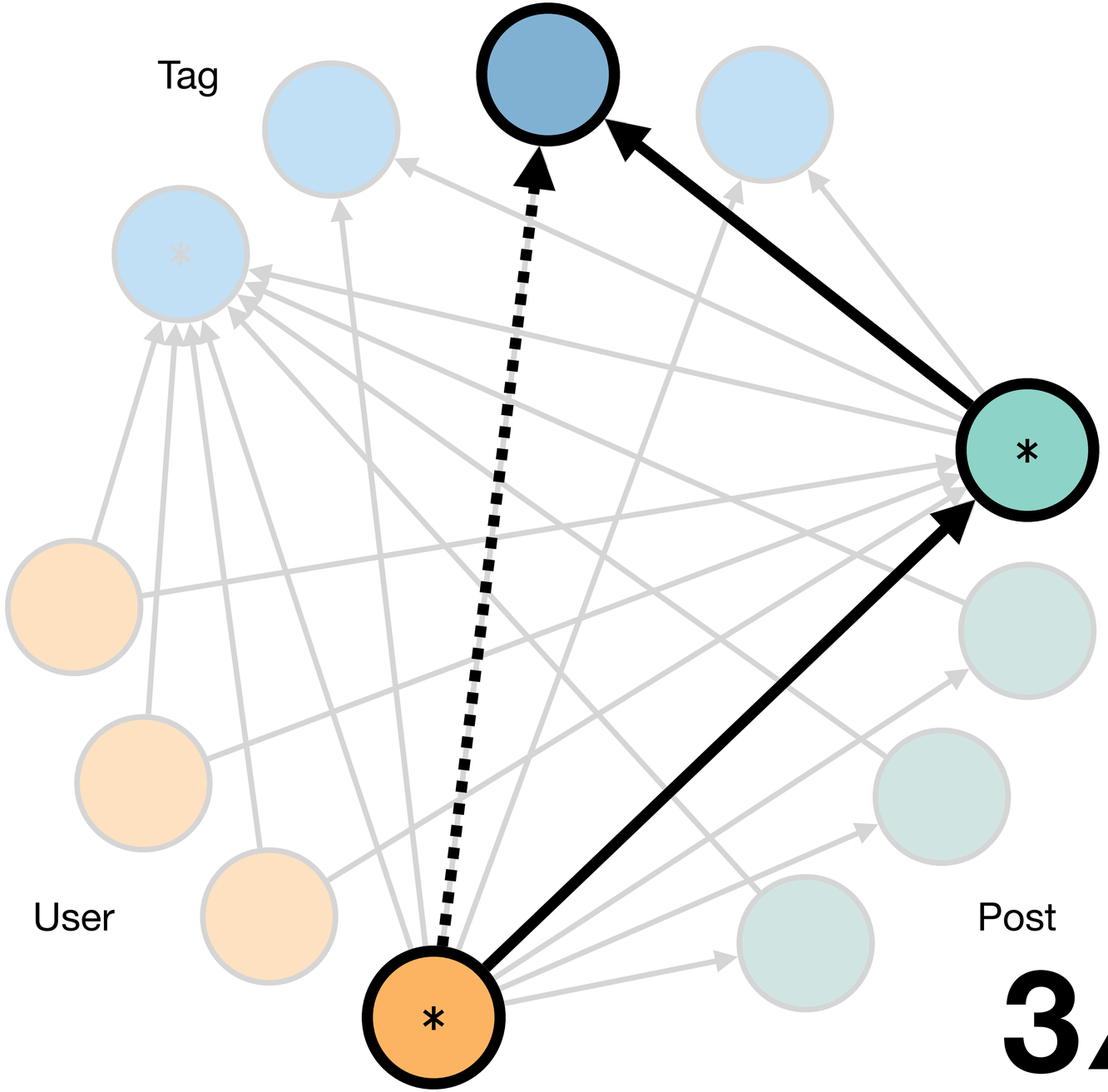




Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

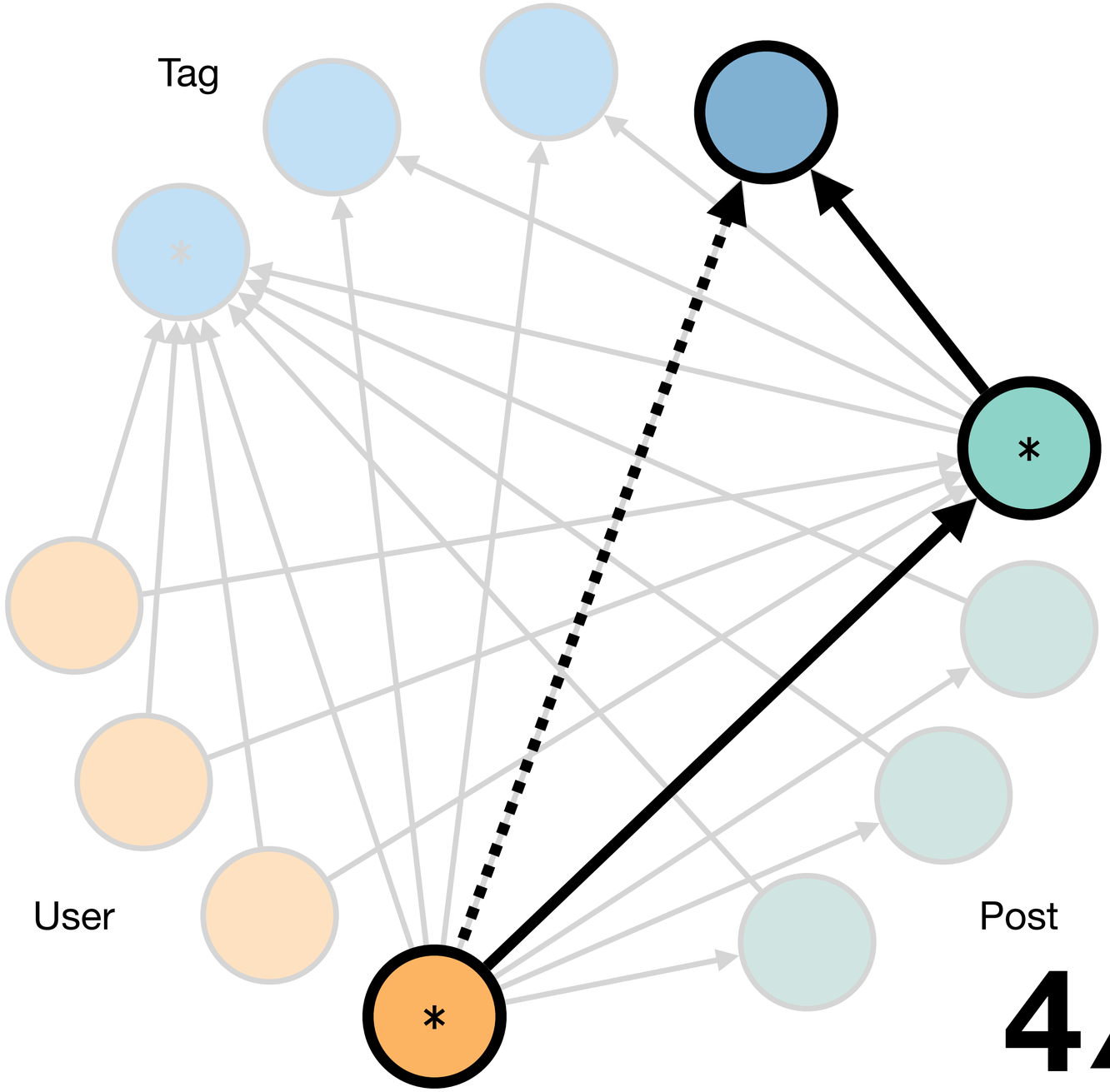
$$n = 2k + 1$$



Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

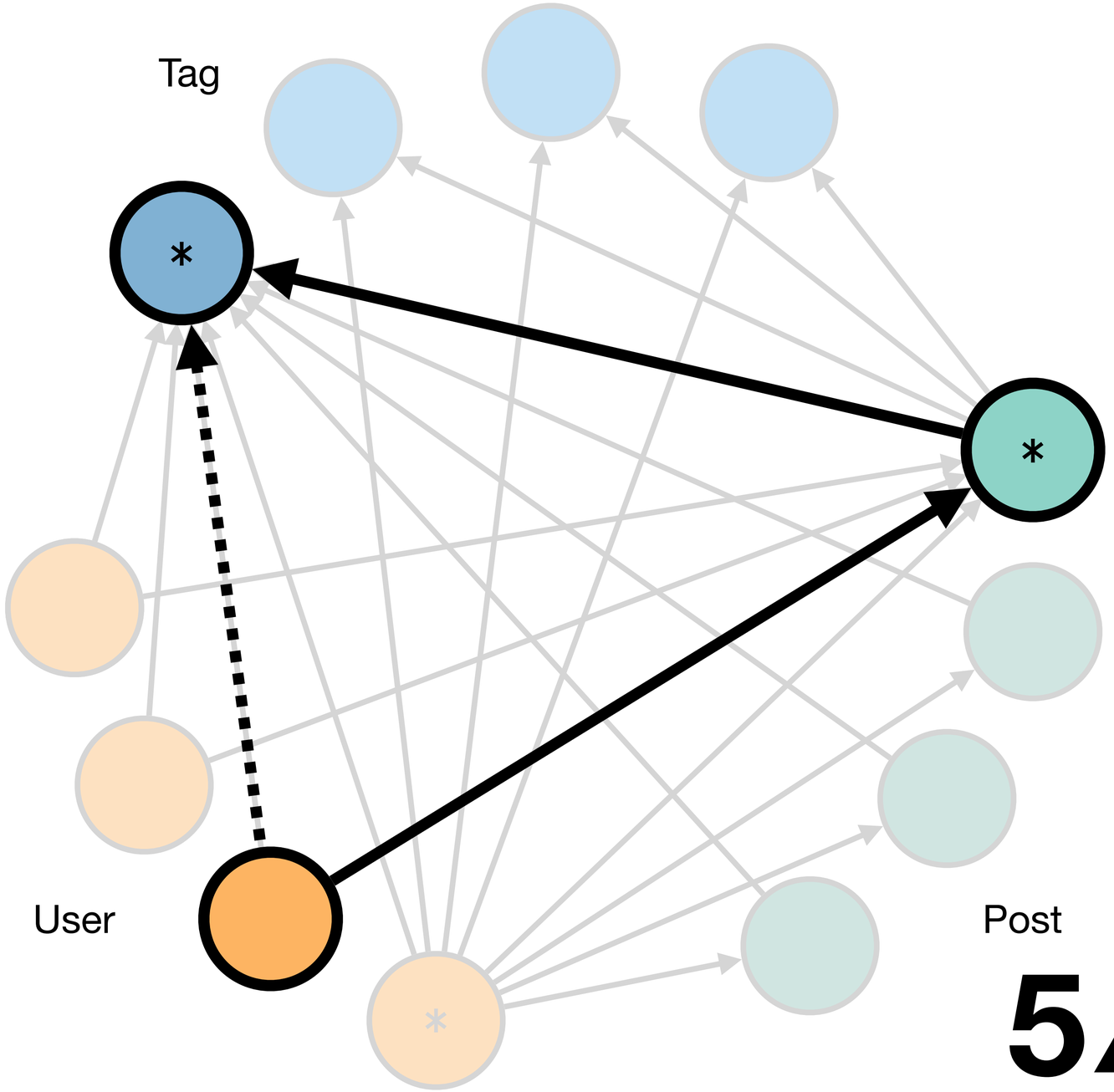
$$n = 2k + 1$$



Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

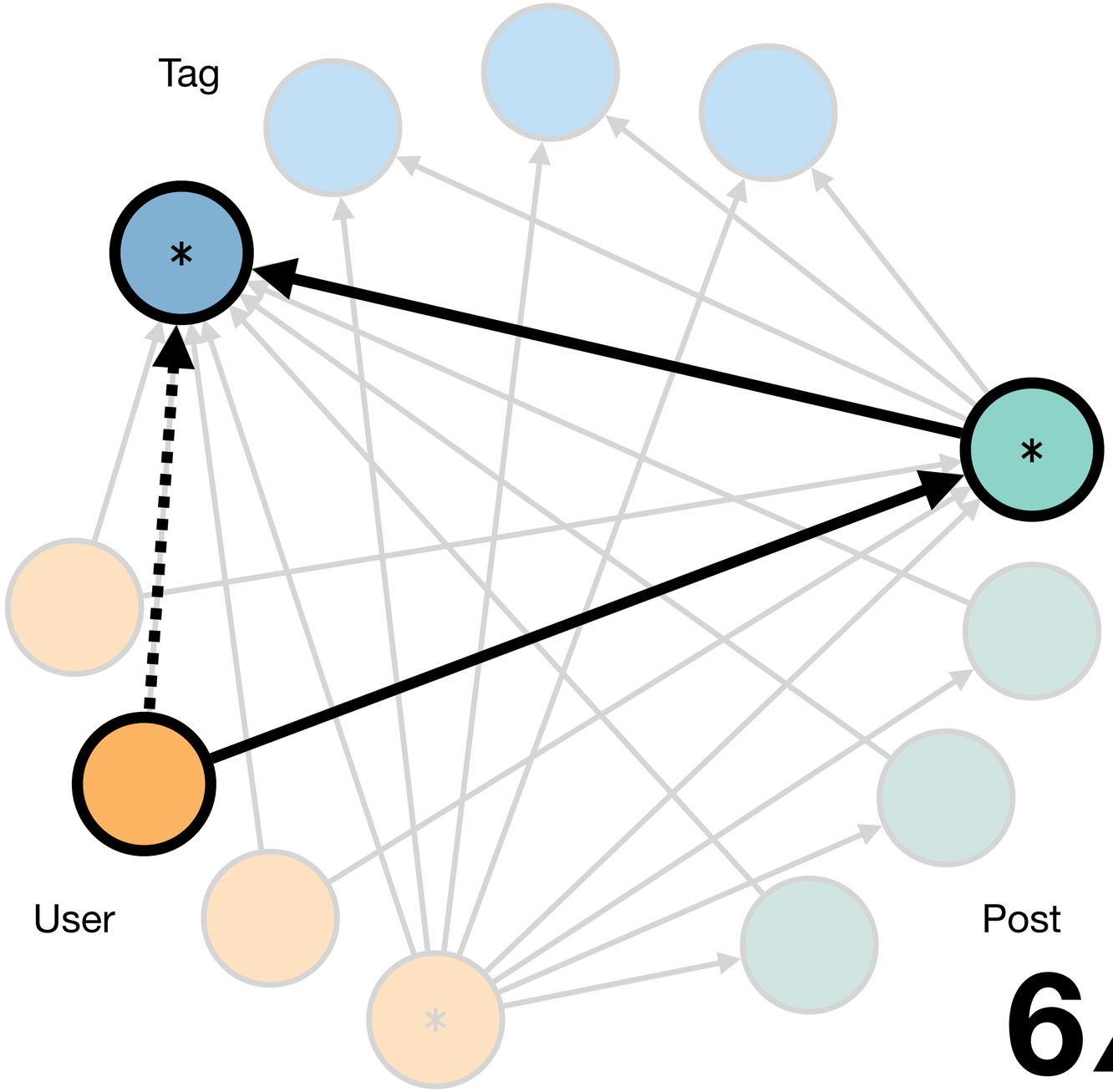
$$n = 2k + 1$$



Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

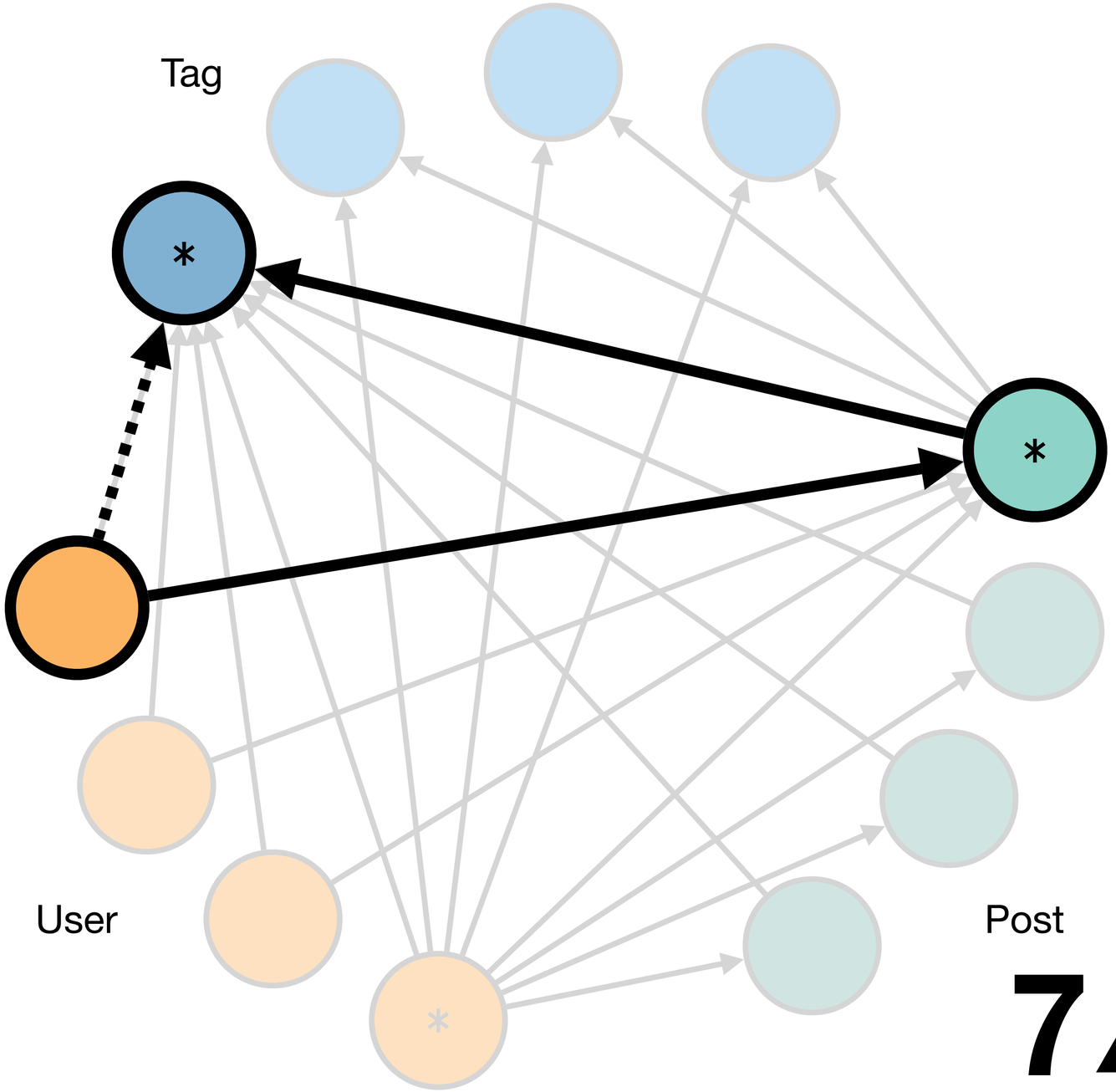
$$n = 2k + 1$$



Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

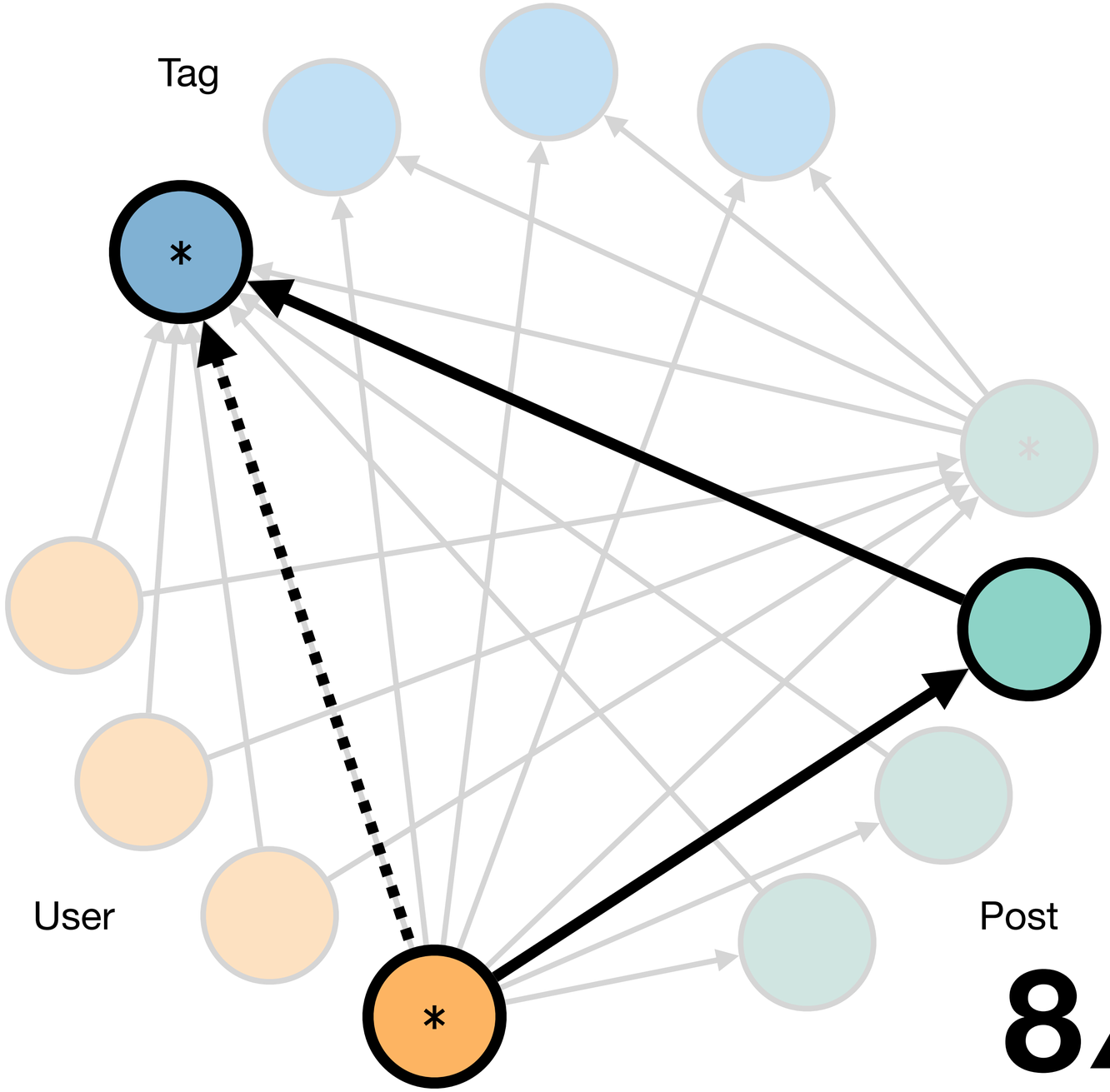
$$n = 2k + 1$$



Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

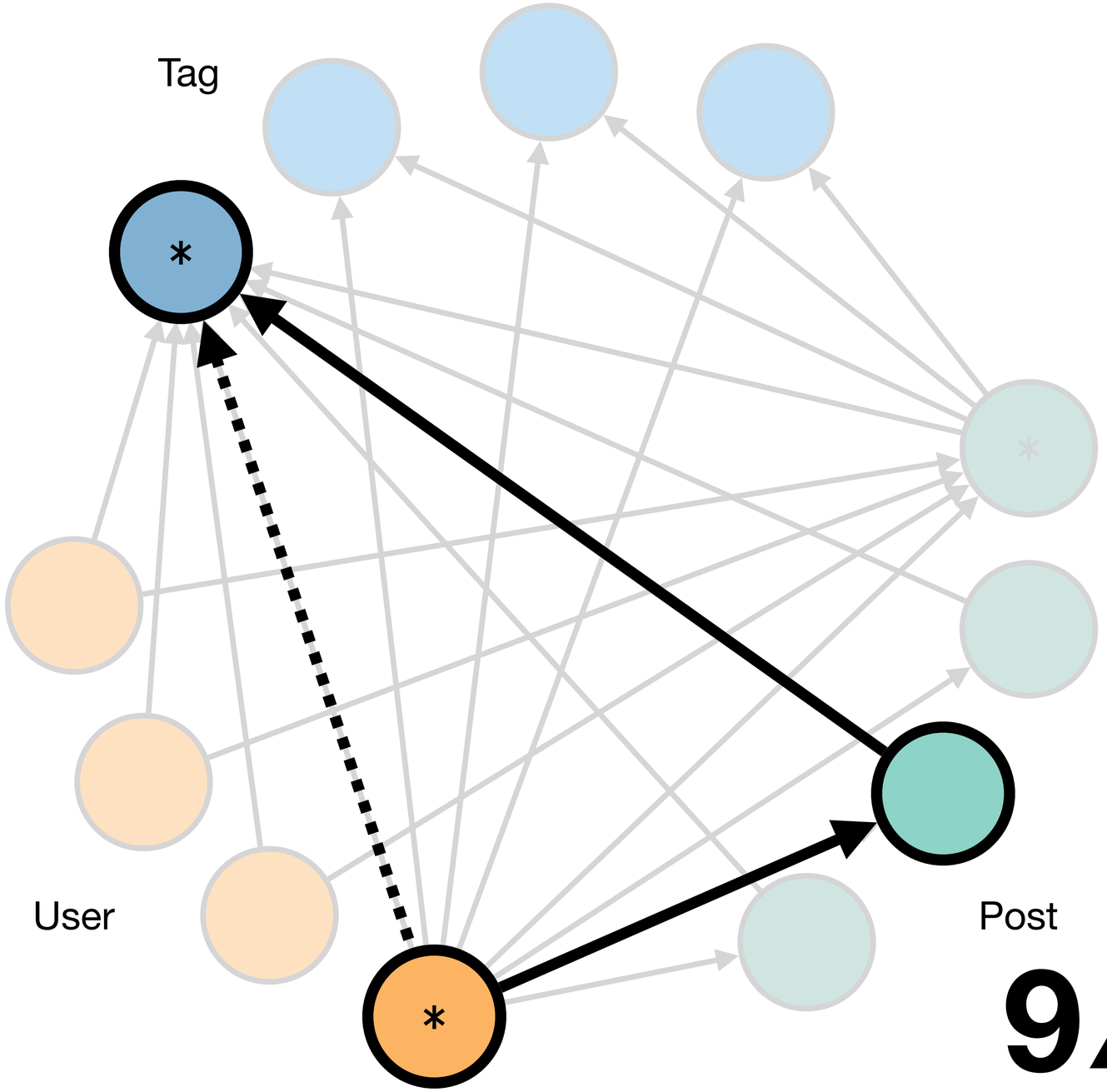
$$n = 2k + 1$$



Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

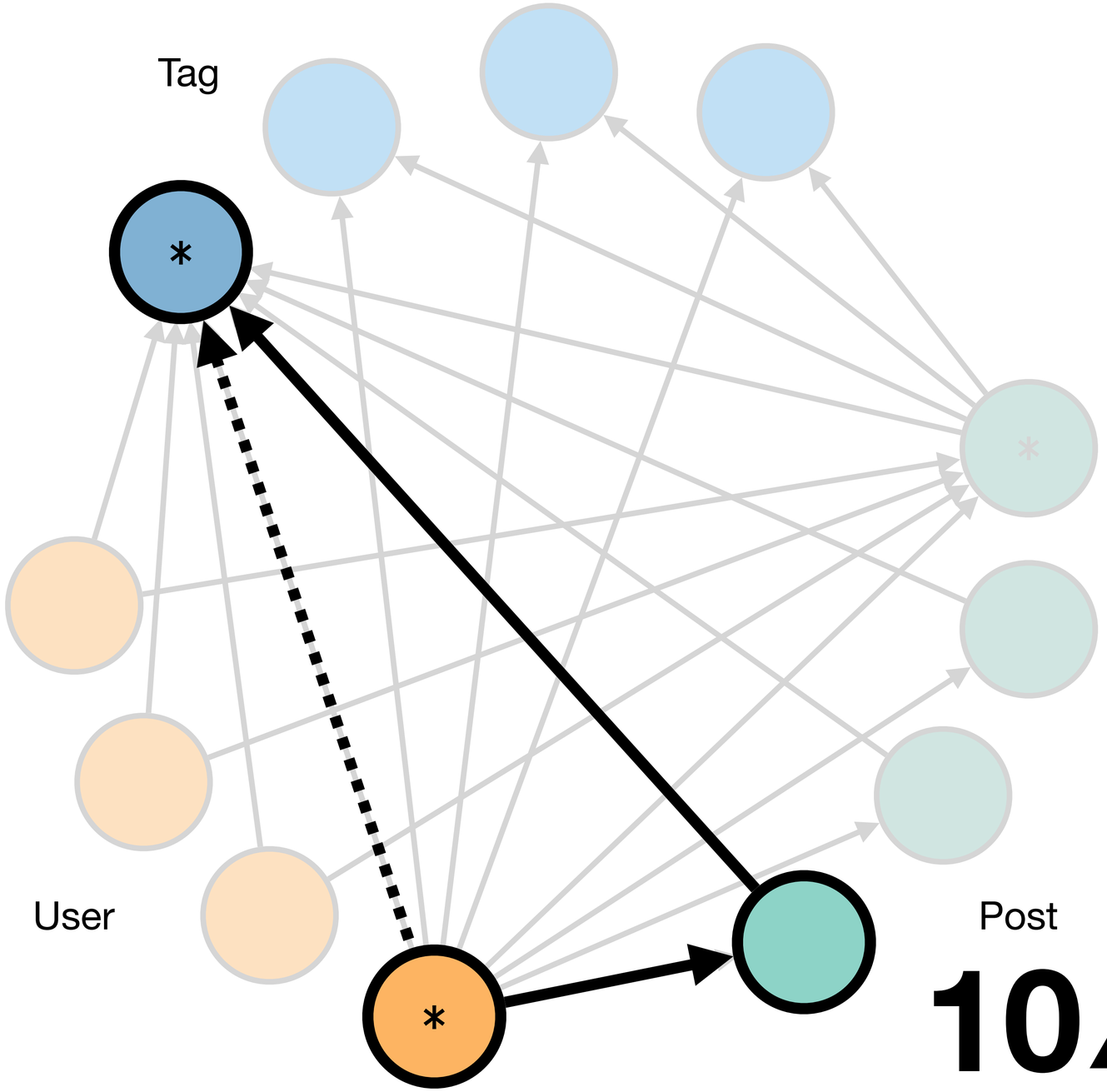
$$n = 2k + 1$$



Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

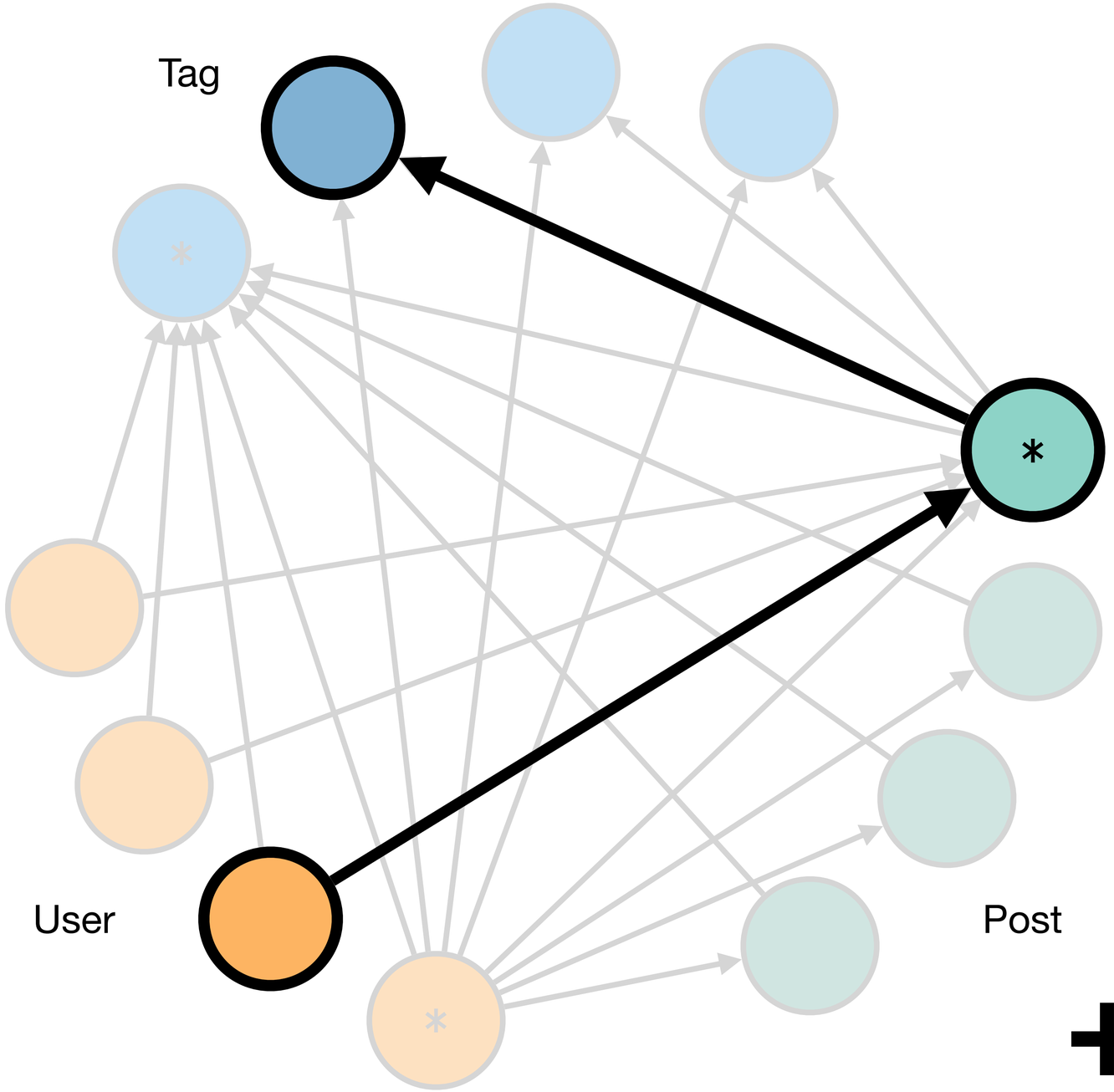
$$n = 2k + 1$$



Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

$$n = 2k + 1$$

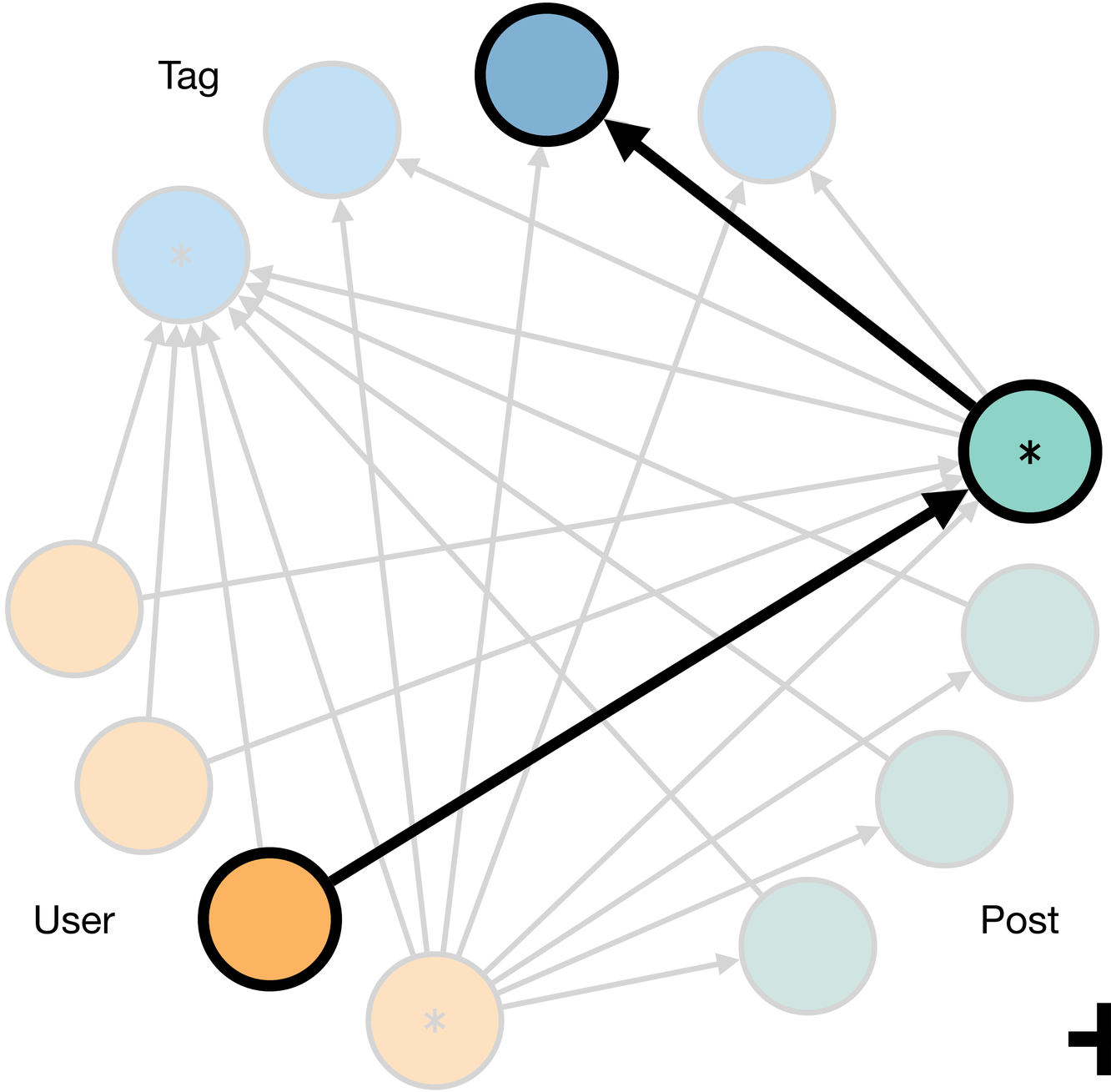


Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

$$n = 2k + 1$$

+1

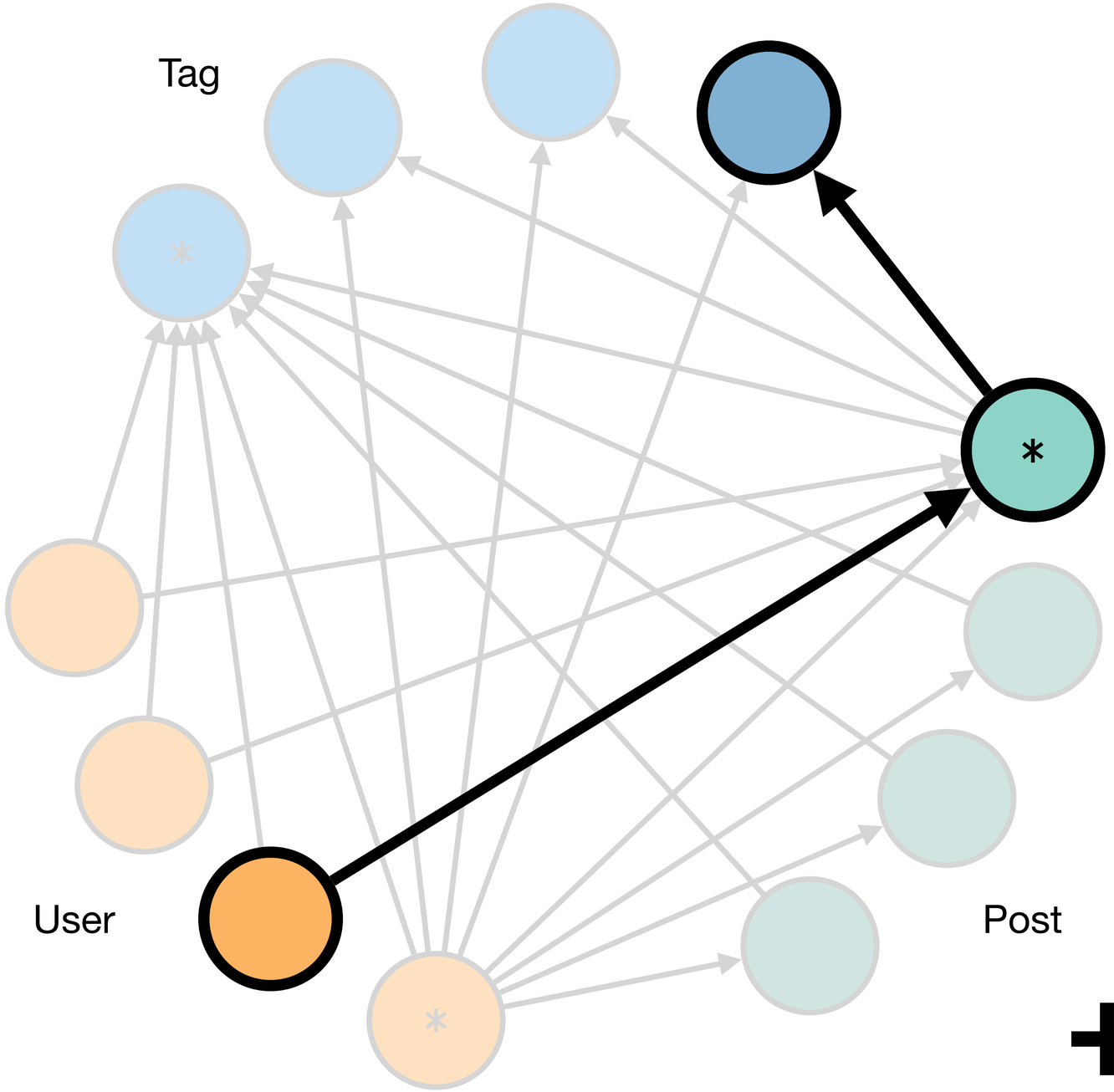


Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

$$n = 2k + 1$$

+2

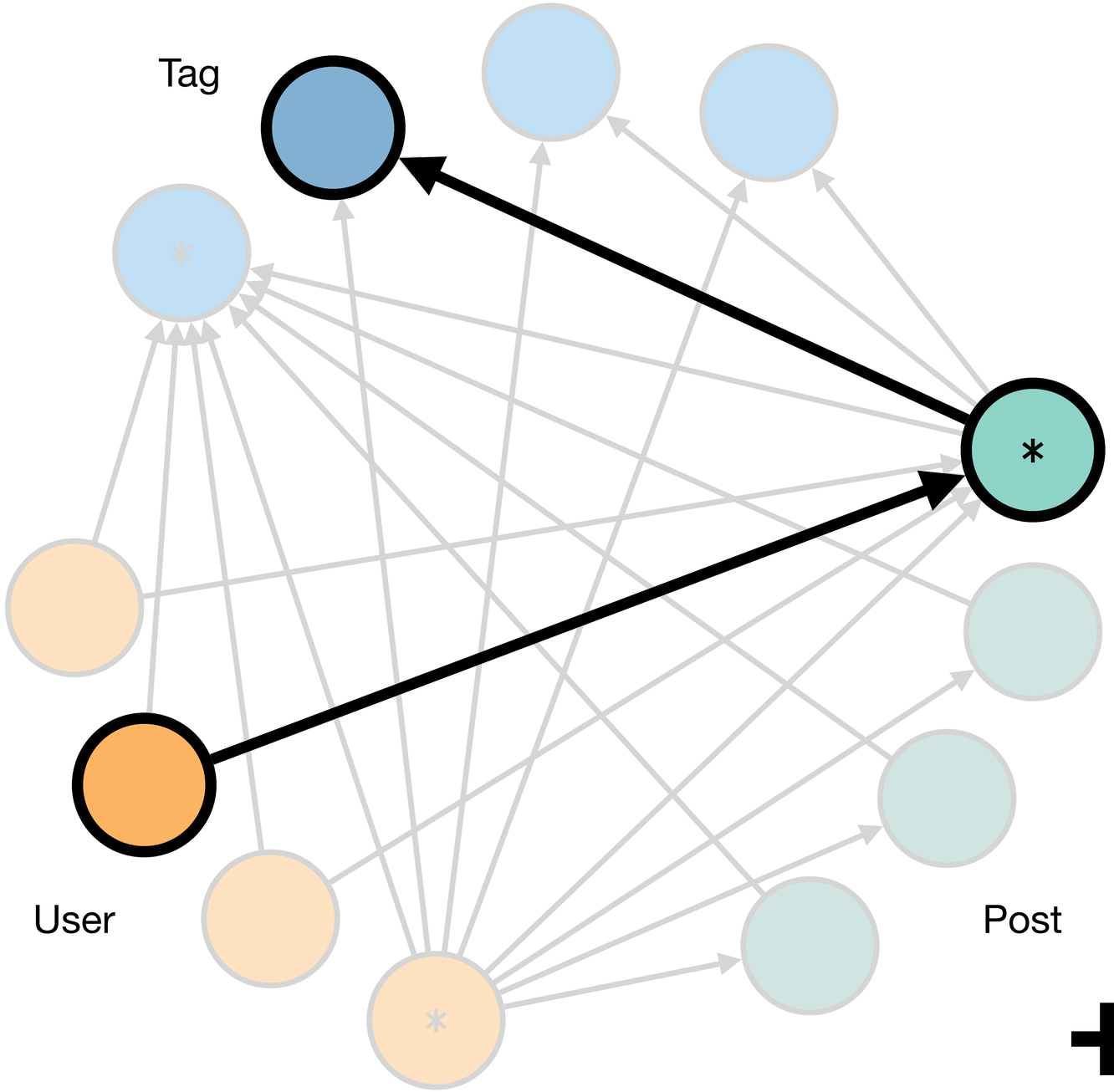


Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

$$n = 2k + 1$$

+3

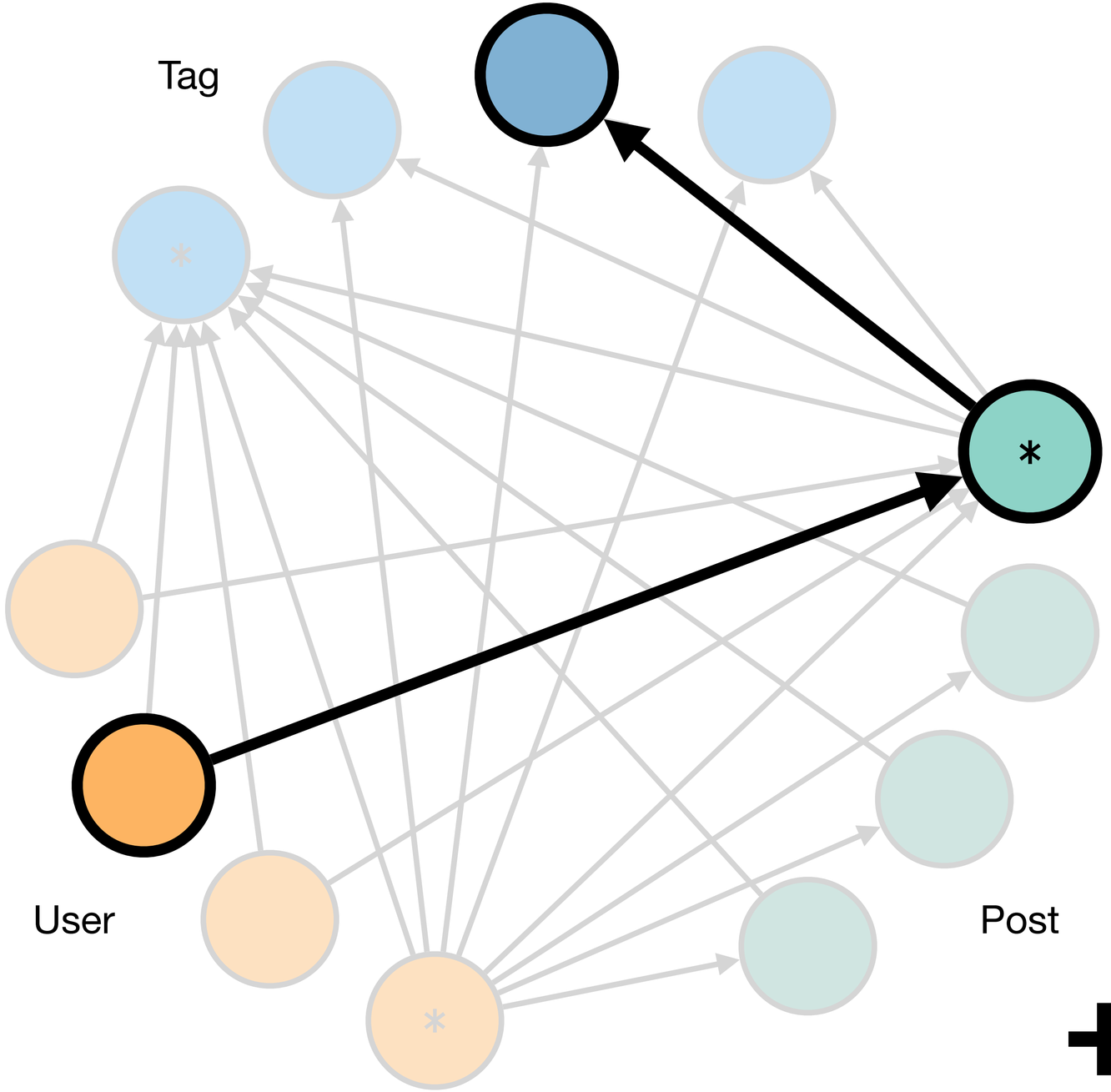


Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

$$n = 2k + 1$$

+4

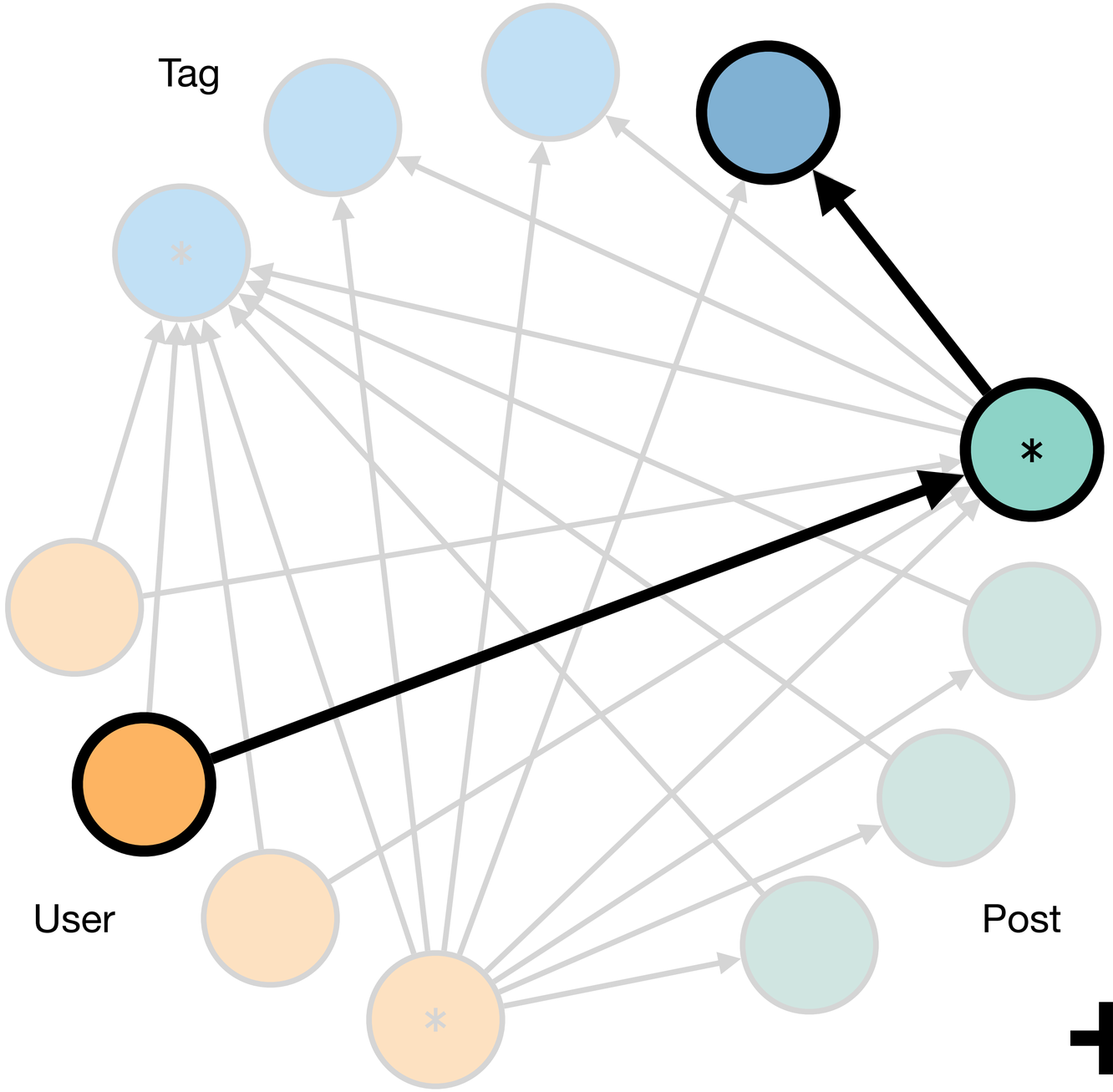


Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

$$n = 2k + 1$$

+5

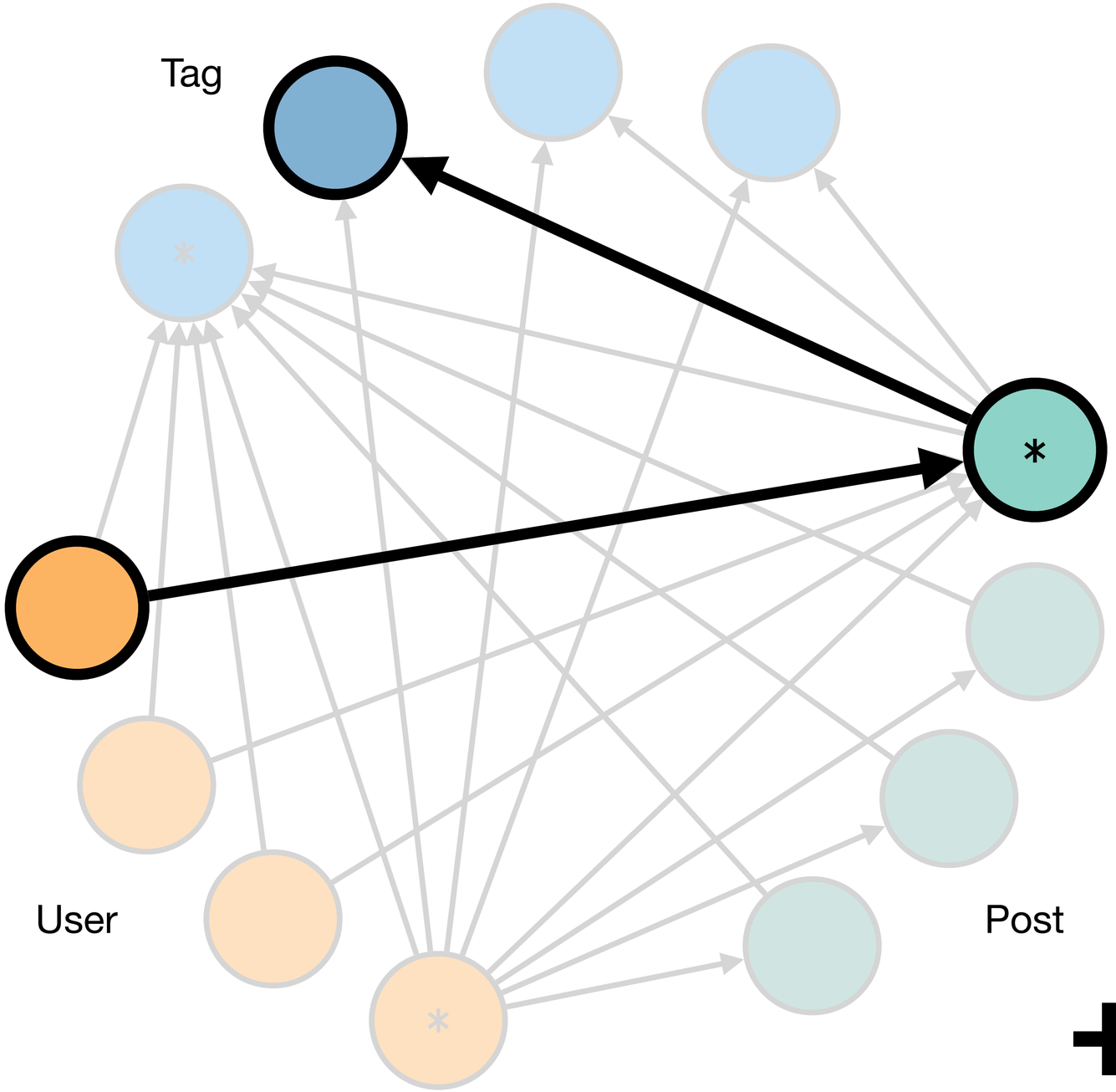


Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

$$n = 2k + 1$$

+6

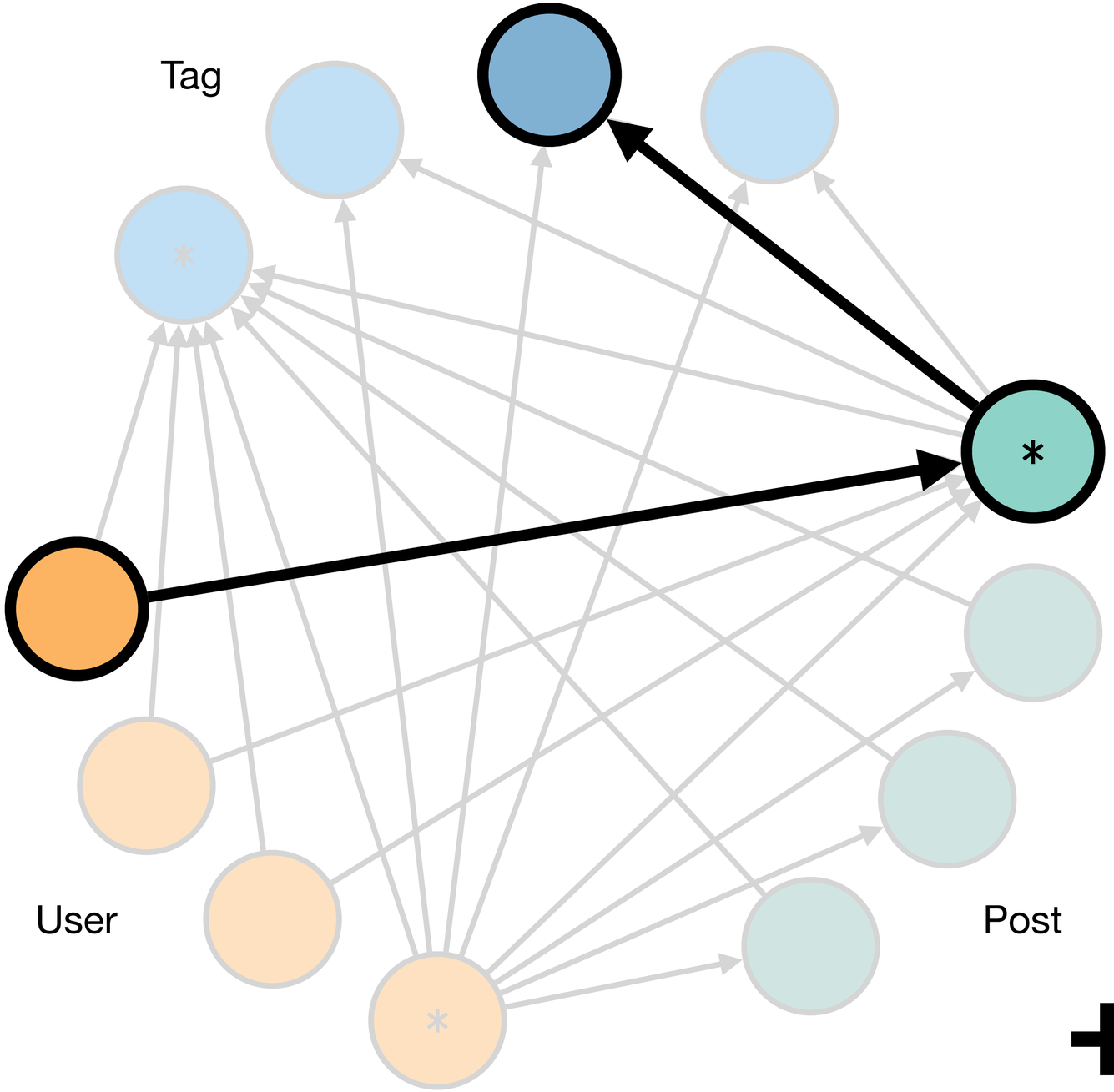


Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

$$n = 2k + 1$$

+7

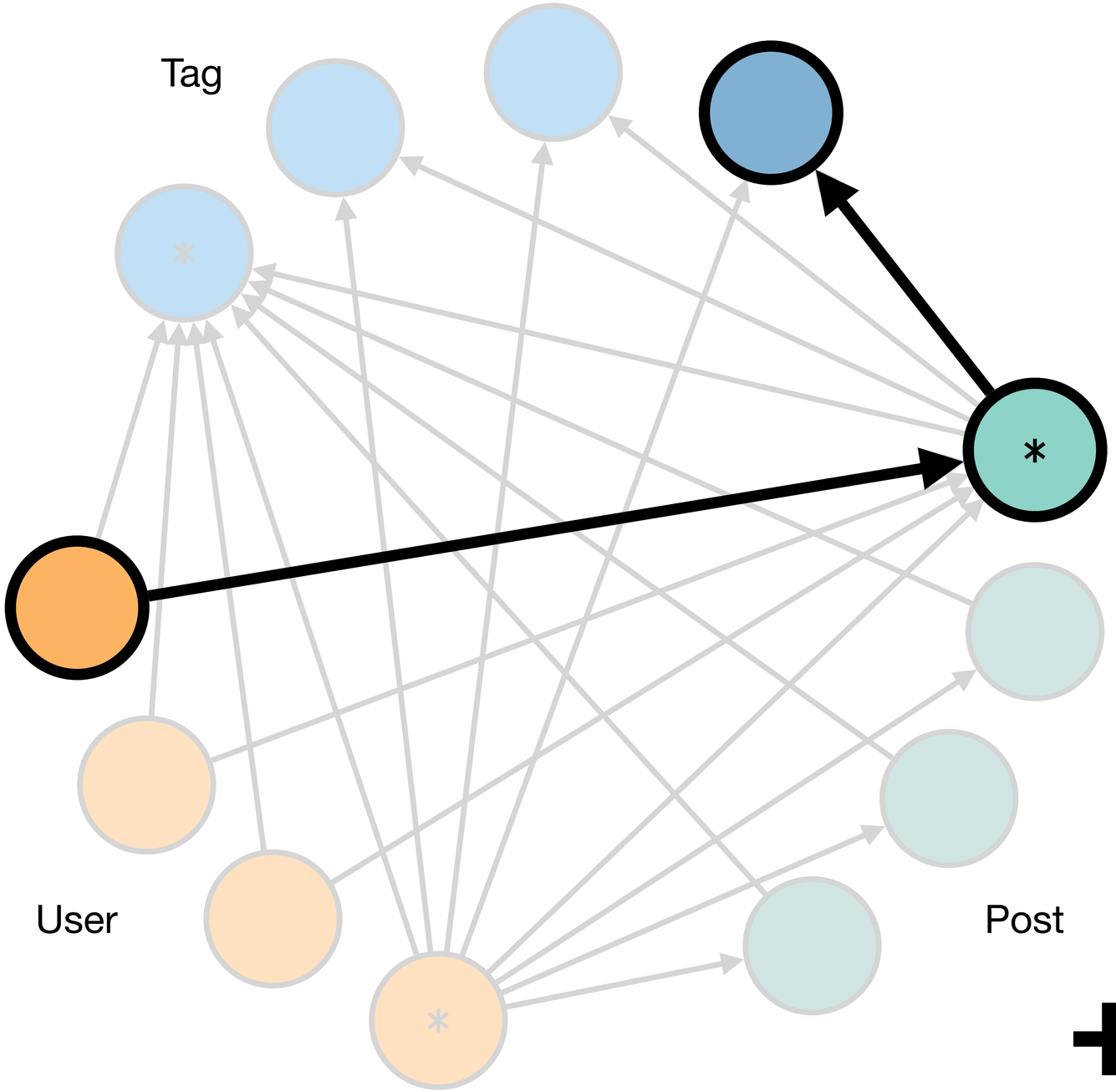


Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

$$n = 2k + 1$$

+8

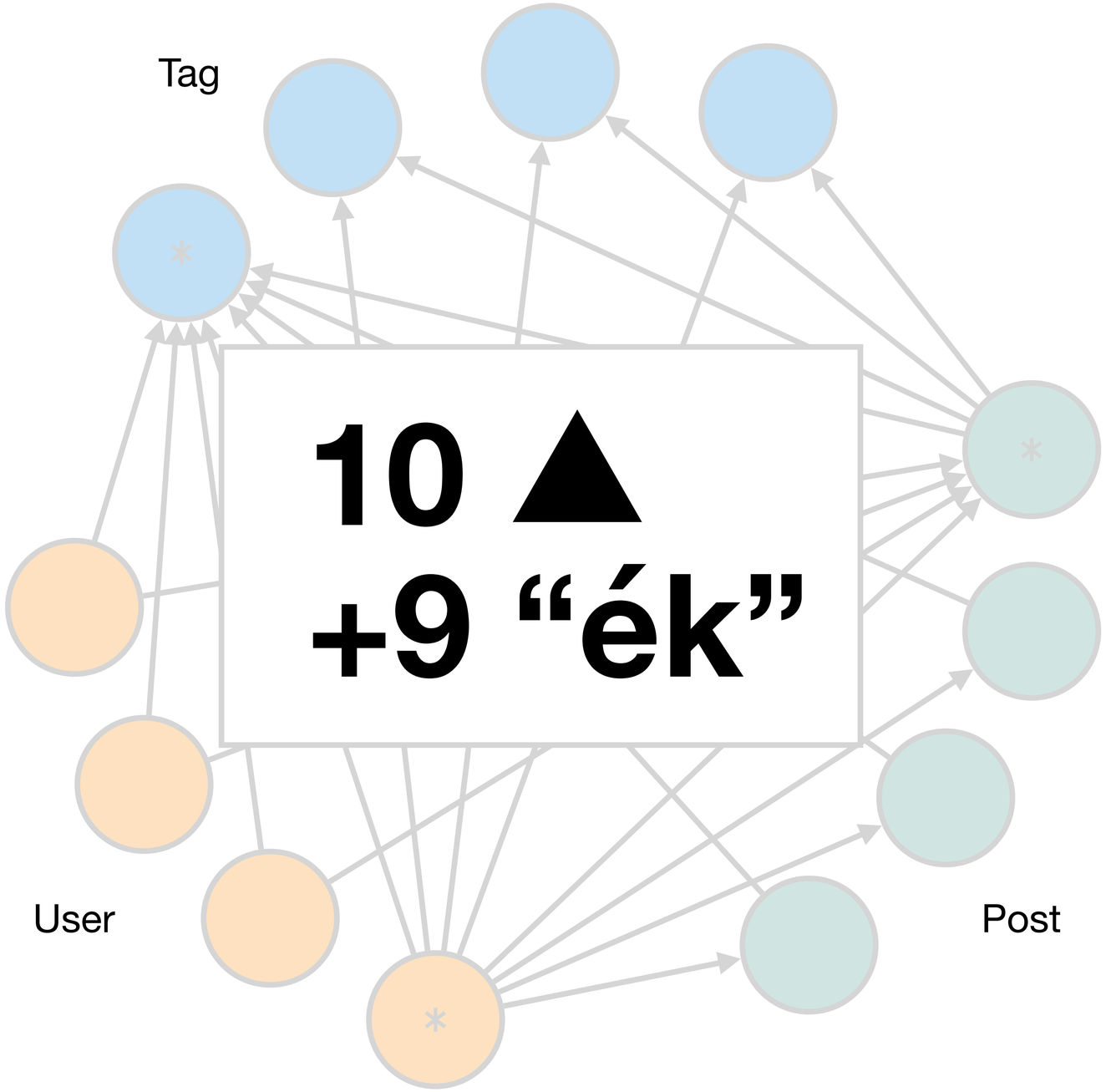


Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

$$n = 2k + 1$$

+9



Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

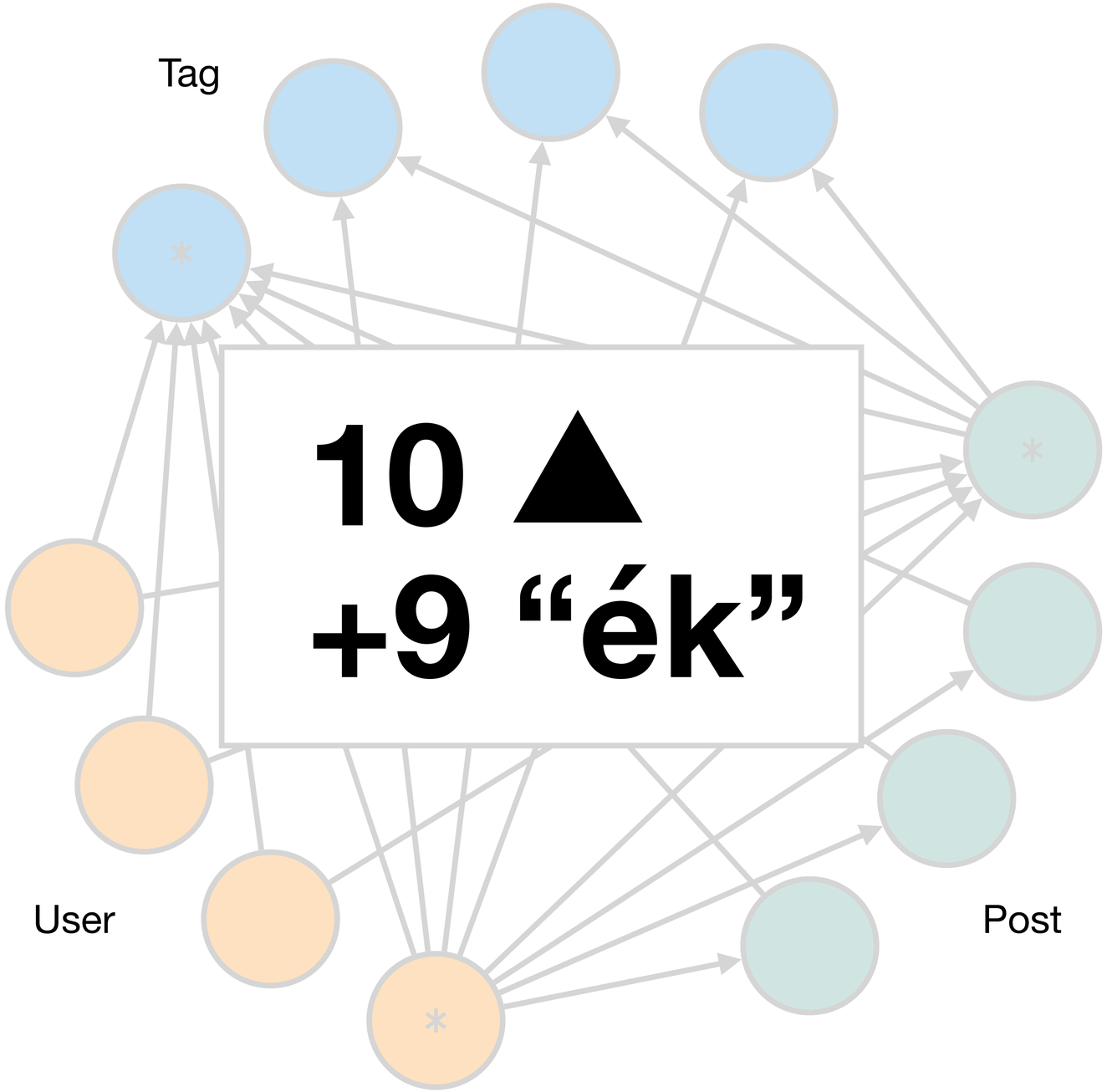
$$n = 2k + 1$$

Háromszögek száma:

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

Első join mérete:

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



Minden csomóponttípusból k sima és 1 kiemelt ($*$) példány van.

Táblák mérete:

$$n = 2k + 1$$

Háromszögek száma:

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

Első join mérete:

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

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

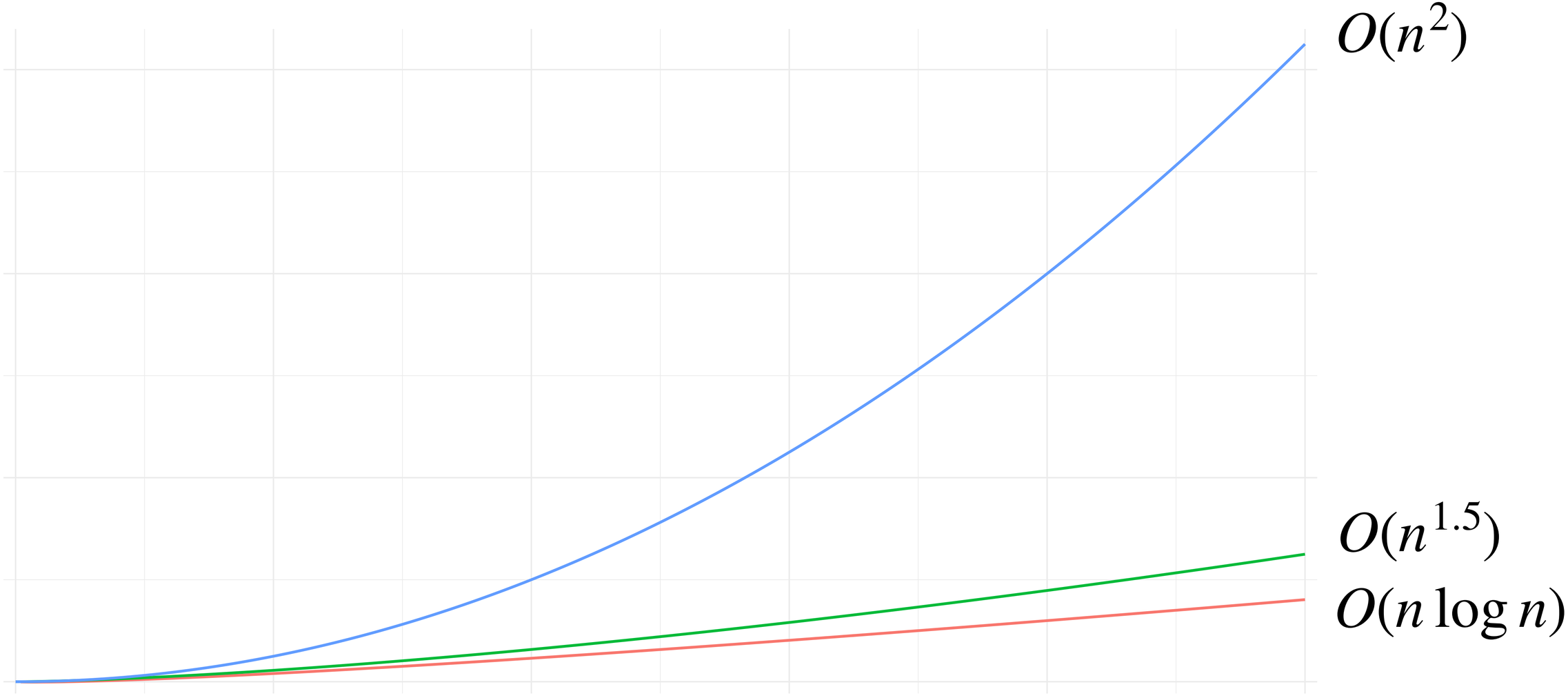
Martin Grohe
Humboldt Universität zu Berlin
Berlin, Germany
grohe@informatik.hu-berlin.de

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

Egy n élű gráfra a $q_{\triangle}$-t join műveletekkel kiszámítva a futásidő $O(n^2)$

$q_{\triangle}$ lekérdezés kimenetének mérete $\Theta(n^{1.5})$

Komplexitás: $n^{1.5}$ 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

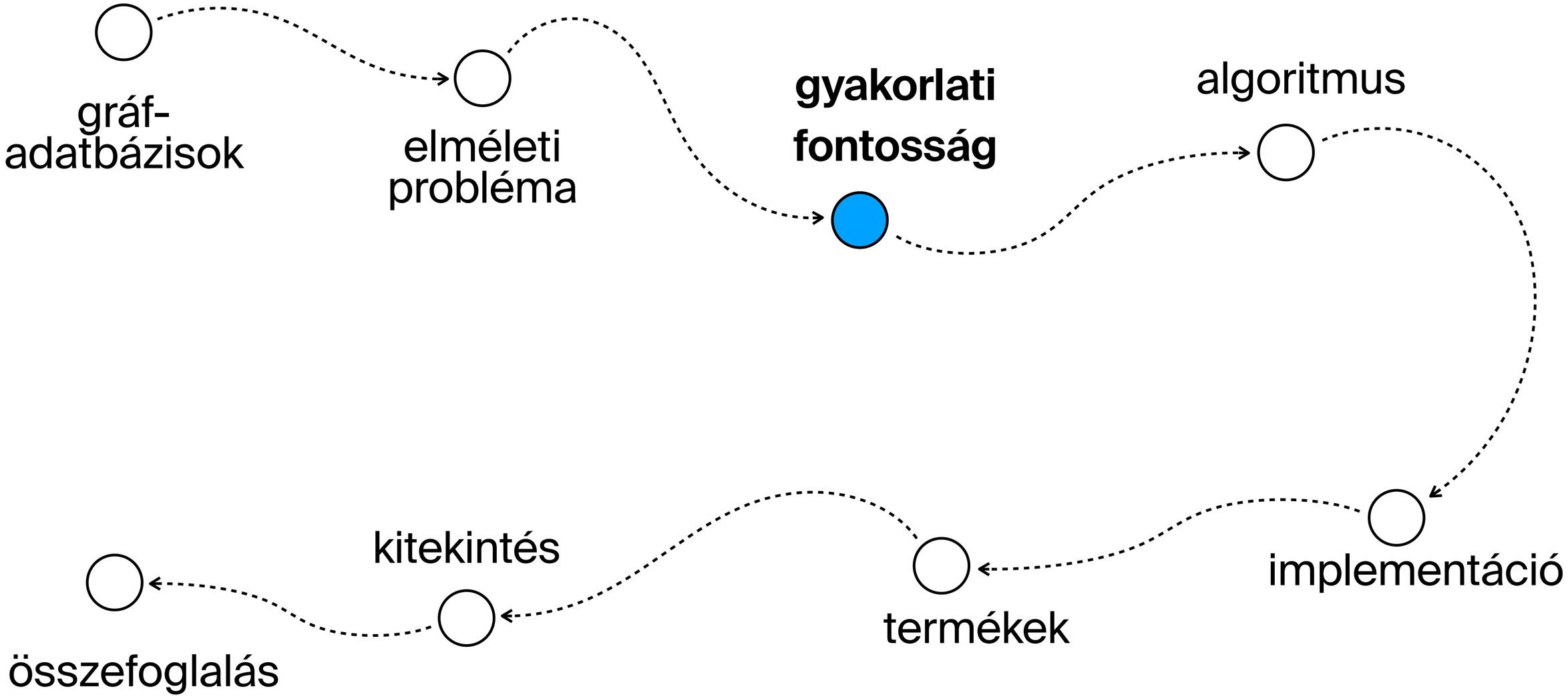
Martin Grohe
Humboldt Universität zu Berlin
Berlin, Germany
grohe@informatik.hu-berlin.de

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

Egy n élű gráfra a $q_{\triangle}$-t join műveletekkel kiszámítva a futásidő $O(n^2)$

$q_{\triangle}$ lekérdezés kimenetének mérete $\Theta(n^{1.5})$

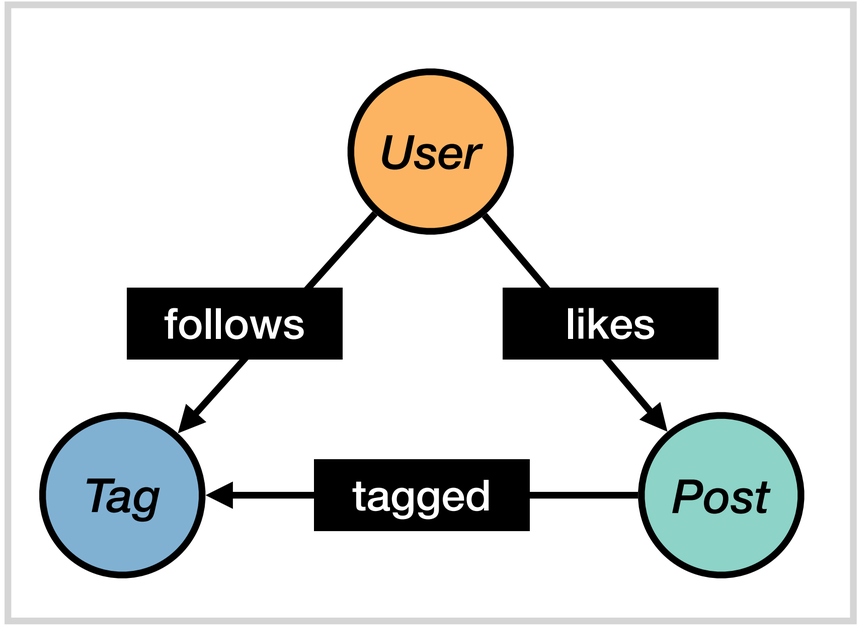
A cikk nem ad algoritmust a join művelet kiszámítására.


gráf-
adatbázisok
elméleti
probléma
gyakorlati
fontosság
algoritmus
implementáció
termékek
kitekintés
összefoglalás

Mikor jön elő ez a probléma?

Feltételei:

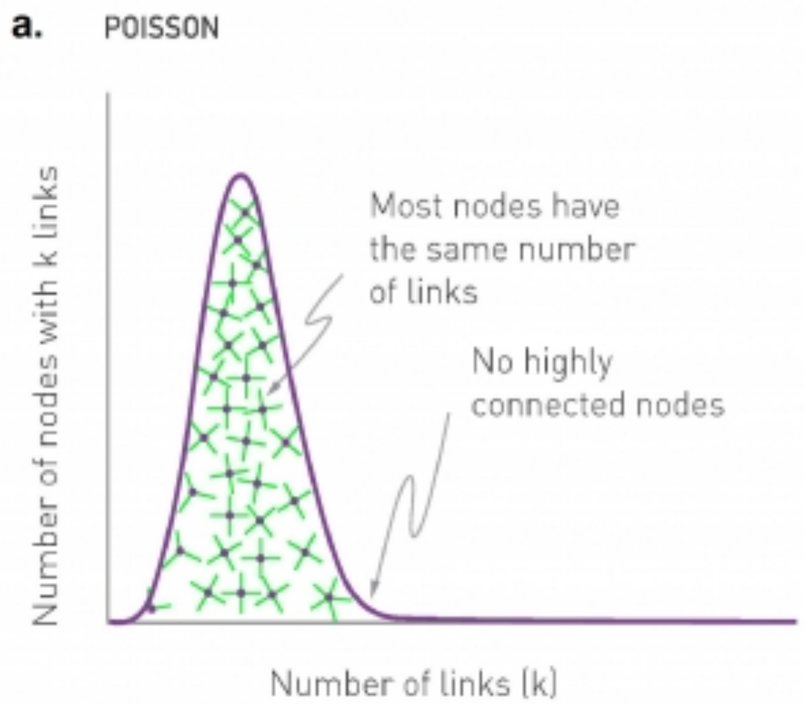
- ciklikus gráfminta
- több-több kardinalitású élek mentén
- amik aszimmetrikus (ferde, *skewed*) fokszám-eloszlásúak

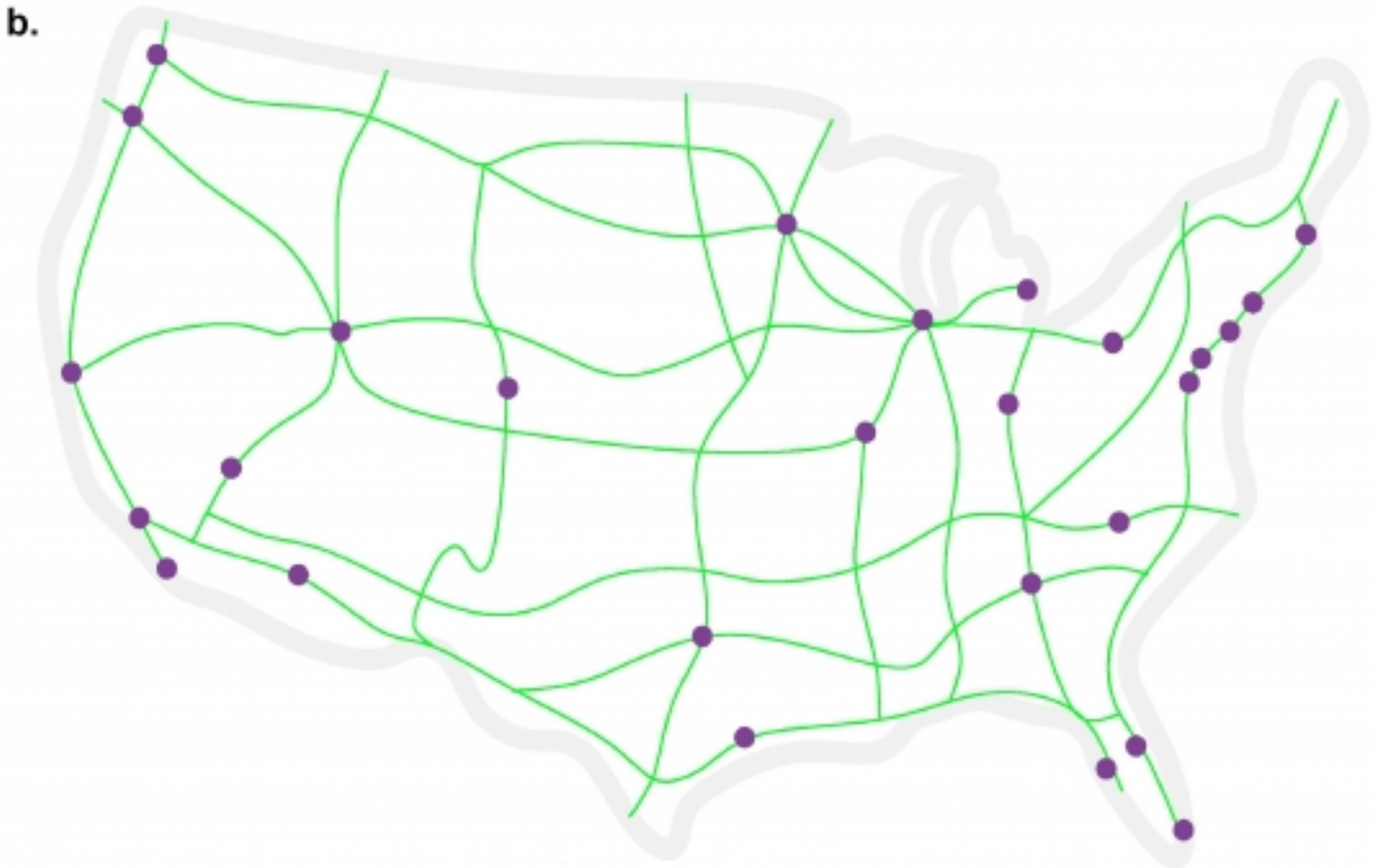


Ezek bármilyen adatbázisban teljesülhetnek,
de gráfadatbázisban gyakoribbak!

Hálózatkutatás *(network science)*

Poisson eloszlás

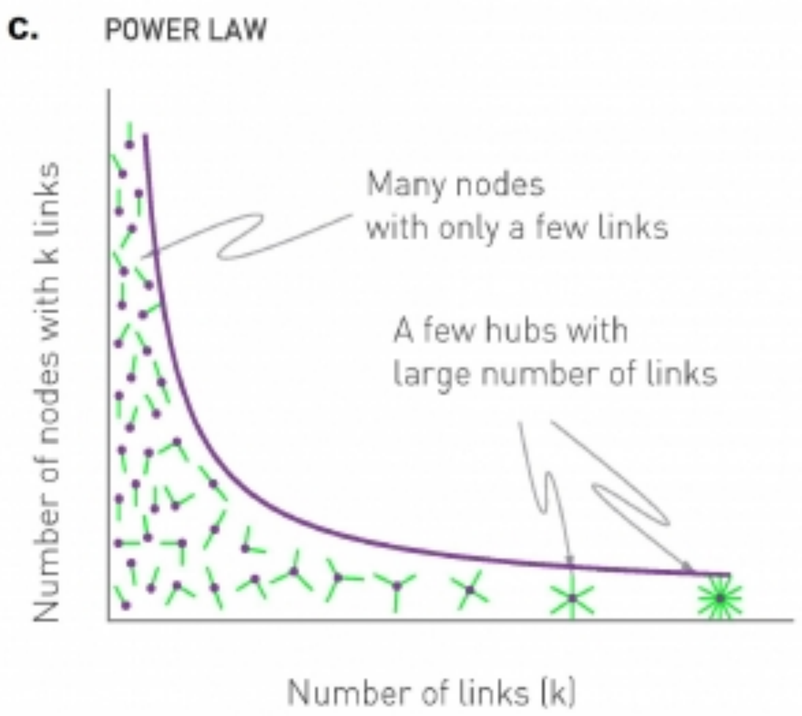




Forrás – Barabási Albert-László: *Network Science*

Hálózatkutatás *(network science)*

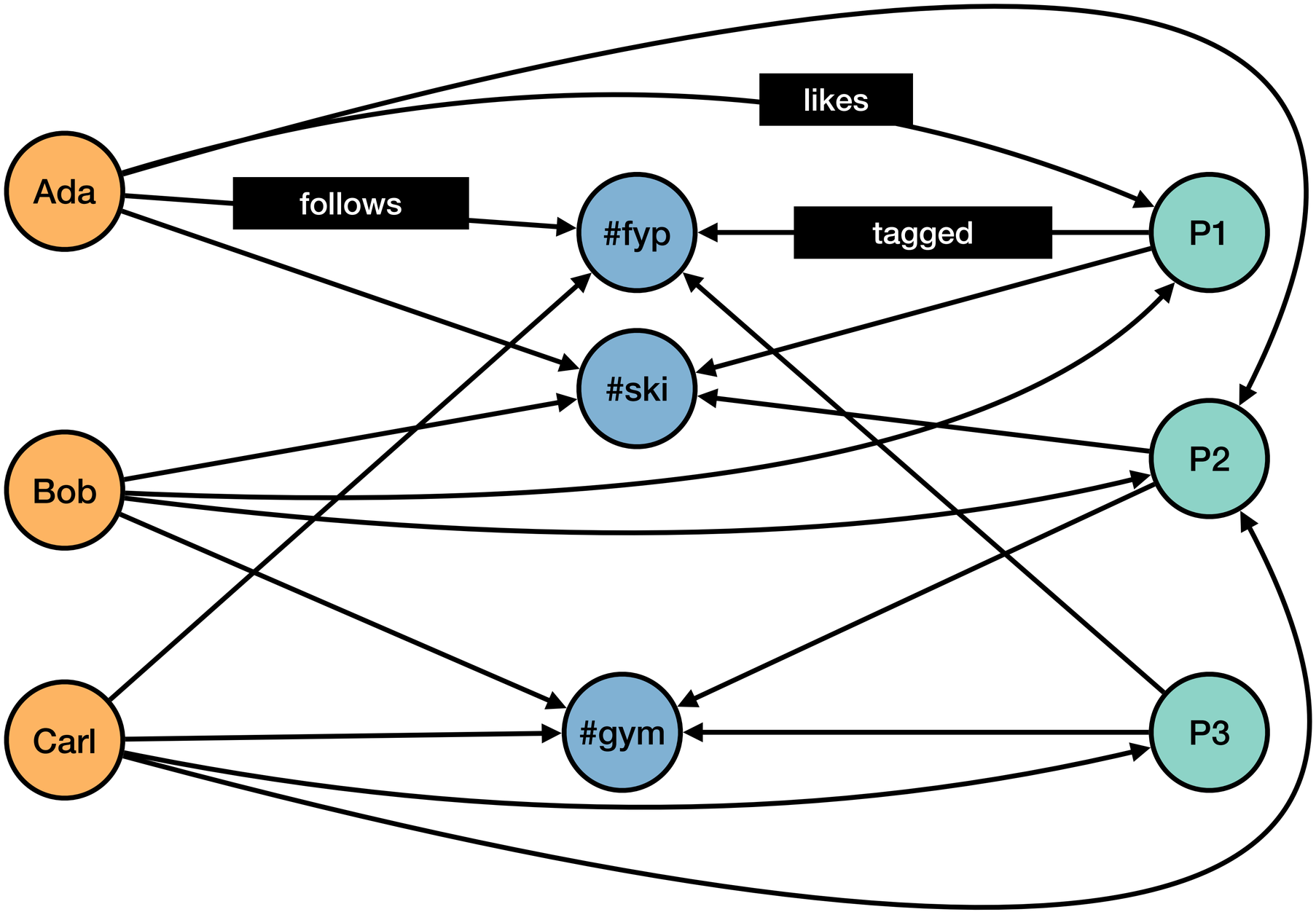
Hatványfüggvény eloszlás *(power law distribution)*

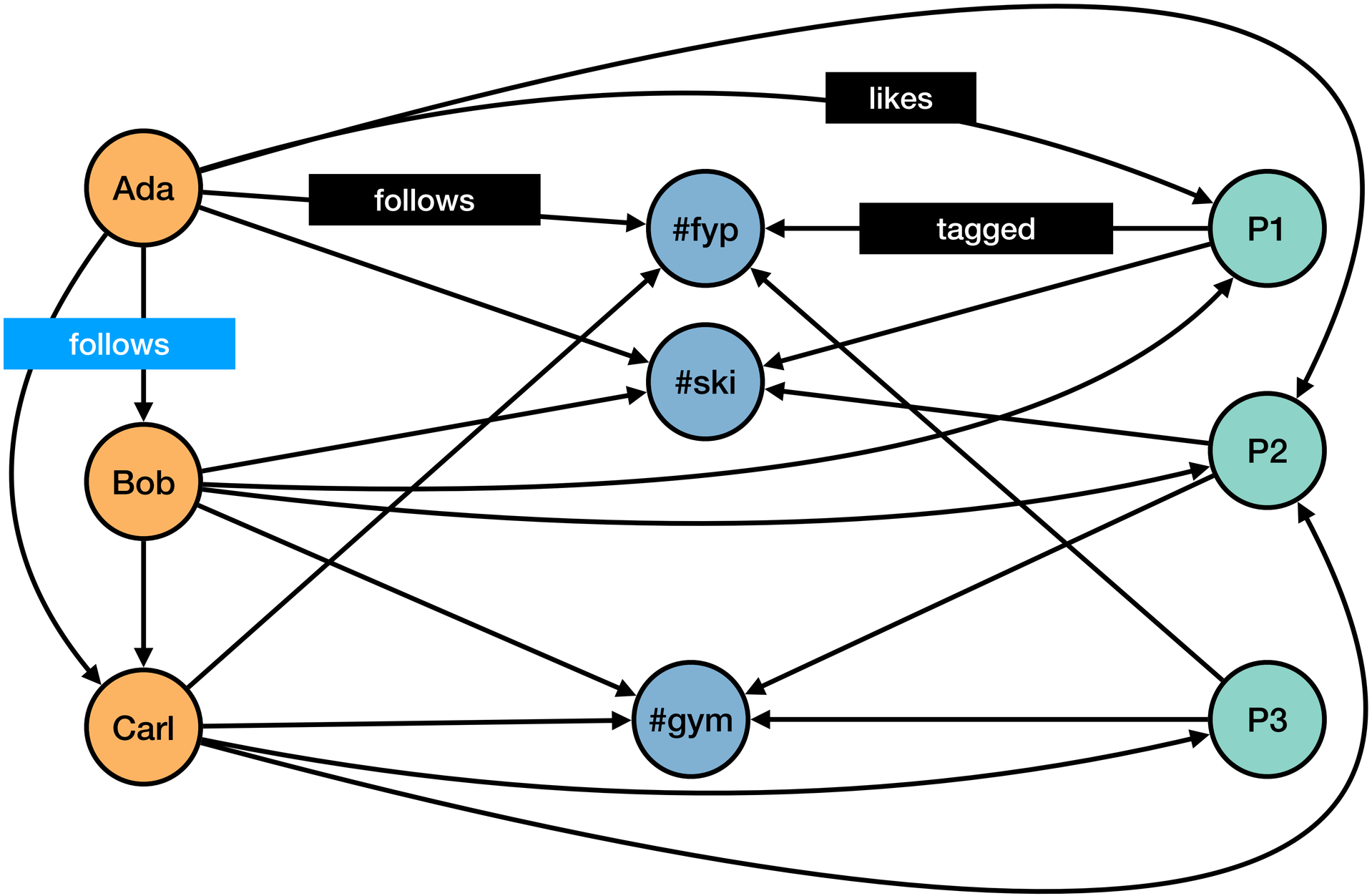


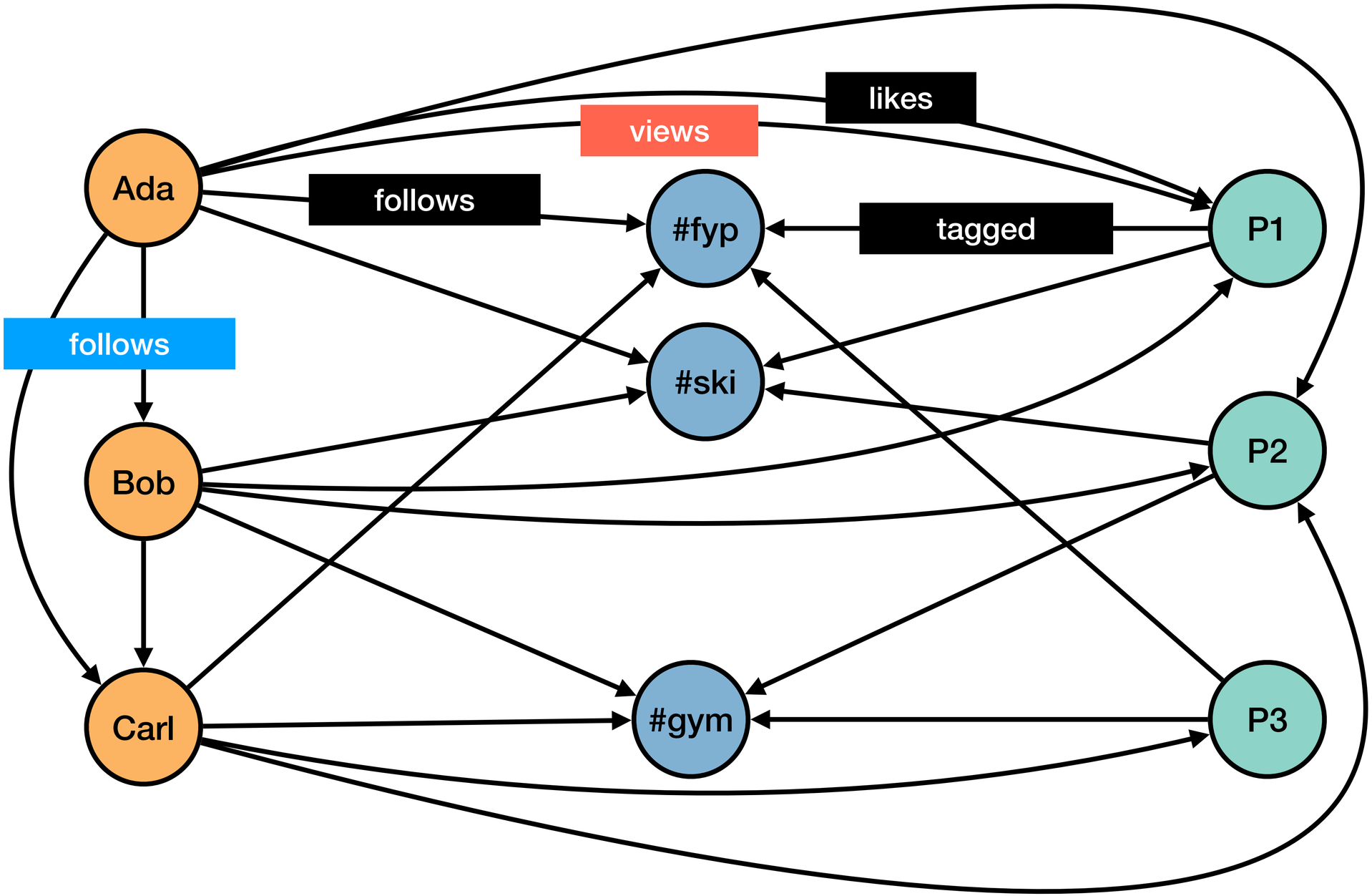
Forrás – Barabási Albert-László: *Network Science*

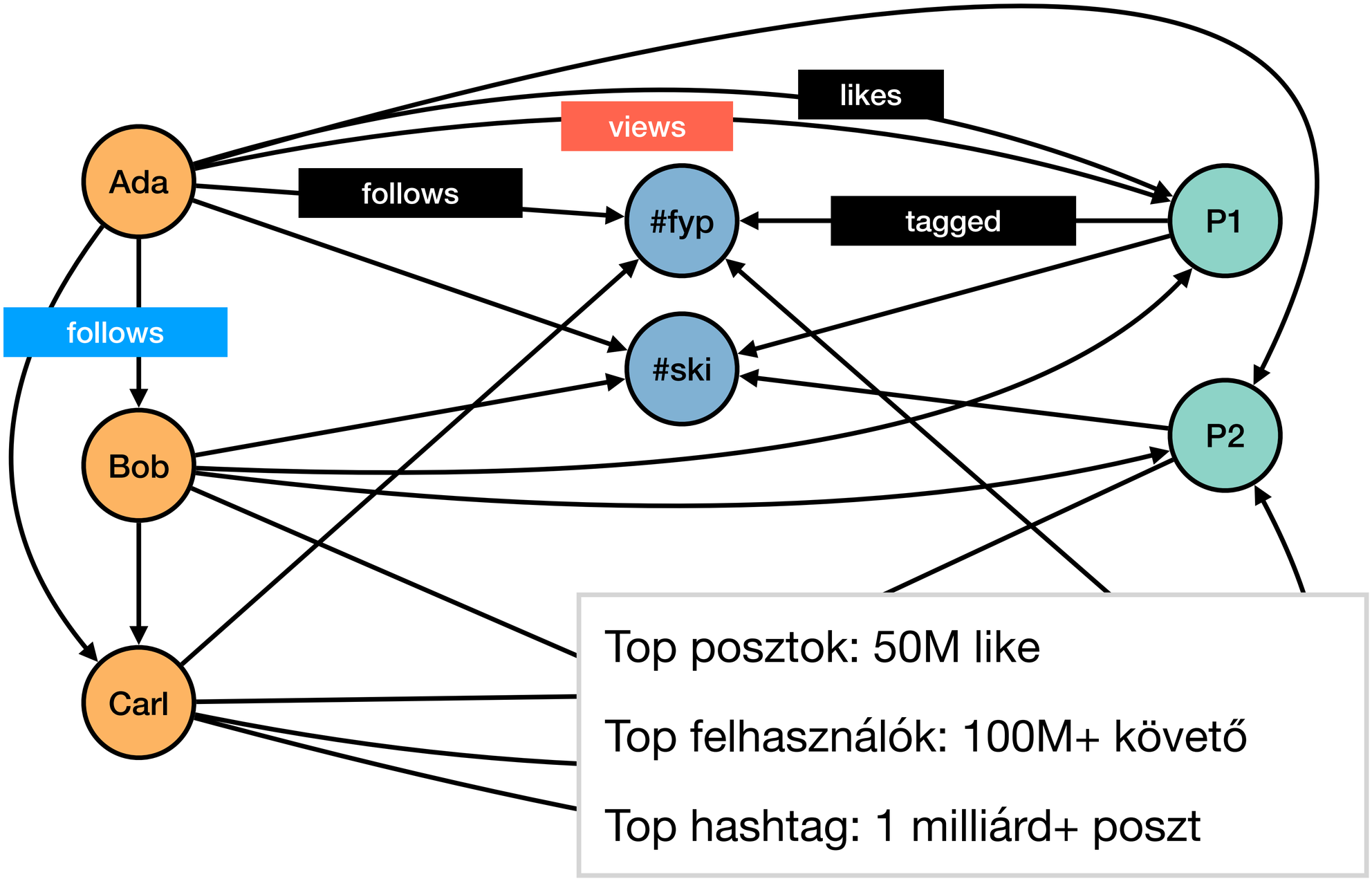
Skálafüggetlen hálózatok

- *Preferential attachment:* “a gazdag még gazdagabb lesz”
- Hatványfüggvény eloszlás *(power law distribution)*
- Skálafüggetlen *(scale-free, small-world)* hálózatok

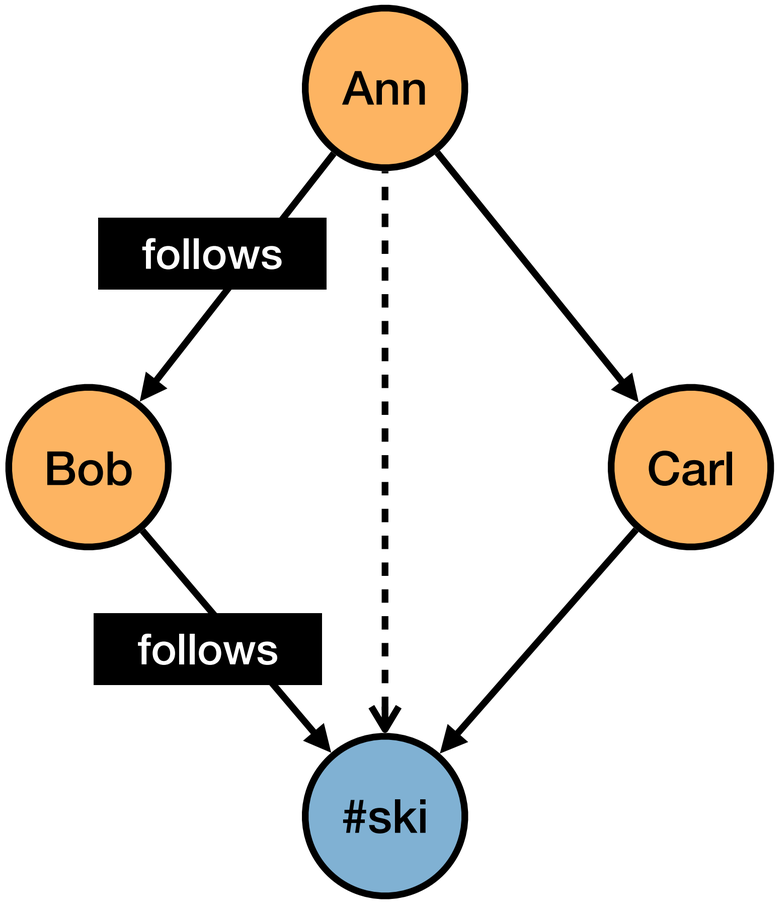

likes
Ada
follows
#fyp
tagged
P1
#ski
Bob
P2
Carl
#gym
P3


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

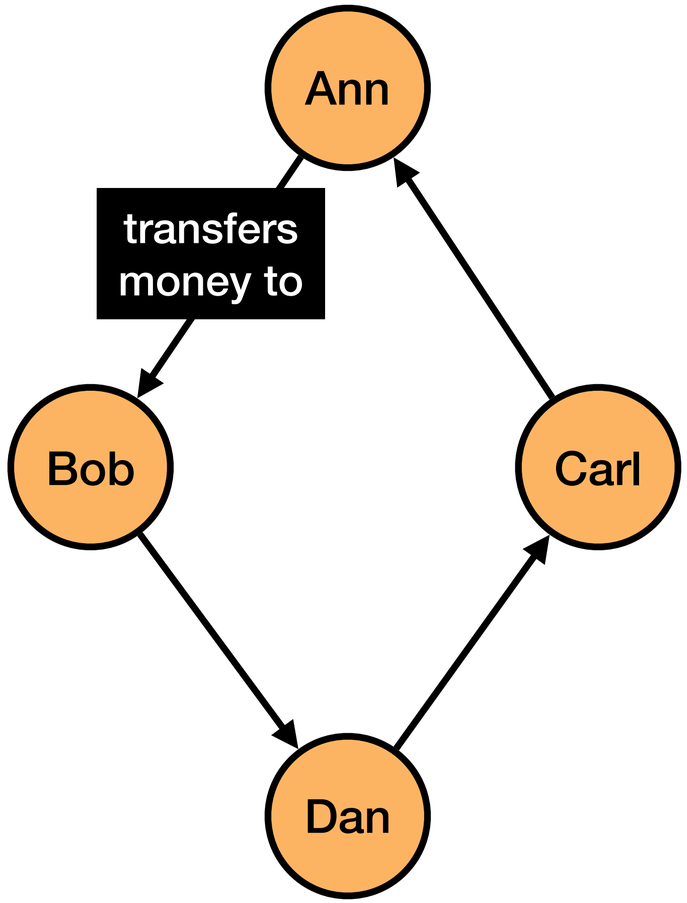

likes
views
Ada
follows
#fyp
tagged
P1
follows
#ski
Bob
P2
Carl
#gym
P3


likes
views
Ada
follows
#fyp
tagged
P1
follows
#ski
Bob
P2
Carl
Top posztok: 50M like
Top felhasználók: 100M+ követő
Top hashtag: 1 milliárd+ poszt

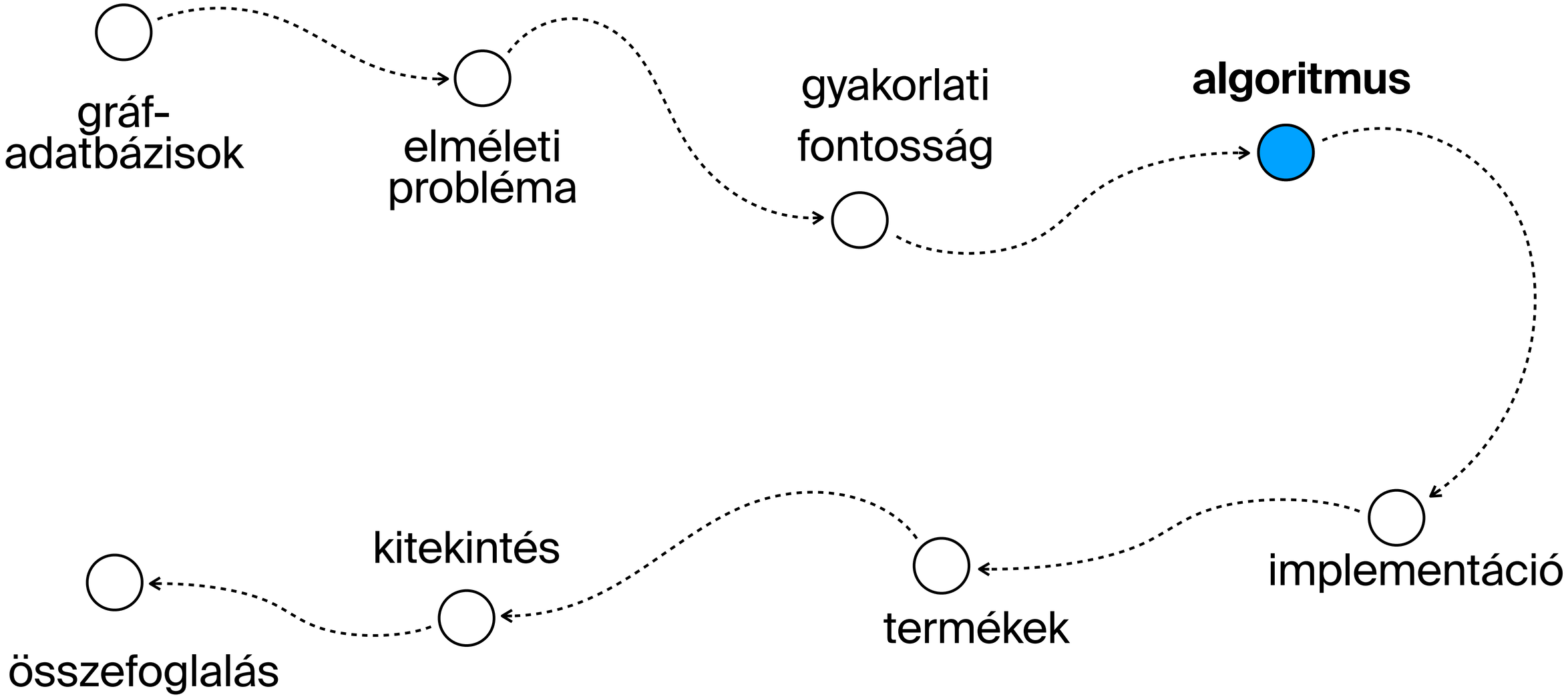
Ciklikus gráflekérdezések a gyakorlatban



ajánlóalgoritmus



pénzmosás felderítése


gráf-
adatbázisok
elméleti
probléma
gyakorlati
fontosság
algoritmus
implementáció
termékek
kitekintés
összefoglalás

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:

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

Algoritmus a $q_{\triangle}$ lekérdezés kiszámítására
$O(n^{1.5})$ lépésszámmal

Az illesztés művelet hagyományosan egy bináris operátor:

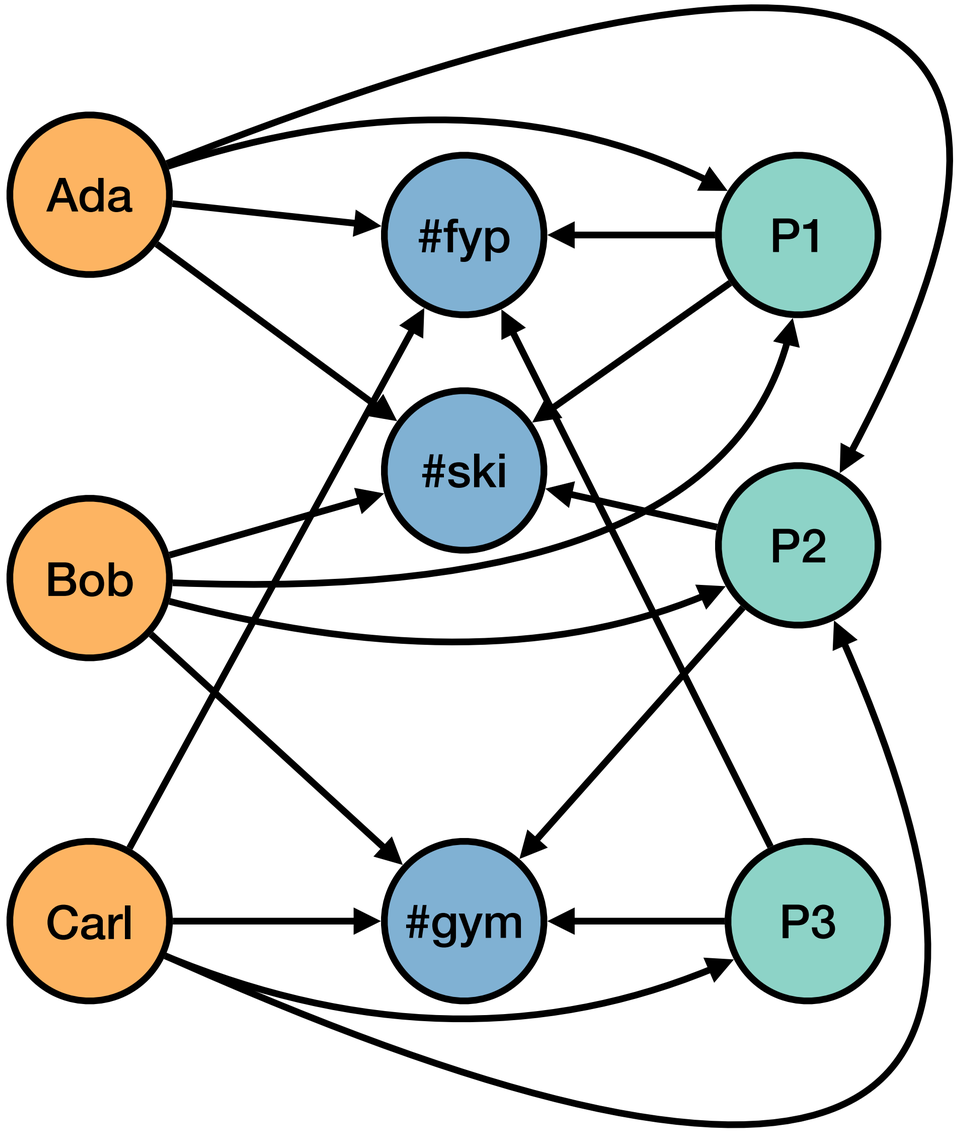
$$\texttt{follows} \bowtie \texttt{tagged} \bowtie \texttt{likes}$$

Használjunk n-áris illesztés *(multiway join)* műveletet:

$$\bowtie (\texttt{follows}, \texttt{tagged}, \texttt{likes})$$

A cikk ad két algoritmust, amik ismételt metszetképzéseket használnak.

Ezek az aszimptotikusan optimális illesztés algoritmusok
(worst-case optimal join algorithm).



follows

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

tagged

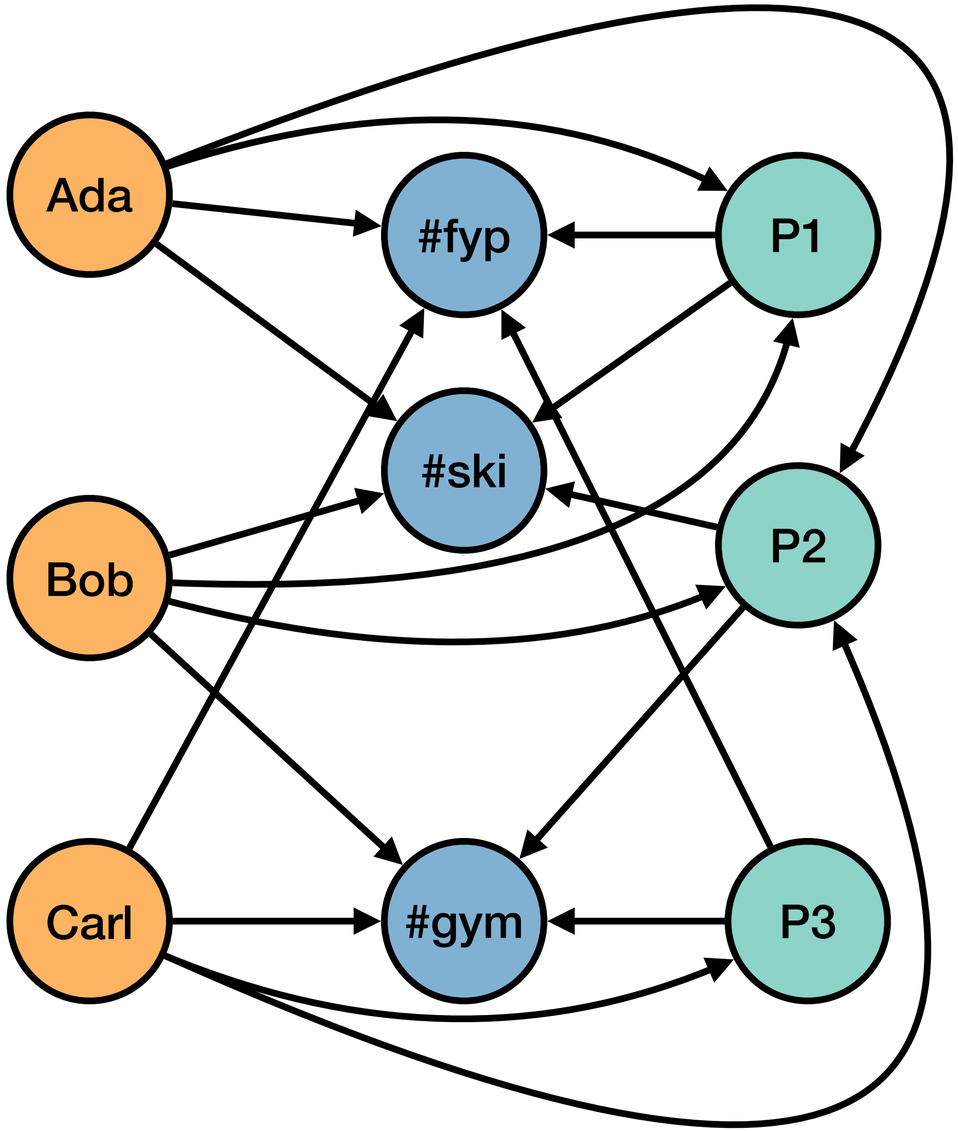
Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

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

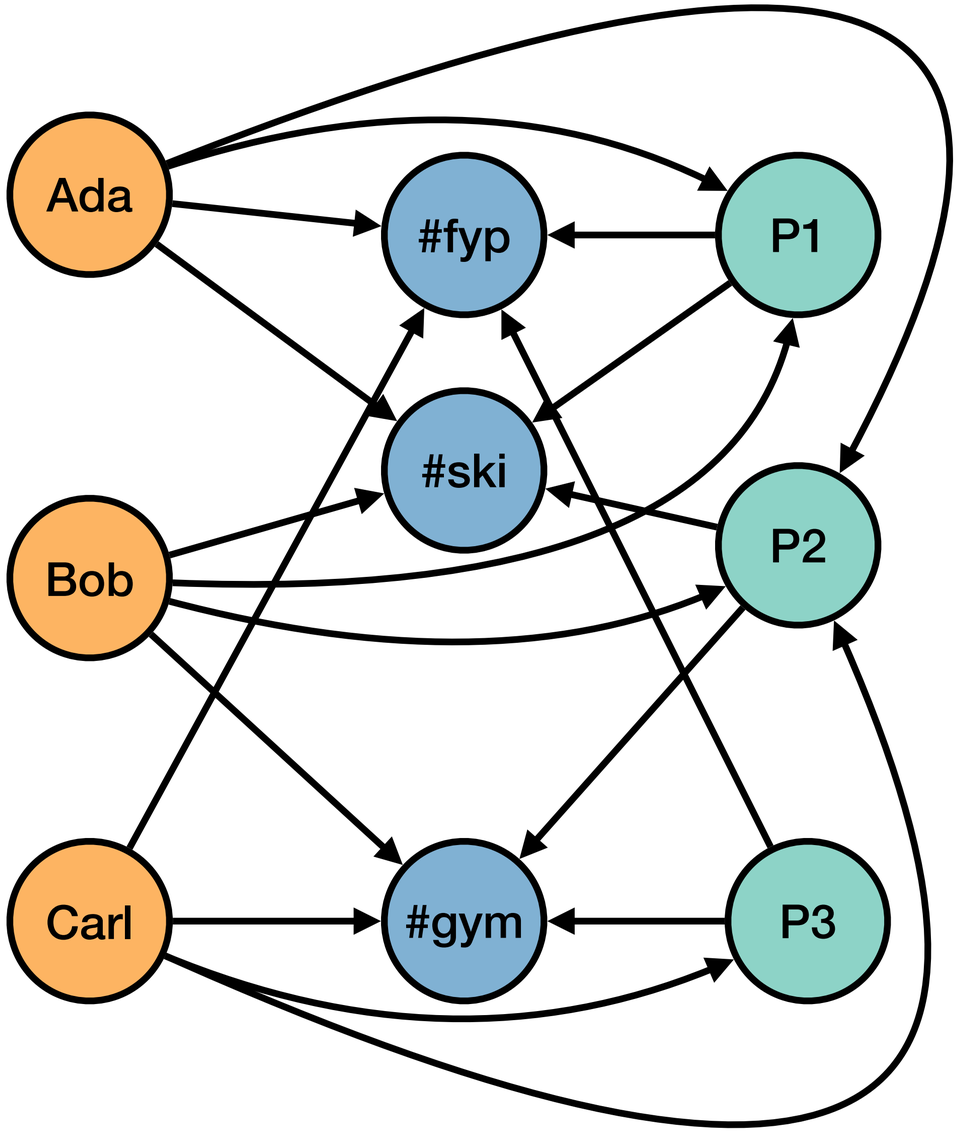
likes

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

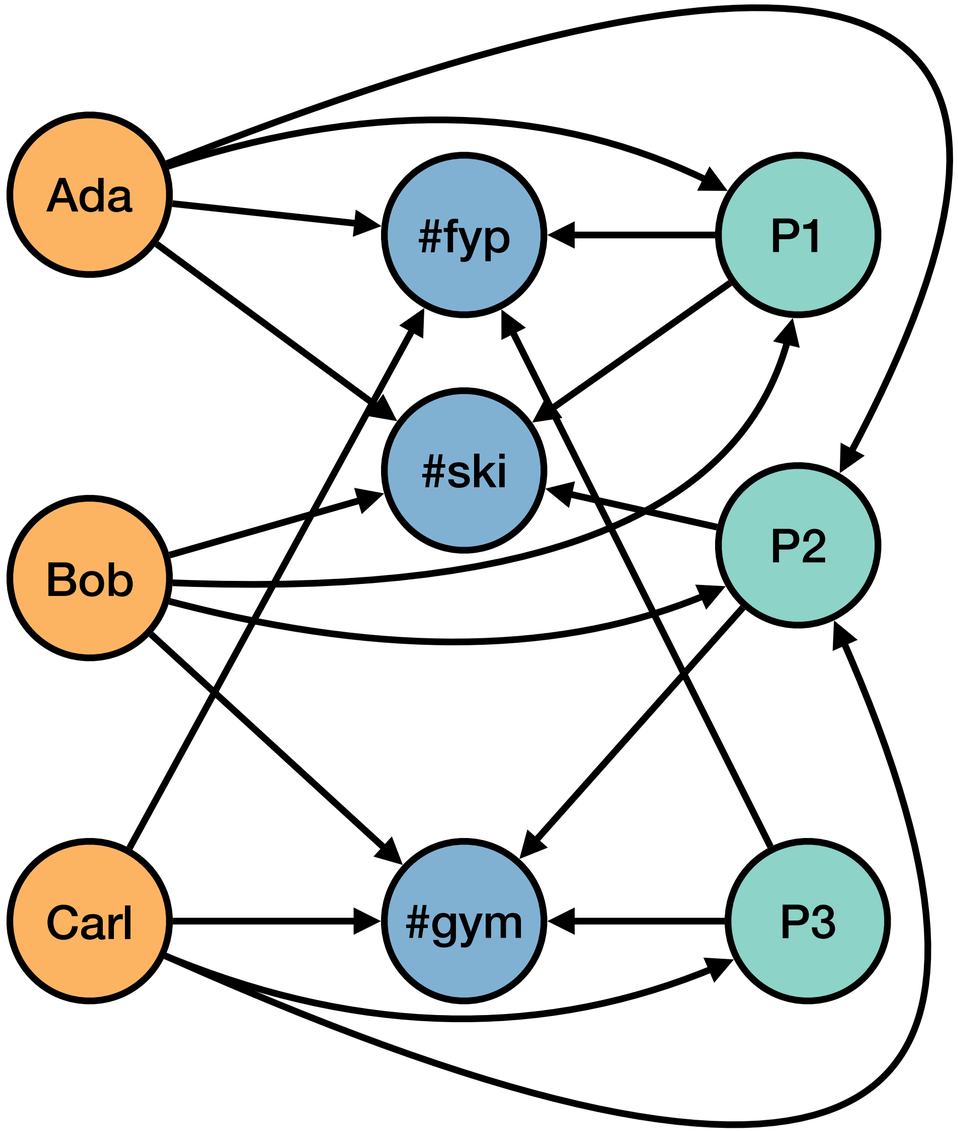
likes

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

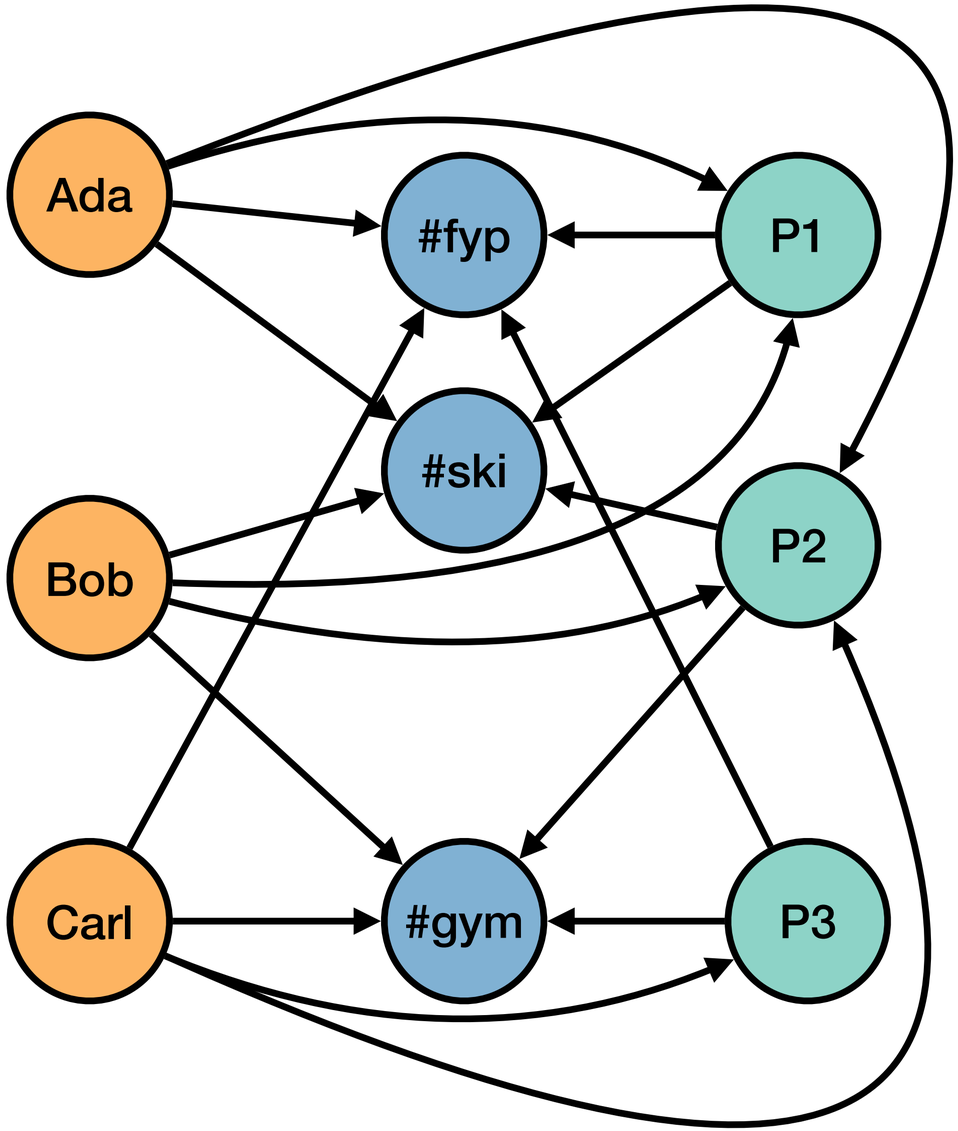
likes

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

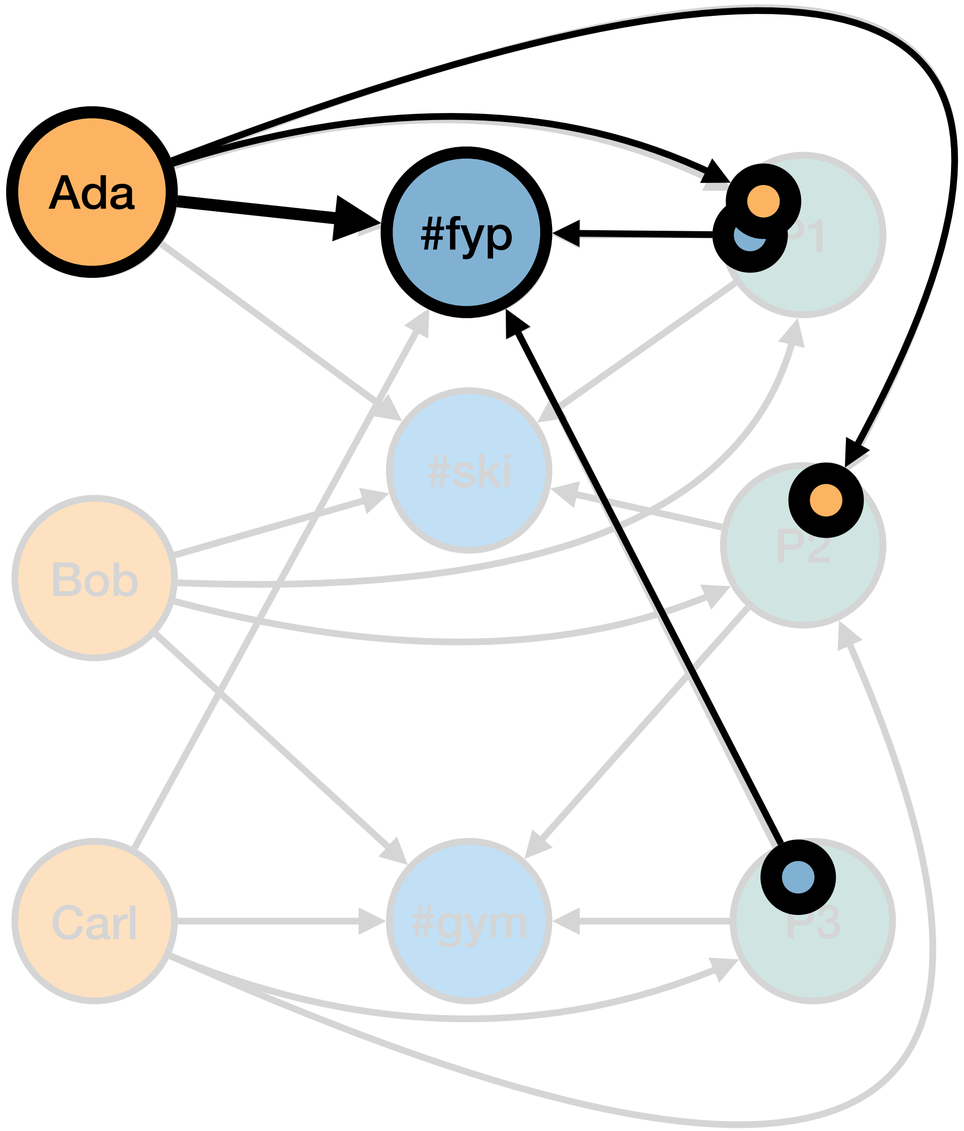
likes

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

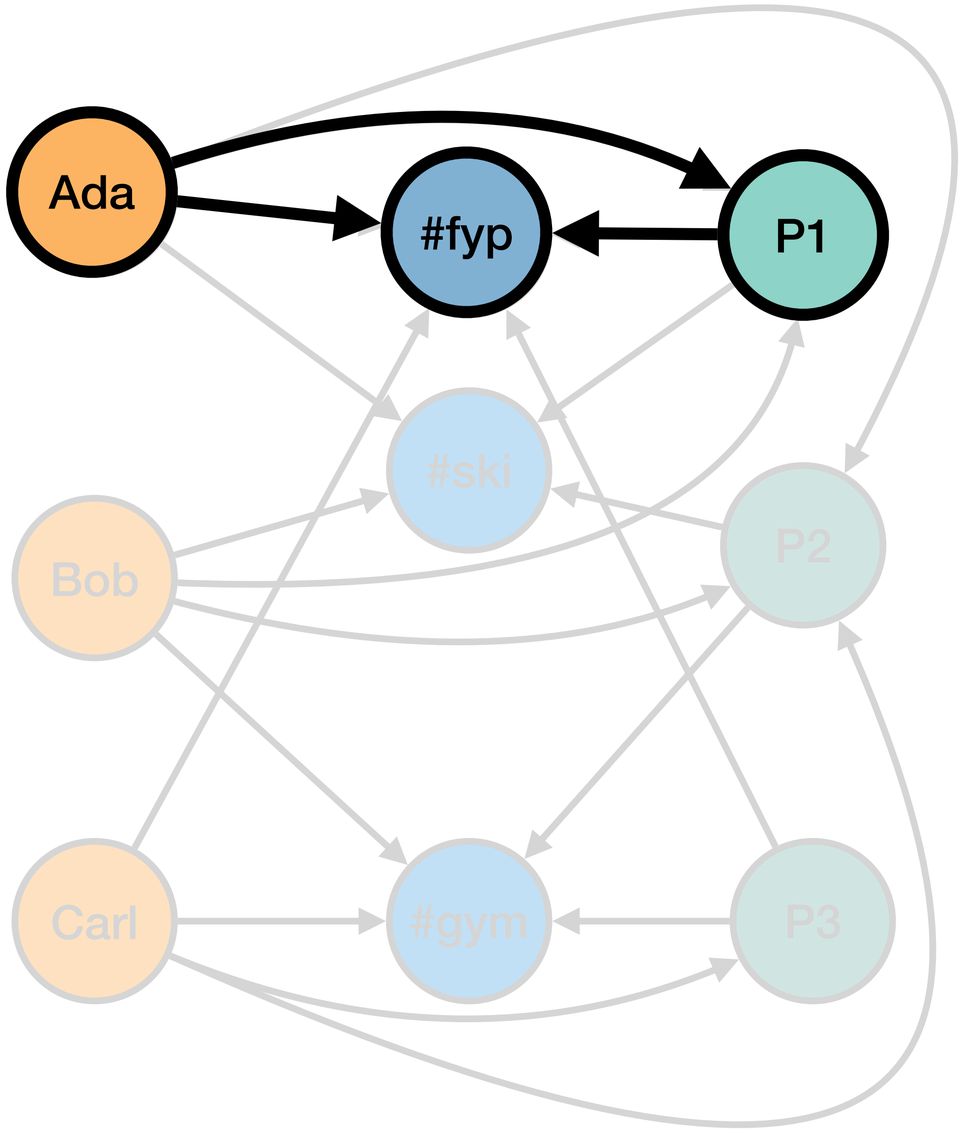
likes

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

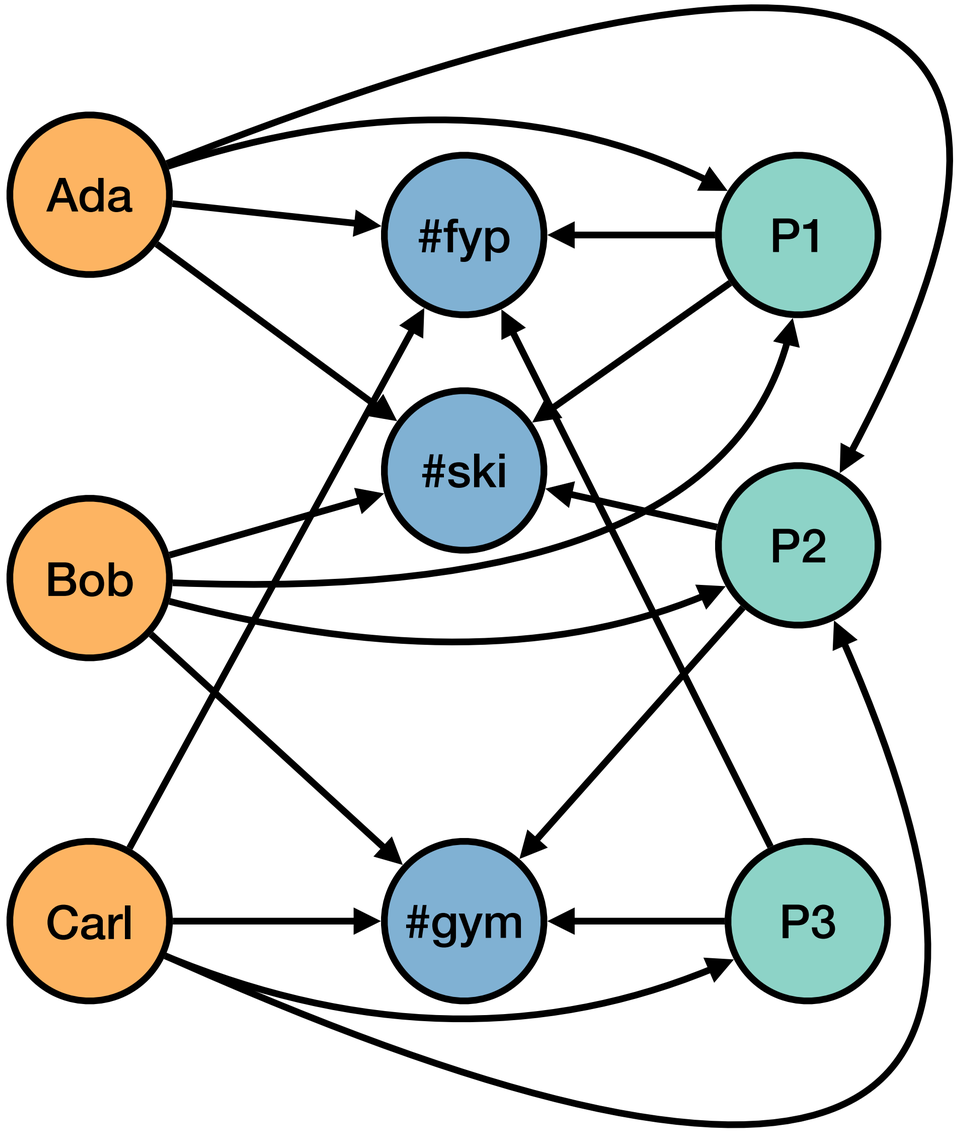
likes

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

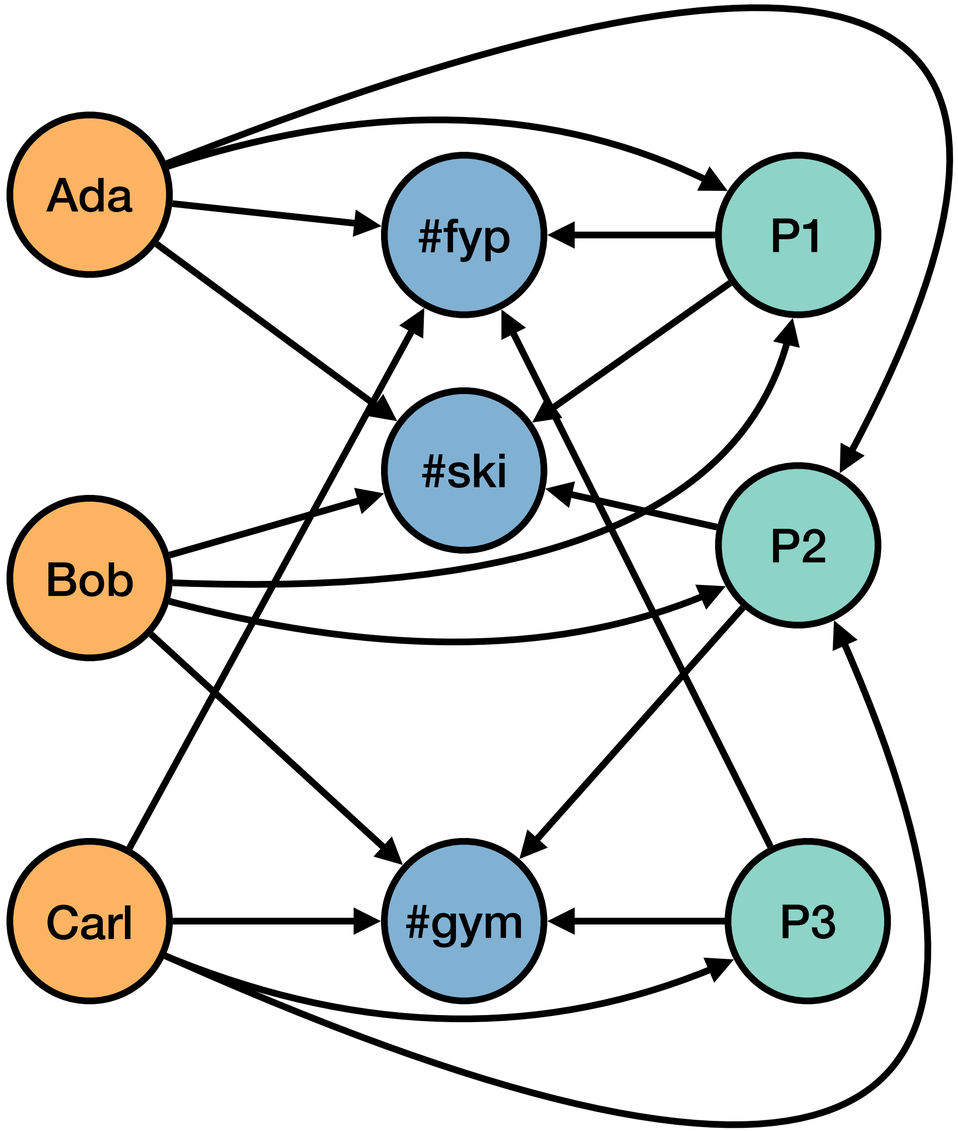
likes

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

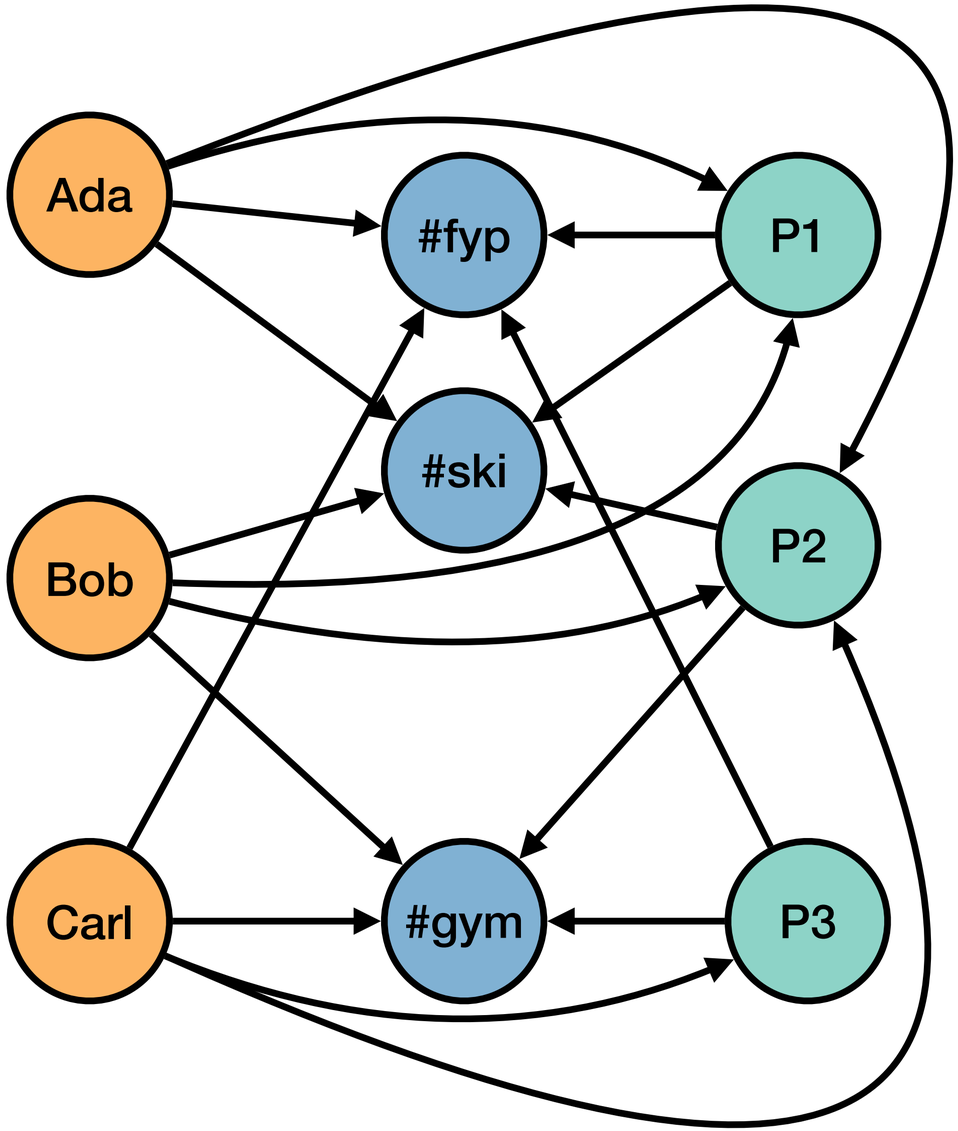
likes

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

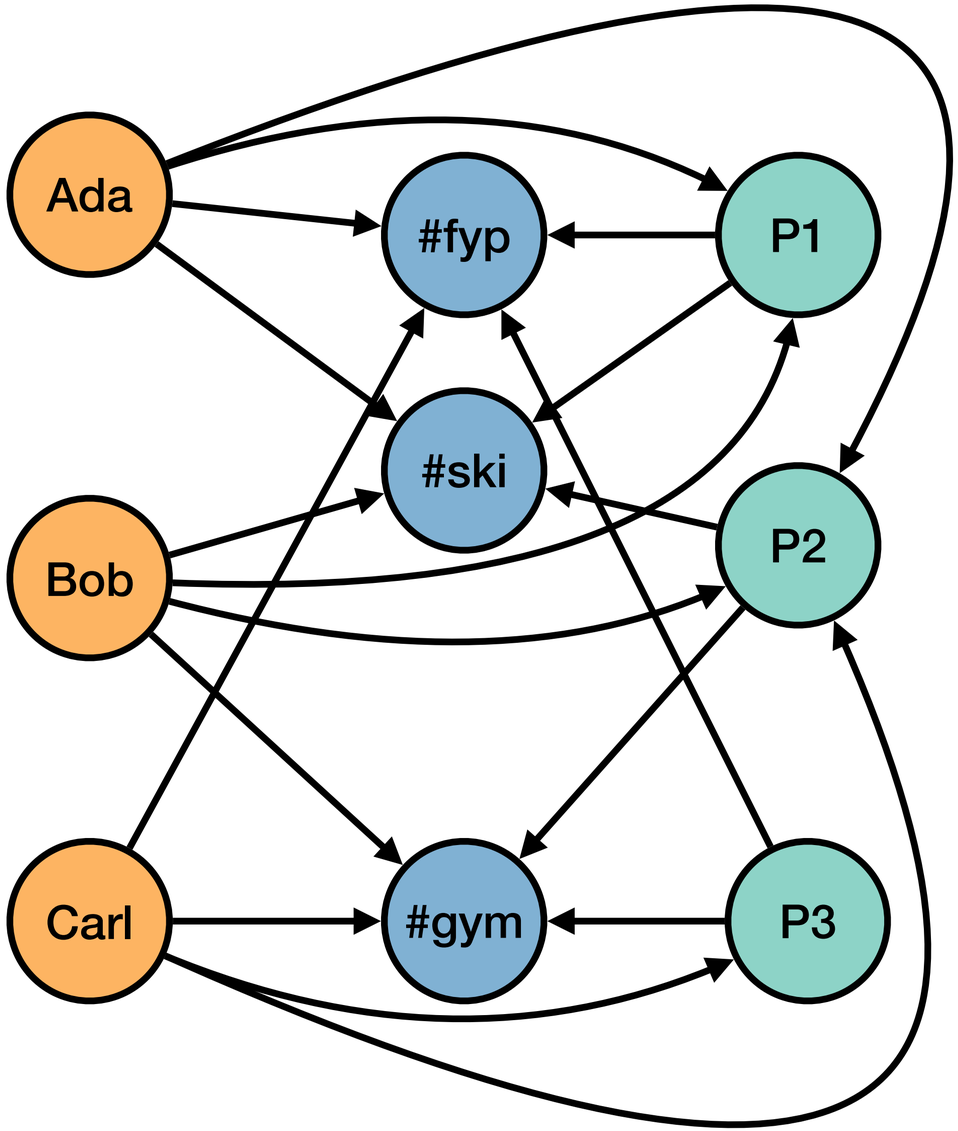
likes

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

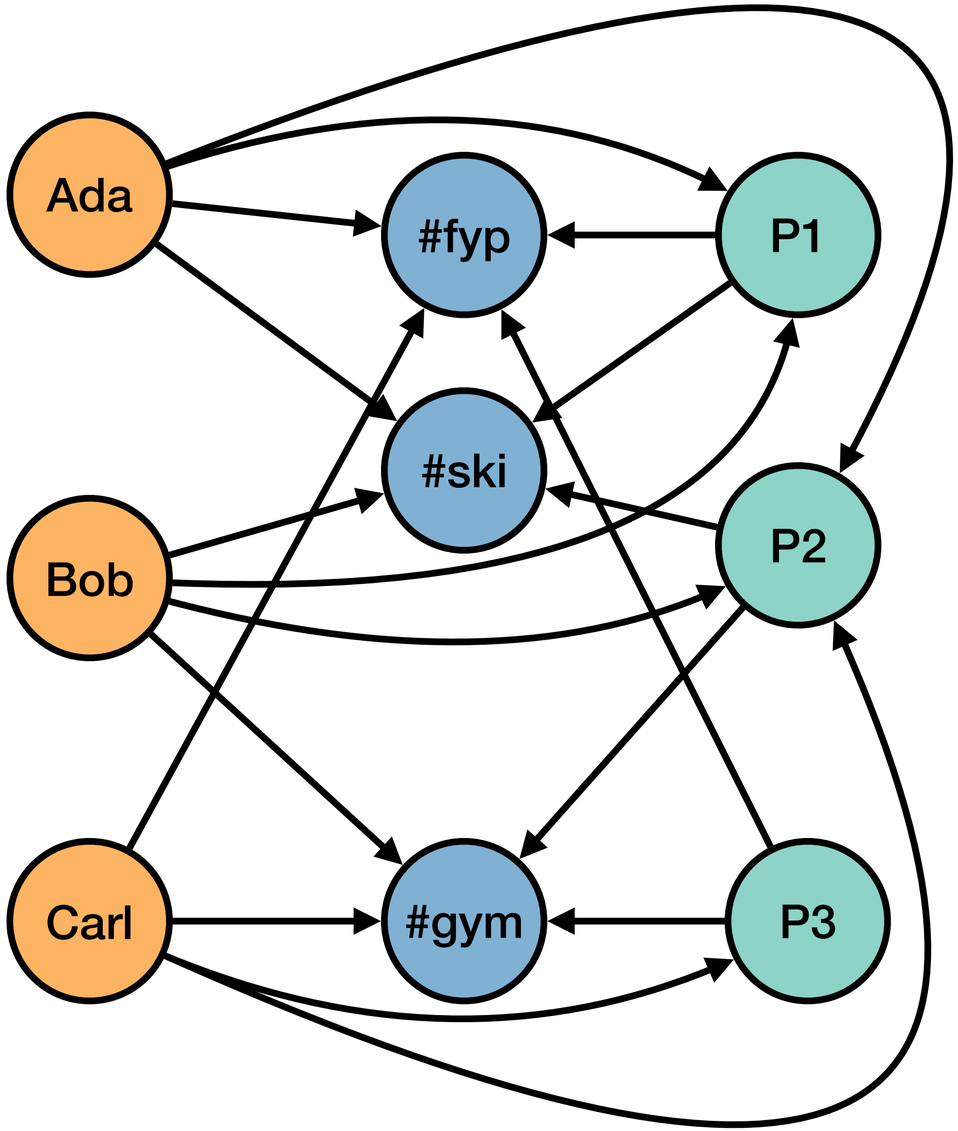
likes

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

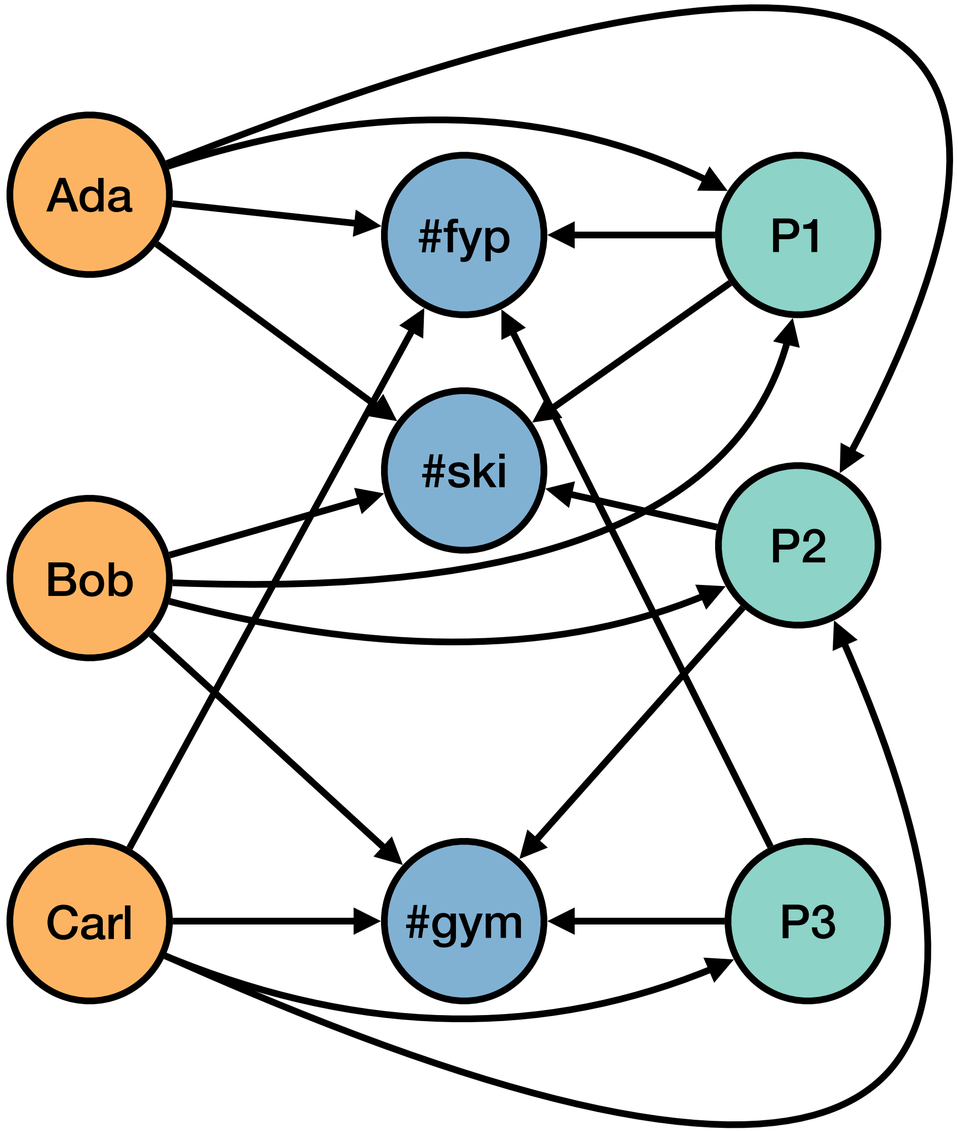
likes

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

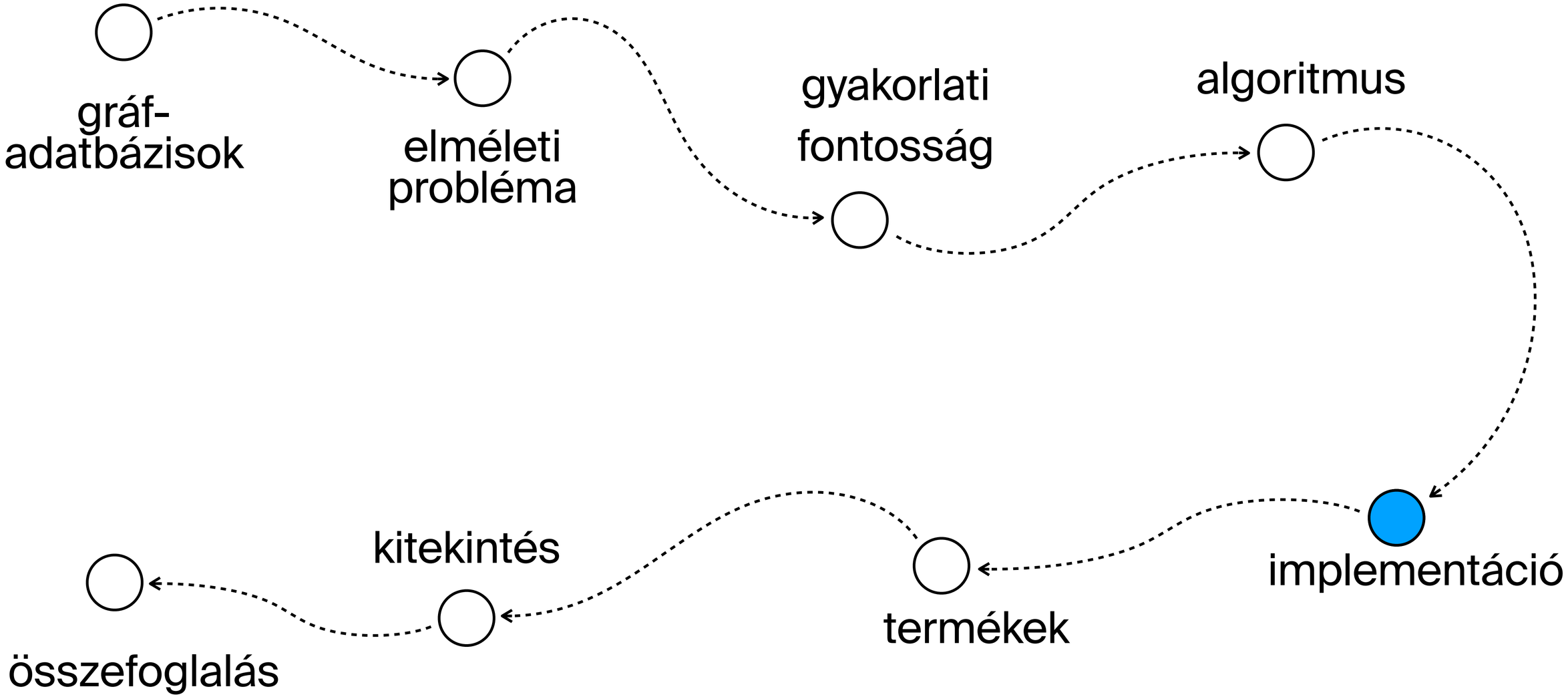
Very Large Databases 2019

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

Mikor válasszunk bináris vs. multiway join műveleteket, hogyan **sorrendezzük** ezeket?


gráf-
adatbázisok
elméleti
probléma
gyakorlati
fontosság
algoritmus
implementáció
termékek
kitekintés
összefoglalás

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

International Conference on Data Engineering 2018

LevelHeaded: A Unified Engine for Business Intelligence and Linear Algebra Querying

Christopher R. Aberger, Andrew Lamb, Kunle Olukotun, Christopher Ré

Stanford University
{caberger,lamb,kunle,chrismre}@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

Sorrendezés helyett hash-tábla alapú hatékony implementáció

[...] existing implementations incur a sizable overhead in practice, primarily since they rely on suitable ordered index structures on their input.

The key component [...] is a novel hash-based worst-case optimal join algorithm

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* $O(n \cdot \sum_{E_j \in \mathcal{E}} |R_j|)$.

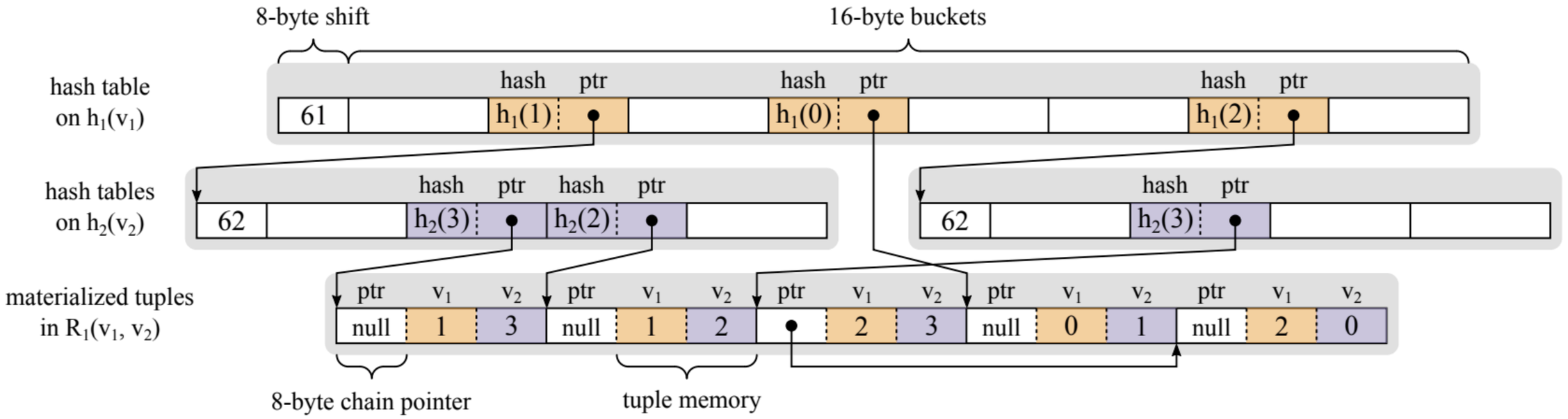
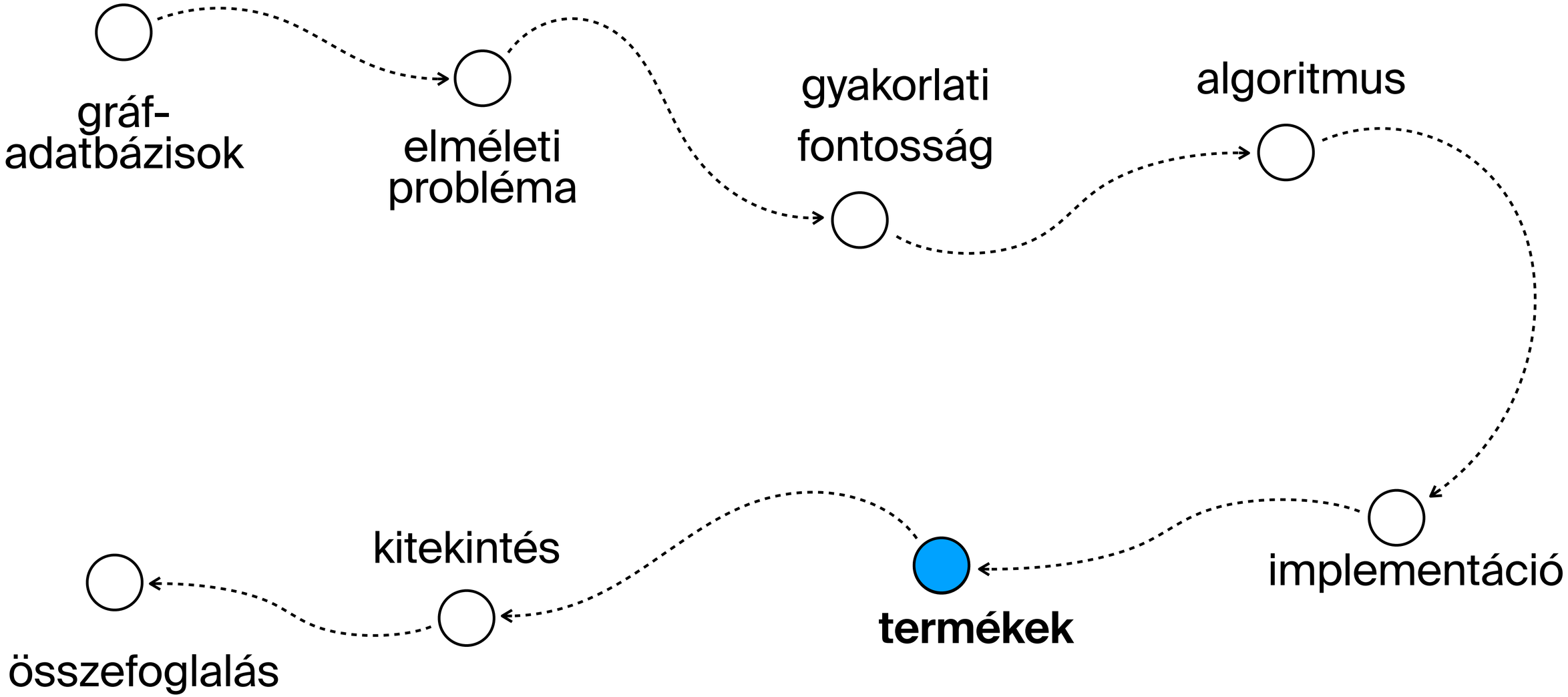


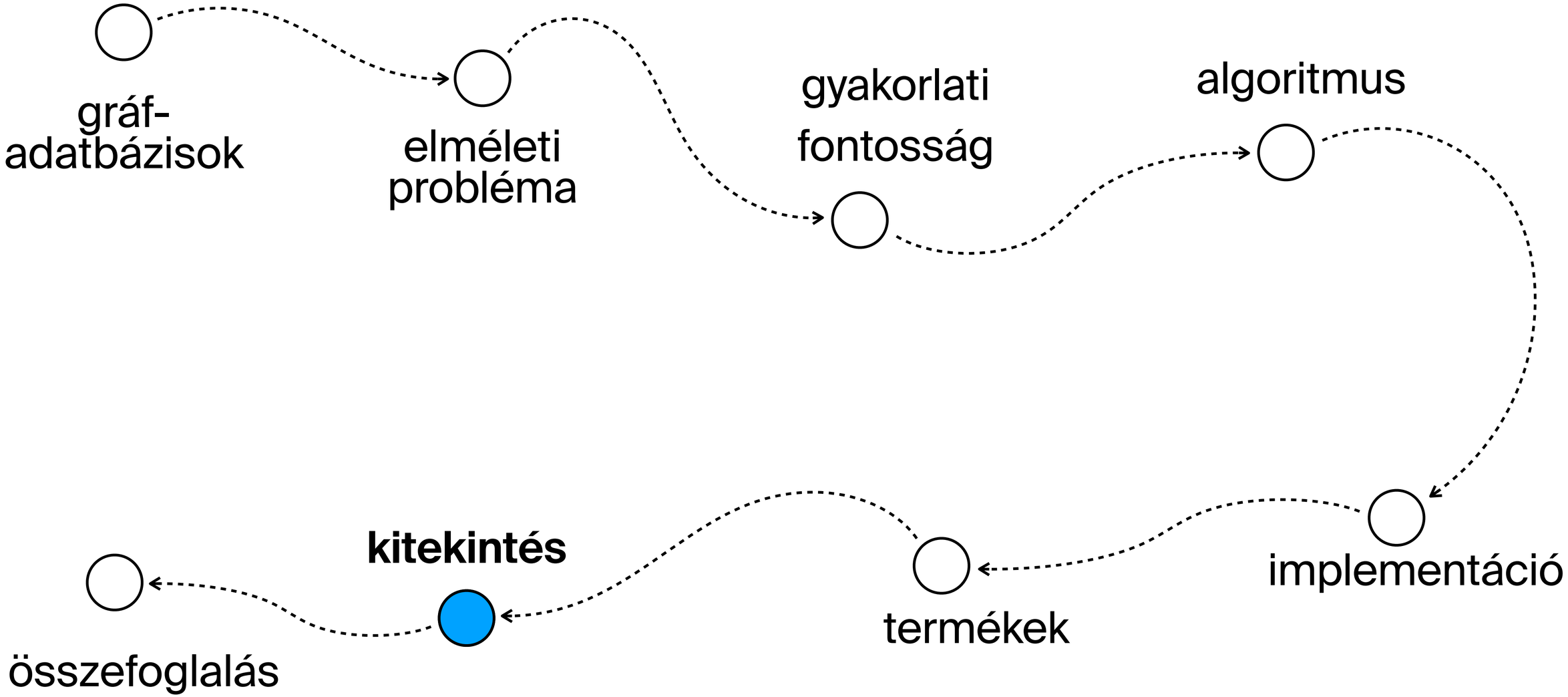
Figure 3: Memory layout of the hash trie in Figure 2. The gray boxes correspond to the individual hash tables and materialized input tuples. No nested hash table is built for the tuple $(0, 1)$ due to singleton pruning.

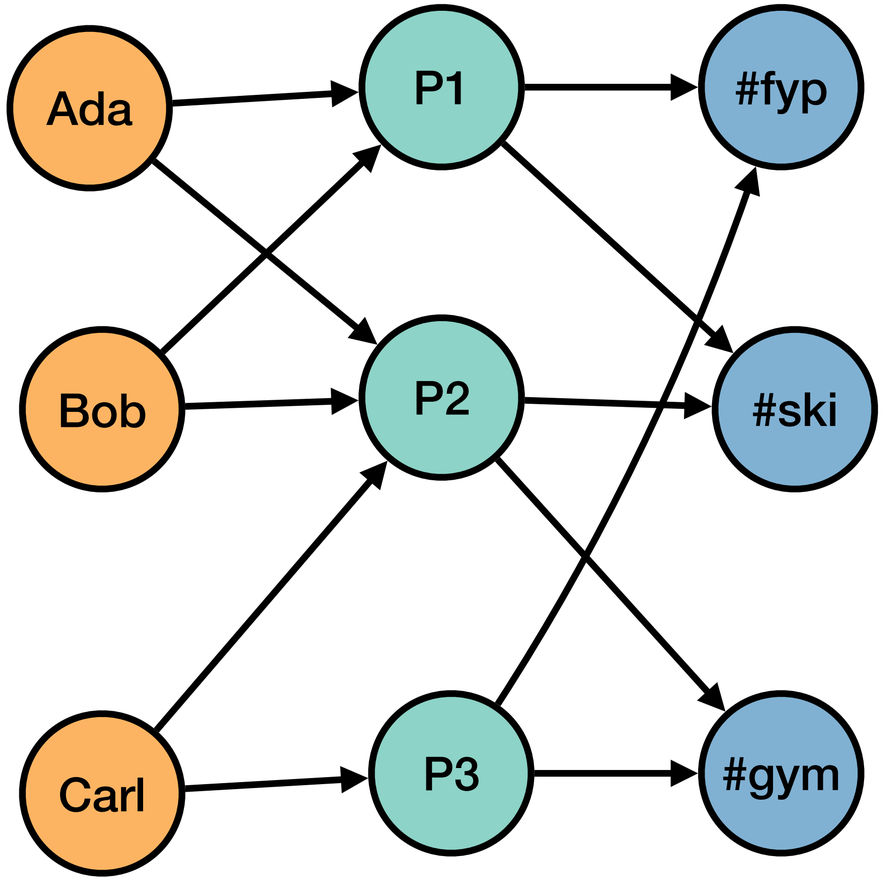

gráf-
adatbázisok
elméleti
probléma
gyakorlati
fontosság
algoritmus
implementáció
termékek
kitekintés
összefoglalás

Multiway joint támogató rendszerek

v0.5
2024 nyár

“v1.0”
2024 ősz


gráf-
adatbázisok
elméleti
probléma
gyakorlati
fontosság
algoritmus
implementáció
termékek
kitekintés
összefoglalás



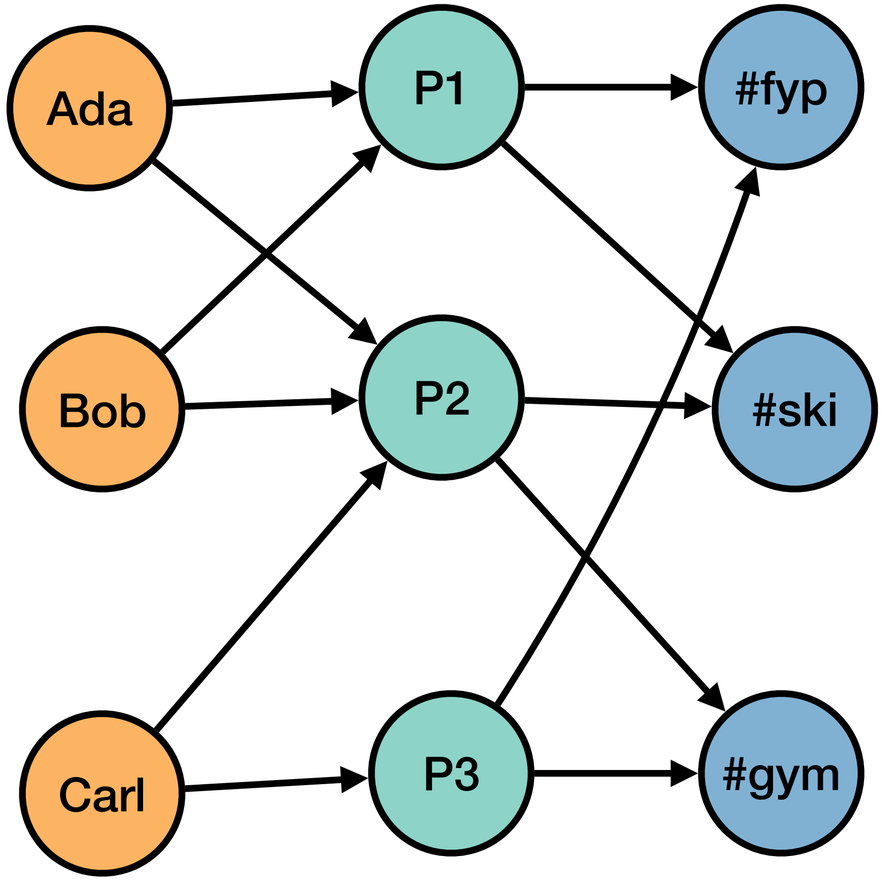

```
SELECT post
FROM likes
NATURAL JOIN tagged
GROUP BY post
HAVING count(distinct user) > count(distinct tag);
```

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

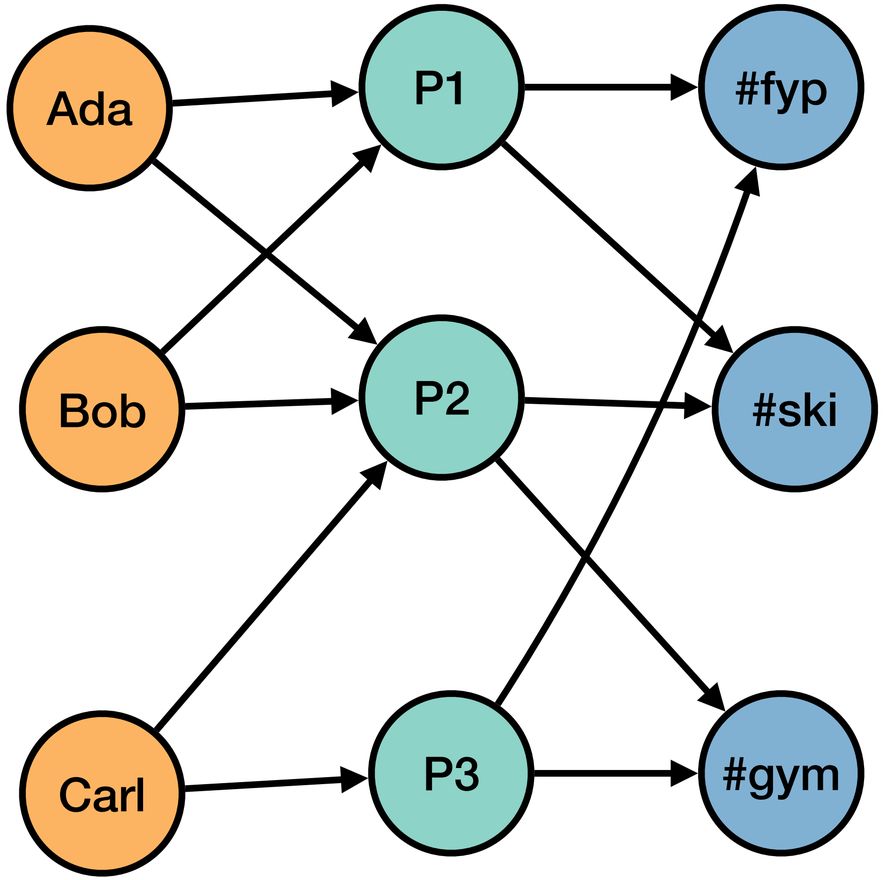
User	Post	Tag
Ada	P1	#ski
Bob	P1	#ski
Ada	P1	#fyp
Bob	P1	#fyp
Ada	P2	#gym
Bob	P2	#gym
Carl	P2	#gym
Ada	P2	#ski
Bob	P2	#ski
Carl	P2	#ski
Carl	P3	#gym
Carl	P3	#fyp

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

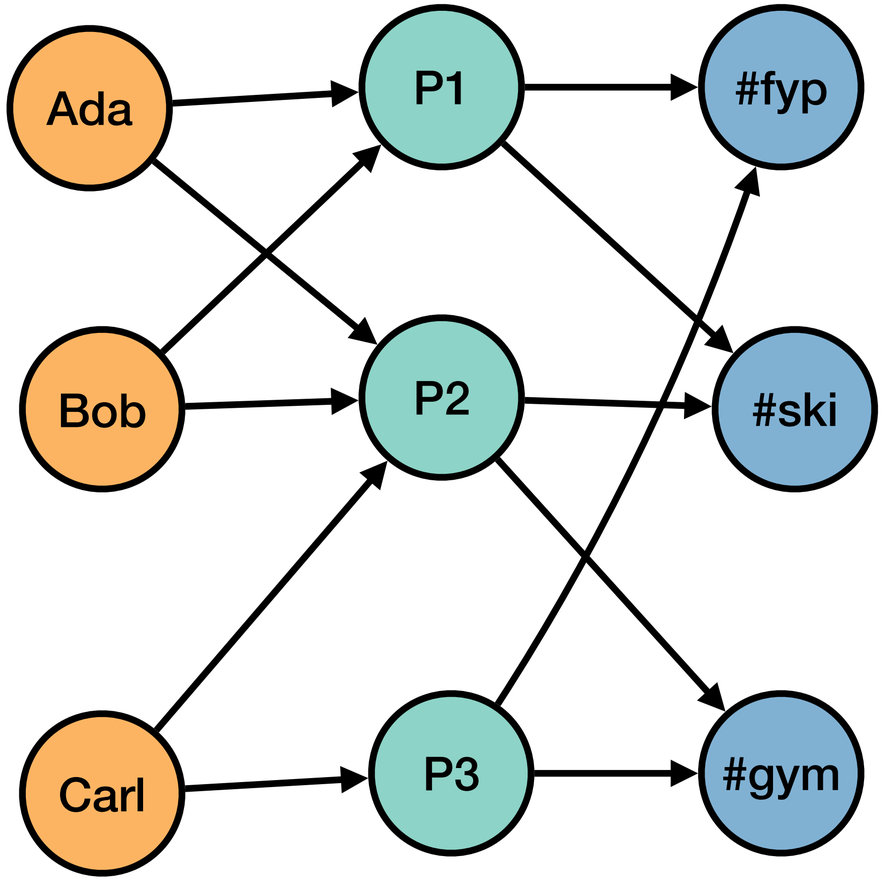
Post	Tag
P1	#fyp
P3	#fyp
P1	#ski
P2	#ski
P2	#gym
P3	#gym

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

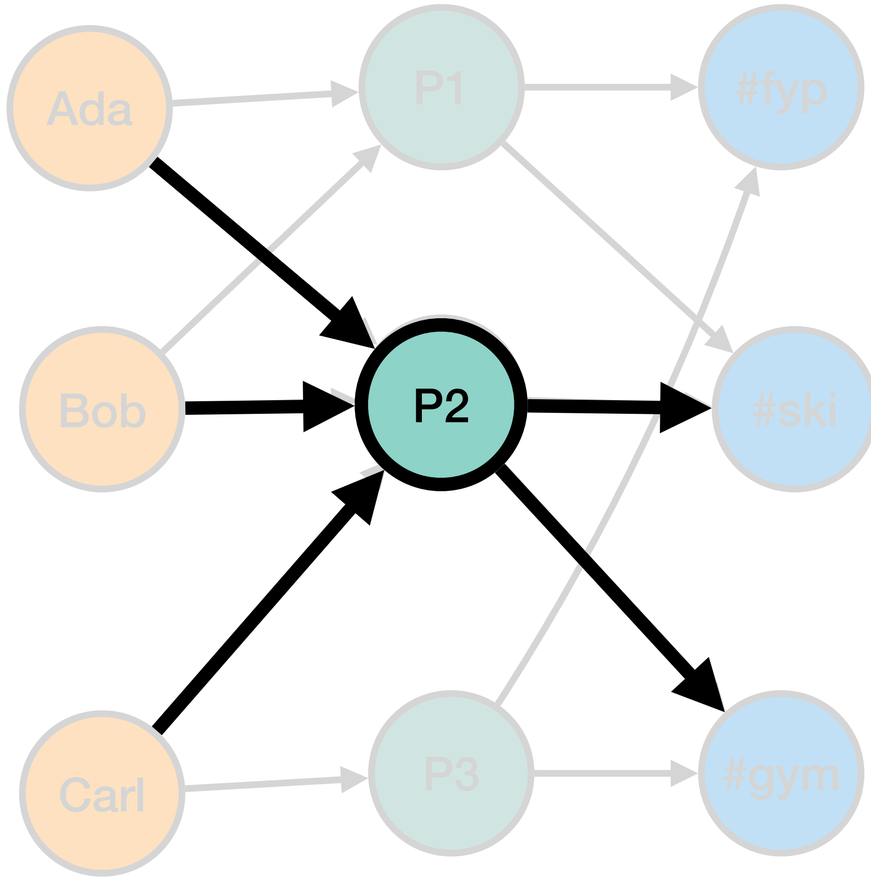
User	Post	Tag
Ada	P1	#ski
Bob	P1	#ski
Ada	P1	#fyp
Bob	P1	#fyp

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

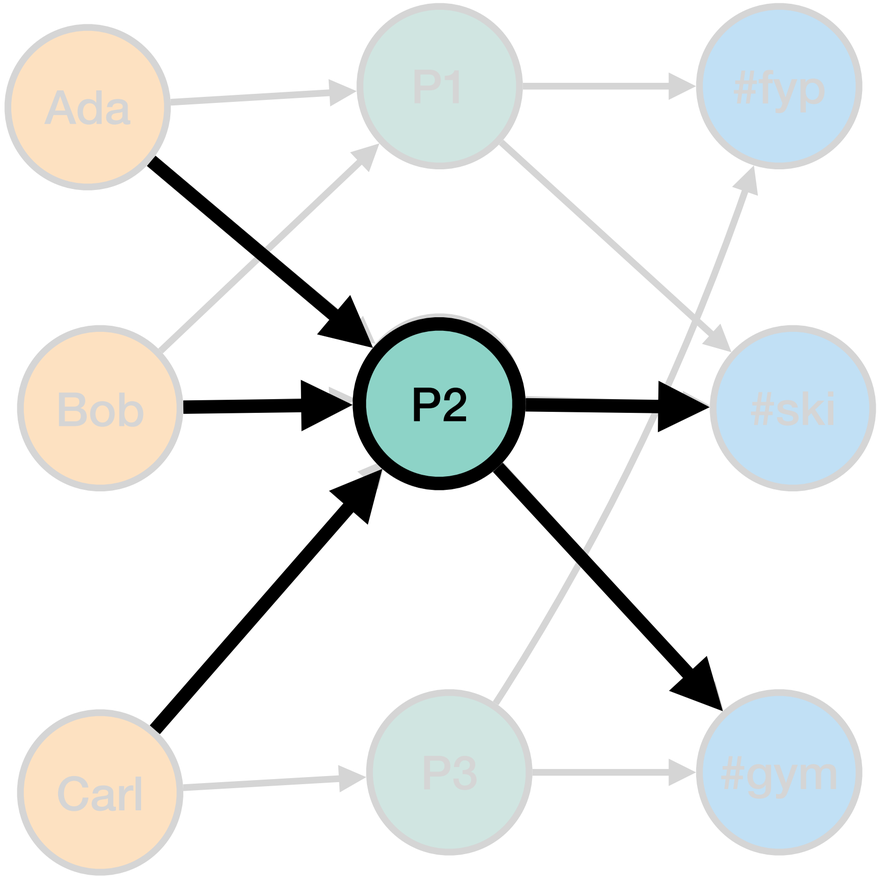
Post
P2

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);
```

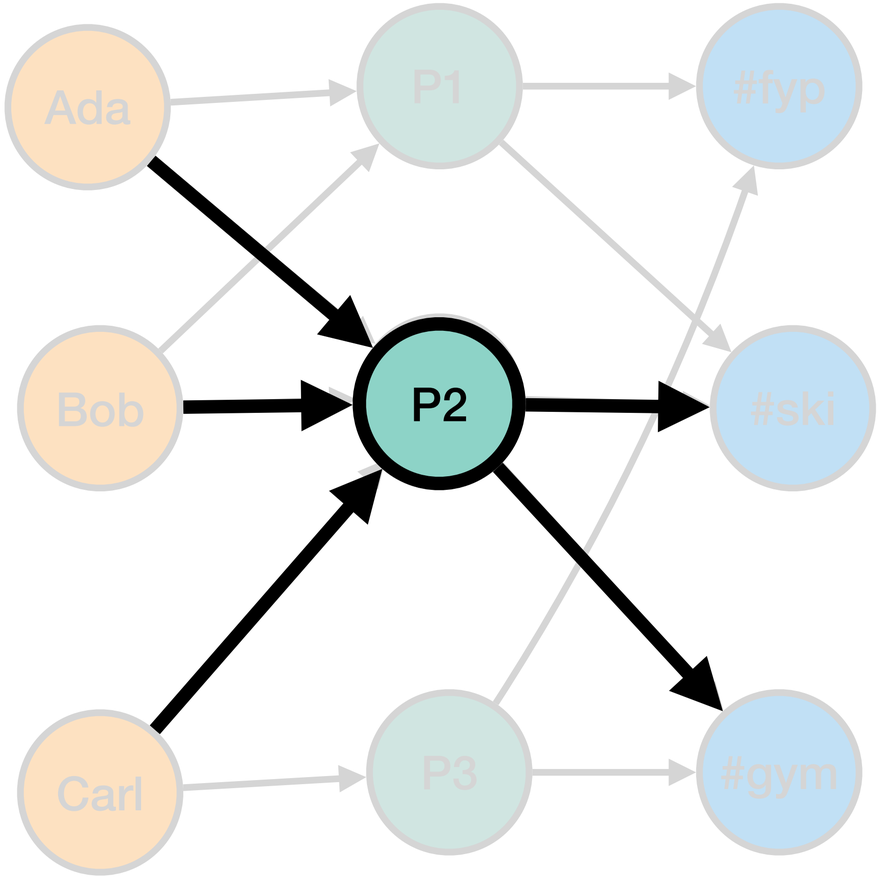
faktorizáció: veszteségmentes tömörítés

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);
```

faktorizáció: veszteségmentes tömörítés

likes

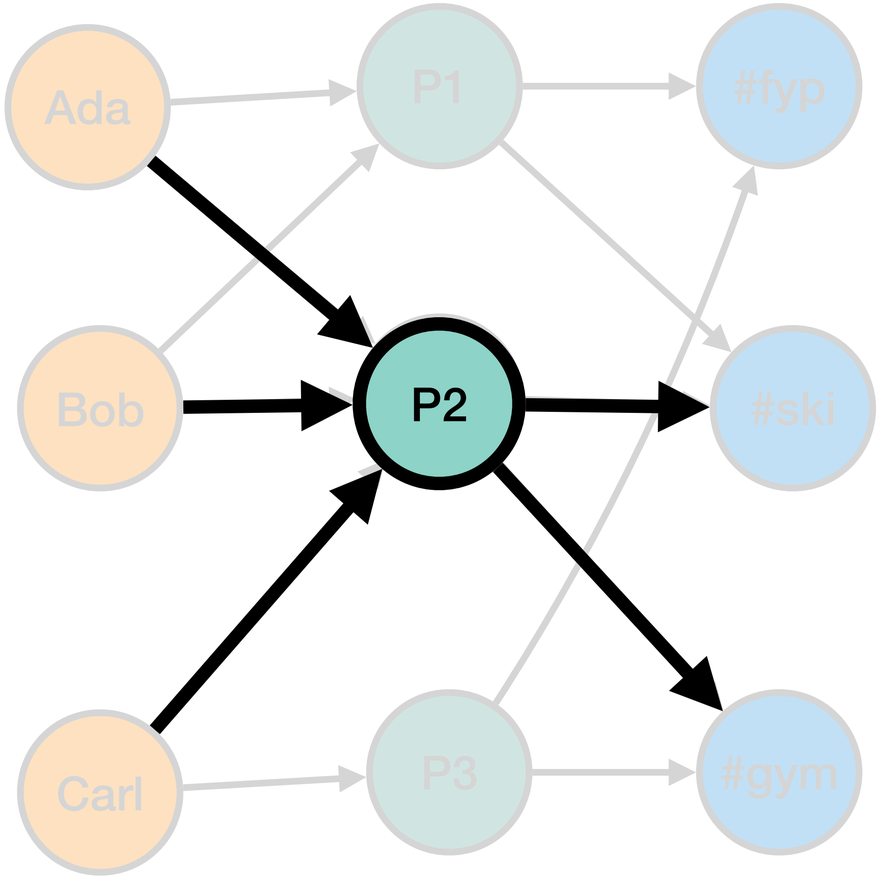
User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

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);
```

faktorizáció: veszteségmentes tömörítés

likes

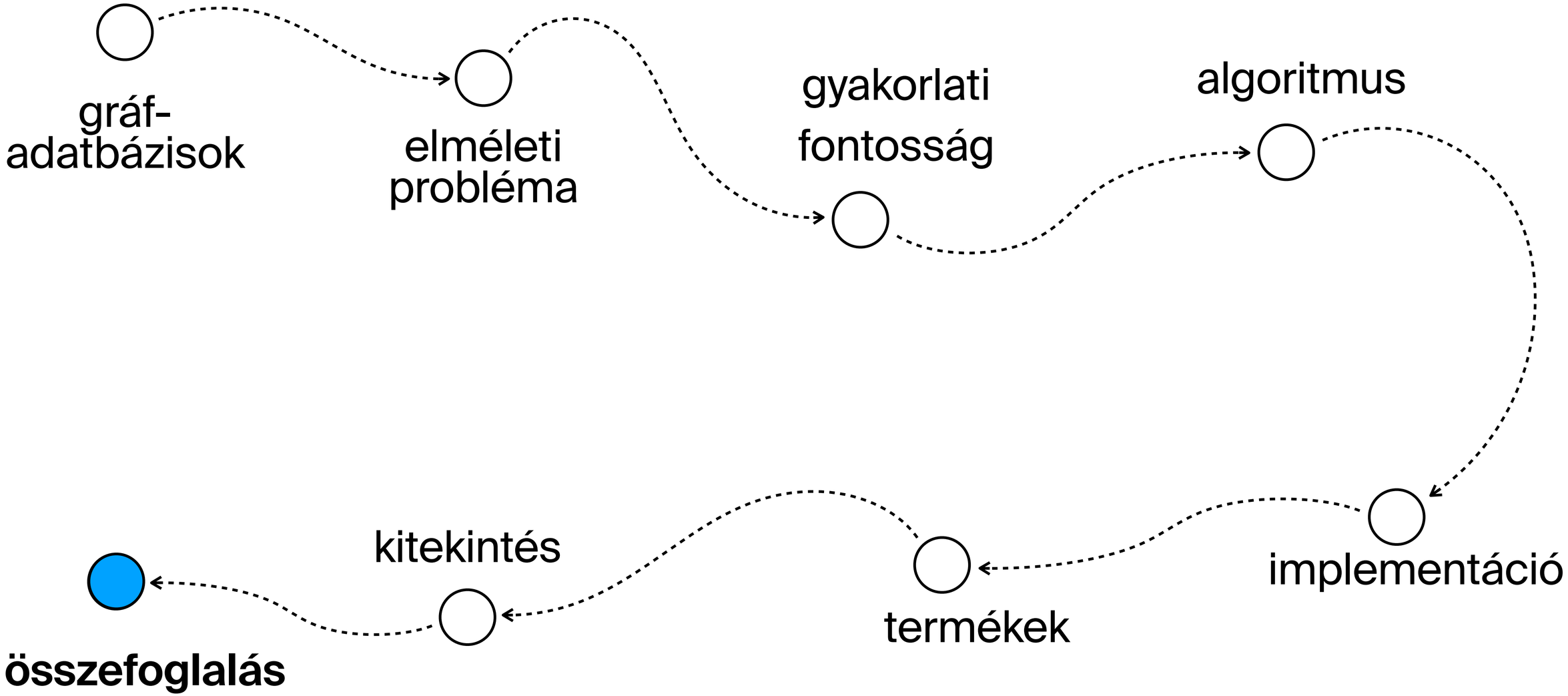
User	Post
Ada	P1
Ada	P2
Bob	P1
Bob	P2
Carl	P2
Carl	P3

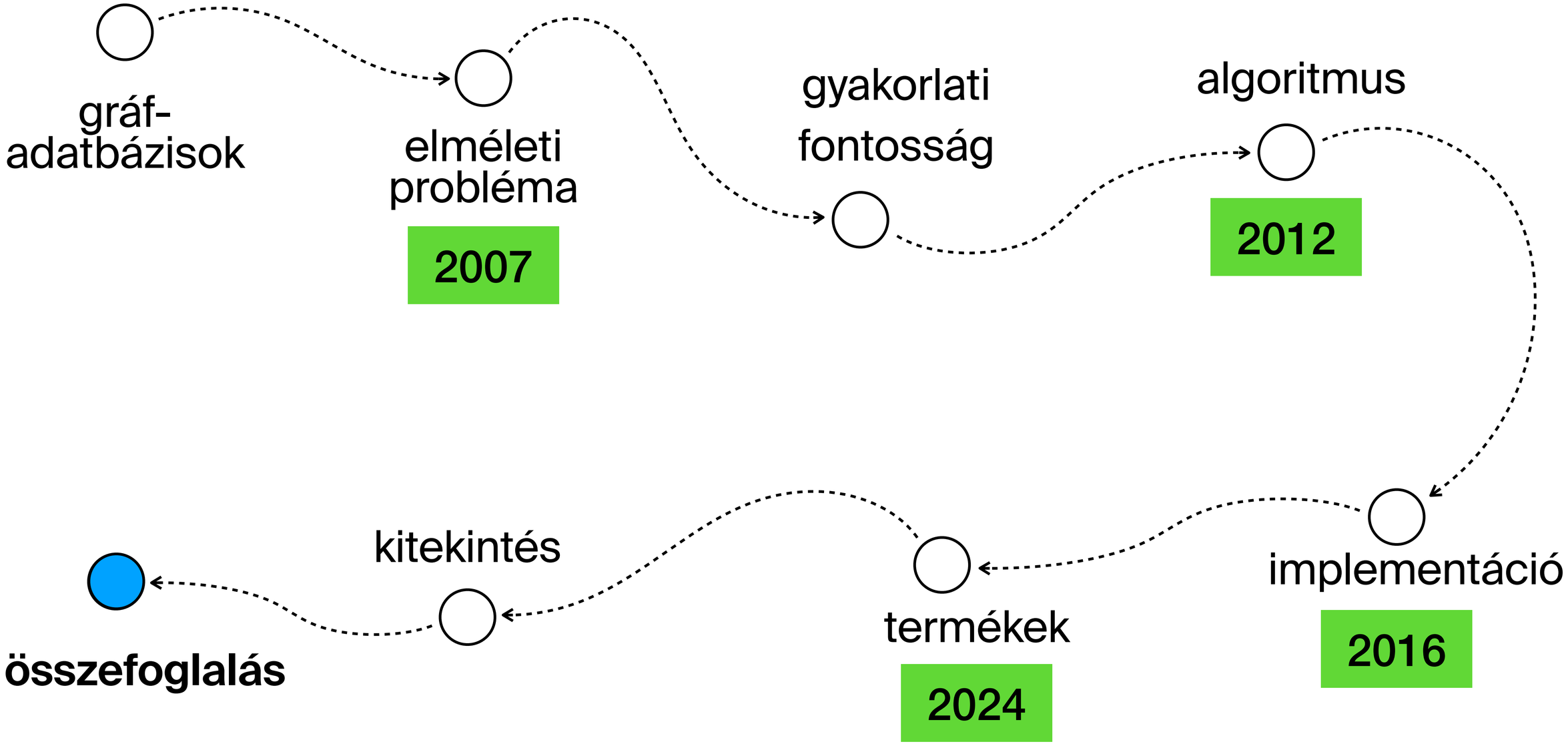
tagged

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}


gráf-
adatbázisok
elméleti
probléma
gyakorlati
fontosság
algoritmus
implementáció
termékek
kitekintés
összefoglalás

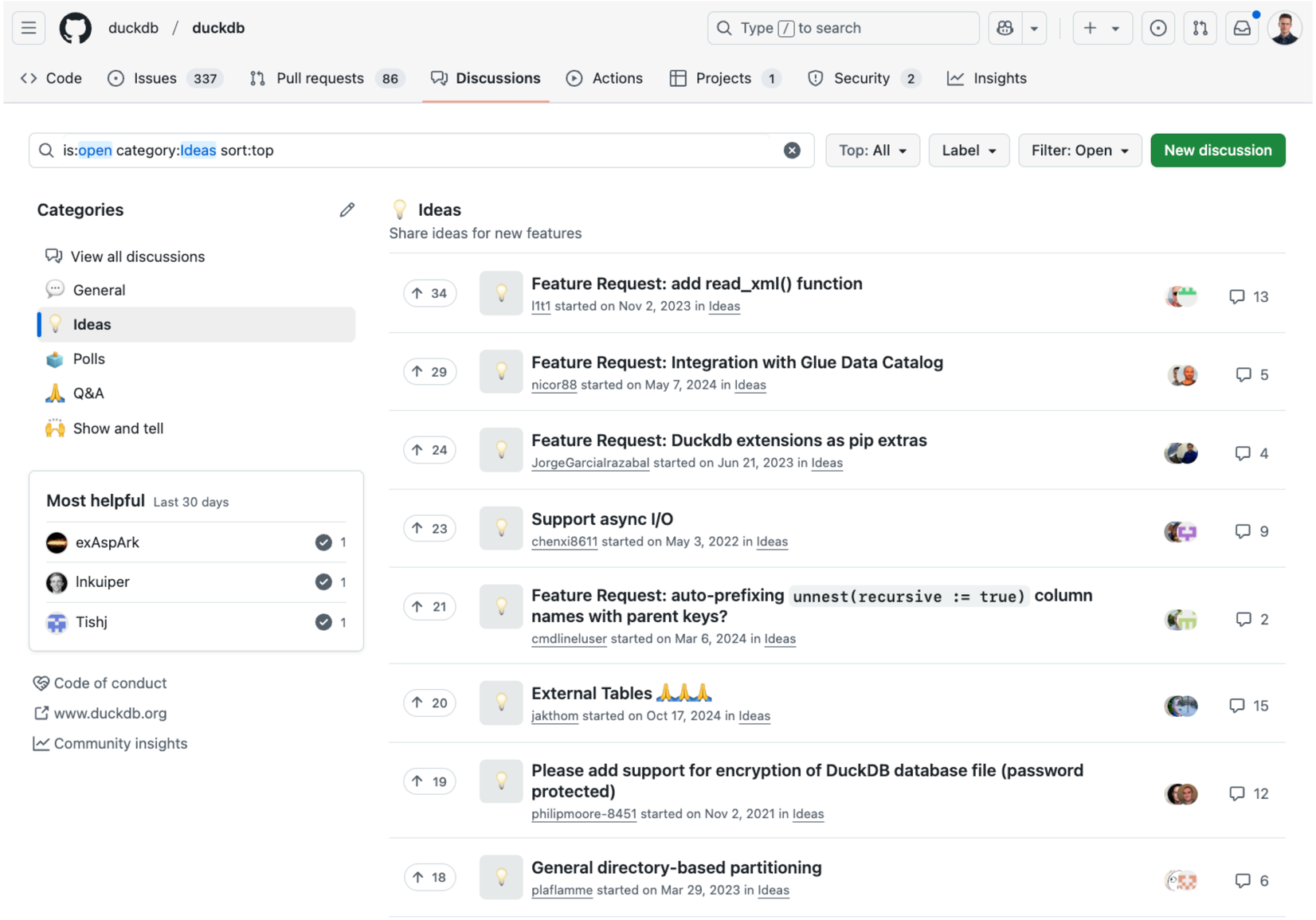

gráf-
adatbázisok
elméleti
probléma
2007
gyakorlati
fontosság
algoritmus
2012
implementáció
2016
termékek
2024
kitekintés
összefoglalás

Összefoglalás

- Gráfadatbázisok: útvonalkeresés és gráfmintaillesztés
- A relációs join algoritmusok területe aktívan kutatott, új algoritmuscsaládokkal
- Az alapkutatástól a termékekig >15 év is eltelhet

Ω

DuckDB


duckdb / duckdb
Type / to search
Code
Issues 337
Pull requests 86
Discussions
Actions
Projects 1
Security 2
Insights
is:open category:Ideas sort:top
Top: All
Label
Filter: Open
New discussion
Categories
View all discussions
General
Ideas
Polls
Q&A
Show and tell
Most helpful Last 30 days
exAspArk 1
lnkuiper 1
Tishj 1
Code of conduct
www.duckdb.org
Community insights
Ideas
Share ideas for new features
34 Feature Request: add read_xml() function
l1t1 started on Nov 2, 2023 in Ideas
13
29 Feature Request: Integration with Glue Data Catalog
nicor88 started on May 7, 2024 in Ideas
5
24 Feature Request: Duckdb extensions as pip extras
JorgeGarcialrazabal started on Jun 21, 2023 in Ideas
4
23 Support async I/O
chenxi8611 started on May 3, 2022 in Ideas
9
21 Feature Request: auto-prefixing unnest(recursive := true) column names with parent keys?
cmdlineluser started on Mar 6, 2024 in Ideas
2
20 External Tables 🙏🙏🙏
jakthom started on Oct 17, 2024 in Ideas
15
19 Please add support for encryption of DuckDB database file (password protected)
philipmoore-8451 started on Nov 2, 2021 in Ideas
12
18 General directory-based partitioning
plaflamme started on Mar 29, 2023 in Ideas
6