Çizge Veritabanları ve Neo4J

Ümit Işıkdağ

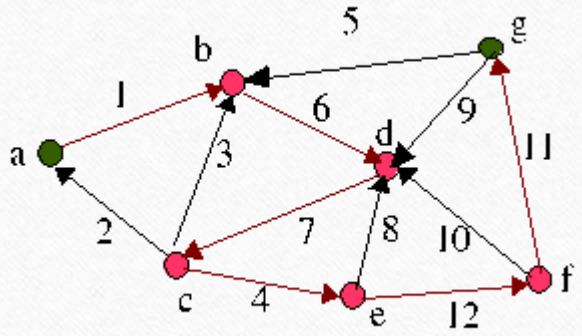
MSGSÜ
Enformatik Bölümü

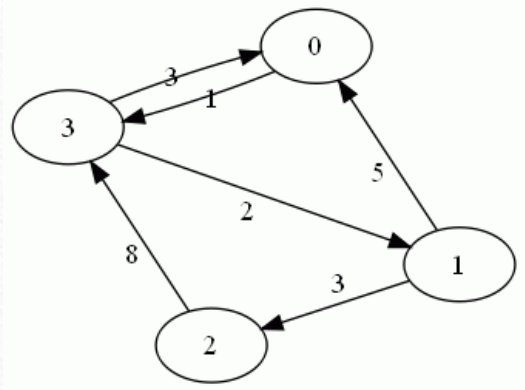
http://www.msgsu.edu.tr/faculties/enformatik-bolumu

https://www.facebook.com/msgsuenformatik

Çizgeler / Graflar

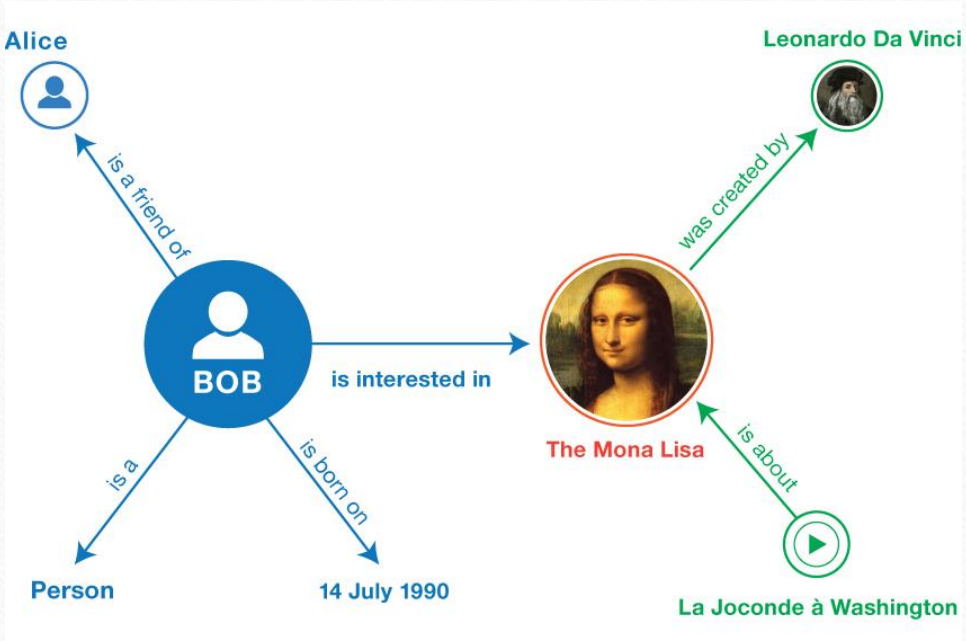
Belirli sayıda düğüm ve hat ile mevcut yada olabilecek bir gerçekliği yansıtan matematiksel gösterimler…

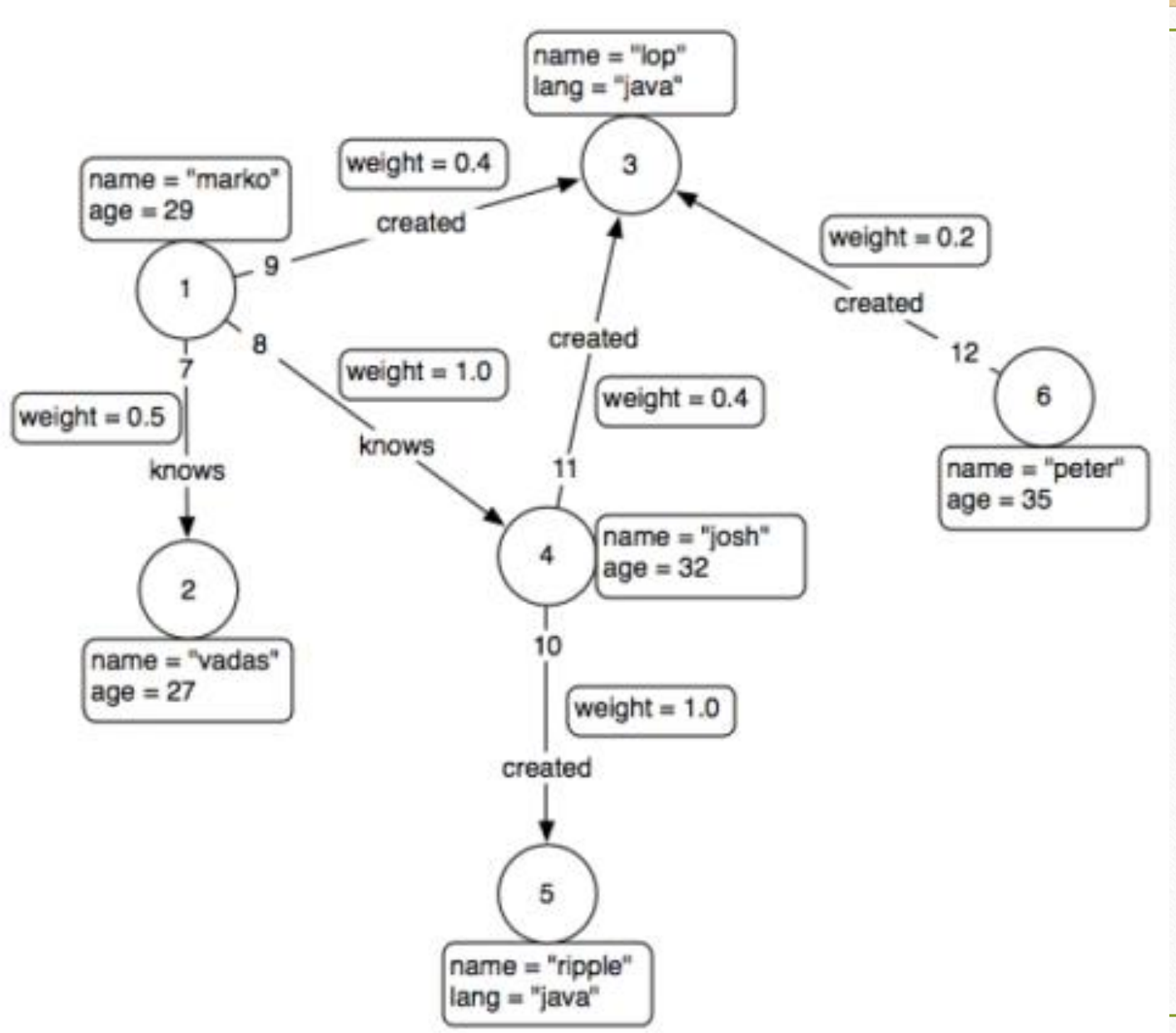




Çizge Veri Tabanı

- Verileri bir çizge yapısını ele alarak saklayan yazılım.
- Veri
  - Düğüm
  - Düğüm Öznitelikleri
  - Hat
  - Hat Öznitelikleri

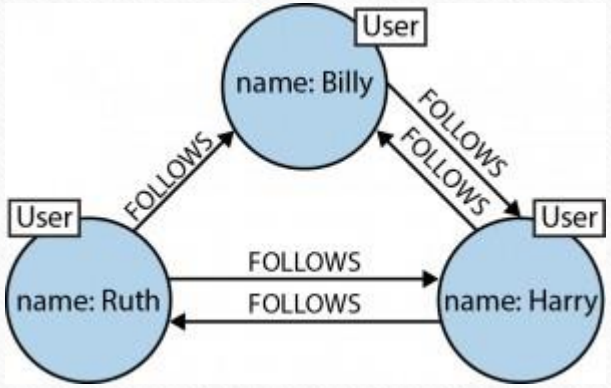



name = "lop"
lang = "java"
3
weight = 0.4
created
name = "marko"
age = 29
9
1
8
7
weight = 0.5
knows
2
name = "vadas"
age = 27
weight = 1.0
knows
created
weight = 0.4
11
4
name = "josh"
age = 32
10
weight = 1.0
created
5
name = "ripple"
lang = "java"
weight = 0.2
created
12
6
name = "peter"
age = 35

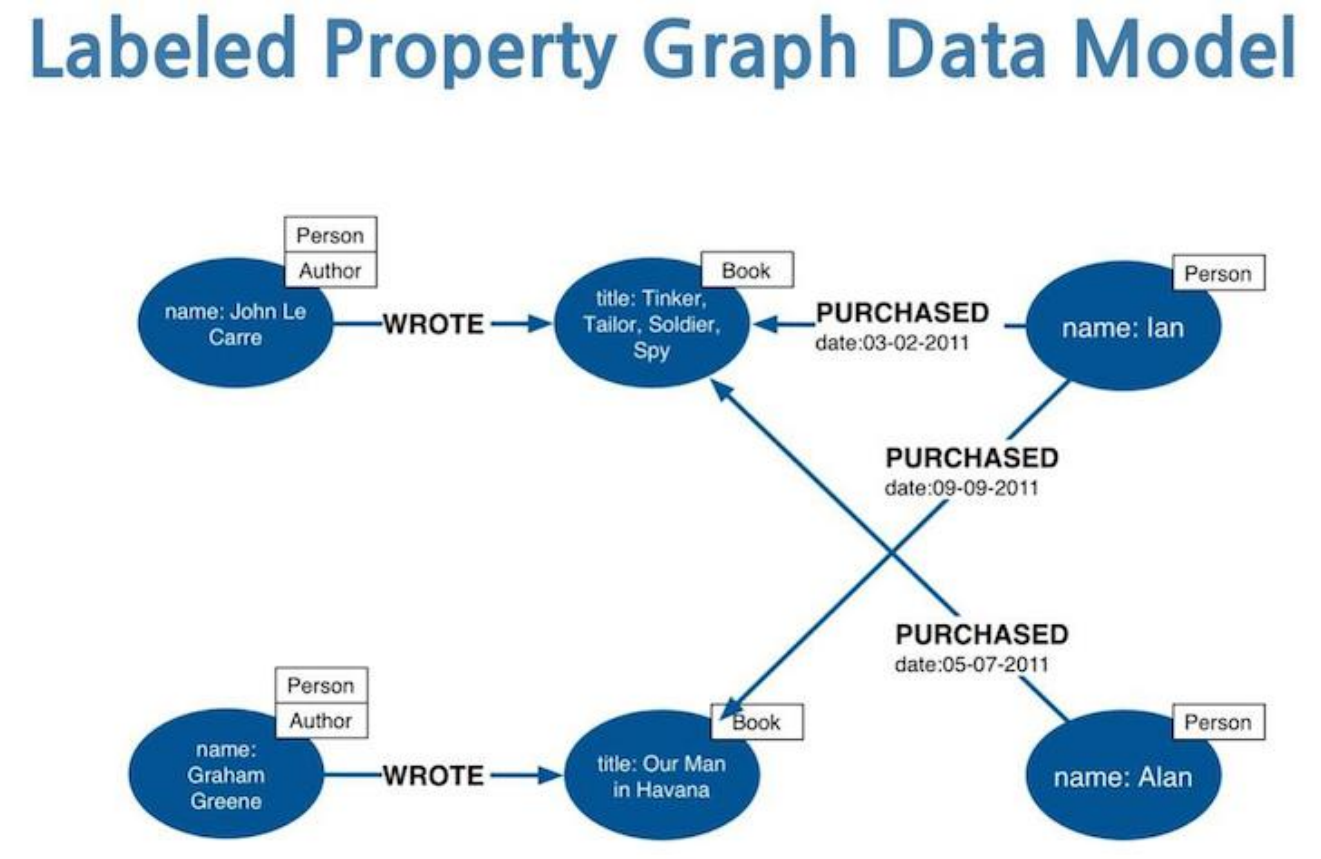
Çizge Veritabanı 3 temel avantajı

- Performans
- Esneklik
- Çeviklik
- «İlişki»ler de saklanır.
- Hem de yönleri ile ☺

Twitter Kullanıcıları



Gelişmiş Çizge Veri Modeli



RDBMS: Varlıklar tablolarda tutulur.

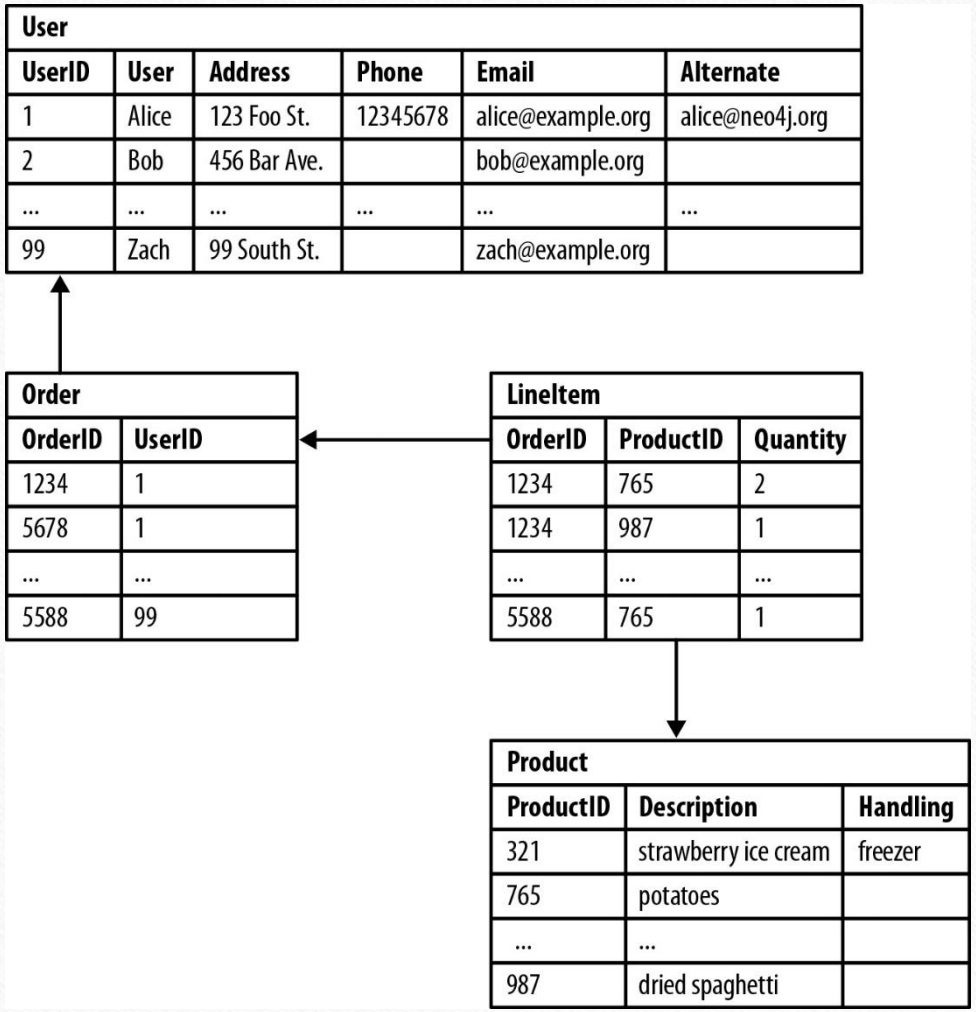
Birincil ve Yabancı Anahtar ile ilişkiler ifade edilir.

Veri tabanı ilişki türlerini saklamaz. Sadece ilişkinin varlığı bilinir.

İlişki Sorgulanmak isteniyor ise ayrı tablolara gereksinim vardır.

Yönler 1 tanedir ve ancak VT şemasından okunarak türetilir.

Pahalı sorgular vardır: «User 1 hangi ürünleri almıştır ?» gibi



User

UserID	User	Address	Phone	Email	Alternate
1	Alice	123 Foo St.	12345678	alice@example.org	alice@neo4j.org
2	Bob	456 Bar Ave.		bob@example.org	
...	...	...	...	...	...
99	Zach	99 South St.		zach@example.org	

Order

OrderID	UserID
1234	1
5678	1
...	...
5588	99

LineItem

OrderID	ProductID	Quantity
1234	765	2
1234	987	1
...	...	...
5588	765	1

Product

ProductID	Description	Handling
321	strawberry ice cream	freezer
765	potatoes	
...	...	
987	dried spaghetti	

Performans Karşılaştırması

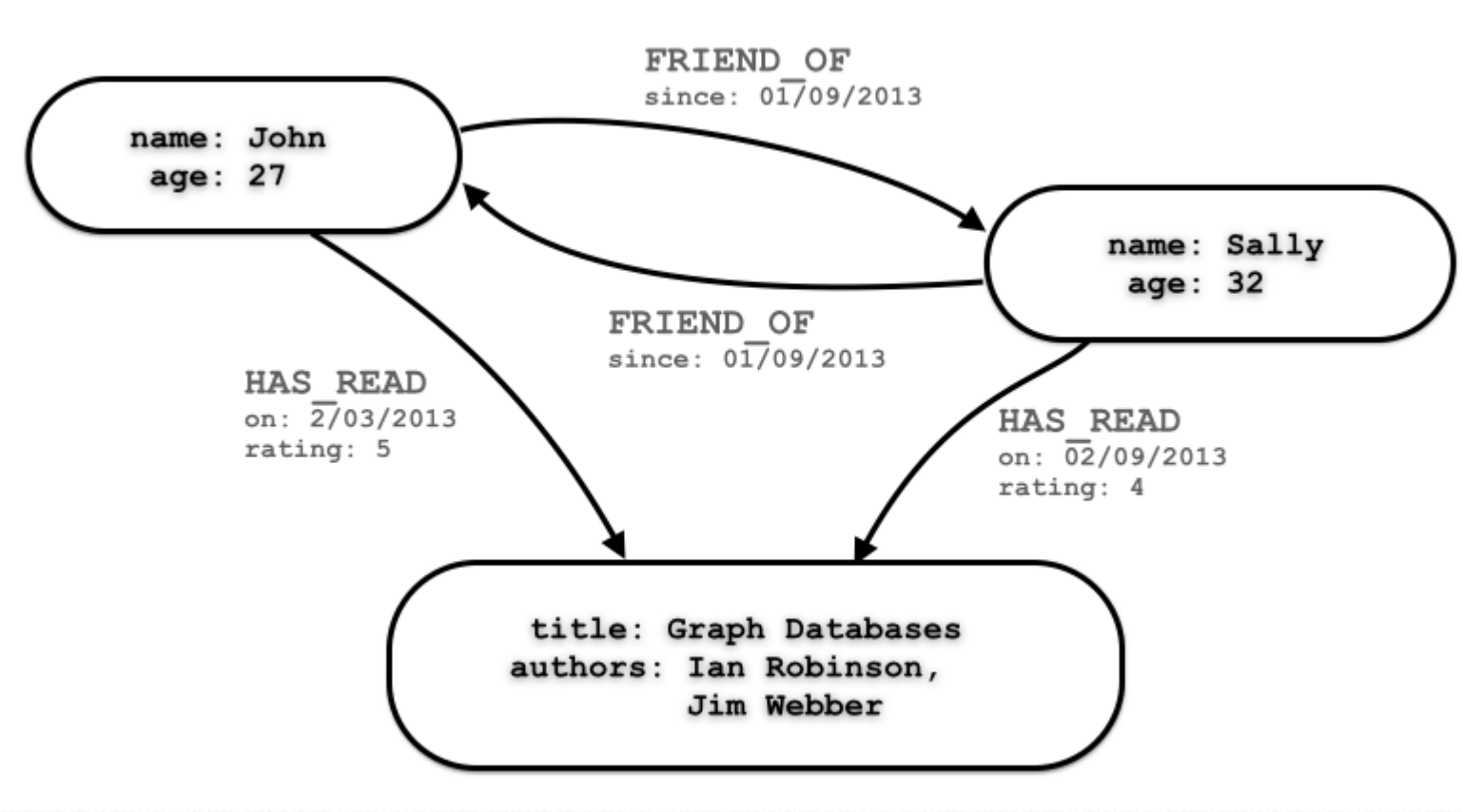
Aranılan veri→ Sosyal Ağ / Arkadaşlar ve arkadaşları ağı
Toplam Ağ Derinliği →5.derece
Veri Hacmi→1.000.000 kişi / 50 arkadaş ortalama

Örnek: Ali nin arkadaşlarını bul (1.derece)
Ali nin arkadaşlarının arkadaşlarını bul (2.derece)

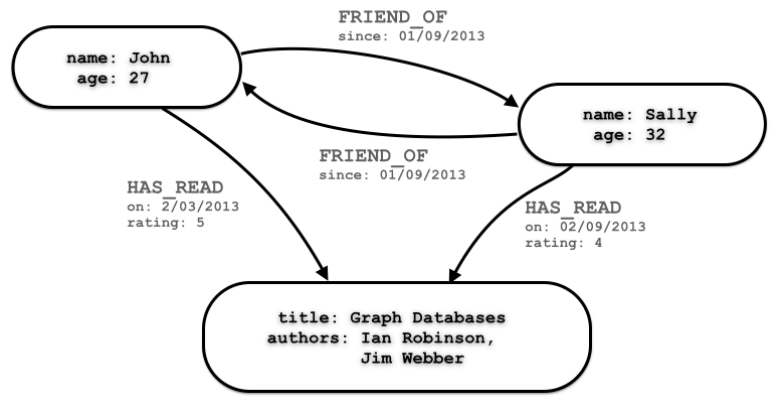
Depth	RDBMS execution time(s)	Neo4j execution time(s)	Records returned
2	0.016	0.01	~2500
3	30.267	0.168	~110,000
4	1543.505	1.359	~600,000
5	Unfinished	2.132	~800,000

RDBMS e veda edelim mi?

- Join leriniz fazla ise
- Sorgularımız uzun cümleler olmuş ise(örneğin 40 satırdan fazla)
- Şemanız sürekli değişmek zorunda kalıyorsa
- Sorgularınıza cevaplar geç geliyorsa
- Graf VT bir deneyin …


FRIEND_OF
since: 01/09/2013
name: John
age: 27
name: Sally
age: 32
FRIEND_OF
since: 01/09/2013
HAS_READ
on: 2/03/2013
rating: 5
HAS_READ
on: 02/09/2013
rating: 4
title: Graph Databases
authors: Ian Robinson,
Jim Webber

```
// Create Sally
CREATE (sally:Person { name: 'Sally', age: 32 })

// Create John
CREATE (john:Person { name: 'John', age: 27 })

// Create Graph Databases book
CREATE (gdb:Book { title: 'Graph Databases', authors: ['Ian Robinson', 'Jim Webber'] })

// Connect Sally and John as friends
CREATE (sally)-[:FRIEND_OF { since: 1357718400 }]->(john)

// Connect Sally to Graph Databases book
CREATE (sally)-[:HAS_READ { rating: 4, on: 1360396800 }]->(gdb)

// Connect John to Graph Databases book

CREATE (john)-[:HAS_READ { rating: 5, on: 1359878400 }]->(gdb)
```



When did John and Sally become Friends?

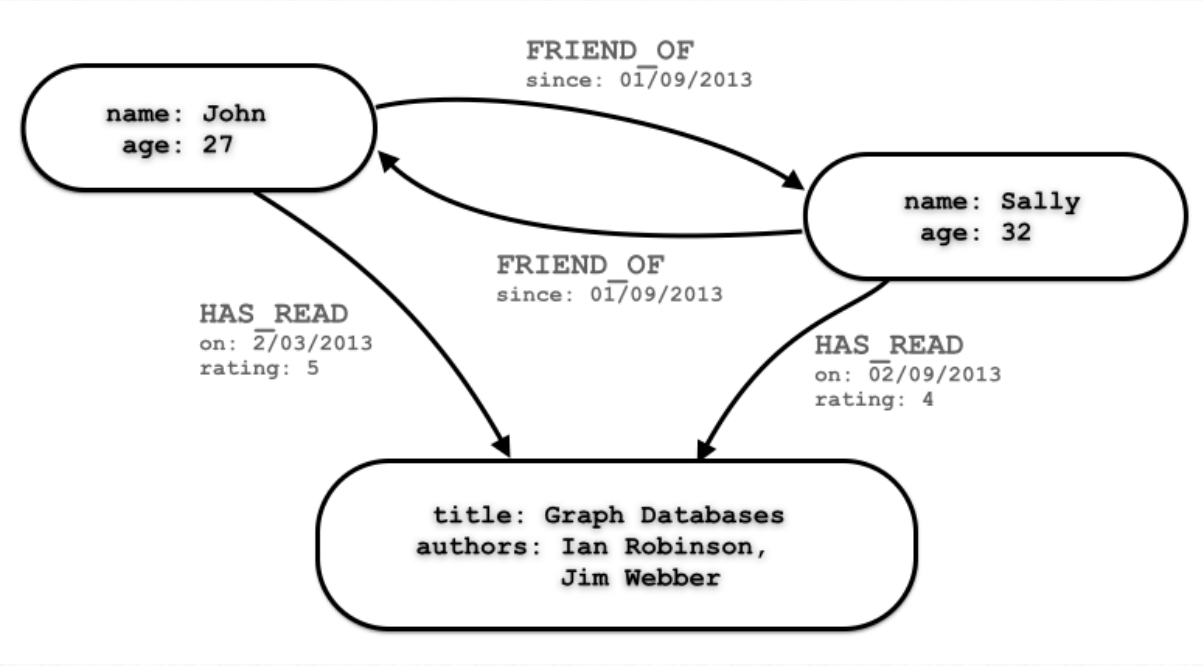
```
MATCH (sally:Person { name: 'Sally' })
MATCH (john:Person { name: 'John' })
MATCH (sally)-[r:FRIEND_OF]-(john)
RETURN r.since as friends_since
```

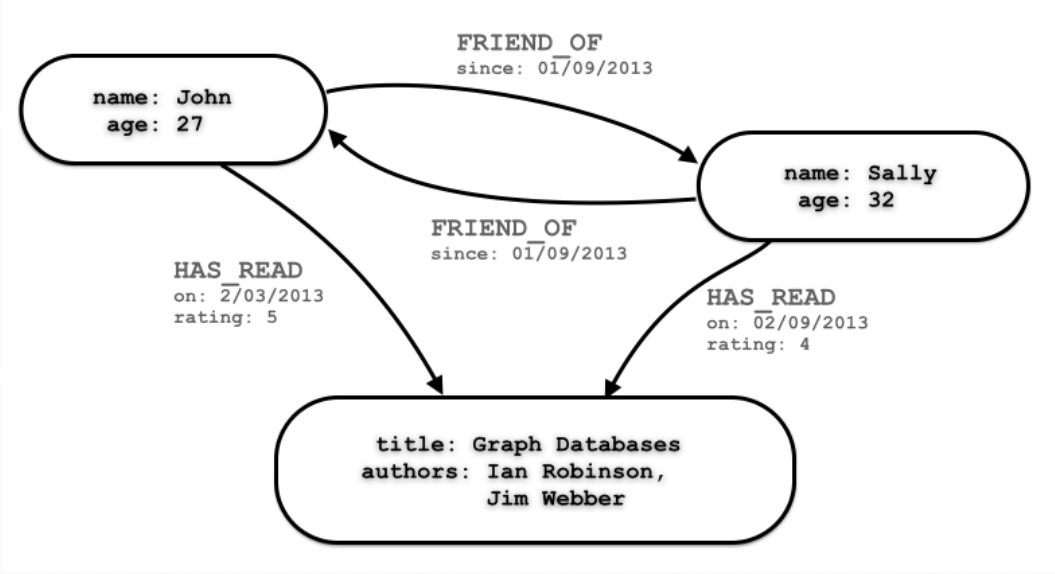
What is the average rating of *Graph Databases*?

```
MATCH (gdb:Book { title: 'Graph Databases' })
MATCH (gdb)<-[r:HAS_READ]-()
RETURN avg(r.rating) as average_rating
```

Who are the authors of *Graph Databases*?

```
MATCH (gdb:Book { title: 'Graph Databases' })
RETURN gdb.authors as authors
```





How old is Sally?

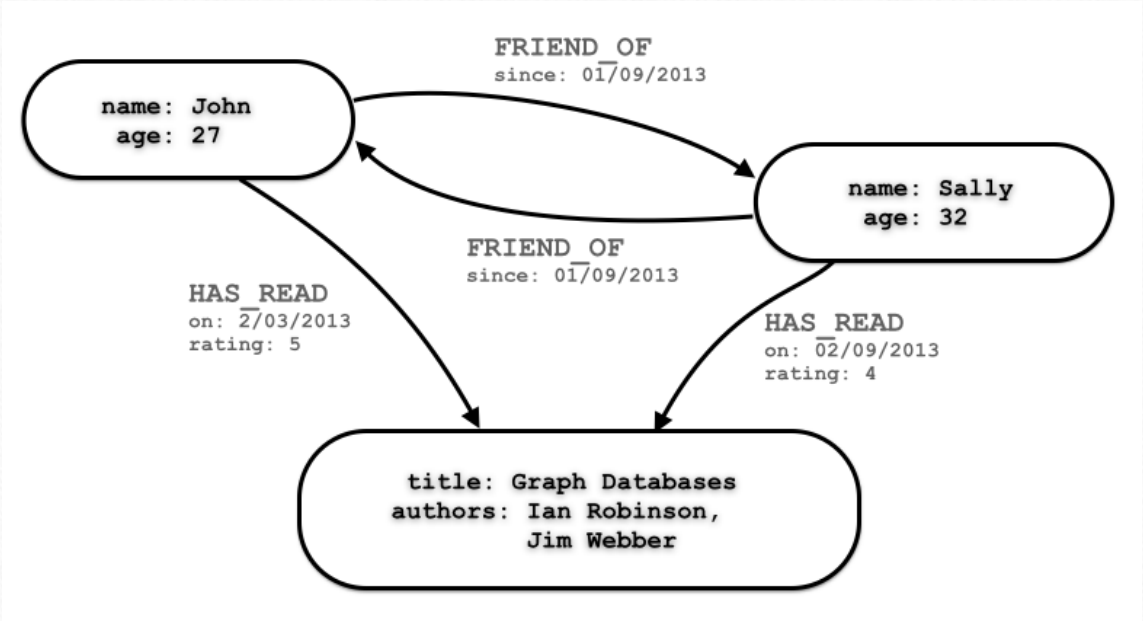
```
MATCH (sally:Person { name: 'Sally' })
RETURN sally.age as sally_age
```

Who is older, Sally or John?

```
MATCH (people:Person)
WHERE people.name = 'John' OR people.name = 'Sally'
RETURN people.name as oldest
ORDER BY people.age DESC LIMIT 1
```

Who Read *Graph Databases* First, Sally or John?

```
MATCH (people:Person) WHERE people.name = 'John' OR people.name = 'Sally'
MATCH (people)-[r:HAS_READ]->(gdb:Book { title: 'Graph Databases' })
RETURN people.name as first_reader
ORDER BY r.on LIMIT 1
```



Nasıl Başlayalım

- http://neo4j.com/download/other-releases/ (Enterprise i indirin)
- Unzip ile dizine kurun (Java yuklu olsun)
- http://localhost :7474

İlk adımlar

CREATE (ee:Person { name: "Emil", from: "Sweden", age: 22 })

1 düğüm var tipi Person ismi ee

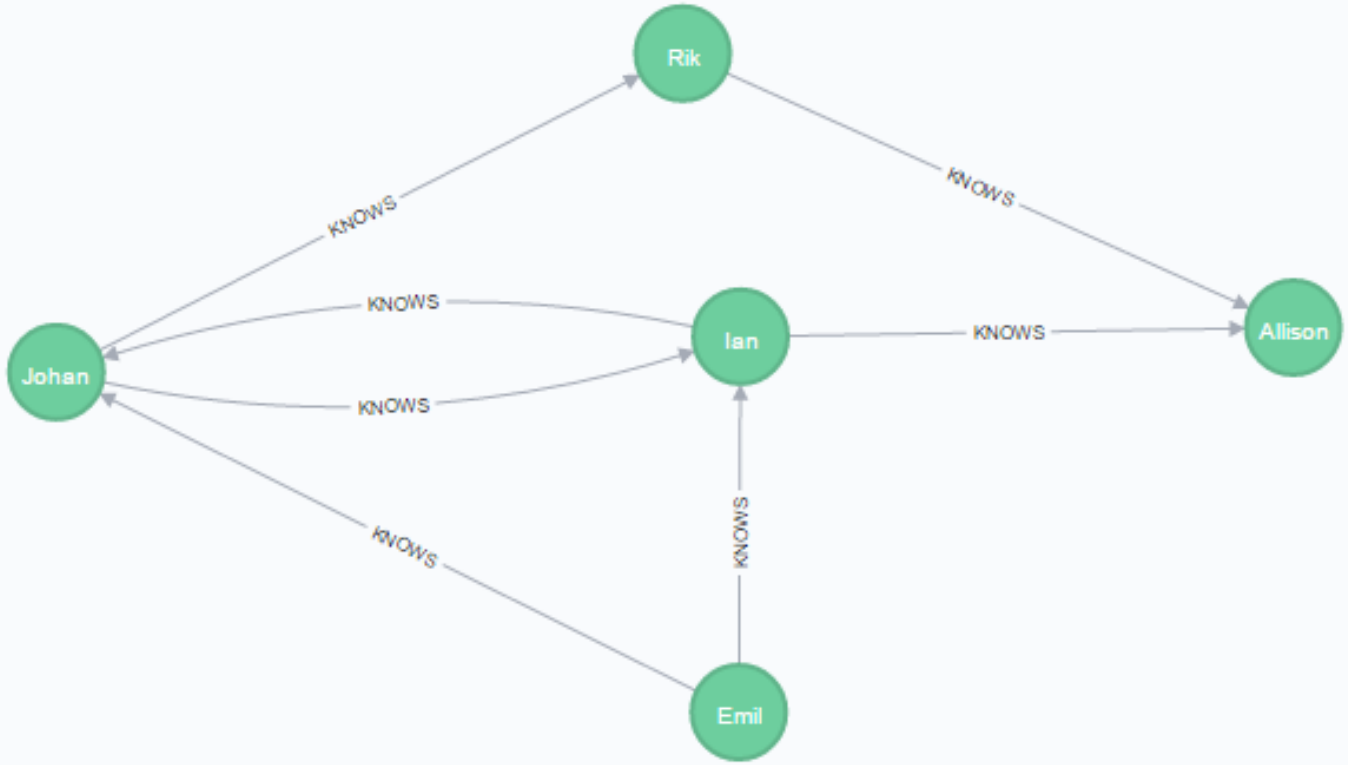
3 öznitelik var name, from ve age

İsmi Emil olanları göster

```
MATCH (ee:Person) WHERE ee.name = "Emil" RETURN ee;
```

```
CREATE
(ee:Person { name: "Emil", from: "Sweden", klout: 99 }),
(js:Person { name: "Johan", from: "Sweden", learn: "surfing" }),
(ir:Person { name: "Ian", from: "England", title: "author" }),
(rvb:Person { name: "Rik", from: "Belgium", pet: "Orval" }),
(ally:Person { name: "Allison", from: "California", hobby: "surfing" }),
(ee)-[:KNOWS {since: 2001}]->(js),
(ee)-[:KNOWS {rating: 5}]->(ir),
(js)-[:KNOWS]->(ir),
(js)-[:KNOWS]->(rvb),
(ir)-[:KNOWS]->(js),
(ir)-[:KNOWS]->(ally),
(rvb)-[:KNOWS]->(ally)
```

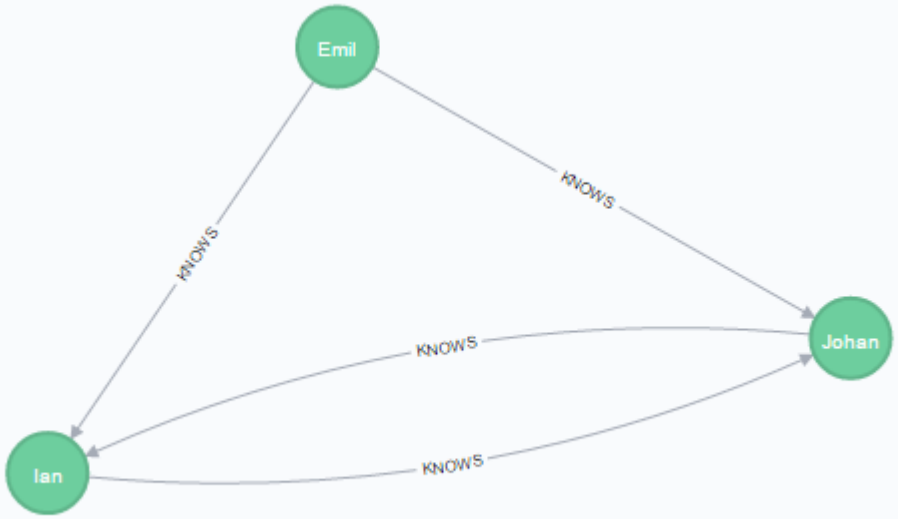
Video’dan devam edelim….


Rik
KNOWS
KNOWS
KNOWS
Johan
KNOWS
Ian
KNOWS
Allison
KNOWS
KNOWS
Emil

//a pattern can be used to find Emil's friends:

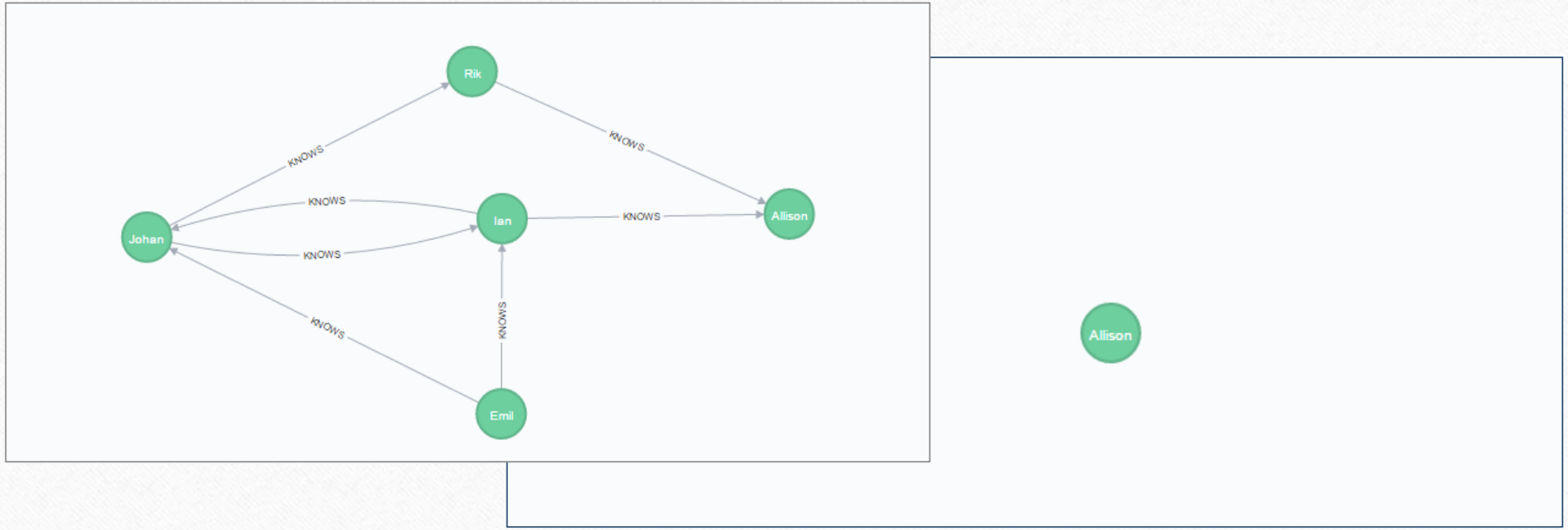
MATCH (ee:Person)-[:KNOWS]-(friends)

WHERE ee.name = "Emil" RETURN ee, friends

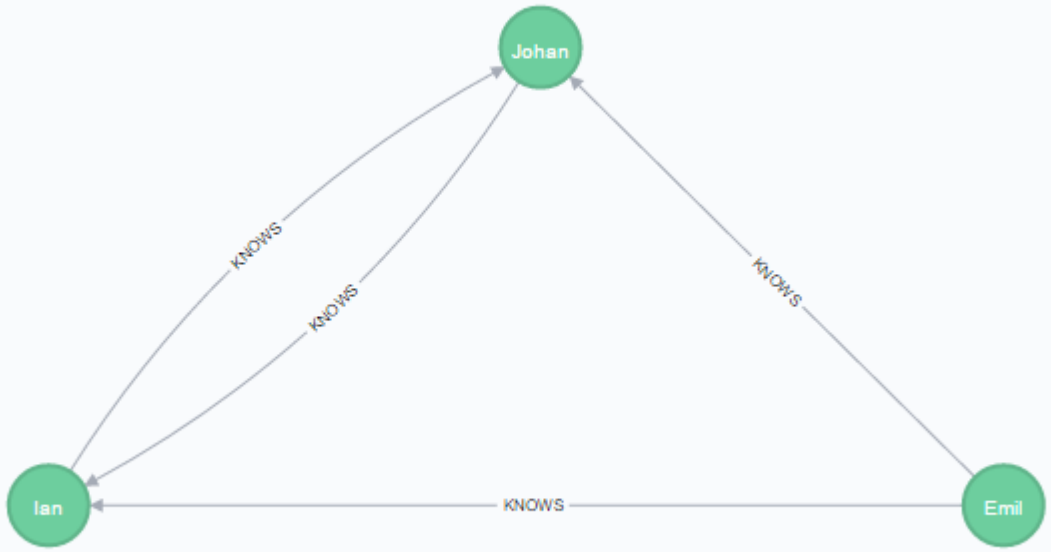


//Johan is learning to surf, so he may want to find a new friend who already does
//Johan ın tanışma ihtimali olan ve sörf yapan birisi ?

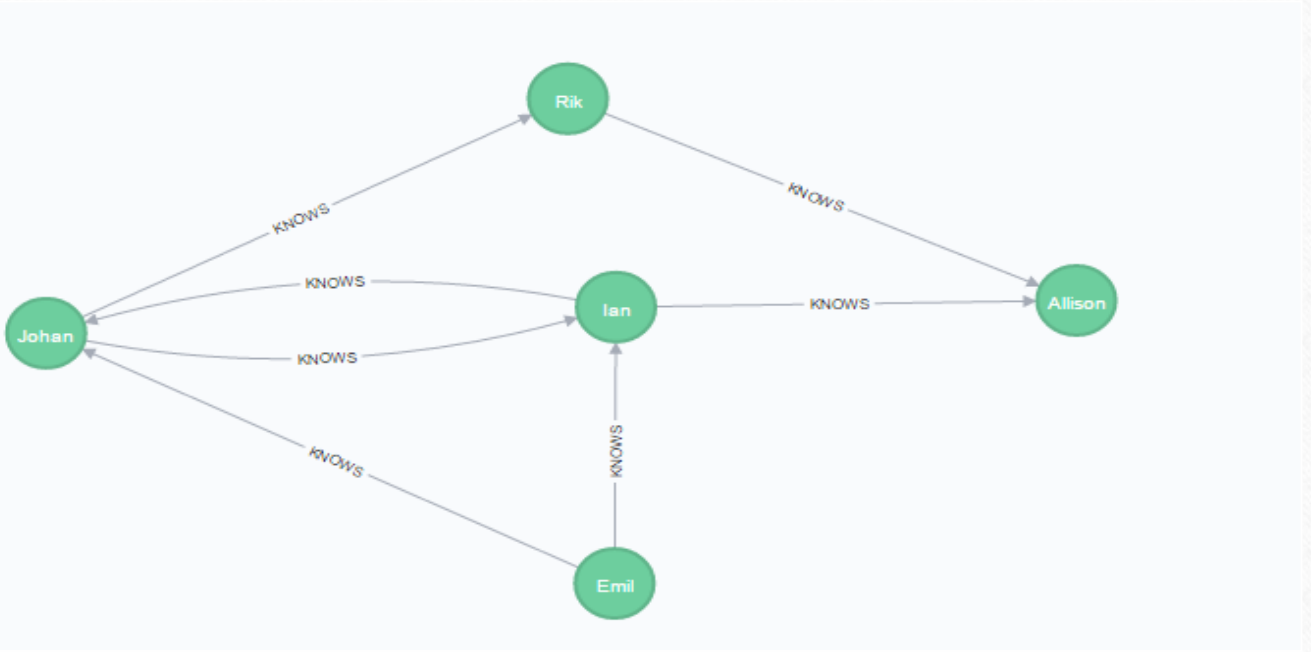
```
MATCH (js:Person)-[:KNOWS]-()-[:KNOWS]-(surfer)
WHERE js.name = "Johan" AND surfer.hobby = "surfing"
RETURN DISTINCT surfer
```



//A clique is a maximal set of vertices that generates a complete subgraph
// Herkesin birbirini tanıdığı 3 lü klik leri leri bulmak



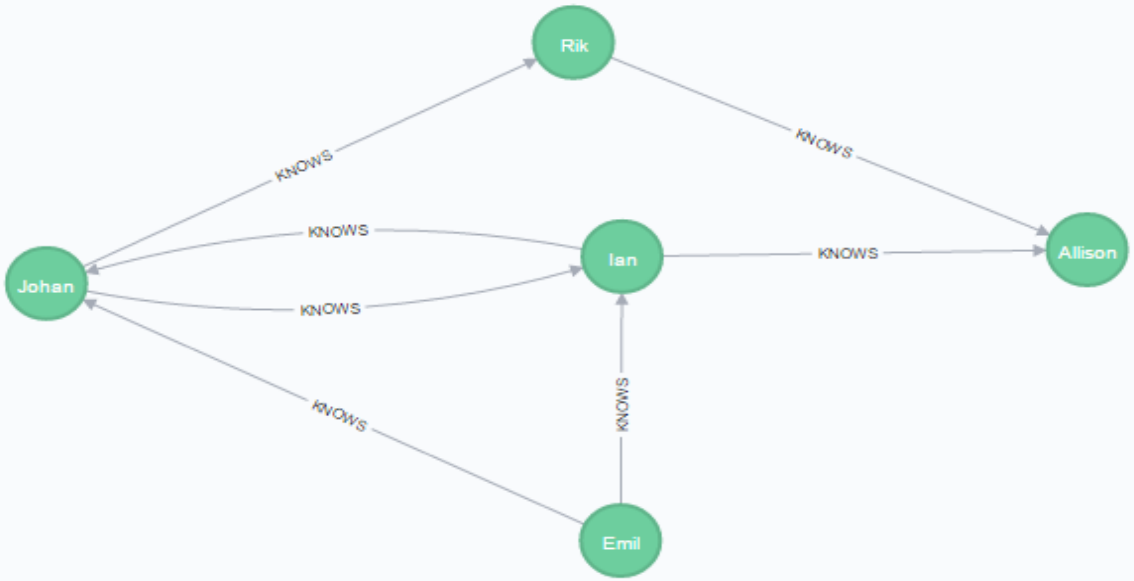
//Degree Centrality (En popüler/sosyal kim)

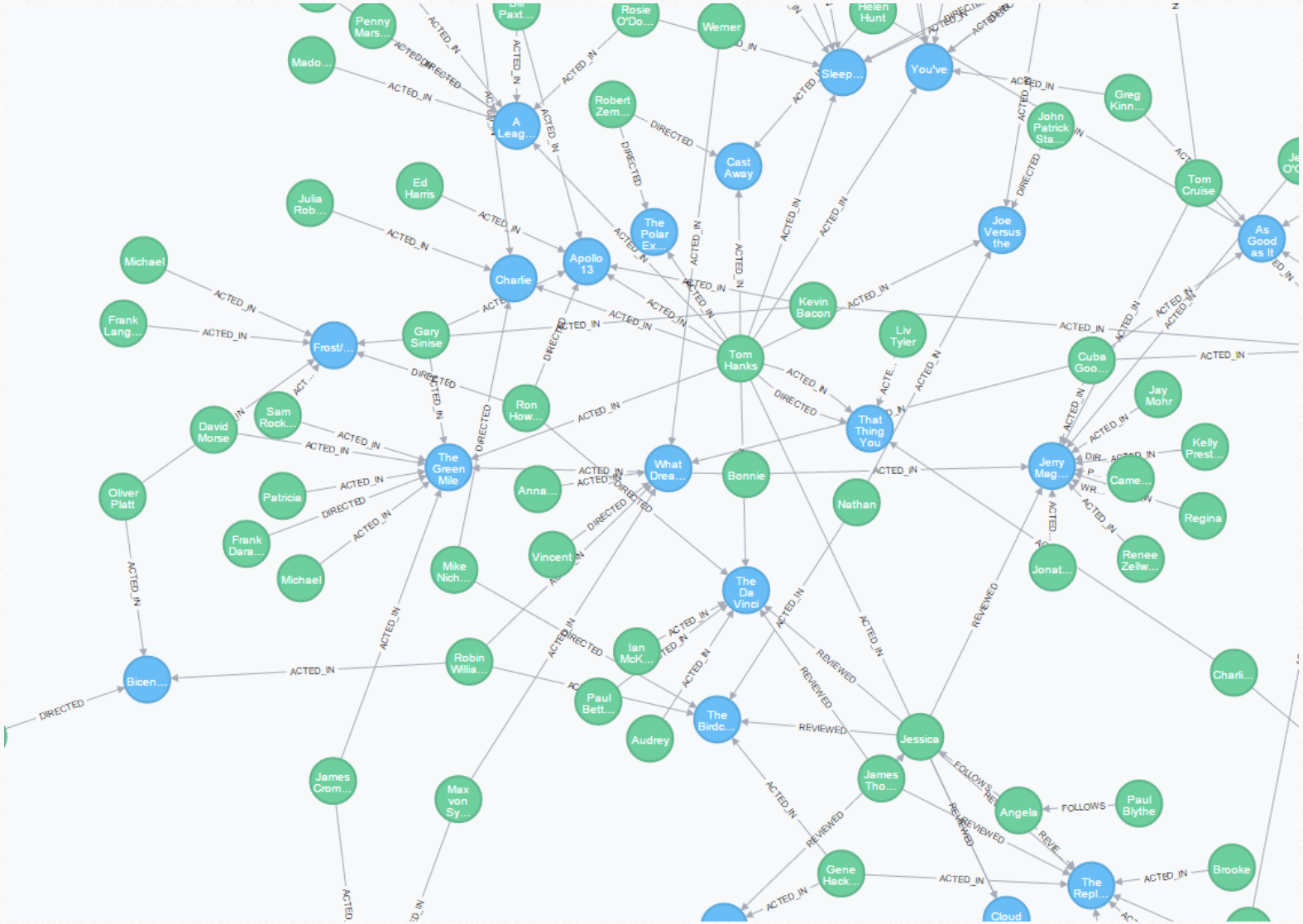


//Graph Theory: betweenness centrality

//BC= equal to the number of shortest paths from all vertices to all others that pass through that node
Bir düğümden o kadar fazla en kısa yol geçer.Bir düğüm ağ da BC kadar en kısa yolun üstünde yer alır

//A node with high betweenness centrality has a large influence on the transfer of items through the network, under the assumption that item transfer follows the shortest paths




Penny Mars...
Mado...
A Leag...
Robert Zem...
Rosie O'Do...
Werner
Helen Hunt
Sleep...
You've
Greg Kinn...
John Patrick Sta...
Tom Cruise
Cast Away
Julia Rob...
Ed Harris
The Polar Ex...
Joe Versus the
As Good as It
Michael
Charlie
Apollo 13
Kevin Bacon
Frank Lang...
Frost/...
Gary Sinise
Liv Tyler
Tom Hanks
Cuba Goo...
Jay Mohr
David Morse
Sam Rock...
Ron How...
That Thing You
Kelly Prest...
The Green Mile
What Drea...
Bonnie
Jerry Mag...
Came...
Oliver Platt
Patricia
Anna...
Nathan
Regina
Frank Dara...
Vincent
Renee Zellw...
Michael
Mike Nich...
Jonat...
The Da Vinci
Bicen...
Robin Willia...
Ian McK...
Charli...
Paul Bett...
The Birdc...
Audrey
Jessica
James Crom...
Max von Sy...
James Tho...
Angela
Paul Blythe
Gene Hack...
The Repl...
Brooke
Cloud
ACTED_IN
DIRECTED
REVIEWED
FOLLOWS

İnsanlar→Person isimli Düğümlerde
Filmler→Movie isimli Düğümlerde

6 farklı ilişki türü ise farklı Hatlar ile ifade edilebilir.

- Produced
- Reviewed
- Follows
- Wrote
- Acted_In
- Directed

Tüm relationshipleri döndür , tabular gösterim

MATCH (n)-[r]->()
RETURN type(r), count(*);

```
$ MATCH (n)-[r]->() RETURN type(r), count(*);
```

Graph

Rows

type(r)	count(*)
PRODUCED	15
REVIEWED	9
FOLLOWS	3
WROTE	10
ACTED_IN	172
DIRECTED	44

```
//Find Tom Hanks
MATCH (tom {name: "Tom Hanks"}) RETURN tom

//Find Cloud Atlas
MATCH (cloudAtlas {title: "Cloud Atlas"}) RETURN cloudAtlas

 //Find 10 people
 MATCH (people:Person) RETURN people.name LIMIT 10

//Find movies released in the 1990s...
MATCH (nineties:Movie) WHERE nineties.released > 1990 AND nineties.released < 2000 RETURN nineties.title
```

```
//List all Tom Hanks movies
MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED_IN]->(tomHanksMovies) RETURN tom,
tomHanksMovies


//Who directed CloudAtlas
MATCH (cloudAtlas {title: "Cloud Atlas"})<-[:DIRECTED]-(directors) RETURN
directors.name


//Tom Hanks co-actors
MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]->(m)<-[:ACTED_IN]-(coActors)
RETURN coActors.name
```