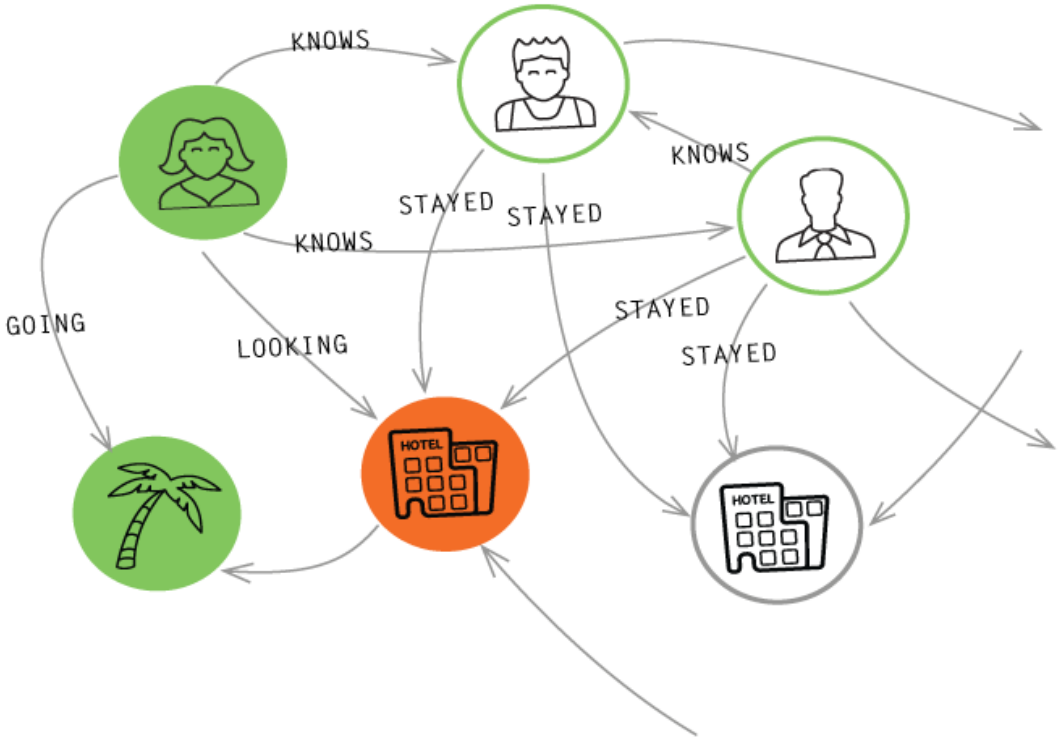
Neo4j

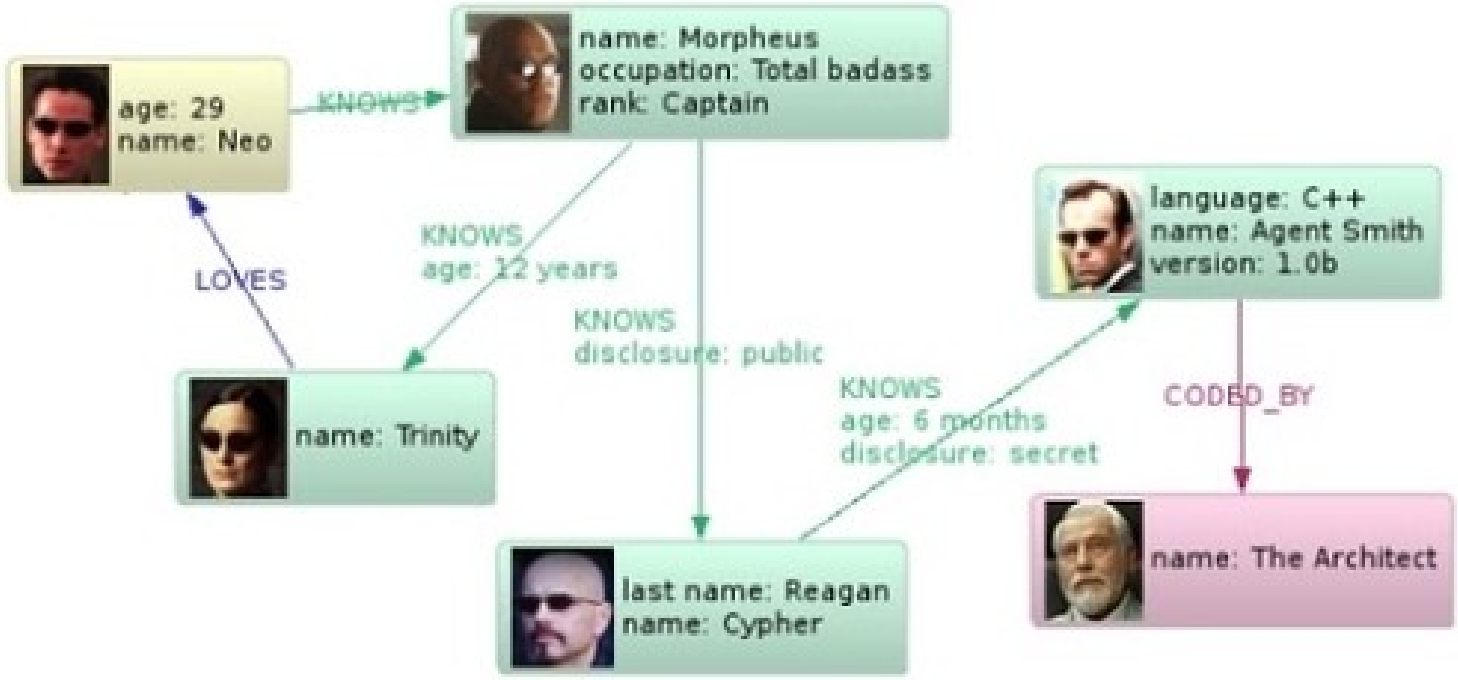


Written by Roni Licher

Last updated Winter 2015-2016

236363 - **Database Systems -** Technion

Graph Database, think different!



- Nodes
- Edges (directed or not)
- Properties

"We found Neo4j to be literally thousands of times faster than our prior MySQL solution, with queries that require 10-100 times less code. Today, Neo4j provides eBay with functionality that was previously impossible."

Volker Pacher, Senior Developer, eBay

Neo4j and Cypher

- Graph database (Like SQL server e.g. PostgreSQL, MySQL)
- Implemented in Java

- Graph query language for Neo4J (Like SQL)
- Declarative

A general query structure

MATCH [Nodes and relationships]
WHERE [Boolean filter statement]
RETURN [DISTINCT] [statements [AS alias]]
ORDER BY [Properties] [ASC\DESC]
SKIP [Number] LIMIT [Number]

First query

Get all nodes of type *Program* that have the name *Hello World!*:

```
MATCH (a : Program)
WHERE a.name = ‘Hello World!’
RETURN a
```

Query relationships

Get all relationships of type *Author* connecting *Programmers* and *Programs*:


```
MATCH (a : Programmer)-[r : Author]->(b : Program)
RETURN r
```

Matching nodes and relationships

Nodes:

(a), (), (:Ntype), (a:Ntype), (a{prop:'value'}) ,
(a:Ntype {prop:'value' })

Relationships:

(a)--(b), (a)-->(b), (a)<--(b), (a)-->(),
(a)-[r]->(b), (a)-[:Rtype]->(b), (a)-[:R1|:R2]->(b),
(a)-[r:Rtype]->(b)

May have more then 2 nodes:

(a)-->(b)<--(c), (a)-->(b)-->(c)

Path:

p = (a)-->(b)

More options:

- Relationship distance:

(a)-[:Rtype*2]->(b) – 2 hops of type Rtype.
(a)-[:Rtype*]->(b) – any number of hops of type Rtype.
(a)-[:Rtype*2..10]-> (b) – 2-10 hops of Rtype.
(a)-[:Rtype* ..10]-> (b) – 1-10 hops of Rtype.
(a)-[:Rtype*2..   ]-> (b) – at least 2 hops of Rtype.

Could be used also as:
(a)-[r*2]->(b) – r gets a sequence of relationships
(a)-[*{prop:val}]->(b)

Operators

- Mathematical
  +, -, *, /,%, ^ (power, not XOR)
- Comparison
  =,<>,<,>,>=,<=, =~ (Regex), IS NULL , IS NOT NULL
- Boolean
  AND, OR, XOR, NOT
- String
  Concatenation through +
- Collection
  Concatenation through +
  IN to check is an element exists in a collection.

More WHERE options

- WHERE others.name IN ['Andres', 'Peter']
- WHERE user.age IN range (18,30)
- WHERE n.name =~ 'Tob.*‘
- WHERE n.name =~ '(?i)ANDR.*‘ - (case insensitive)
- WHERE (tobias)-->()
- WHERE NOT (tobias)-->()
- WHERE has(b.name)
- WHERE b.name? = 'Bob'
  (Returns all nodes where name = 'Bob' plus all nodes without a name
  property)

Functions:

- On paths:
  - MATCH shortestPath((a)-[*]-(b))
  - MATCH allShorestPath((a)-[*]-(b))
  - Length(path) – The path length or 0 if not exists.
  - RETURN relationships(p) - Returns all relationships in a path.

- On collections:
  - RETURN a.array, filter(x IN a.array WHERE length(x)= 3)

FILTER - returns the elements in a collection that comply to a predicate.

  - WHERE ANY (x IN a.array WHERE x = "one“) – at least one
  - WHERE ALL (x IN nodes(p) WHERE x.age > 30) – all elements
  - WHERE SINGLE (x IN nodes(p) WHERE var.eyes = "blue") – Only one

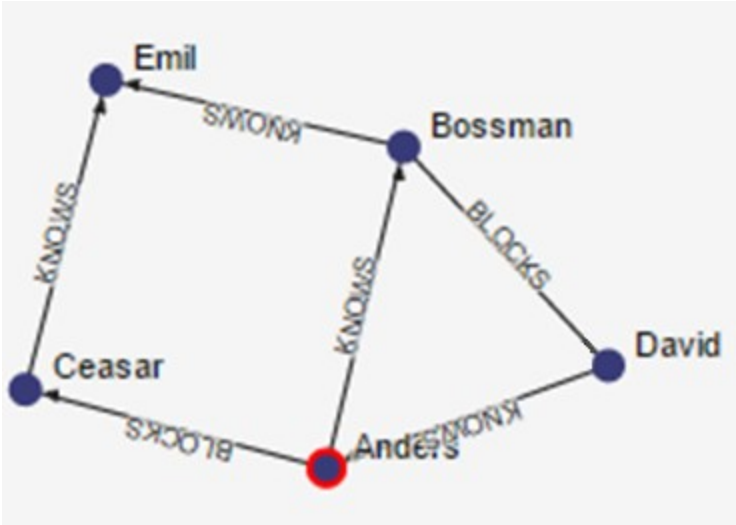
* nodes(p) – nodes of the path p

With

- Manipulate the result sequence before it is passed on to the following query parts.
- Usage of WITH :
  - Limit the number of entries that are then passed on to other MATCH clauses.
  - Introduce aggregates which can then be used in predicates in WHERE.
  - Separate reading from updating of the graph. Every part of a query must be either read-only or write-only.

With

```
MATCH (david { name: "David" })--(otherPerson)-->()
WITH otherPerson, count(*) AS foaf
WHERE foaf > 1
RETURN otherPerson
```



What will be returned?

The person connected to David with the at least one outgoing relationship.

(2 {name:"Anders"})

More collections options

- ```
  MATCH (user)
  RETURN count(user)
  ```
- ```
  MATCH (user)
  RETURN count(DISTINCT user.name)
  ```
- ```
  MATCH (user)
  RETURN collect(user.name)
  ```
  Collection from the values, ignores NULL.
- ```
  MATCH (user)
  RETURN avg(user.age)
  ```
  Average numerical values. Similar functions are sum, min, max.

Example

Student	Lecturer	Course
Name	Name	Name
ID	ID	Catalogue_Number
Address		Syllabus

Dersi alan öğrencilerin isimlerini listeleyin.

```
MATCH (c:Course)
WITH collect(c) AS courses
MATCH (s:Student)
WHERE ALL (x in courses WHERE (s)-[:Studies]->(x))
RETURN s.name
```

Example

Student	Lecturer	Course
Name	Name	Name
ID	ID	Catalogue_Number
Address		Syllabus

"Roy" la en az 2 en fazla 4 dersi ortak olan öğrencilerin isimleri

```
MATCH (s:Student)-[:Studies*2..4]-
>(:Student{Name:"Roy"})
RETURN DISTINCT s.name
```

Example

Student	Lecturer	Course
Name	Name	Name
ID	ID	Catalogue_Number
Address		Syllabus

İki farklı öğrenci arasındaki mesafe işlevini şu şekilde tanımlarız:
1. Ortak bir kurs alınmışsa, A ve B öğrencileri 1 uzaktadır.
2. A ve B öğrencileri n = 1'dir, eğer n en küçük sayı ise, Öğrenci C'nin var olması için A'nın n-1 mesafede olması ve C'nin B'nin 1 olması gerekir.
3. Böyle bir n mevcut değilse, mesafeyi 0 olarak tanımlayacağız.

Kimliği 12345 ve 67890 olan iki öğrenci arasındaki mesafe

```
MATCH p=shortestPath((s1:Student {ID:'12345'})-[:Studies*]-
(s2:Student {ID:'67890'}))
RETURN length(p)/2
```

Example

Student	Lecturer	Course
Name	Name	Name
ID	ID	Catalogue_Number
Address		Syllabus

En az 3 ders veren tüm öğretim elemanlarının isimleri

```
MATCH (l:Lecturer)-[:Teaches]->(c:Course)
WITH l, count(c) as numcourses
WHERE numcourses >= 3
RETURN l.name
```

EXAMPLES OF NETWORKS AND THEIR COMPONENTS

NETWORK	VERTICES	VERTEX ATTRIBUTES	EDGES	EDGE ATTRIBUTES
Airlines Network	**Airports**	Footfall, Terminals, Staff, City population, International/Domestic, Freight, Hangar capacity	**Airplanes / Routes**	Frequency, # Passengers, Plane Type, Fuel Usage, Distance covered, Empty seats
Banking Network	**Account Holders**	Name, demographics, KYC Document, Products, Account status, balance and other details	**Transactions**	Type, Amount, Authentication (pass/OTP), Time, Location, Device
Social Network	**Users**	Name, demographics, # connections, likes, circles belong to, subscriptions	**Interactions**	Medium (like/comment/direct message), time, duration, type of content, topic
Physician Network	**Doctors**	Demographics, speciality, experience, affiliation (type and size), Weekly patient intake	**Patients**	Demographics, Diagnosis history, visit frequency, purpose, referred to, insurance
Supply Chain Network	**Warehouses**	Location, size, capacity, storage type, connectivity, manual/automated	**Trucks**	Load capacity, # wheels, year of make, geographical permit, miles travelled. Maintenance cost, driver experience

Graphs in Data Analytics

Marketing Analytics – Graphs can be used to figure out the most influential people in a Social Network. Advertisers and Marketers can estimate the biggest bang for the marketing buck by routing their message through the most influential people in a Social Network

Banking Transactions – Graphs can be used to find unusual patterns helping in mitigating Fraudulent transactions. There have been examples where Terrorist activity has been detected by analyzing the flow of money across interconnected Banking networks

Supply Chain – Graphs help in identifying optimum routes for your delivery trucks and in identifying locations for warehouses and delivery centres

Pharma – Pharma companies can optimize the routes of the salesman using Graph theory. This helps in cutting costs and reducing the travel time for salesman

Telecom – Telecom companies typically use Graphs (Voronoi diagrams) to understand the quantity and location of Cell towers to ensure maximum coverage

Graph Theory concepts

connectivity - the minimum number of elements (nodes or edges) that need to be removed to disconnect the remaining nodes from each other. For network flow problems.
degree - number of edges incident to a vertex
eigenvector centrality - is a measure of the influence of a node in a network. Google's PageRank and the Katz centrality are variants of this
closeness centrality - a measure of centrality of a node to a network, calculated as the sum of the length of the shortest paths between the node and all other nodes in the graph. Thus the more central a node is, the closer it is to all other nodes.
Link analysis - In network theory, used to evaluate relationships (connections) between graph nodes.
eccentricity of a vertex - maximum distance from a vertex to all other vertices
radius of a connected graph - the minimum value of eccentricity from all vertices
diameter of a connected graph - Unlike the radius of the connected graph here we basically used the maximum value of eccentricity from all vertices

Uygulamalar

py2neo-example.ipynb

Py2neo is a client library and toolkit for working with Neo4j from within Python applications and from the command line.

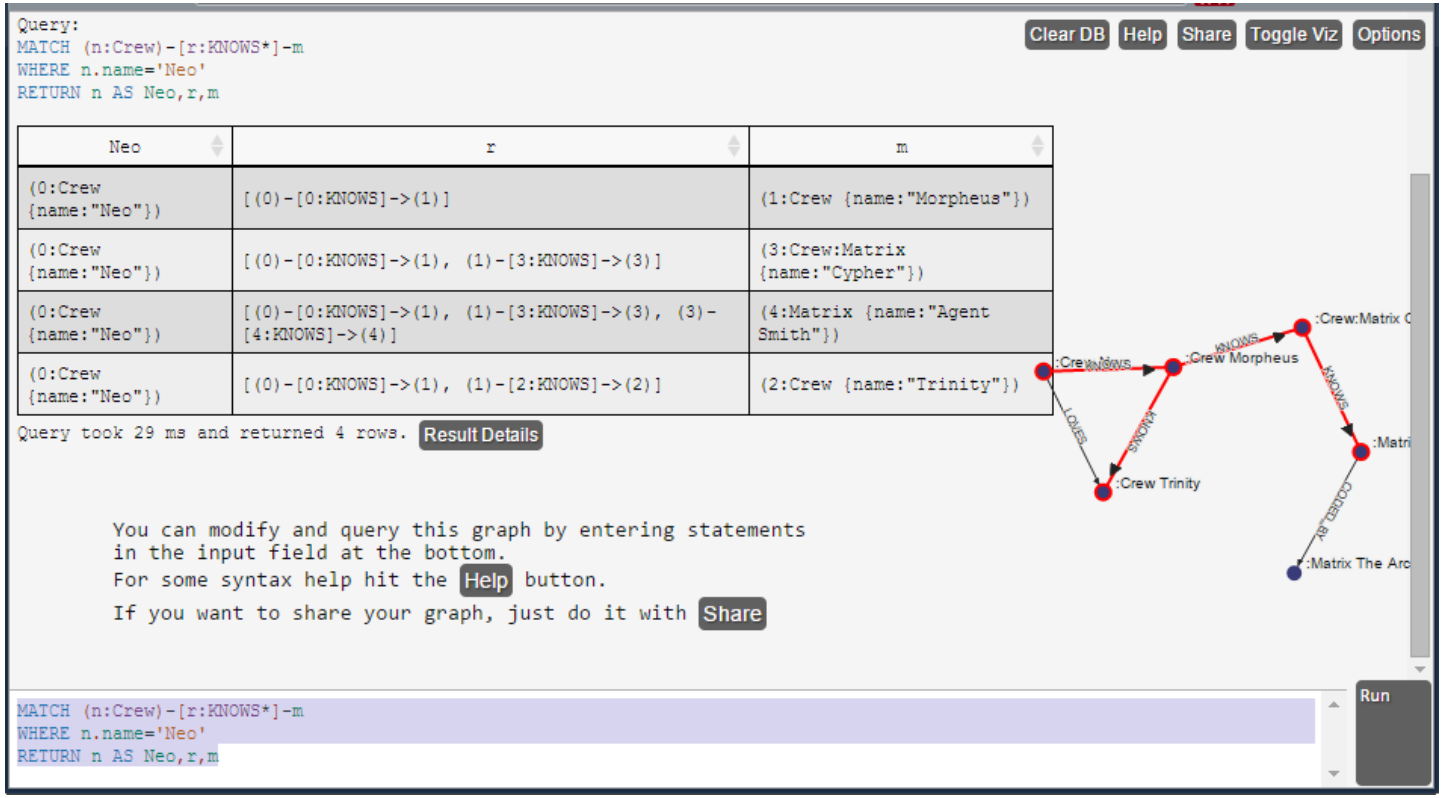
networkx-example.ipynb

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

Learn more...

Check Neo4j online version:

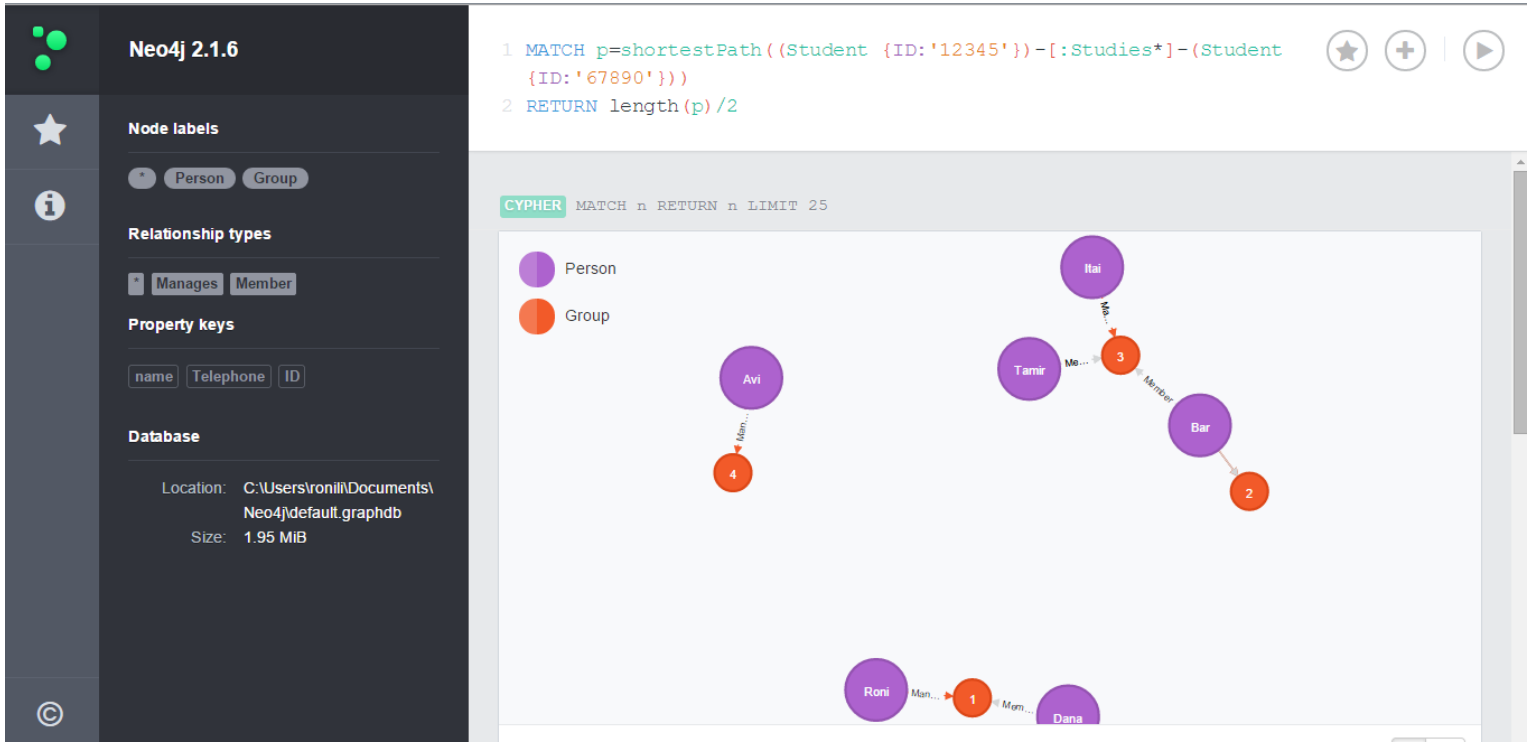
http://console.neo4j.org/



Learn more...

Download Neo4j for free:

http://neo4j.com/download/



Learn more...

Read the Neo4j manual:

http://neo4j.com/docs/stable/

Cypher tutorials:

http://neo4j.com/developer/cypher-query-language/

More Neo4j developers tutorials:

http://neo4j.com/developer/get-started/