Gráf adatbázisok
NoSql, neo4j

Gombos Gergő

Áttekintés

- Miért használjunk gráfot?
- Mi a gráf?
- Hogy dolgozzunk gráfadatbázisokkal (neo4j)

Gráf az egész világ

Közösségi hálók

Gráf az egész világ

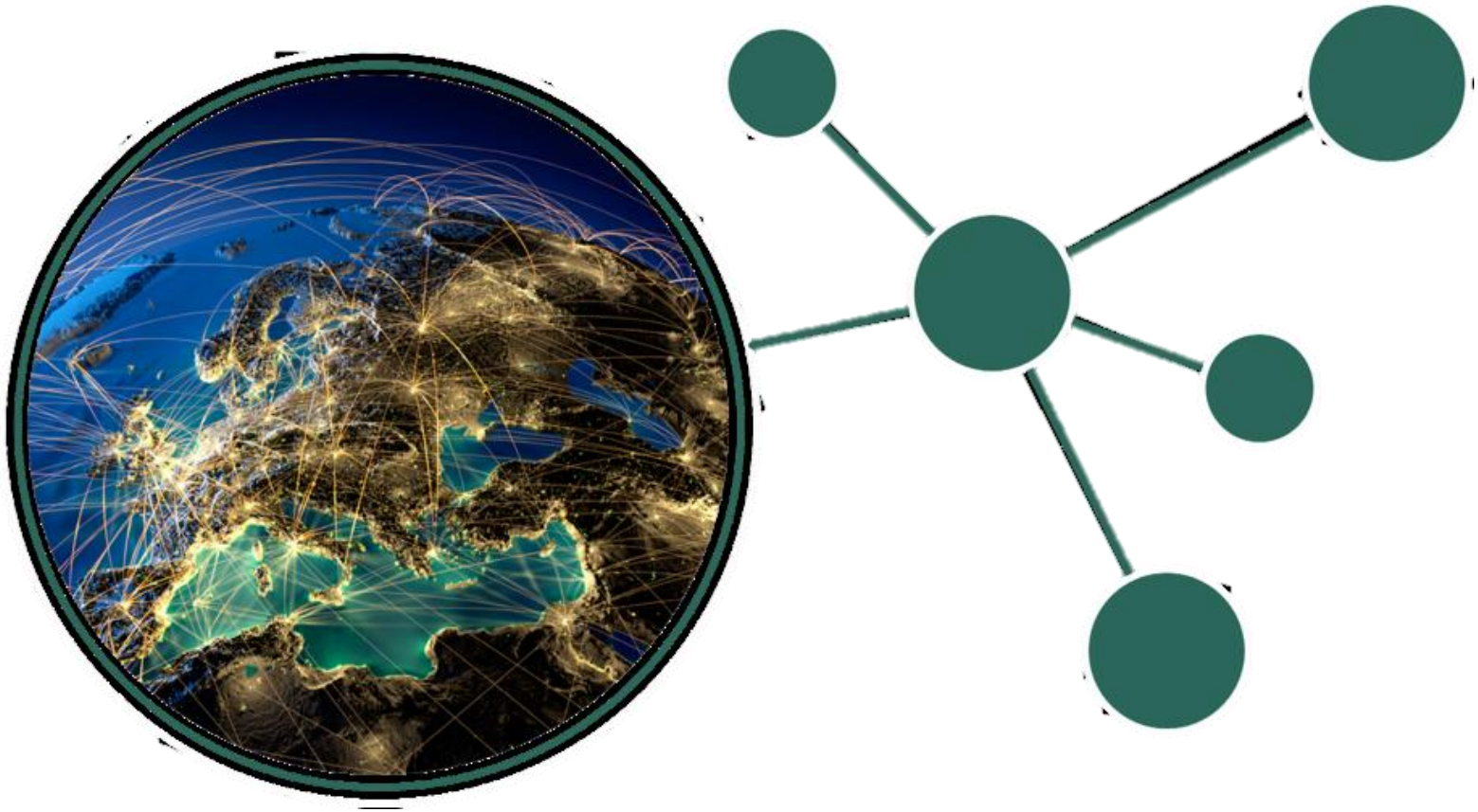
- Útvonal tervezés

Gráf az egész világ

- Ajánló rendszerek

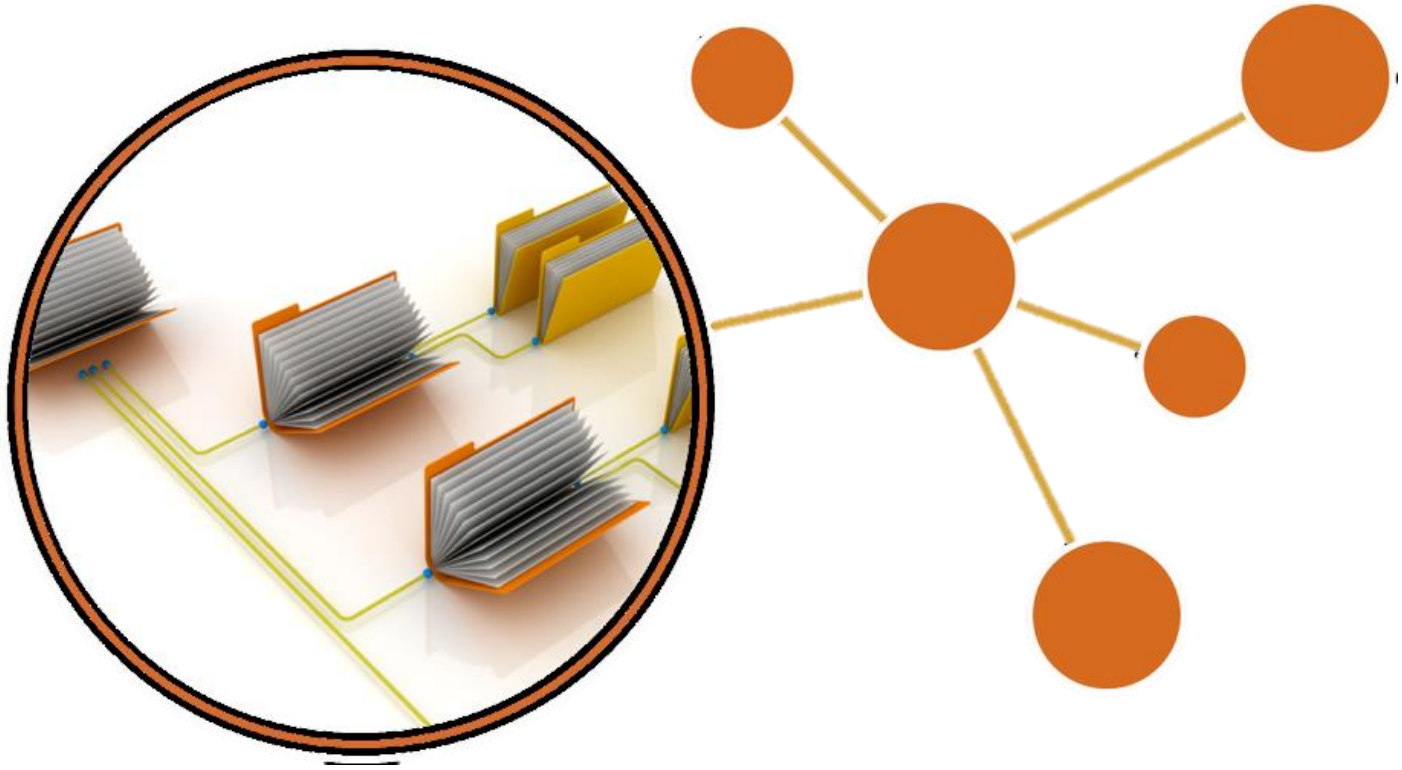
Gráf az egész világ

- Logisztika



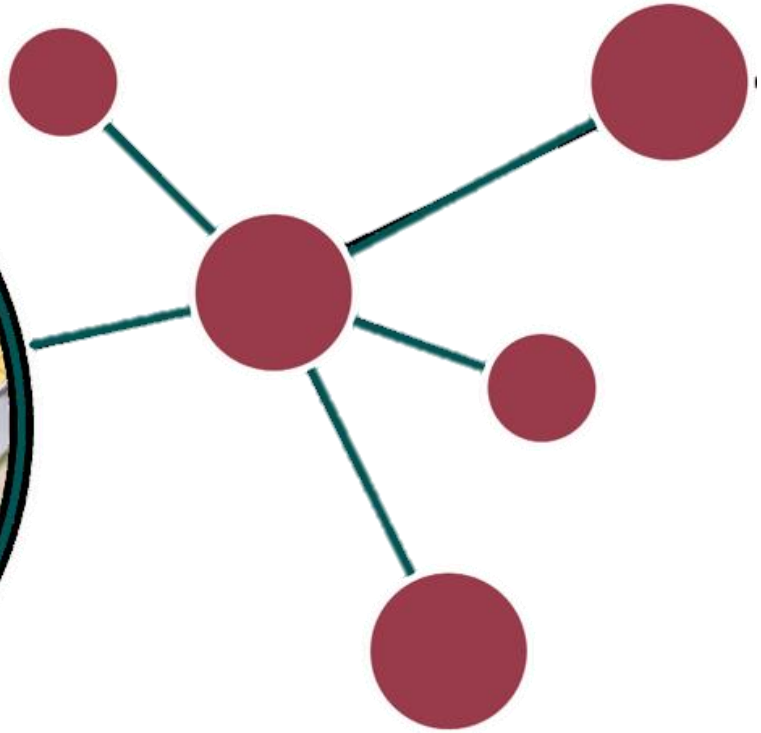
Gráf az egész világ

- Hozzáférések

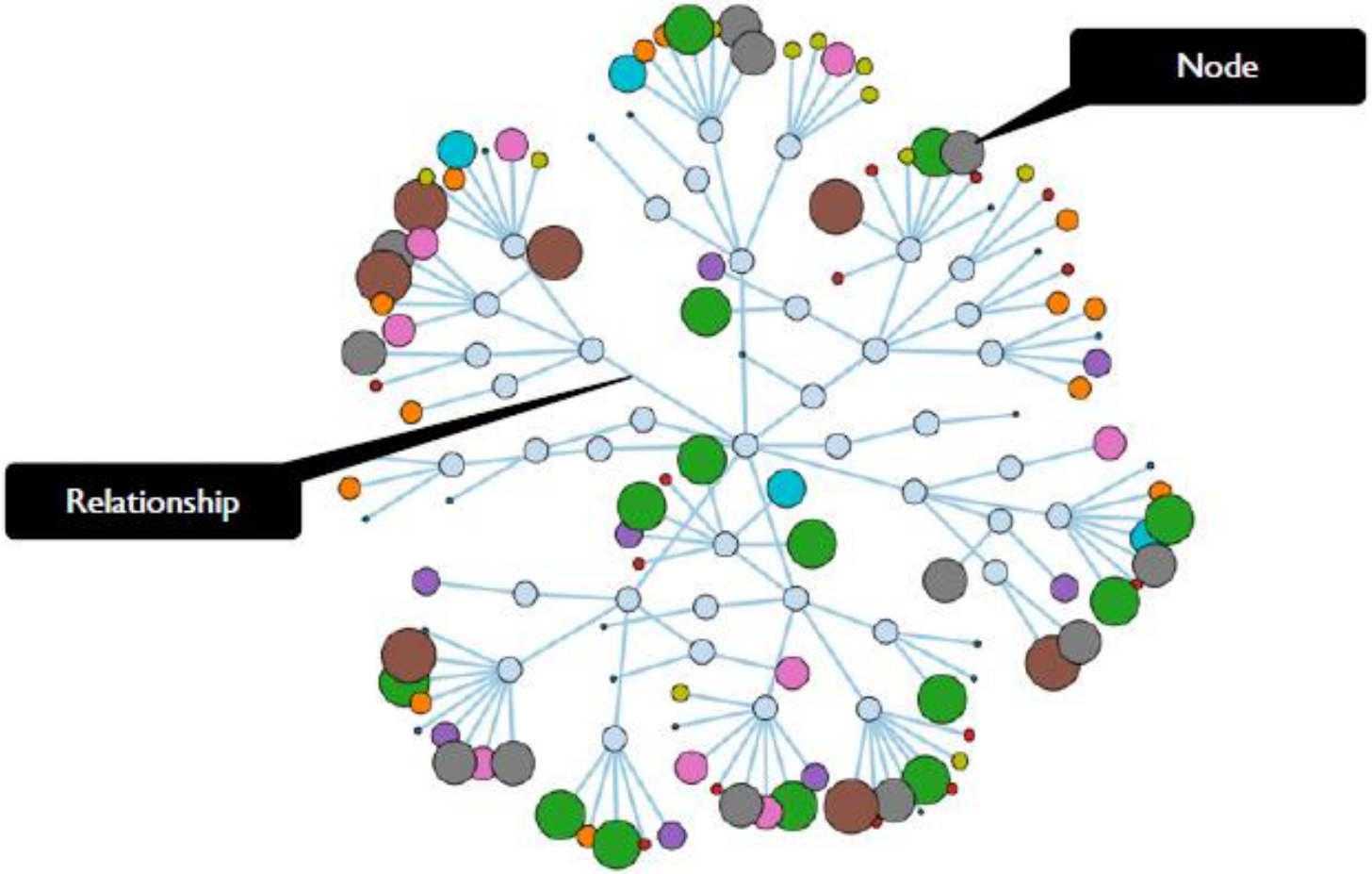


Gráf az egész világ

- Csalás detektálás



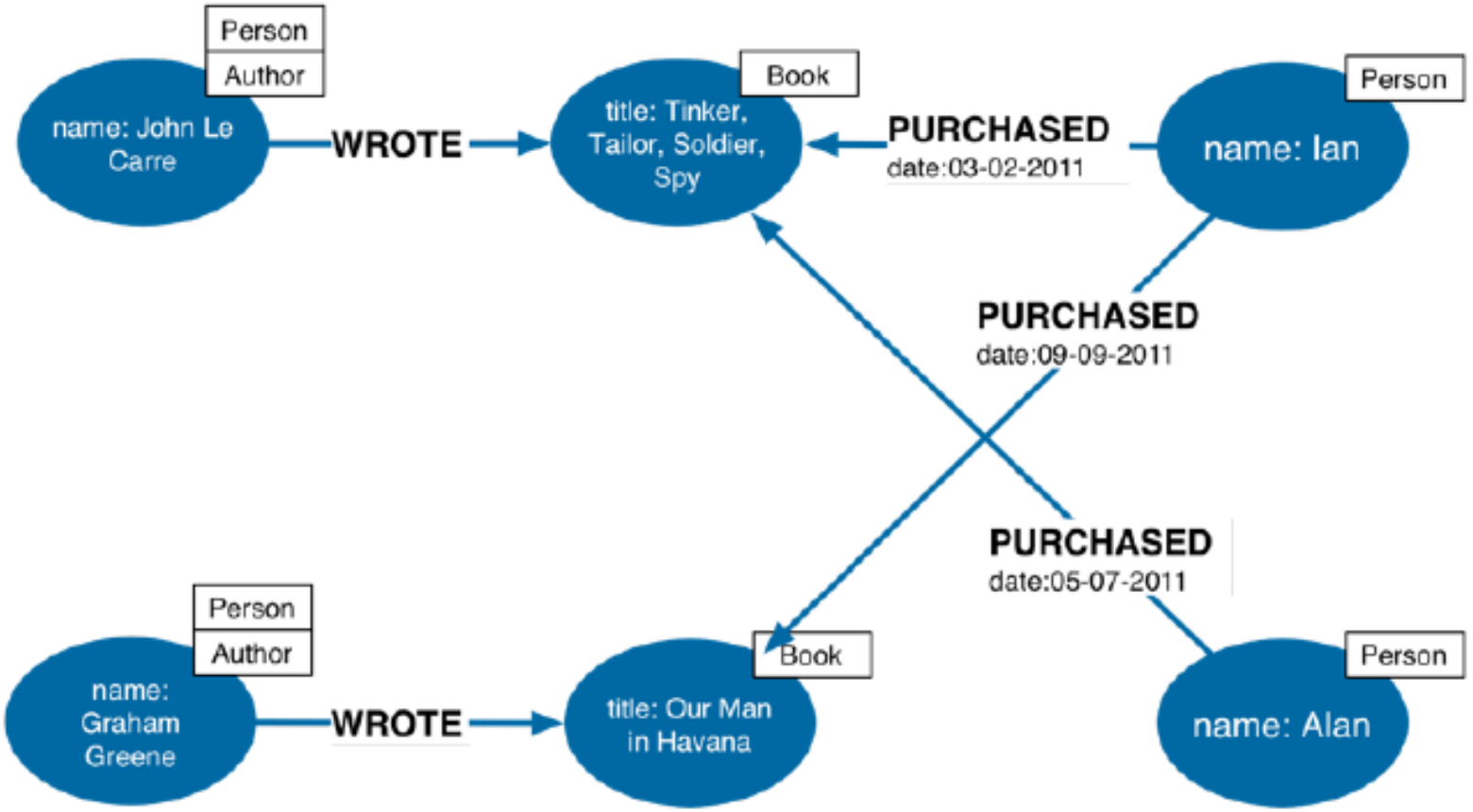
Mi is a gráf?



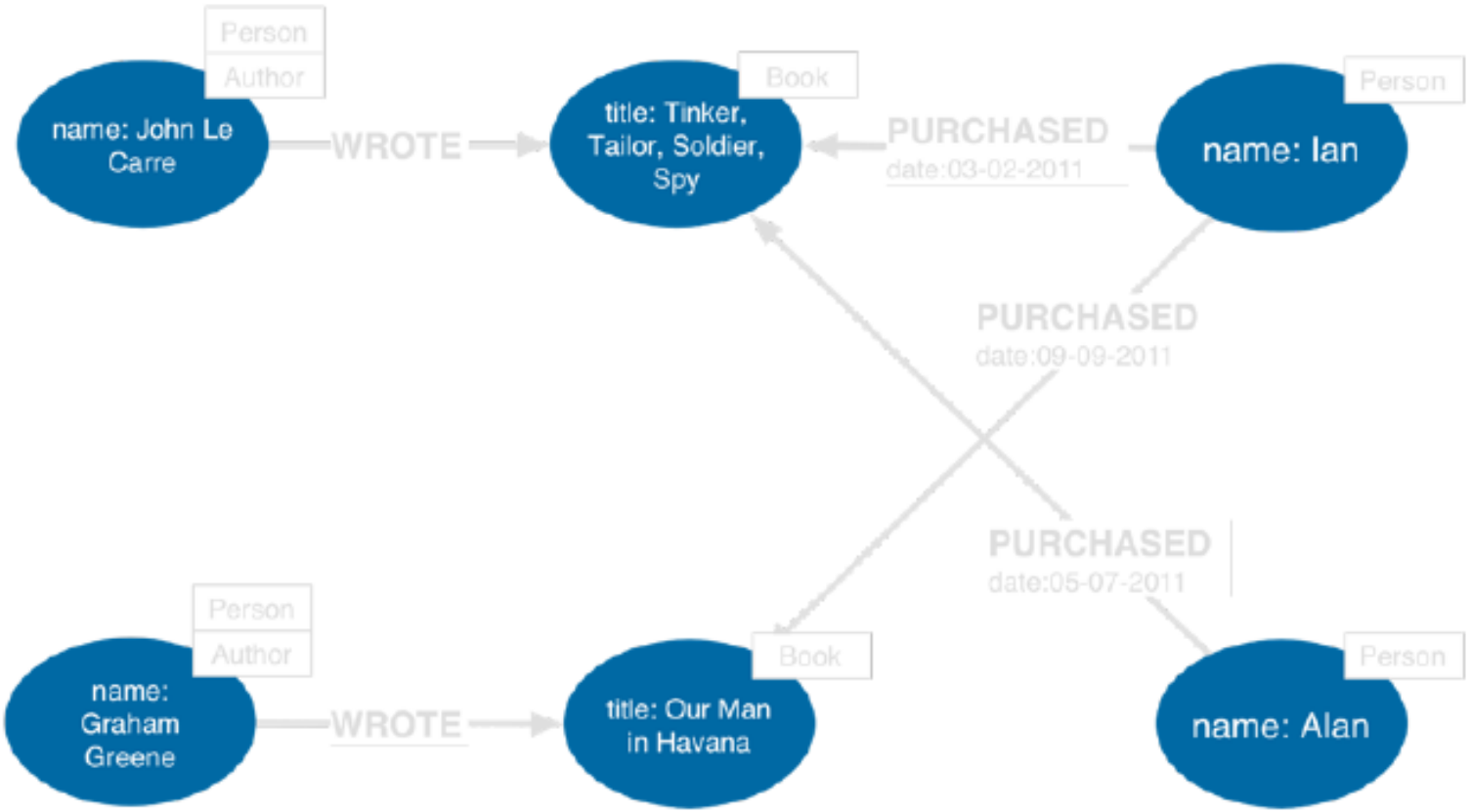
Property gráfok

- Építő elemei:
  - Csúcsok
    - Entitások
  - Élek
    - Kapcsolatok az entitások között
  - Tulajdonságok
    - Attribútumok és metainformációk
  - Címkék
    - Csoportosítás

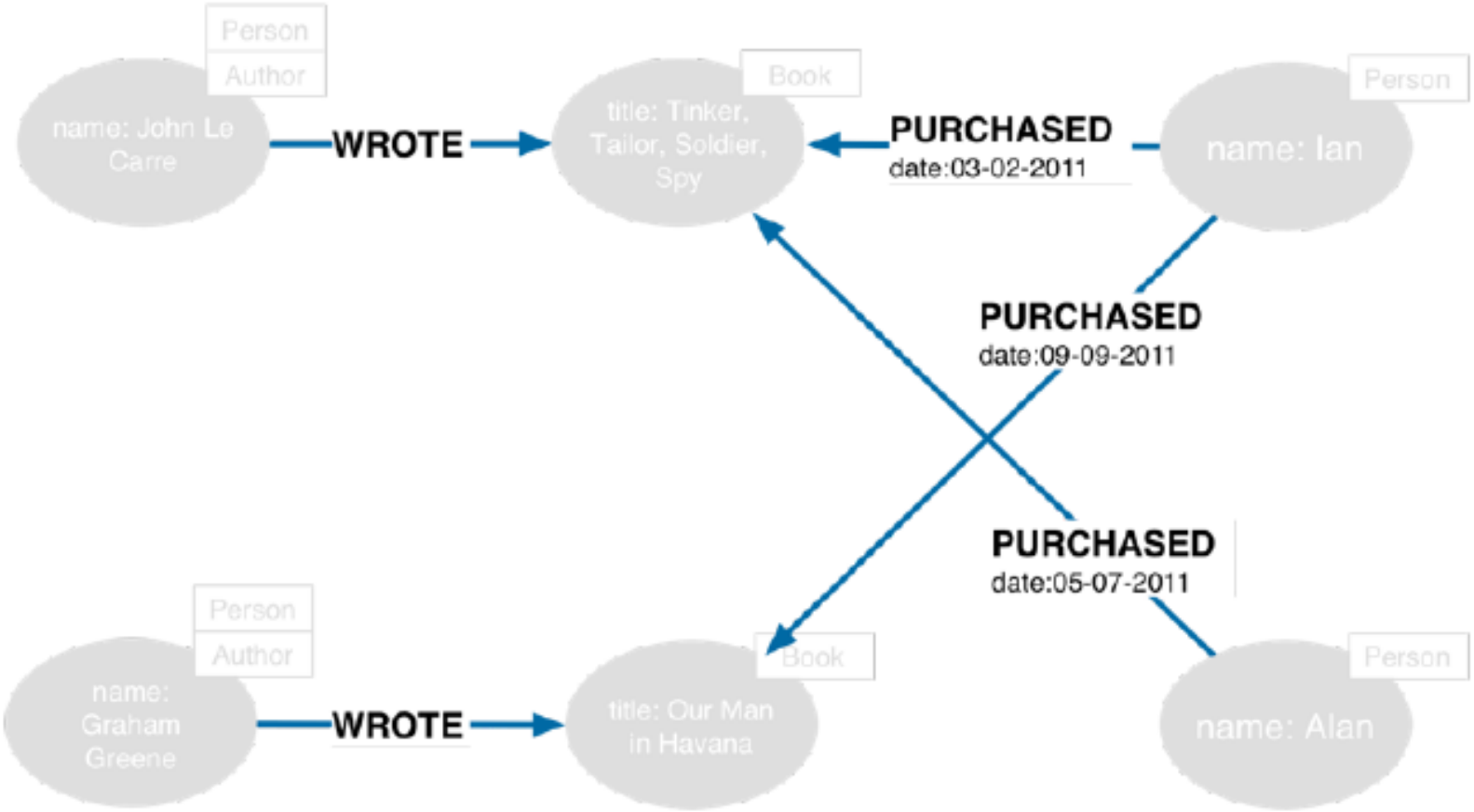
Példa Property gráf



Csúcsok

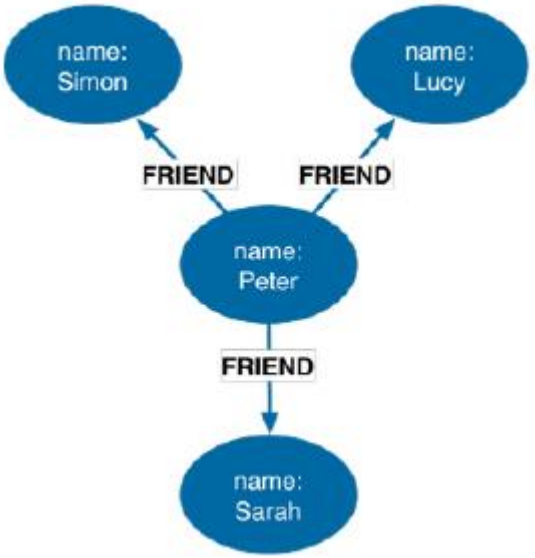

Person
Author
name: John Le Carre
WROTE
Book
title: Tinker, Tailor, Soldier, Spy
PURCHASED
date:03-02-2011
Person
name: Ian
PURCHASED
date:09-09-2011
PURCHASED
date:05-07-2011
Person
Author
name: Graham Greene
WROTE
Book
title: Our Man in Havana
Person
name: Alan

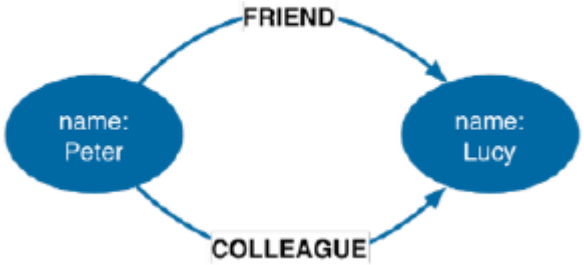
Élek


Person
Author
name: John Le Carre
WROTE
Book
title: Tinker, Tailor, Soldier, Spy
PURCHASED
date:03-02-2011
Person
name: Ian
PURCHASED
date:09-09-2011
PURCHASED
date:05-07-2011
Person
Author
name: Graham Greene
WROTE
Book
title: Our Man in Havana
Person
name: Alan

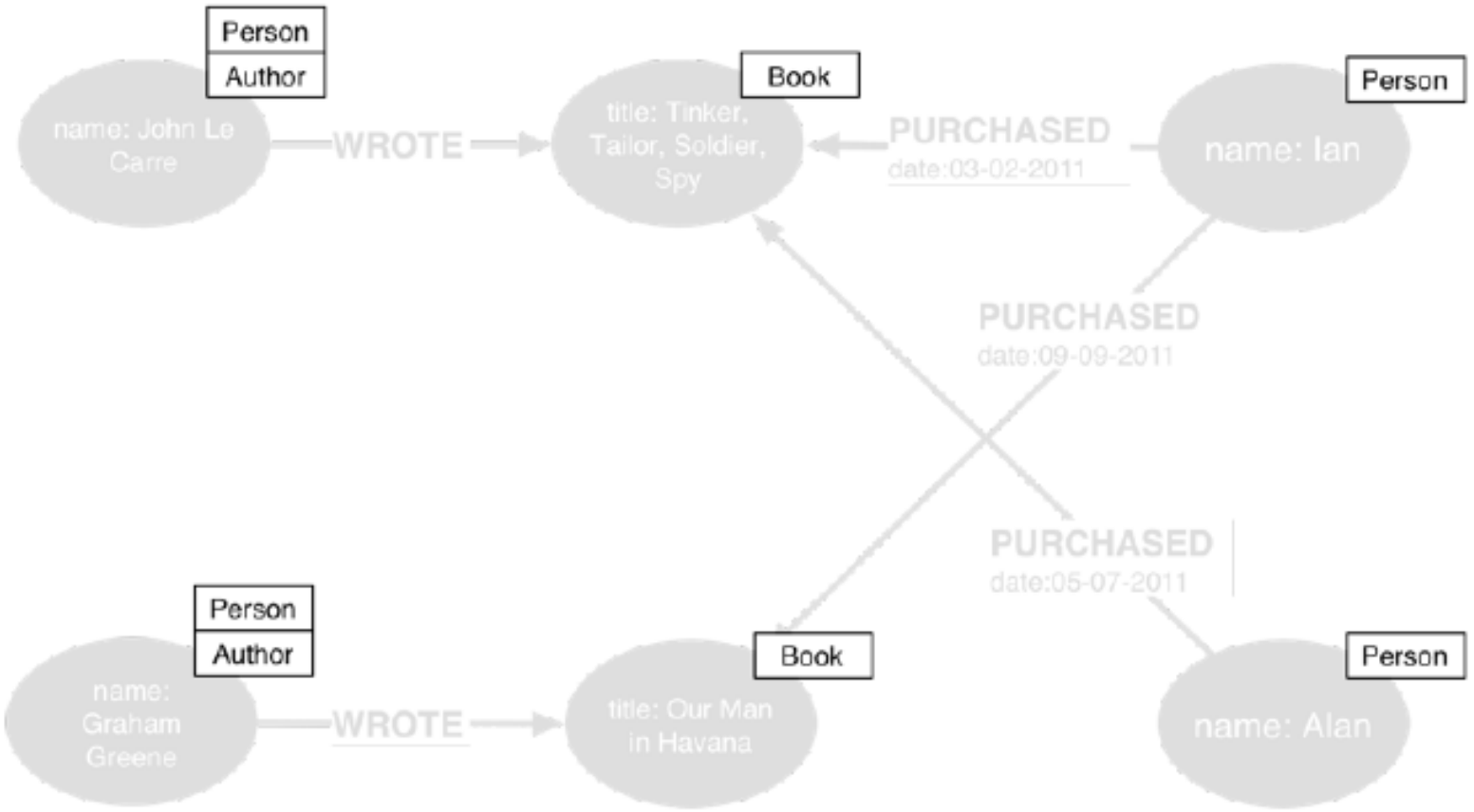
Élek

- Egy csúcsnak több éle is lehet
- Lehetnek
  - párhuzamos élek
  - hurokélek

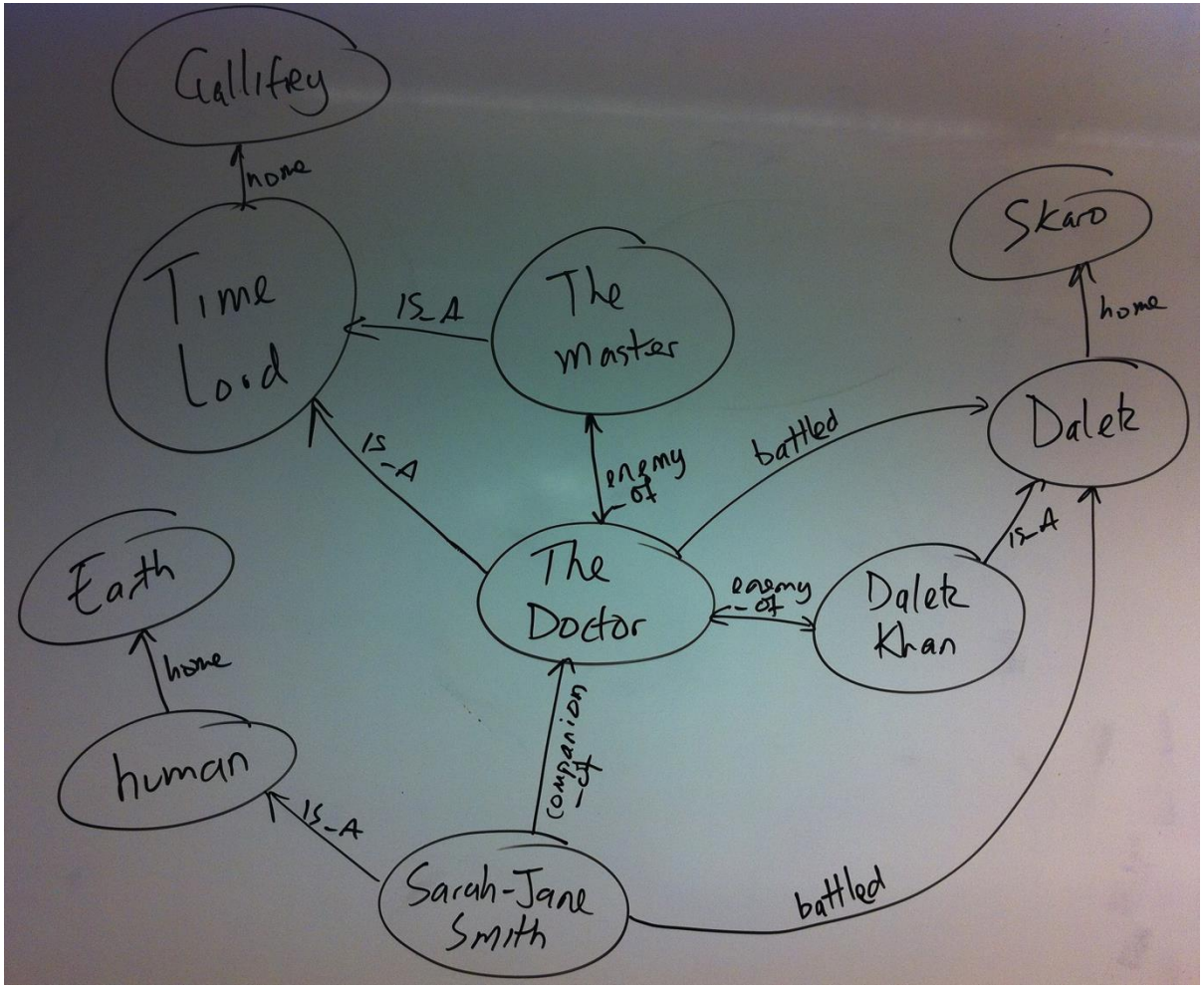




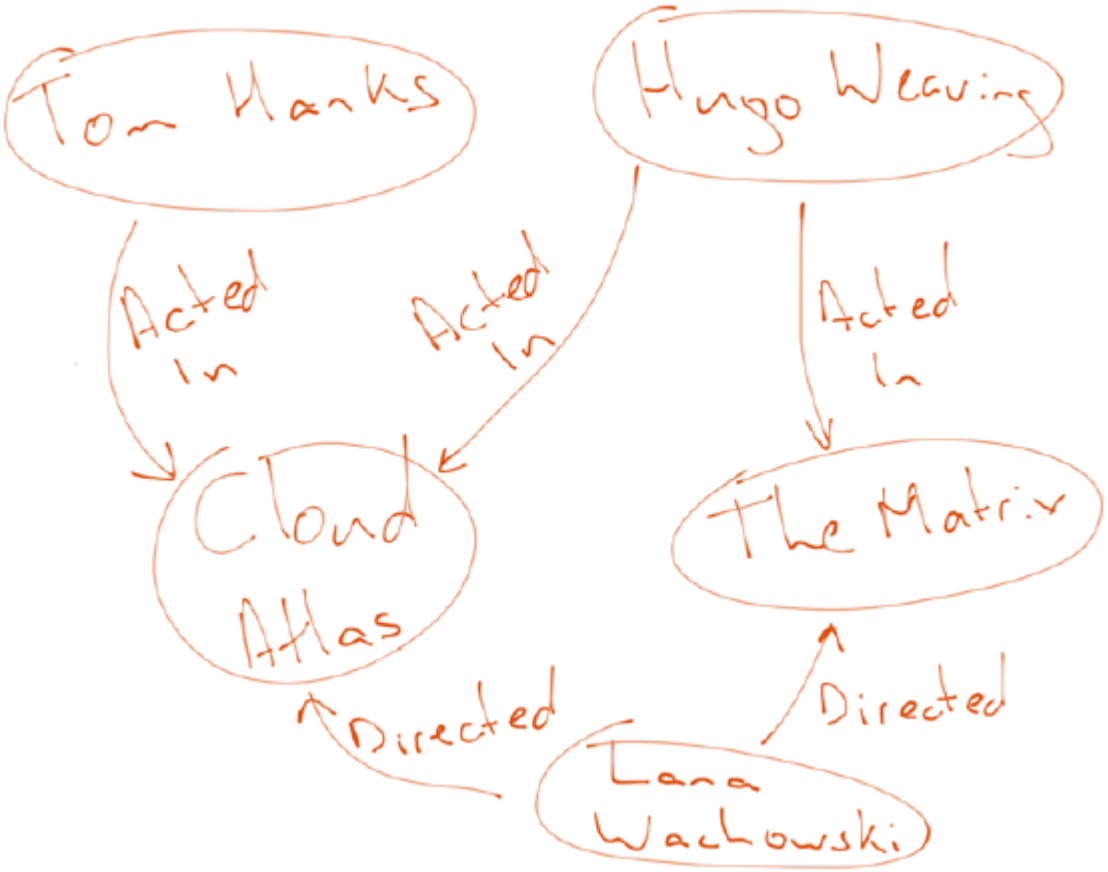
Címkék


Person
Author
name: John Le Carre
WROTE
Book
title: Tinker, Tailor, Soldier, Spy
PURCHASED
date:03-02-2011
Person
name: Ian
PURCHASED
date:09-09-2011
PURCHASED
date:05-07-2011
Person
Author
name: Graham Greene
WROTE
Book
title: Our Man in Havana
Person
name: Alan

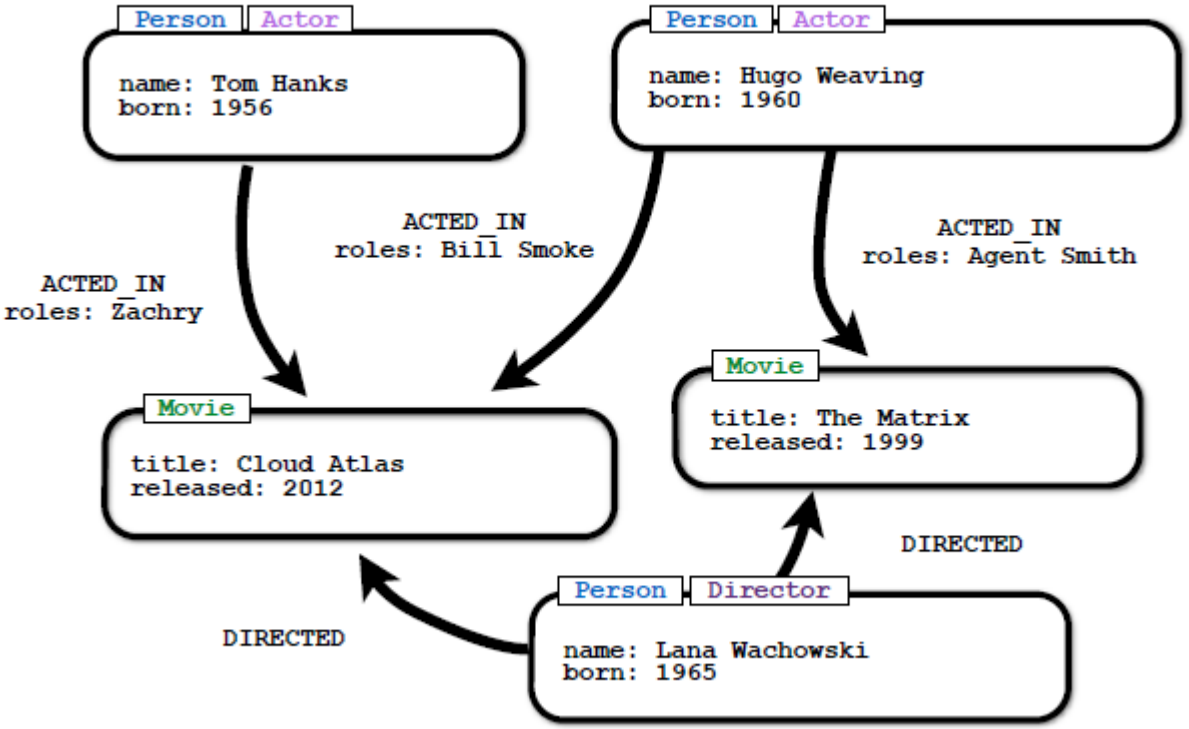
A gráfok könnyen ábrázolhatóak



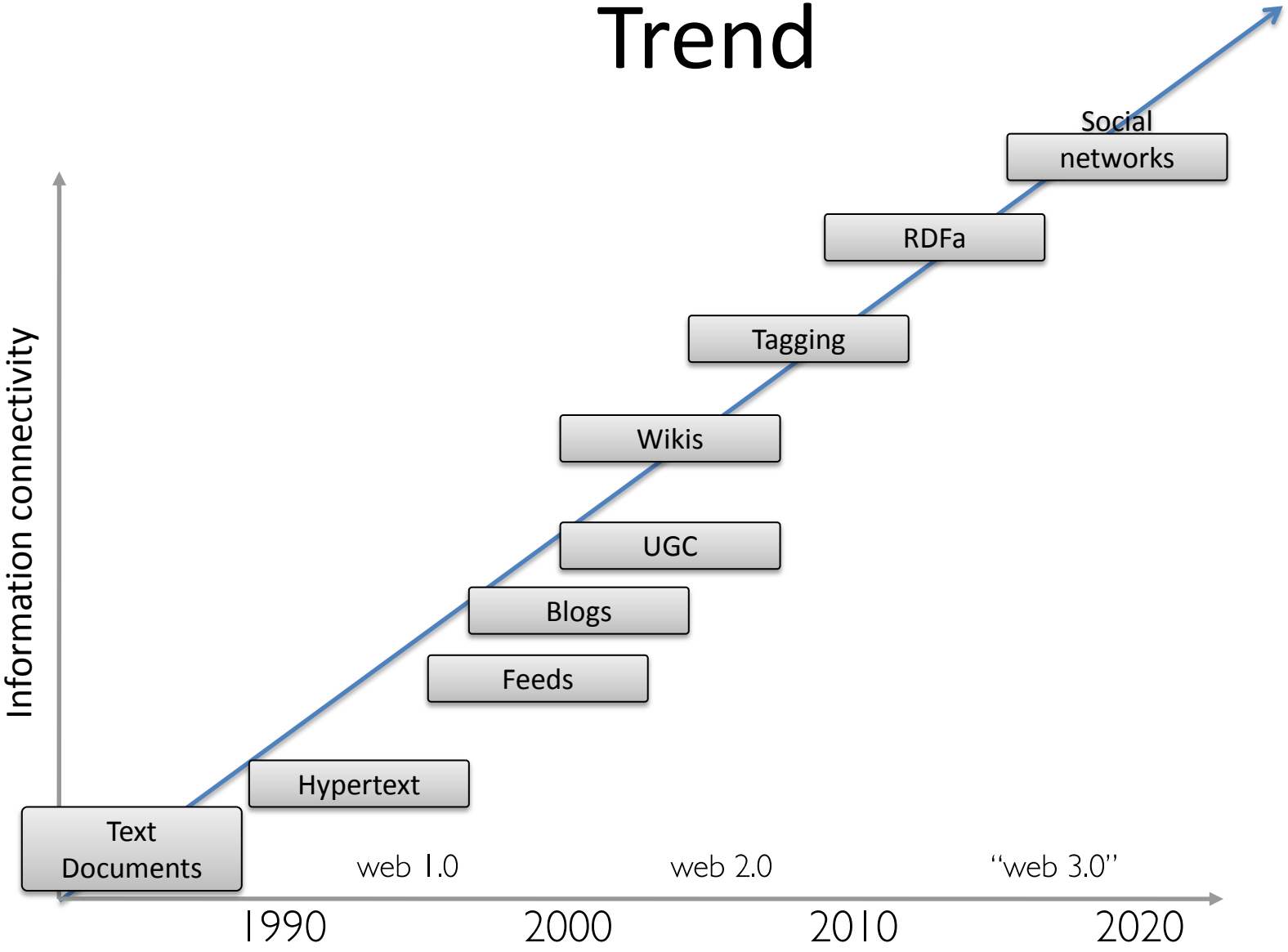
Egyszerű modell tervezés



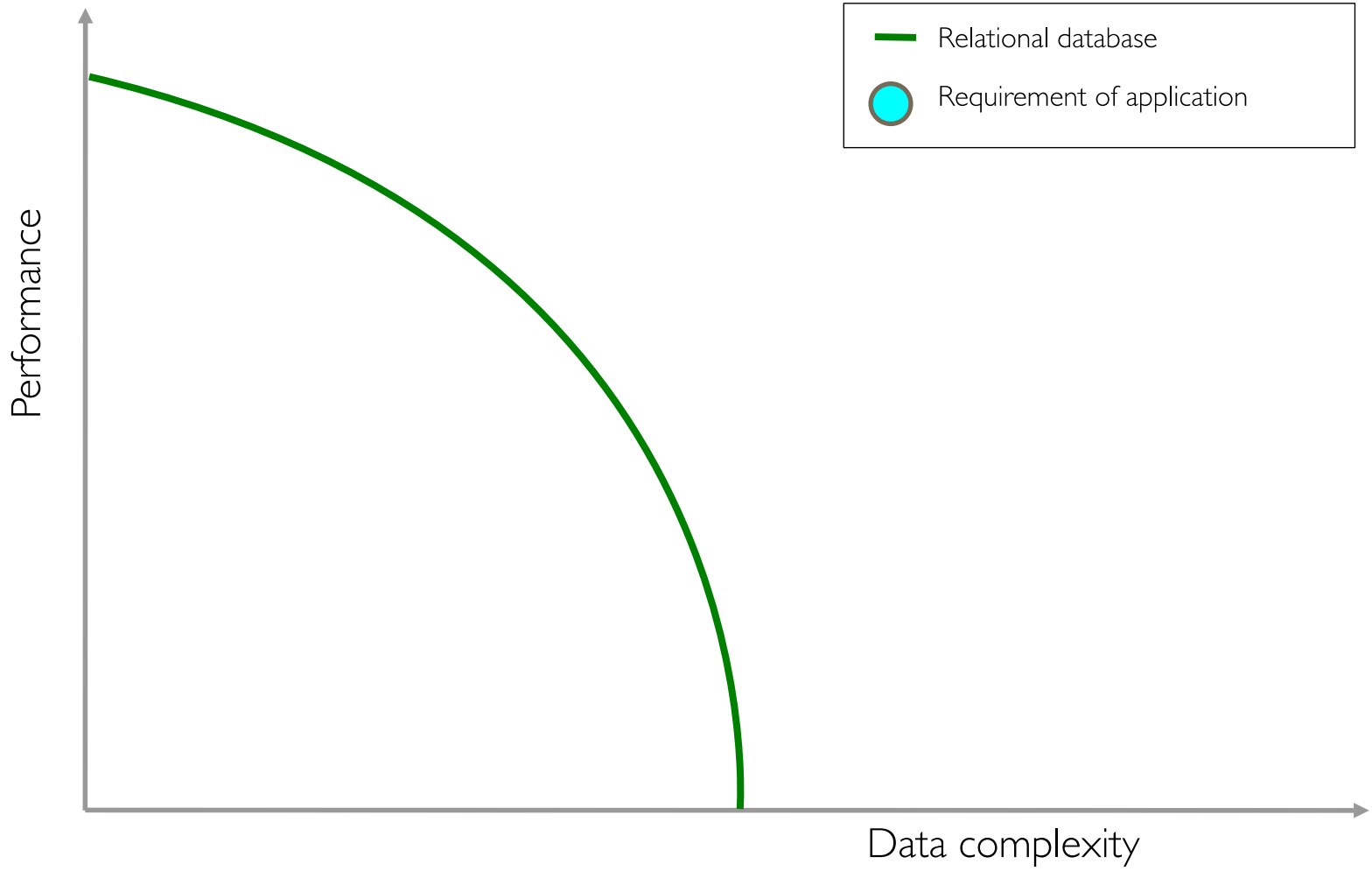
Egyszerű modell tervezés



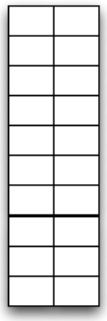
Trend


Social networks
RDFa
Tagging
Wikis
UGC
Blogs
Feeds
Hypertext
Text Documents
Information connectivity
web 1.0
web 2.0
"web 3.0"
1990
2000
2010
2020

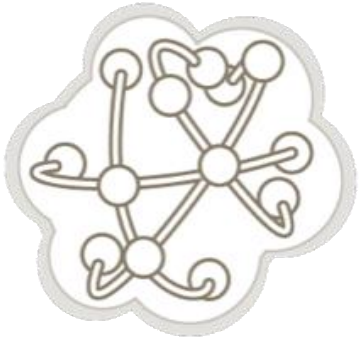
RDBMS teljesítmény



Key-Value

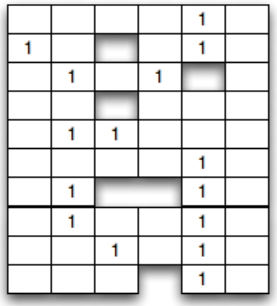


Graph DB

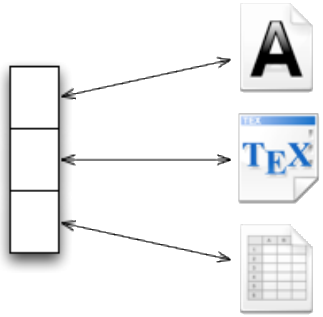


Négy NOSQL kategória

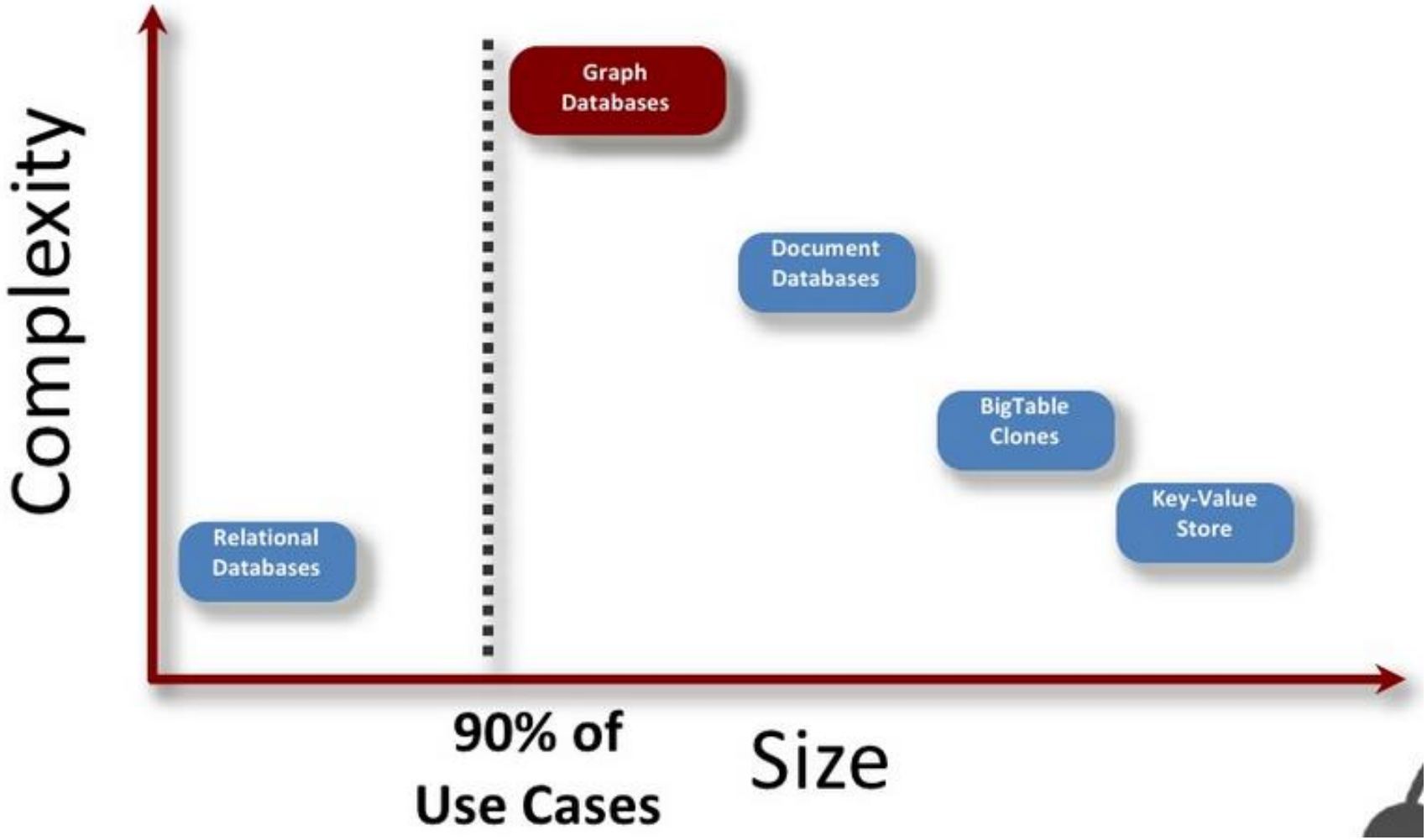
BigTable



Document



Living in a NOSQL World



Kik használnak?

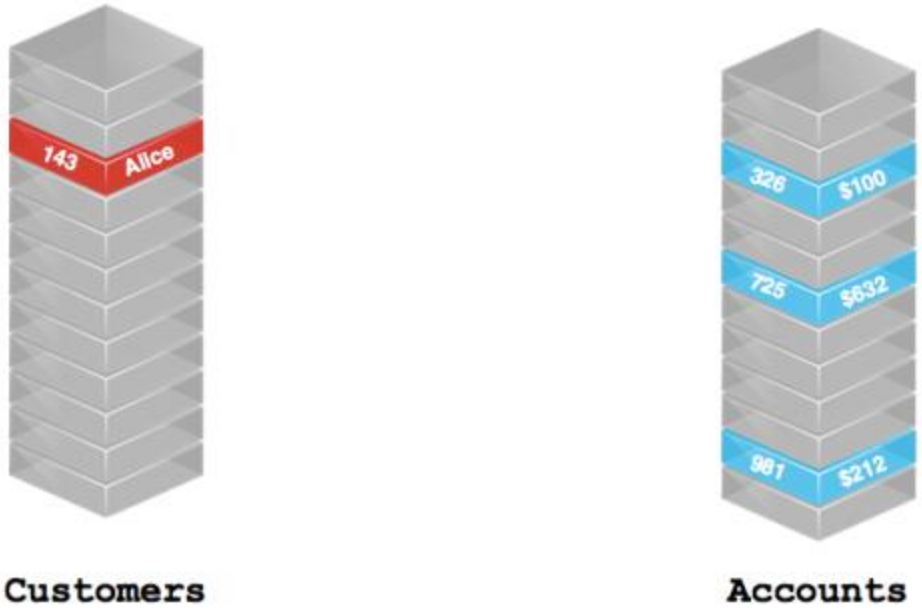
- Facebook
  - Gráf keresés, közös témák keresése
- Google
  - Tudás gráf, keresésekhez
- Twitter
  - Felhasználó ajánlás
- További
  - Ajánlás, logisztika, stb.

RDBMS vs. Gráfadatbázisok

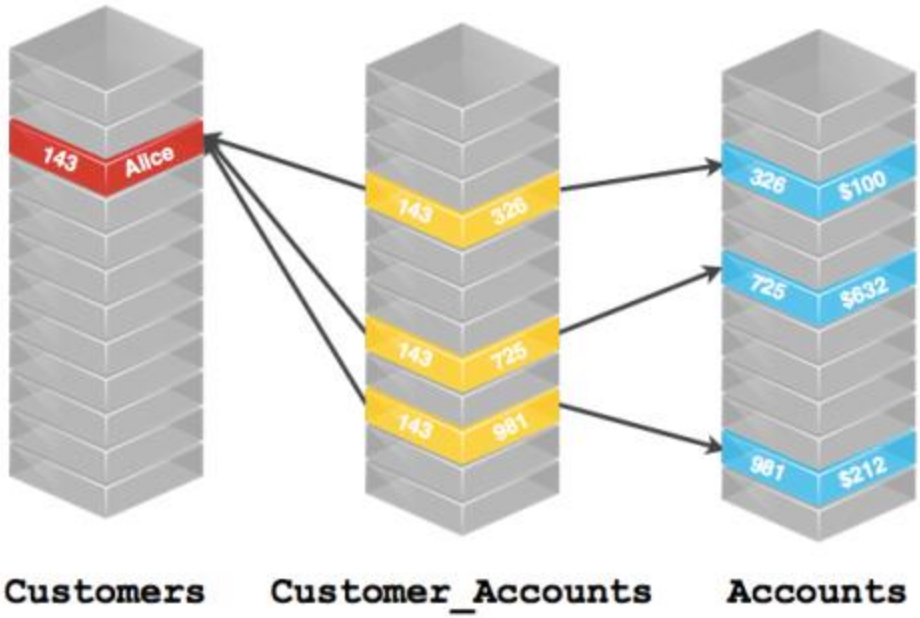
- Relációs adatbázisok
  - Kapcsolatok a táblák elemei között: foreign-keys
    - Kötött séma
    - Az összekapcsolások a lekérdezéskor számítódnak ki
  - Több-a-többhöz kapcsolatok: kapcsoló táblák
- Gráfadatbázisok
  - Minden csúcs explicit össze van kapcsolva a szomszédaival

Forrás: http://neo4j.com/developer/graph-db-vs-rdbms/

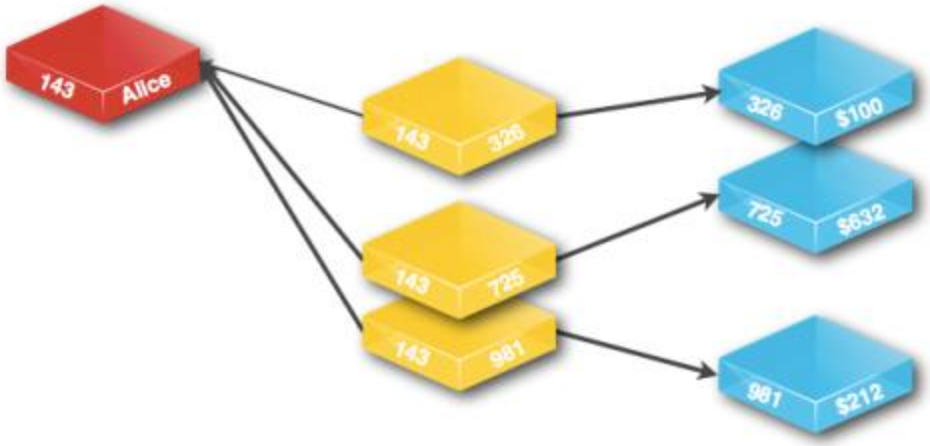
RDBMS vs. Gráf



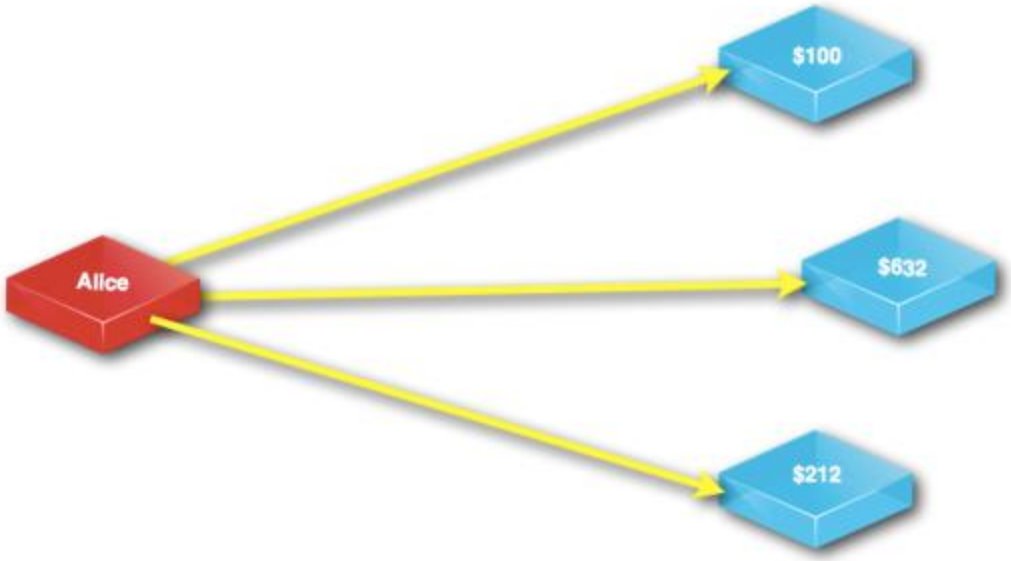
RDBMS vs Gráf



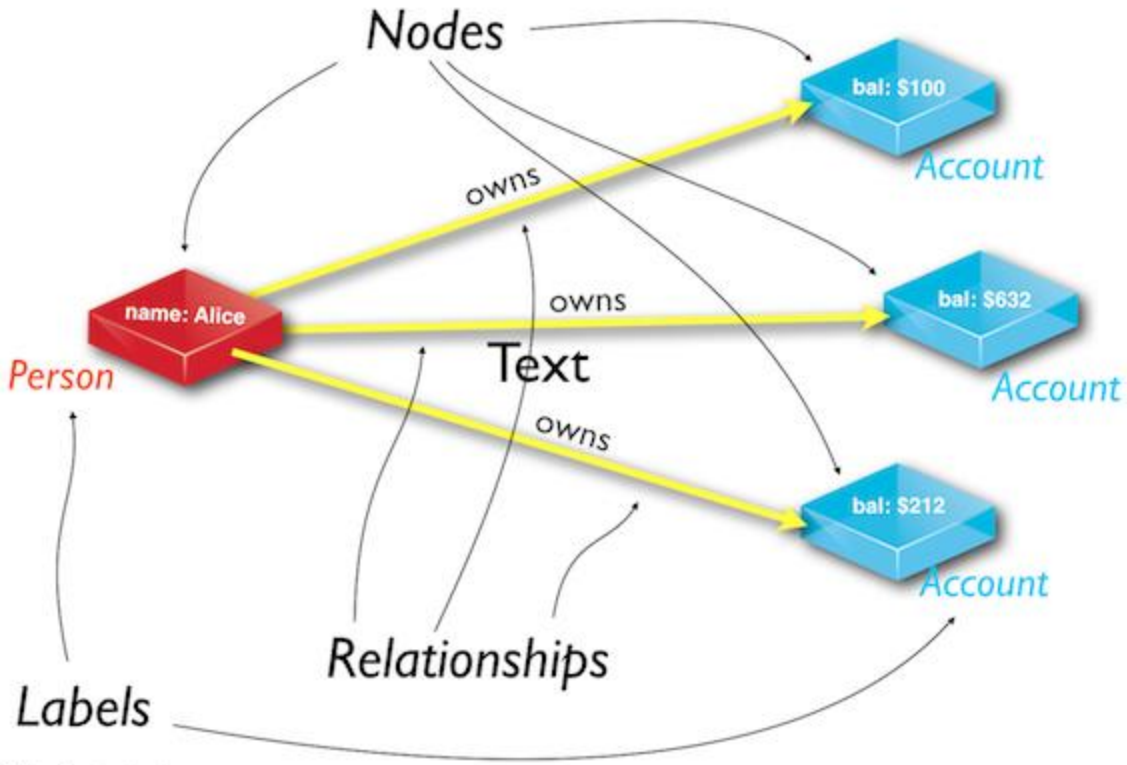
RDBMS vs Gráf



RDBMS vs Gráf

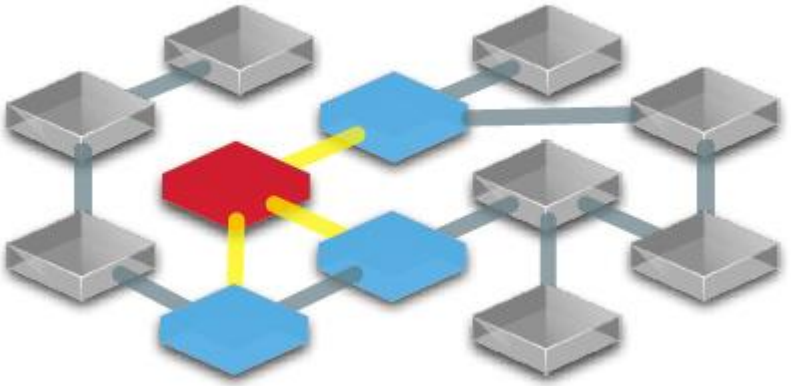


RDBMS vs Gráf



RDBMS vs Gráf

- További kapcsolatokkal nagyobb gráf keletkezik.
- További személyek.
- További számlák.



Közösségi Hálózat „kapcsolat” Teljesítmény

- Mérés:
  - ~1000 személy
  - Átlag 50 barát emberenként
  - `pathExists(a,b)` úthossz limit 4

	# személyek	futásidő
Relációs adatbázis	1000	2000ms

Közösségi Hálózat „kapcsolat” Teljesítmény

- Mérés:
  - ~1000 személy
  - Átlag 50 barát emberenként
  - `pathExists(a,b)` úthossz limit 4

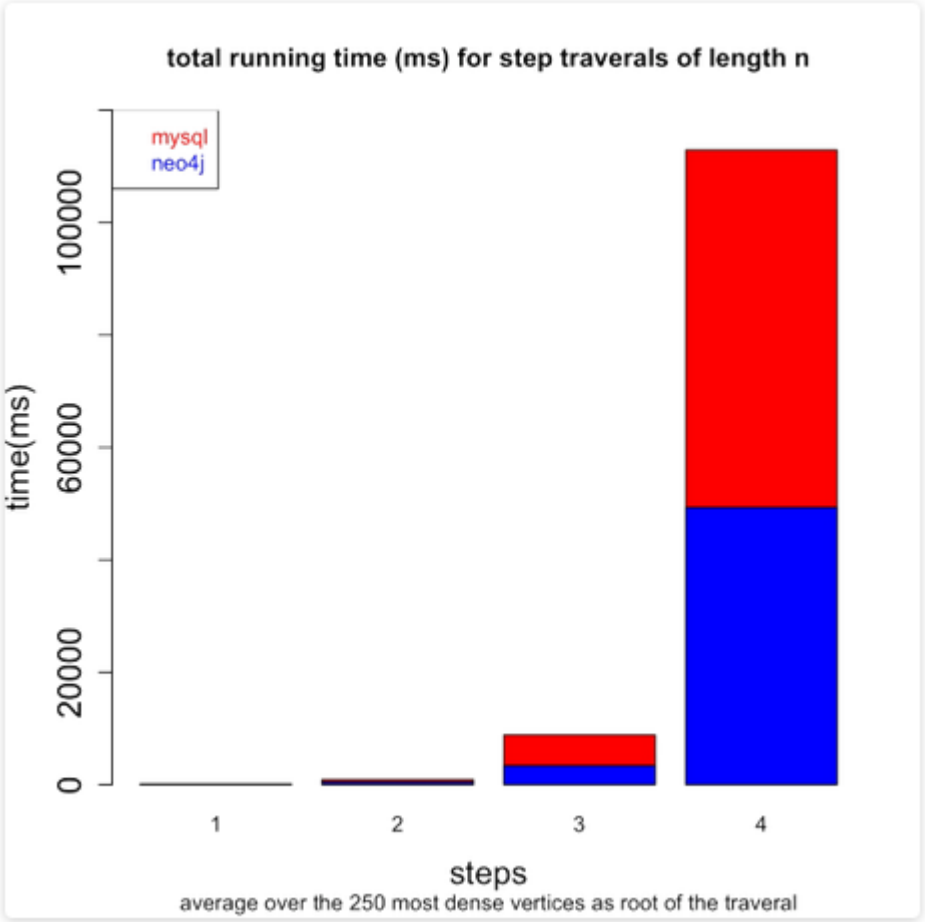
	# személyek	futásidő
Relációs adatbázis	1000	2000ms
Neo4j	1000	2ms

Közösségi Hálózat „kapcsolat” Teljesítmény

- Mérés:
  - ~1000 személy
  - Átlag 50 barát emberenként
  - `pathExists(a,b)` úthossz limit 4

	# személyek	futásidő
Relációs adatbázis	1000	2000ms
Neo4j	1000	2ms
Neo4j	1000000	2ms

MySQL vs Neo4j




```
CREATE TABLE graph (
 outV INT NOT NULL,
 inV INT NOT NULL
);
CREATE INDEX outV_index USING BTREE ON graph (outV);
CREATE INDEX inV_index USING BTREE ON graph (inV);
```

SELECT a.inV
FROM graph as a
WHERE a.outV=?

SELECT b.inV
FROM graph as a, graph as b
WHERE a.inV=b.outV and a.outV=?

Forrás: http://java.dzone.com/articles/mysql-vs-neo4j-large-scale

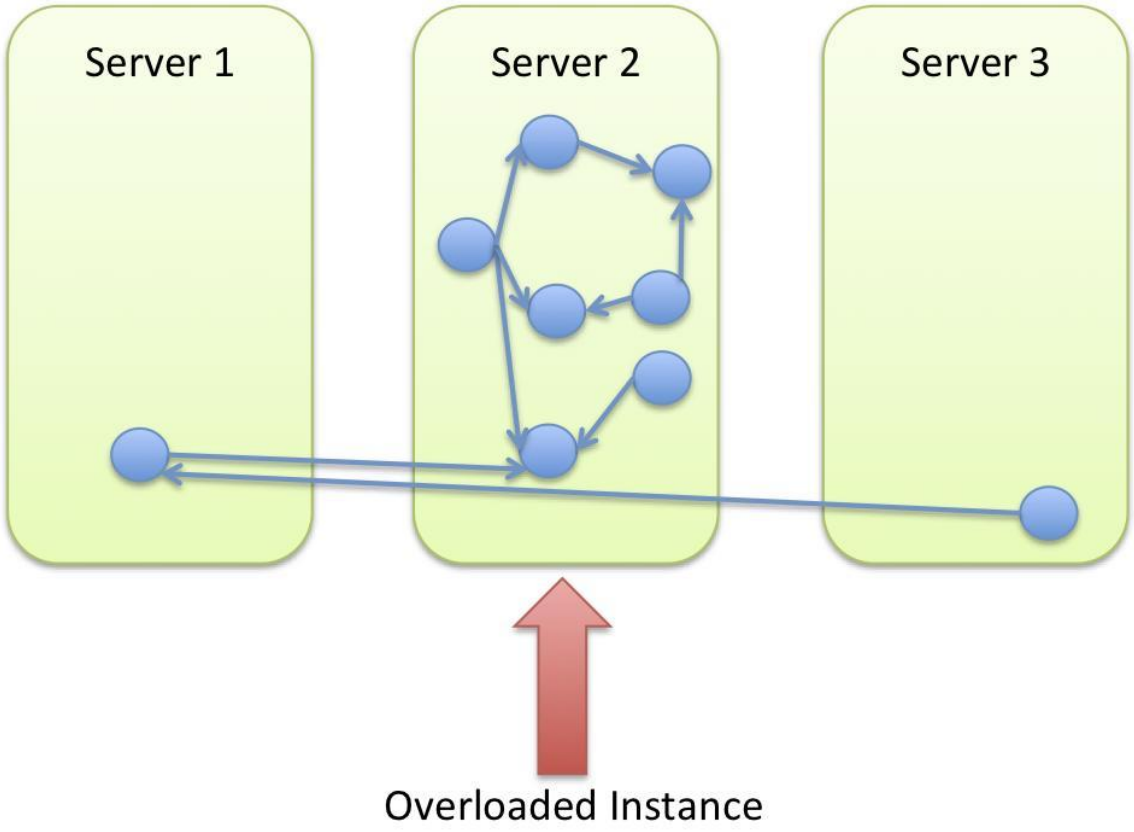
Gráfadatbázisok klaszterben

- A NoSql adatbázisok klasztereken futnak
- Klaszter:
  - Számítógépek összekapcsolt csoportja
- Egy klaszter skálázható új node-okkal

- Hogyan tároljunk gráfot a klaszterben?

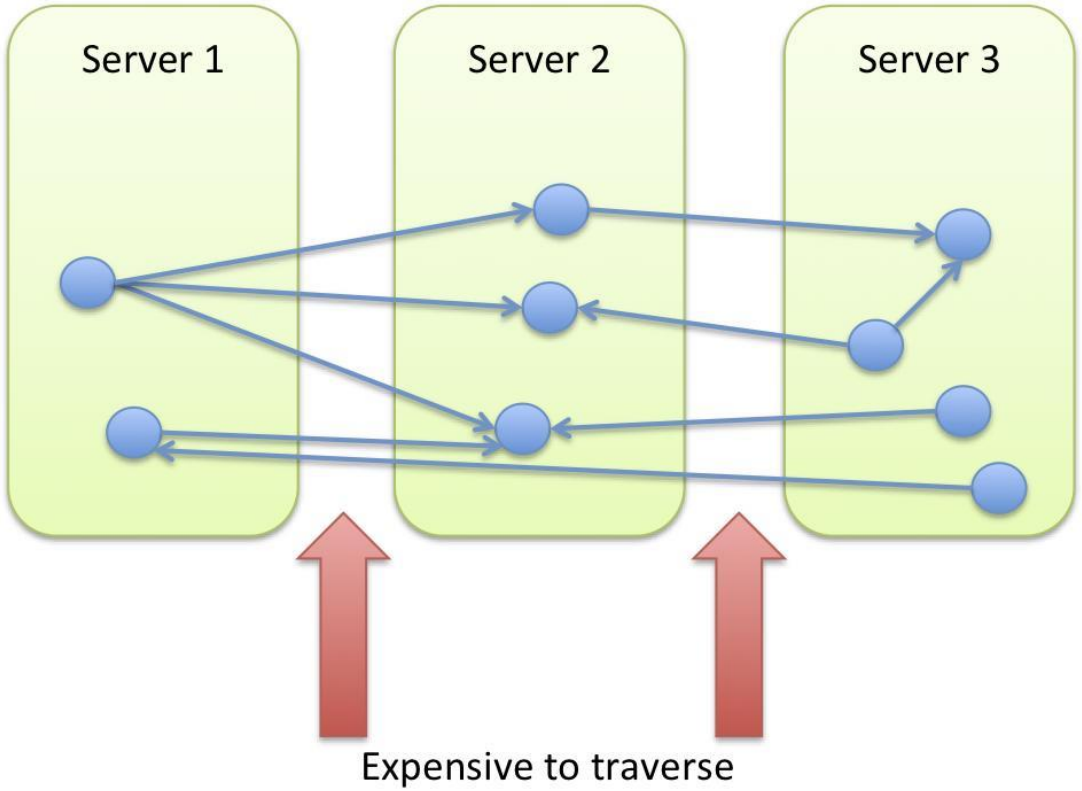
“Fekete lyuk” server

- Előny:
  - Kevés kommunikáció a node-ok között
- Hátrány:
  - Túlterhelt szerver



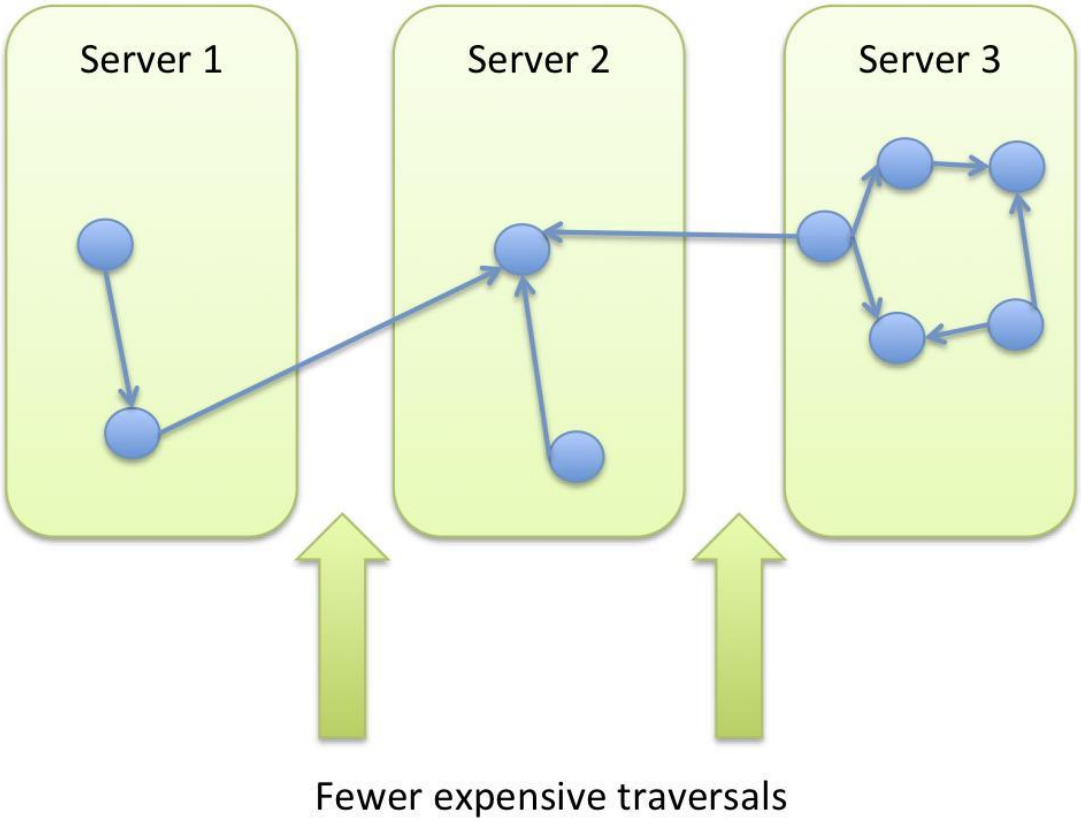
Csevegő hálózat

- Előny:
  - Egyenletes terhelés eloszlás
- Hátrány:
  - Sok kommunikáció a node-ok között



Minimális vágás

- Legjobb megoldás:
  - Terhelés eloszlik
  - Kevés kommunikáció a node-ok között



Neo4j

- Gráfadatbázis
  - Sémafüggetlen
  - ACID tranzakciós modell
  - Skálázható
  - Java, beágyazható
  - Magas rendelkezésre állás (commercial)
  - Deklaratív lekérdezőnyelv
  - REST API

Gráfadatbázisok: Előnyök és Hátrányok

- Előny
  - Erős adatmodellezés, általánosabb mint az RDBMS
  - Gyorsabb lekérdezés összekapcsolt adatokra mint az RDBMS-ben
  - Könnyű lekérdezhetőség
- Hátrány:
  - „Gráfos” gondolkodás
  - Globális lekérdezés / csúcsszintű számítás
  - Bináris adatok tárolása

Egyszerű kapcsolatok

- **Nodes**

(a)	actors
(m)	movies
( )	anonymous node

- **Relationships**

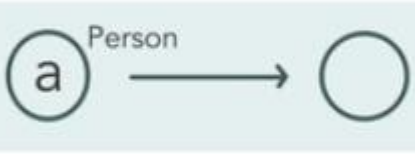
-[r]->	"r„ kapcsolat
(a)-[r]->(m)	actors "r" kapcsolatban a movies-zal
-[:ACTED_IN]->	ACTED_IN típusú kapcsolat
(a)-[:ACTED_IN]->(m)	actors ACTED_IN kapcsolatban
(d)-[:DIRECTED]->(m)	director DIRECTED kapcsolatban

Egyszerű kapcsolatok

Connected Nodes


```
(a)-->( )
```

Labels




```
(a:Person)
```

Connected Nodes


```
(a)-->(b)
```

Relationships


```
(a)-[r]->( )
```

Relationships Have Properties


```
(a)-[:ACTED_IN]->(m)
```

Attribútumok

- Node-oknak, kapcsolatoknak lehetnek attribútumai. Az attribútumok kulcs-érték párok.
  - Pl: actor, név

- **Node attribútunok**
  (m {title:"The Matrix"}) Movie title attribútummal
  (a {name:"Keanu Reeves",born:1964})

- **Relationship with Properties**
  (a)-[:ACTED_IN {roles:["Neo"]}]->(m)
  Kapcsolat (*ACTED_IN) roles* attribútummal

Címkék

- A címkék segítenek különbséget tenni a node-ok között. Egy node rendelkezhet több címkével is.
  - Pl: *Clint Eastwood: Actor, Director*.

- **Címkék**
  (a:Person)
  (a:Person {name:"Keanu Reeves"})
  (a:Person)-[:ACTED_IN]->(m:Movie)

Cypher

- Cypher a Neo4j lekérdező nyelve

- Cypher
  - dekleratív
  - SQL inspirált

Cypher

Lekérdezés

- **MATCH**: Az adott gráfminta keresése a gráfban.

  WHERE: Lekérdezés szűrése.

  RETURN: Visszatérési érték, projekció, aggregálás.

  ORDER BY: Rendezzük az eredményt.

  SKIP/LIMIT: Lapozás.

Cypher

Update utasítás

- **CREATE**: Csúcs és él létrehozás.
  MERGE: Egyedi node létrehozás.
  CREATE UNIQUE: Eygedi kapcsolat létrehozás.
  DELETE: Csúcsok, kapcsolatok törlése.
  SET: Cimkék és tulajdonságok beállítása.
  REMOVE: Címkék és tulajdonságok törlése.
  FOREACH: Egy adott listában lévő elemeken való módosítás.
  WITH: A lekérdezés felosztása különböző részekre és az eredmény átadása a következő lekérdezésnek.

Cypher

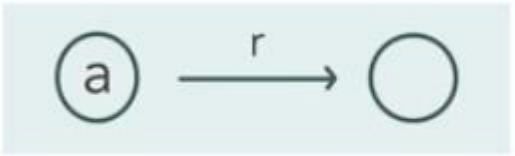
Two Nodes, One Relationship


```
MATCH (a)-->(b)
RETURN a, b;
```

Returning a Property


```
MATCH (a)-->( )
RETURN a.name;
```

Naming a Relationship




```
MATCH (a)-[r]->( )
RETURN a.name, type (r);
```

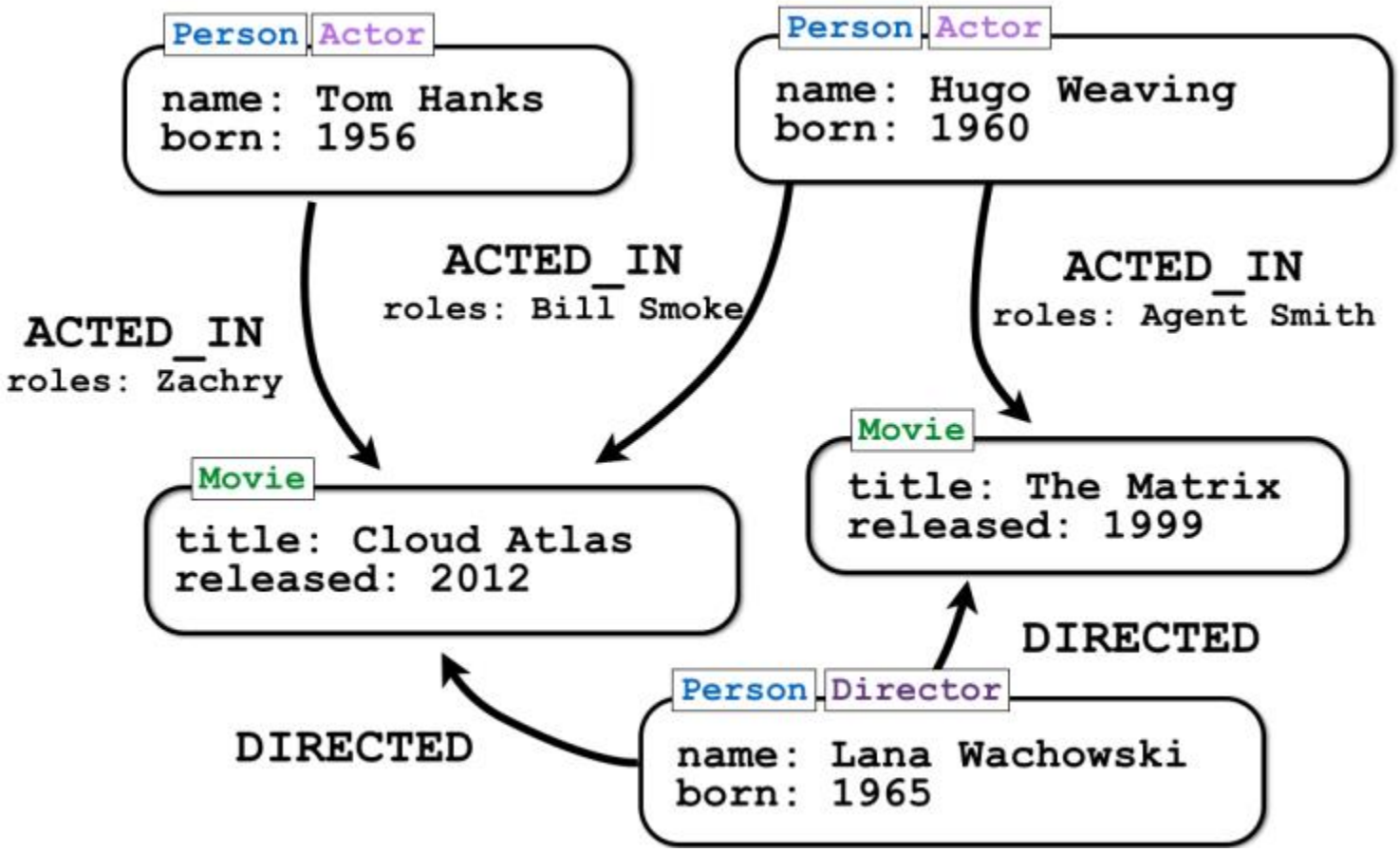
Matching by Relationship


```
MATCH(a)-[:ACTED_IN]->(m)
RETURN a.name, m.title;
```

Match Node Label

```
MATCH (tom:Person)
WHERE tom.name="Tom Hanks"
RETURN tom;
```

Példa gráf

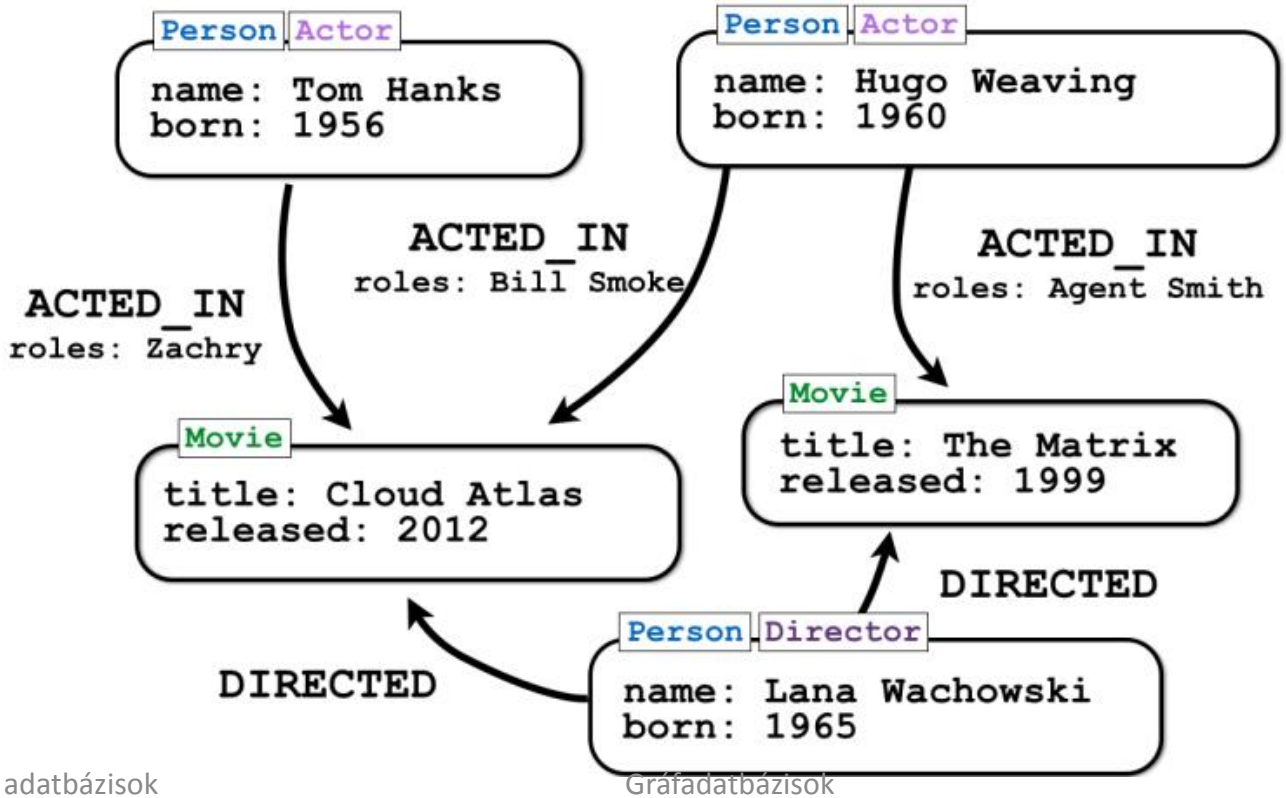

Person
Actor
name: Tom Hanks
born: 1956
Person
Actor
name: Hugo Weaving
born: 1960
ACTED_IN
roles: Zachry
ACTED_IN
roles: Bill Smoke
ACTED_IN
roles: Agent Smith
Movie
title: Cloud Atlas
released: 2012
Movie
title: The Matrix
released: 1999
DIRECTED
DIRECTED
Person
Director
name: Lana Wachowski
born: 1965

Kérdezzük le az összes „Matrix” karaktert

MATCH (m:Movie)<-[r:ACTED_IN]-(a:Person)

WHERE m.title=„The Matrix”

RETURN r.roles, actor.name



Paths

- Több utat is írhatunk egyszerre

```
MATCH (a)-[:ACTED_IN]->(m)<-[:DIRECTED]-(director)
RETURN a.name, m.title, d.name
```

- Ha túl sok van, akkor olvashatatlan

```
MATCH (a)-[:ACTED_IN]->(m),
      (m)<-[:DIRECTED]-(director)
RETURN a.name, m.title, d.name
```

Paths

- Utakat el lehet nevezni, és eredményként visszakapni

```
MATCH p=(a)-[:ACTED_IN]->(m)<-[:DIRECTED]-(director)
RETURN p;
```

- Ill. az uton lévő csomópontokat

```
MATCH p=(a)-[:ACTED_IN]->(m)<-[:DIRECTED]-(director)
RETURN nodes(p);
```

Egyszerűsítés

- Példa lekérdezés

```
MATCH (p:Person)
WHERE p.name=„Tom Hanks”
RETURN p;
```

- Egyszerűsítve

```
MATCH (p:Person {name=„Tom Hanks”})
RETURN p;
```

Limit, Skip, Disttinct, Order by

- Skip: eredmények kihagyása

```
MATCH (a)-[:ACTED_IN]->(m)
RETURN a.name, m.title
SKIP 10
LIMIT 10;
```

- Rendezés

```
MATCH (p:Person)
RETURN p.name
ORDER BY p.born
LIMIT 10;
```

Distinct

```
MATCH (a)-[:ACTED_IN]->()
RETURN distinct a
ORDER BY a.born
LIMIT 10;
```

Complex lekérdezés, Index

- Komplex lekérdezés

```
MATCH (g:Person {name: „Gene Hackman”})-[:ACTED_IN]->(movie),
(other)-[:ACTED_IN]->(movie),
(robin:Person {name:”Robin Williams”})
WHERE NOT (robin)-[:ACTED_IN]->(movie)
RETURN DISTINCT other;
```

- Index

```
CREATE INDEX ON :Movie(title);


CREATE INDEX ON :Person(name);
```

Aggregáció

- Aggregációk
  - Count
  - Min, Max
  - Avg
- Collect: összegyűjtés

```
MATCH (a:Person)-[:ACTED_IN]->(m)
RETURN a.name, collect(m.title);
```

Gráf módosítás

- Mivel sémafüggetlen, bármely csúcshoz és élhez adhatunk bármilyen attribútumot

```
MATCH (m:Movie)
WHERE m.title=„Mystic River”
SET M.TAGLINE= „BLABLA”
RETURN m;
```

- Módosítani is tudunk attribútumot

```
MATCH (m:Movie)
WHERE m.title=„Mystic River”
SET M.released= 2003
RETURN m;
```

Merge

- Élek logikai vagy kapcsolata

```
MATCH (a)-[:ACTED_IN|DRECTED]->()<-[:ACTED_IN|DIRECTED]-(b)
RETURN a,b;
```

- Élek egyesítése

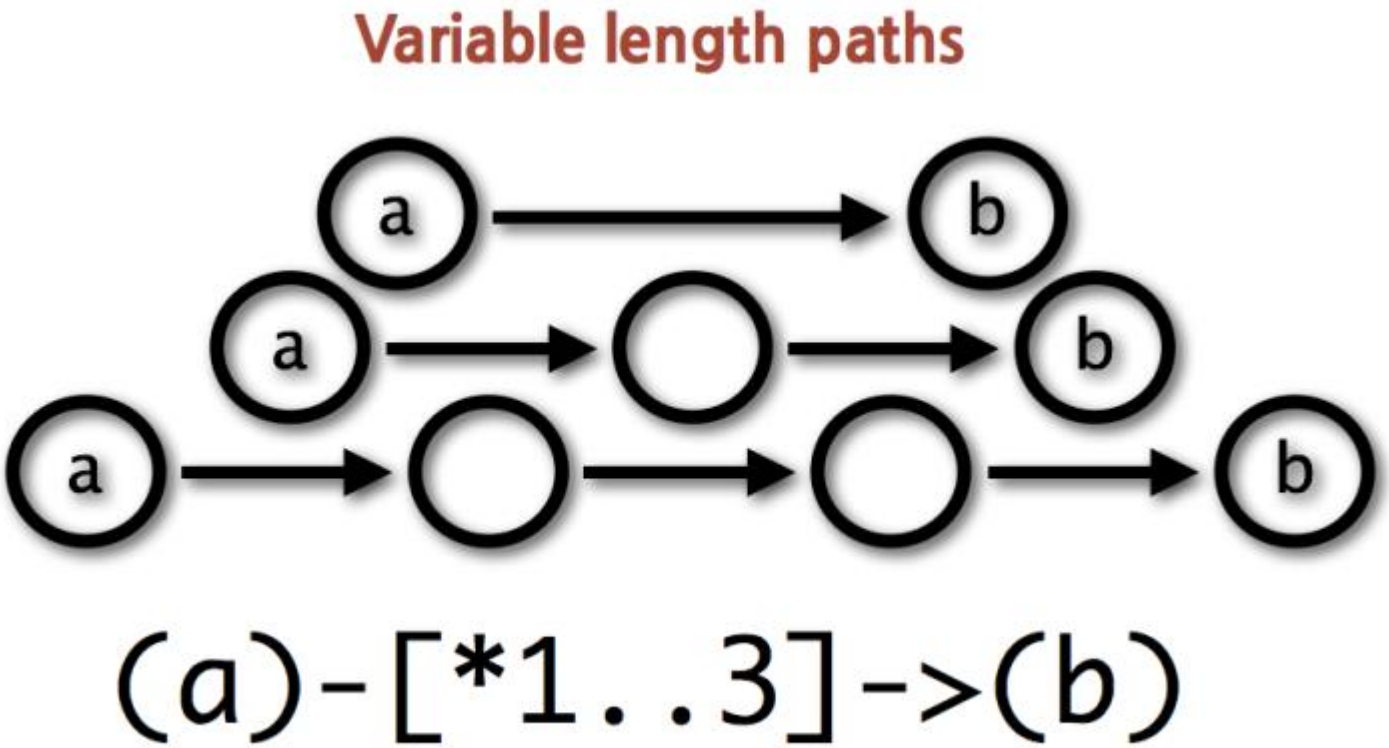
```
MATCH (a)-[:ACTED_IN|DRECTED]->()<-[:ACTED_IN|DIRECTED]-(b)
WHERE NOT a-[:KNOWS]-(b)
MERGE (a)-[:KNOWS]->(b);
```

Optional

- Töröljük magunkat, és kapcsolatainkat!

```
MATCH (me:Person {name=„My Name”})
OPTIONAL MATCH (me)-[r]-()
DELETE me, r;
```

Változó úthossz



neotechnology

shortestPath

- Legrövidebb út függvény

```
MATCH p=shortestPath( (keanu:Person)-[:KNOWS*]->(kevin:Person) )
WHERE keanu.name=„Keanu Reeves” and kevin.name=„Kevin Bacon”
RETURN length(p);
```

```
MATCH (keanu:Person {name=„Keanu Reeves”}),
          (kevin:Person {name=„ Kevin Bacon”}),
MATCH p=shortestPath( (keanu)-[:KNOWS*]->(kevin) )
RETURN length(p);
```

Köszönöm a Figyelmet!