
# Graph Databases

Γιάννης Δήμου

# Περίληψη

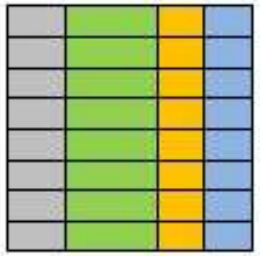
- Λίγο Background: Είδη βάσεων δεδομένων
- Πως προέκυψε η ανάγκη για graph DBs?
- Πολυπλοκότητα και μέγεθος
- Τυπικά πεδία εφαρμογής
- Πότε έχει νόημα να χρησιμοποιήσει κανείς graph DBs
- Πως λειτουργούν
- Graph query languages
- Οι πιο διαδεδομένες υλοποιήσεις graph DBs
- Παράδειγμα: neo4j
  - Χαρακτηριστικά
  - Η cypher query language
  - Αρχιτεκτονική
  - Απόδοση
  - APIs και Connectors
- Παρεμπιπτόντως: Azure Cosmos DB
- Συμπεράσματα

# Λίγο background: Είδη βάσεων δεδομένων

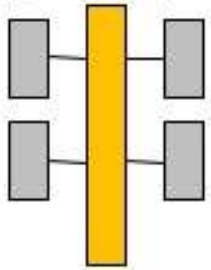
To dump large volumes from RDBMSs
and run Business Intelligence workloads

Model-based,
transaction oriented,
normalized

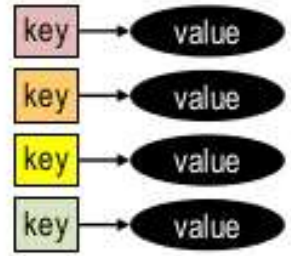
**Relational**



**Analytical (OLAP)**

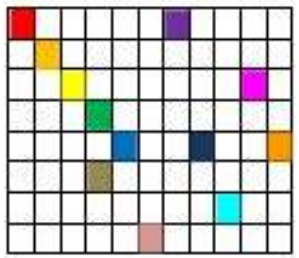


**Key-Value**



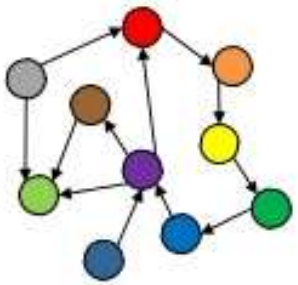

A dictionary with pointers
to potentially dissimilar
data objects

**Column-Family**



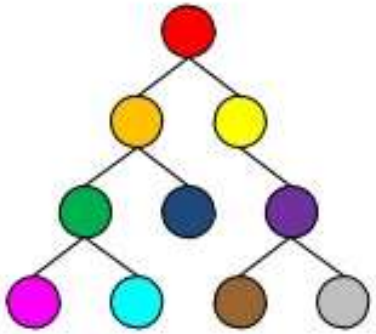
Sparse matrices
made of rows with varying column sets.
Good for write heavy logging, IoT etc.
petabyte-scale workloads.

**Graph**



Model-less document DBs
to intuitively and efficiently
query associative datasets

**Document**



A key-value DB where
each object is a parseable
XML or JSON file

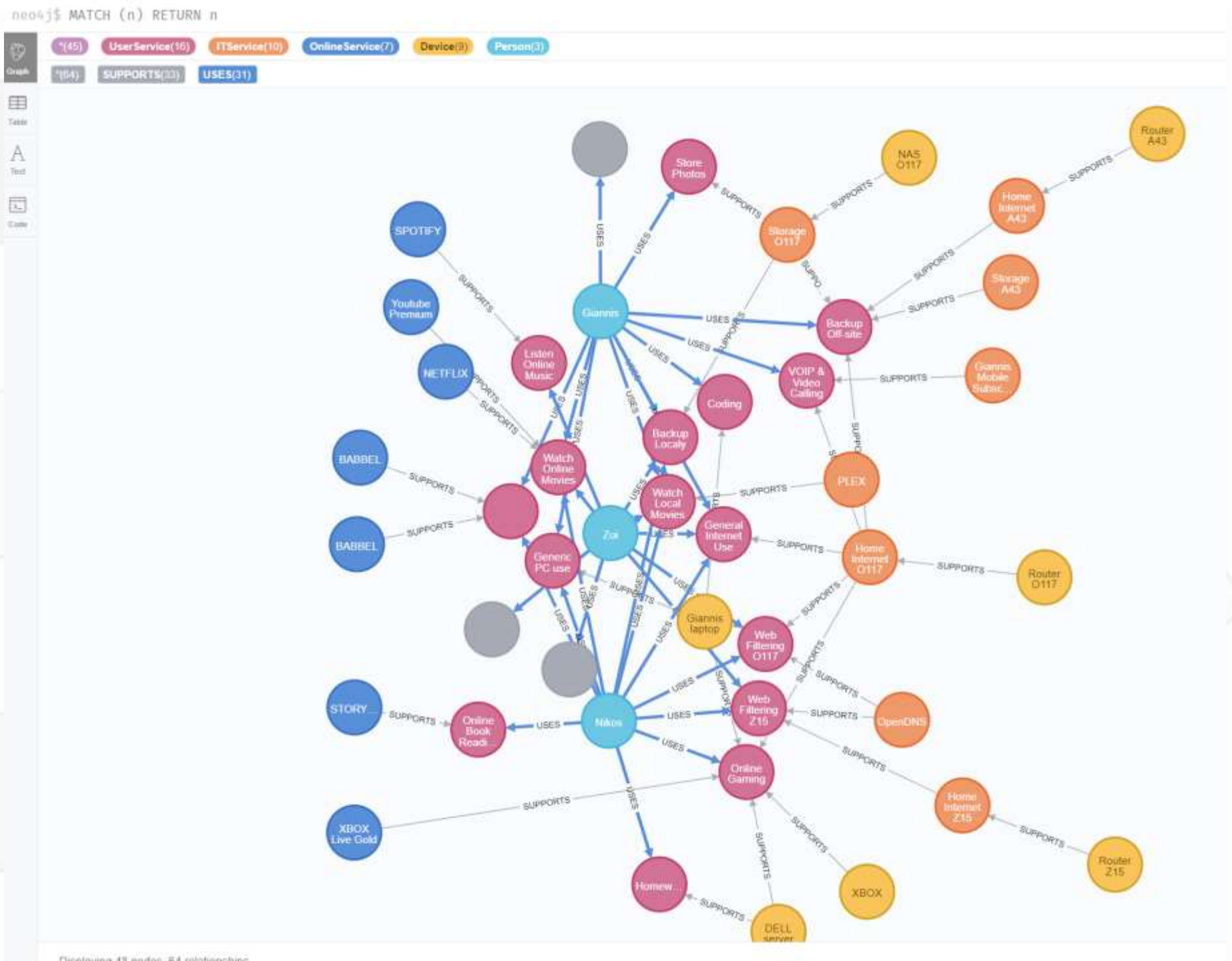
# Πως προέκυψε η ανάγκη για graph DBs;
# Ποιό ειναι τελικά το πρόβλημα που παν να λύσουν;

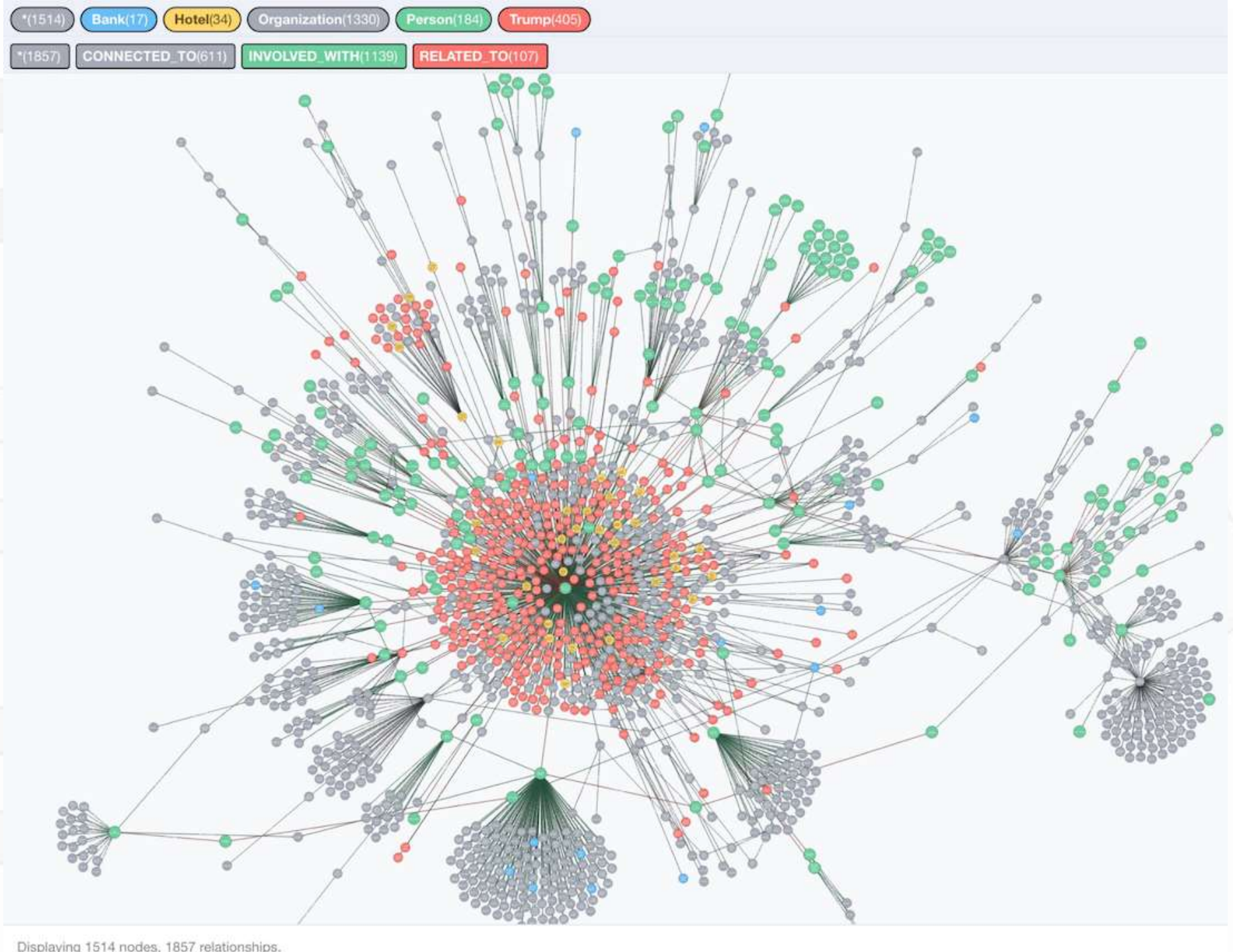
Το πρόβλημα αφορά κυρίως **ΠΟΛΥΠΛΟΚΟΤΗΤΑ**

…και δευτερευόντως **ΜΕΓΕΘΟΣ**

Δυσκολία στη μοντελοποίηση κάποιων προβλημάτων με σχεσιακές βασεις (περίπλοκες σχέσεις, δυσνόητα SQL schemas)

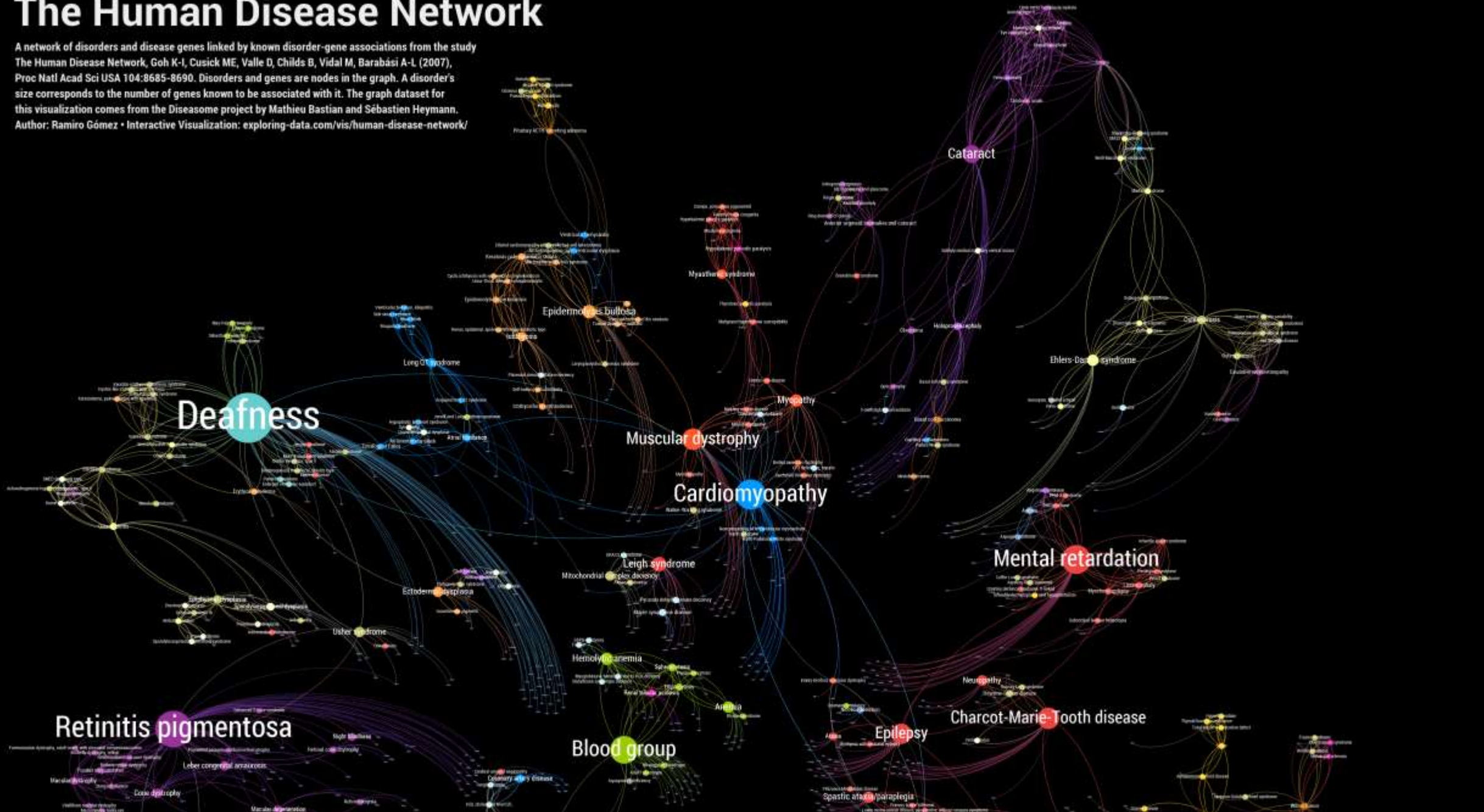
Πολύ χρονοβόρες αναζητήσεις/queries, (βαθιές αναδρομικές αναζητήσεις μπορούν να υπάρχουν ακόμη και σε απλά SQL schemas)

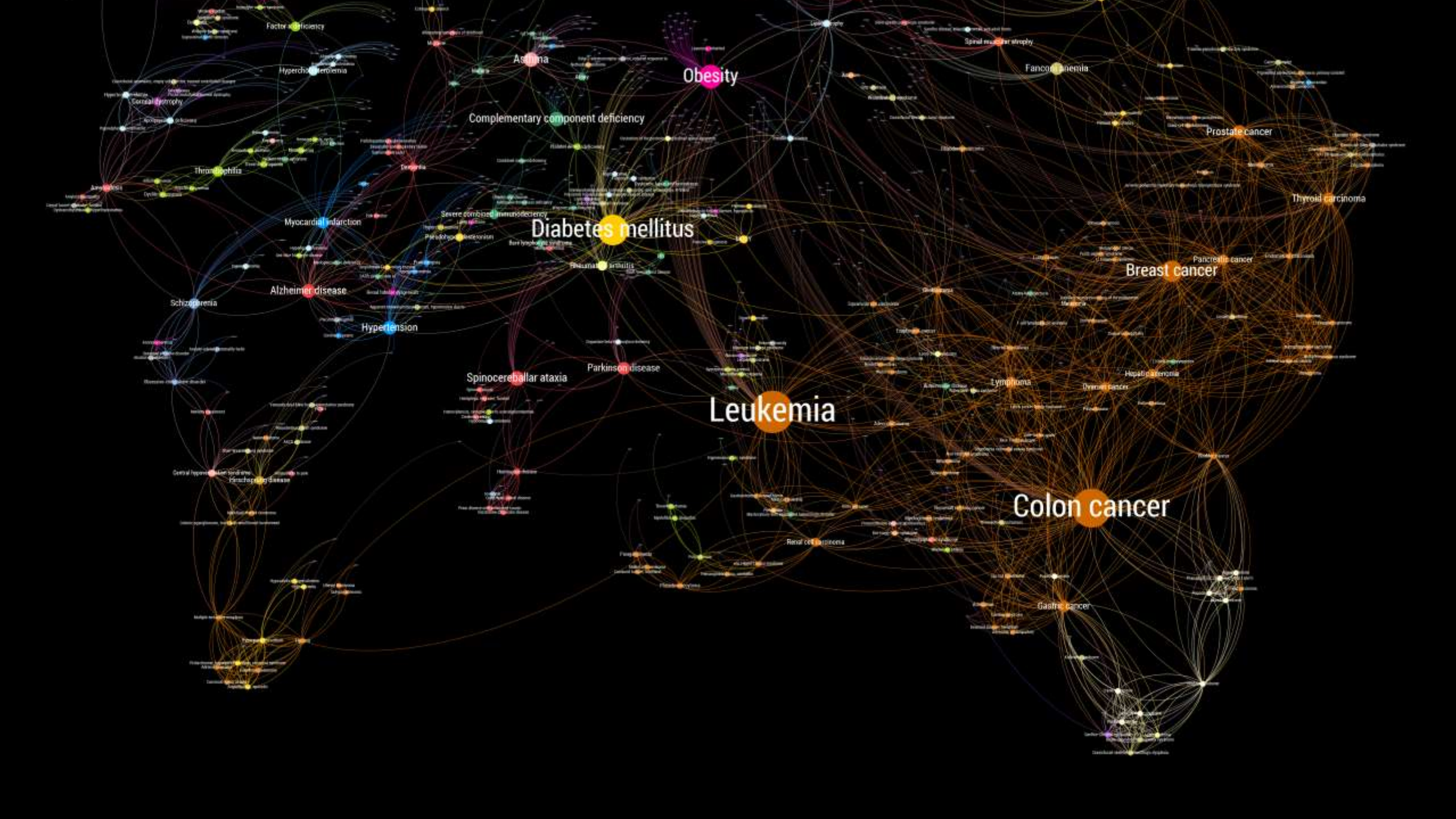

neo4j$ MATCH (n) RETURN n
*(45)
UserService(16)
ITService(10)
OnlineService(7)
Device(9)
Person(3)
*(84)
SUPPORTS(33)
USES(31)
Graph
Table
Text
Code
Giannis
Zoe
Nikos
Store Photos
NAS O117
Router A43
Home Internet A43
Storage O117
Storage A43
Backup Off-site
SPOTIFY
Youtube Premium
NETFLIX
Listen Online Music
Watch Online Movies
Backup Locally
Coding
VOIP & Video Calling
Giannis Mobile Subsc...
PLEX
Watch Local Movies
General Internet Use
Home Internet O117
Router O117
BABBEL
BABBEL
Generic PC use
Giannis laptop
Web Filtering O117
Web Filtering Z15
OpenDNS
STORY
Online Book Reads
Online Gaming
Home Internet Z15
Router Z15
XBOX Live Gold
XBOX
Homew...
DELL
SUPPORTS
USES


*(1514)
Bank(17)
Hotel(34)
Organization(1330)
Person(184)
Trump(405)
*(1857)
CONNECTED_TO(611)
INVOLVED_WITH(1139)
RELATED_TO(107)
Displaying 1514 nodes, 1857 relationships.

# The Human Disease Network

A network of disorders and disease genes linked by known disorder-gene associations from the study The Human Disease Network, Goh K-I, Cusick ME, Valle D, Childs B, Vidal M, Barabási A-L (2007), Proc Natl Acad Sci USA 104:8685-8690. Disorders and genes are nodes in the graph. A disorder's size corresponds to the number of genes known to be associated with it. The graph dataset for this visualization comes from the Diseasome project by Mathieu Bastian and Sébastien Heymann.
Author: Ramiro Gómez • Interactive Visualization: exploring-data.com/vis/human-disease-network/

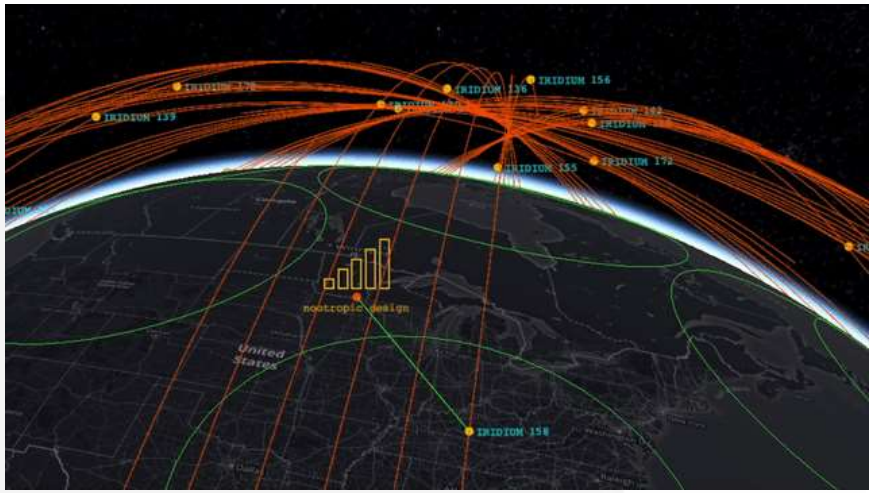



Factor X deficiency
Asthma
Obesity
Hypercholesterolemia
Corneal dystrophy
Complementary component deficiency
Spinal muscular atrophy
Fanconi anemia
Prostate cancer
Thrombophilia
Dementia
Thyroid carcinoma
Myocardial infarction
Severe combined immunodeficiency
Diabetes mellitus
Pancreatic cancer
Breast cancer
Rheumatoid arthritis
Alzheimer disease
Schizophrenia
Hypertension
Spinocerebellar ataxia
Parkinson disease
Lymphoma
Ovarian cancer
Hepatic adenoma
Leukemia
Colon cancer
Renal cell carcinoma
Gastric cancer

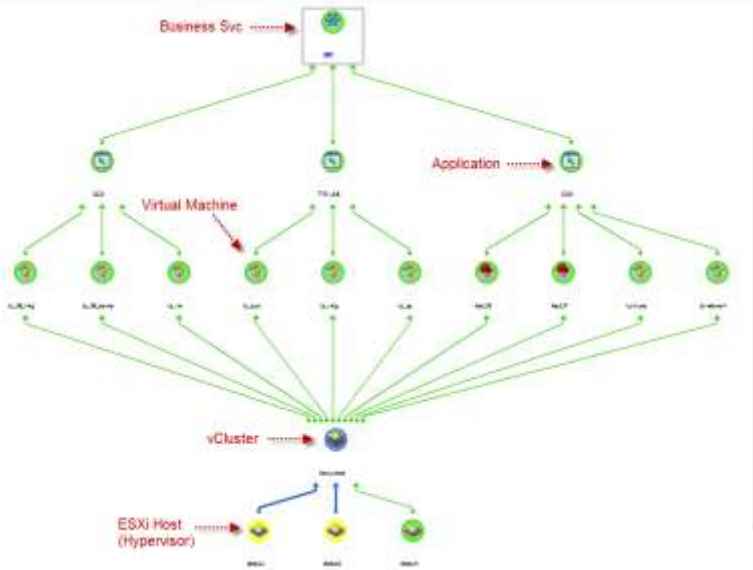
# Τυπικά πεδία εφαρμογής

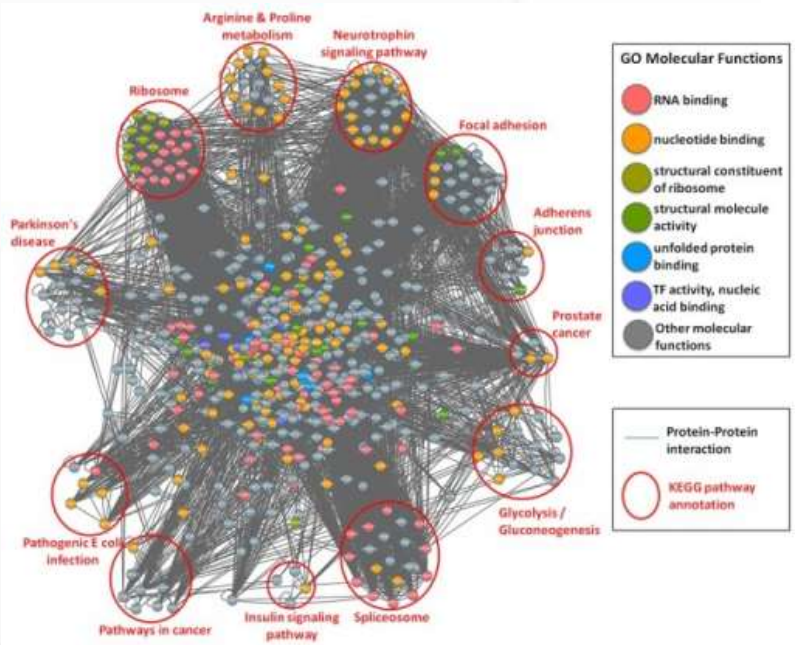
Ύπάρχουν προβλήματα που από τη φύση τους βασίζονται σε γράφους (**graph related problems**), όπως:

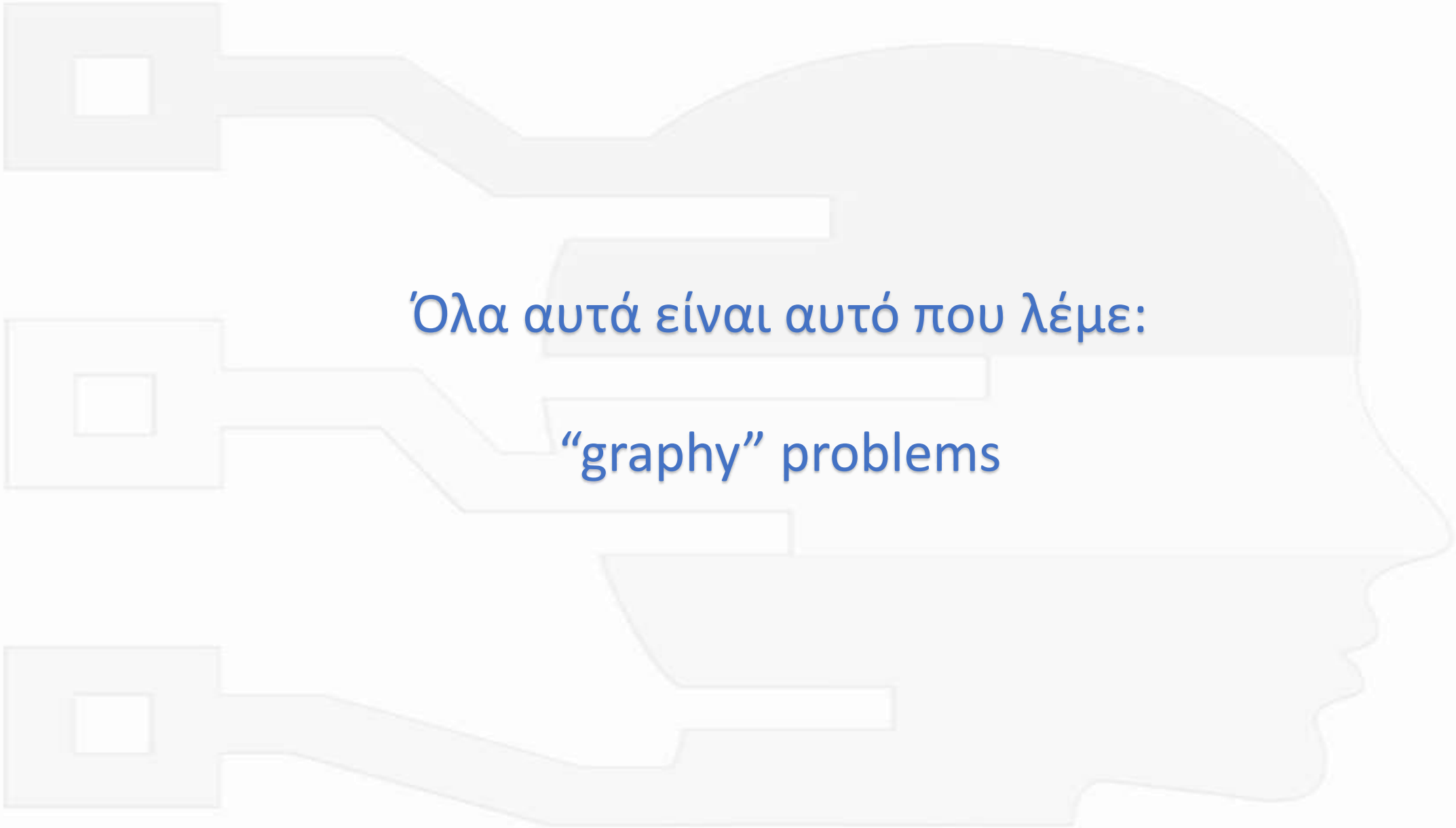
- Δρομολόγηση κλήσεων
- Protein regulatory networks έρευνα φαρμάκων
- Network outage impact analysis
- Master data management
- Ανάλυση social networks
- Fraud detection
- Προτεινόμενα προϊόντα σε online αγορές









Όλα αυτά είναι αυτό που λέμε:

“graphy” problems

# Πότε έχει νόημα να χρησιμοποιήσει κανείς graph DBs

*“Αν το μόνο που έχεις ειν’ ένα σφυρί,*
*τα πάντα σου φαίνονται σαν καρφιά.”*

*Αν το μόνο που ξέρεις είν’ SQL,*
*όλα τα datasets σου μοιάζουν relational.*

Αλλά δεν είναι.

Ο πραγματικός κόσμος δεν ταιριάζει απαραίτητα με το μοντέλο που χουμε στο νου μας.
Τα μοντέλα μας πρέπει να ταιριάζουν στον πραγματικό κόσμο.

Άρα –**στις περιπτώσεις που ταιριάζει-** μπορούμε να εκμεταλλευτούμε κάποια συγκεκριμένα πλεονεκτήματα των graph DBs για να:

- Αποθηκεύουμε και να κάνουμε πολύ γρήγορες και περιπλοκες αναζητήσεις σε highly associative data
- Μοντελοποιούμε με πιο απλό κ διαισθητικό τρόπο graphy προβλήματα και το μοντέλο μας να επιδέχεται live αλλαγές
- Διαχειριζόμαστε μεγάλα graphy datasets χωρίς να τρώμε μεγάλο (εκθετικό) αντίκτυπο στην απόδοση του συστήματος

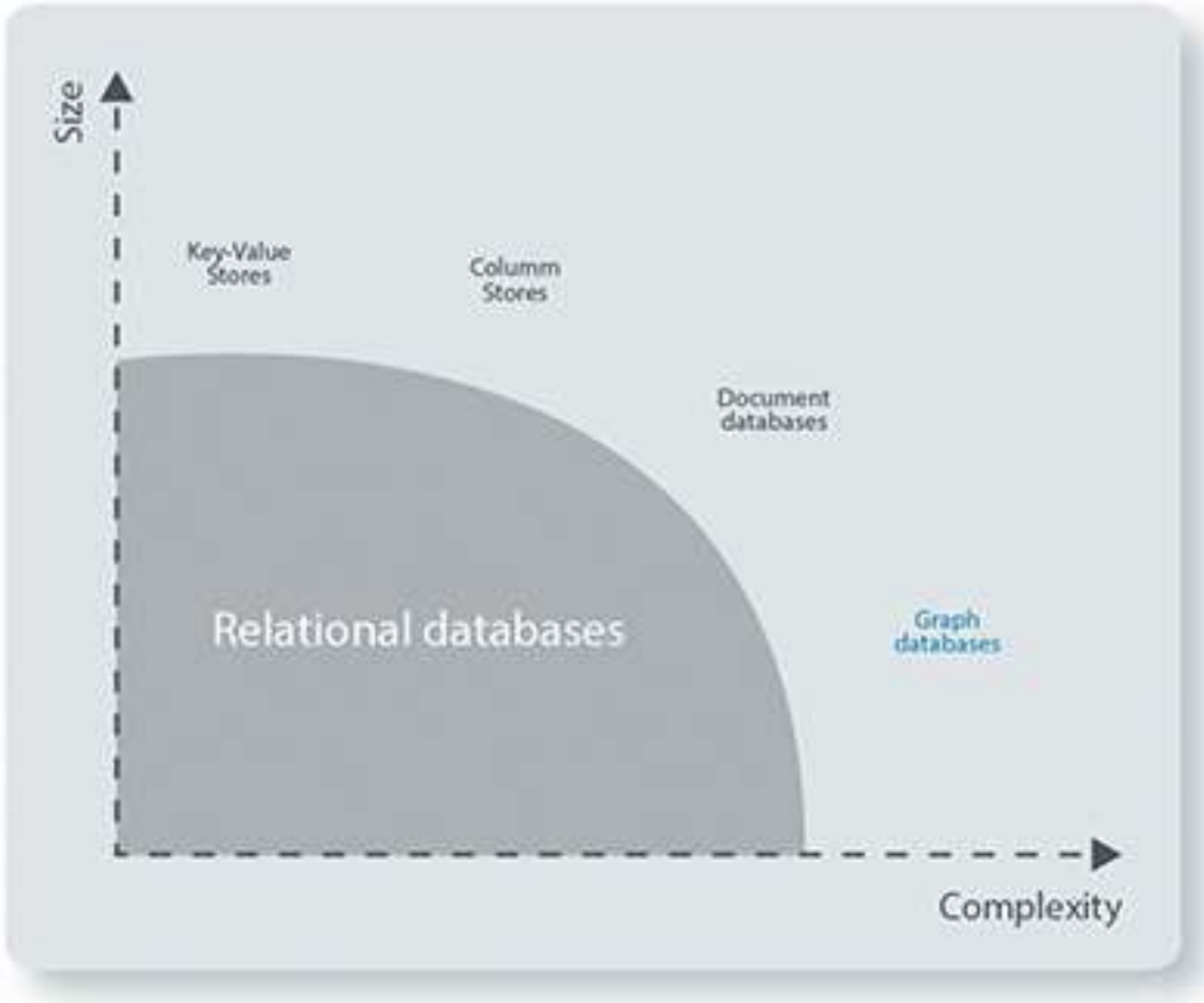
# Πότε έχει νόημα να χρησιμοποιήσει κανείς graph DBs

Έχει νόημα για:

Περίπλοκα μοντέλα με μεγάλη πυκνότητα σχέσεων μεταξύ των οντοτήτων/entities

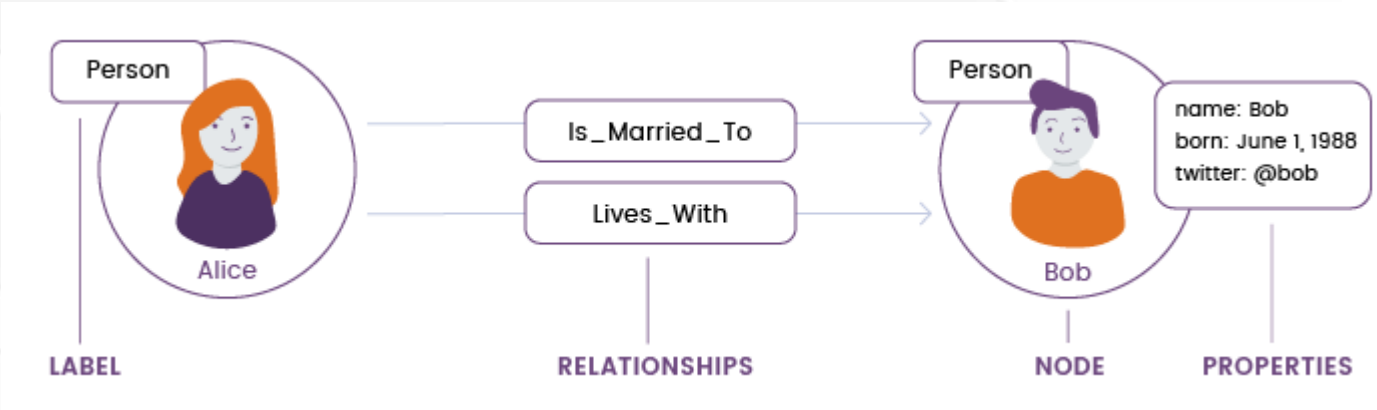
ή

Όταν ξέρουμε από πριν ότι η εφαρμόγή μας θα τρέχει βαριά queries πάνω στις σχέσεις των οντοτήτων της βάσης μας.



# Πως λειτουργούν οι graph DBs;

Μοντελοποιούν το πρόβλημα (data domain) σαν ένα συνολο απο κόμβους (nodes) και σχέσεις (relationships) με labels και properties.




| RDBMS | Graph Database |
|---|---|
| Tables | Graphs |
| Rows | Nodes |
| Columns and Data | Properties and their values |
| Constraints | Relationships |
| Joins | Traversal |

# Οπότε ειν’ απλά πολύ βελτιστοποιημένες in-memory databases?

Όχι.

Είναι ένας συνδυασμός απο:

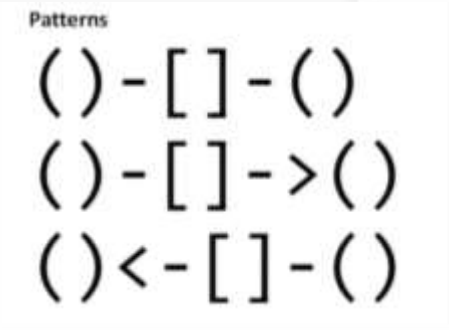
Ένα διαφορετικό τρόπο μοντελοποίησης προβλημάτων

+

Γρήγορες, scalable graph engines (=γρήγορο graph traversal)

+

Query languages που μας επιτρέπουν να αλληλεπιδρούμε με διαισθητικό τρόπο με μοντέλα γράφων



# Graph query languages

Don’t get too excited. No standard yet…. ☹

Σε τυχαία σειρά:

- Gremlin (Apache TinkerPop specific)
- PGQL (for Oracle Spatial and Graph)
- SPARQL (generic for document DBs)
- Cypher (neo4j specific)
- SQL/PGQ Property Graph Query (SQL standard proposed extension to accept graph patterns in SELECT statements)
- GQL (ISO working group to formally extend the SQL standard, work in progress)

# Οι πιο γνωστές υλοποιήσεις graph DBs

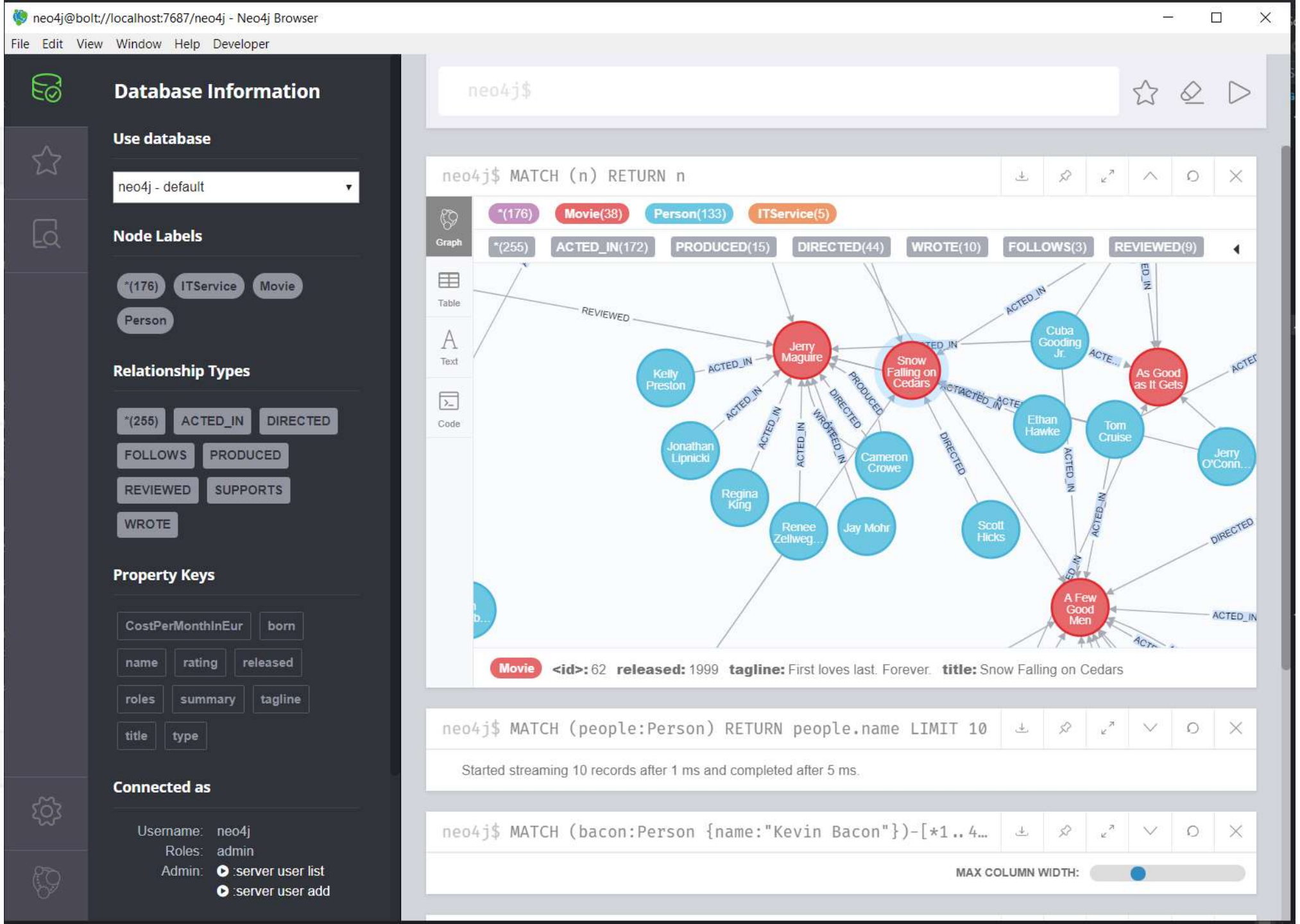
| Rank | | | DBMS | Database Model | Score | | |
|---|---|---|---|---|---|---|---|
| Nov 2020 | Oct 2020 | Nov 2019 | | | Nov 2020 | Oct 2020 | Nov 2019 |
| 1. | 1. | 1. | Neo4j | Graph | 53.53 | +2.20 | +3.00 |
| 2. | 2. | 2. | Microsoft Azure Cosmos DB | Multi-model | 32.50 | +0.49 | +0.52 |
| 3. | 3. | ↑ 4. | ArangoDB | Multi-model | 5.37 | -0.18 | +0.36 |
| 4. | 4. | ↓ 3. | OrientDB | Multi-model | 5.30 | -0.17 | -0.09 |
| 5. | 5. | 5. | Virtuoso | Multi-model | 2.54 | -0.03 | -0.10 |
| 6. | 6. | ↑ 7. | Amazon Neptune | Multi-model | 2.43 | -0.05 | +0.83 |
| 7. | 7. | ↓ 6. | JanusGraph | Graph | 2.37 | -0.03 | +0.58 |
| 8. | 8. | 8. | GraphDB | Multi-model | 2.11 | +0.01 | +0.97 |
| 9. | 9. | ↑ 14. | FaunaDB | Multi-model | 1.78 | 0.00 | +1.17 |
| 10. | 10. | ↓ 9. | Dgraph | Graph | 1.62 | -0.06 | +0.58 |

Πηγή: db-engines.com, November 2020
Αθροιστικά δεδομένα δημοτικότητας από StackExchange, LinkedIn, GoogleTrends

# Παράδειγμα:


- Κατασκευασμένη εξ αρχής ως graph database
- Γραμμένη σε java
- Αποθηκέυει τα δεδομένα σαν ένα σετ απο nodes και relationships
- Σχεδιασμένη γύρω απο μια συγκεκριμένη βελτιστοποίηση:
  Κάθε κόμβος/node, αποθηκεύει pointers προς όλους τους κόμβους με τους οποίους συνδέεται. Αυτό επιτρέπει πολύ γρήγορες αναζητήσεις σε σχέσεις πολλαπλών βημάτων, τάξεις μεγέθους πιο γρήγορα απ’ ότι με κλασσικές βάσεις SQL/RDBMSs.
- Διαχειρίζεσαι τη βάση με Cypher query language (επινοημένη ειδικά για τη neo4j)

# Neo4j Desktop



# Η Cypher query language

Neo4j's query language

Λειτουργεί σαν a data definition language (DDL) για δημιουργία μοντέλων με graph data
Και για την σύνταξη queries για αναζήτηση με συγκεκριμένα patterns σχέσεων.

Π.χ. Πως δημιουργούμε έναν κόμβο/node με τη Cypher.

```
() //anonymous node (no label or variable) can refer to any node in the database
(p:Person) //using variable p and label Person
(:Technology) //no variable, label Technology
(work:Company) //using variable work and label Company
```

Πως δημιουργούμε μια σχέση με τη Cypher.

```
//data stored with this direction
CREATE (p:Person)-[:LIKES]->(t:Technology)

//query relationship backwards will not return results
MATCH (p:Person)<-[:LIKES]-(t:Technology)

//better to query with undirected relationship unless sure of direction MATCH
(p:Person)-[:LIKES]-(t:Technology)
```

# Η Cypher query language παράδειγμα: Movie Graph

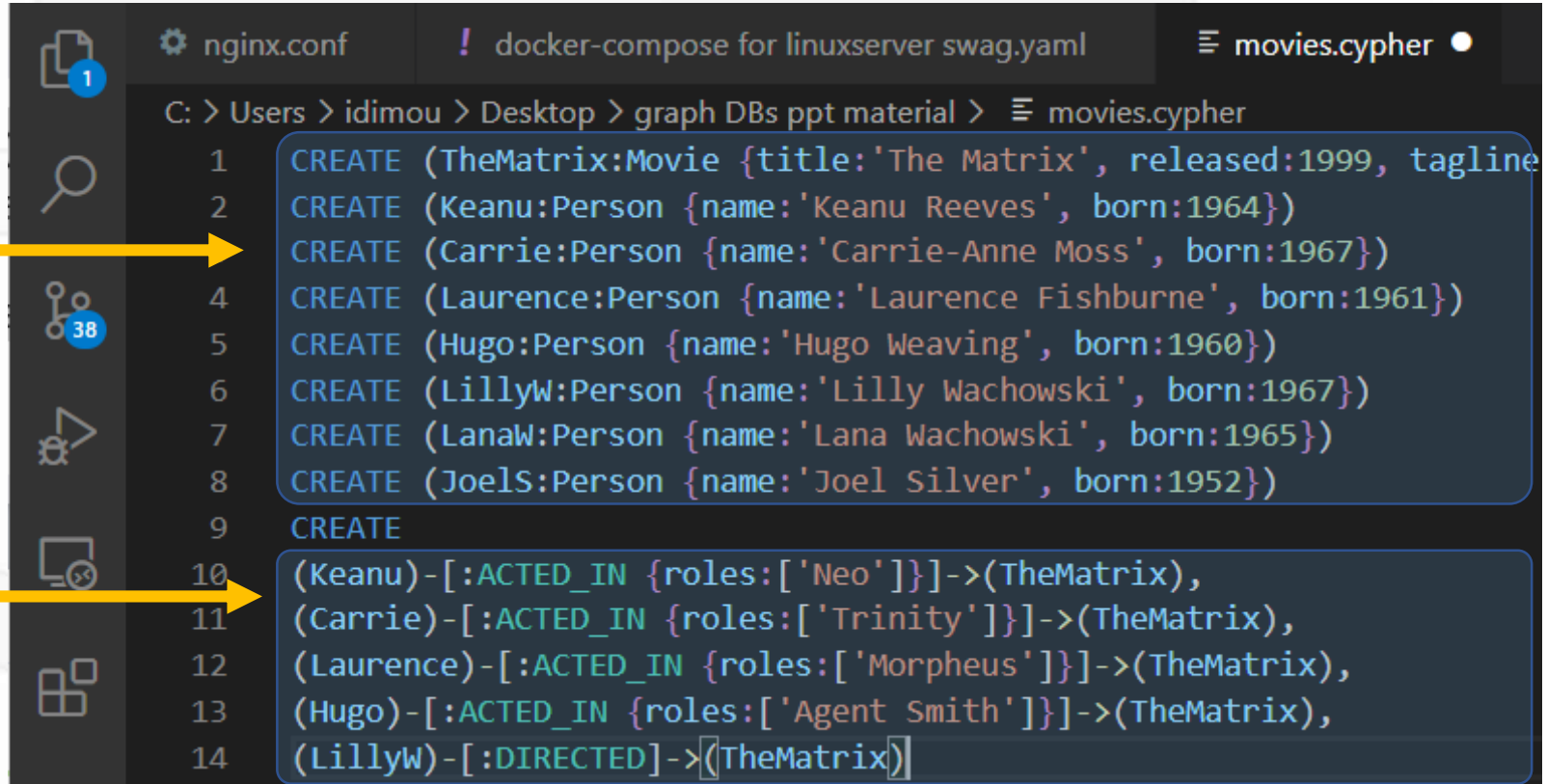
Έστω οτι θέλουμε να στήσουμε μια μικρή εφαρμογή που να περιέχει ηθοποιούς και σκηνοθέτες και τις ταινίες στις οποίες έχουν συνεργαστεί. Για να το κάνουμε αυτό θα πρέπει:

1. Να δημιουργήσουμε (create) τις έγγραφές για ηθοποιούς, σκηνοθέτες και ταινίες (3 διαφορετικά ειδη κομβων) στη graph DB
2. Να μπορούμε να κάνουμε απλές αναζητήσεις π.χ. για μεμονωμένους ηθοποιούς ή ταινίες
3. Να κανουμε πιο περίπλοκες αναζητήσεις π.χ. για ηθοποιούς που χουν συνεργαστει σε ταινίες
4. Να βρίσκουμε ελάχιστες αποστάσεις (Bacon Path)

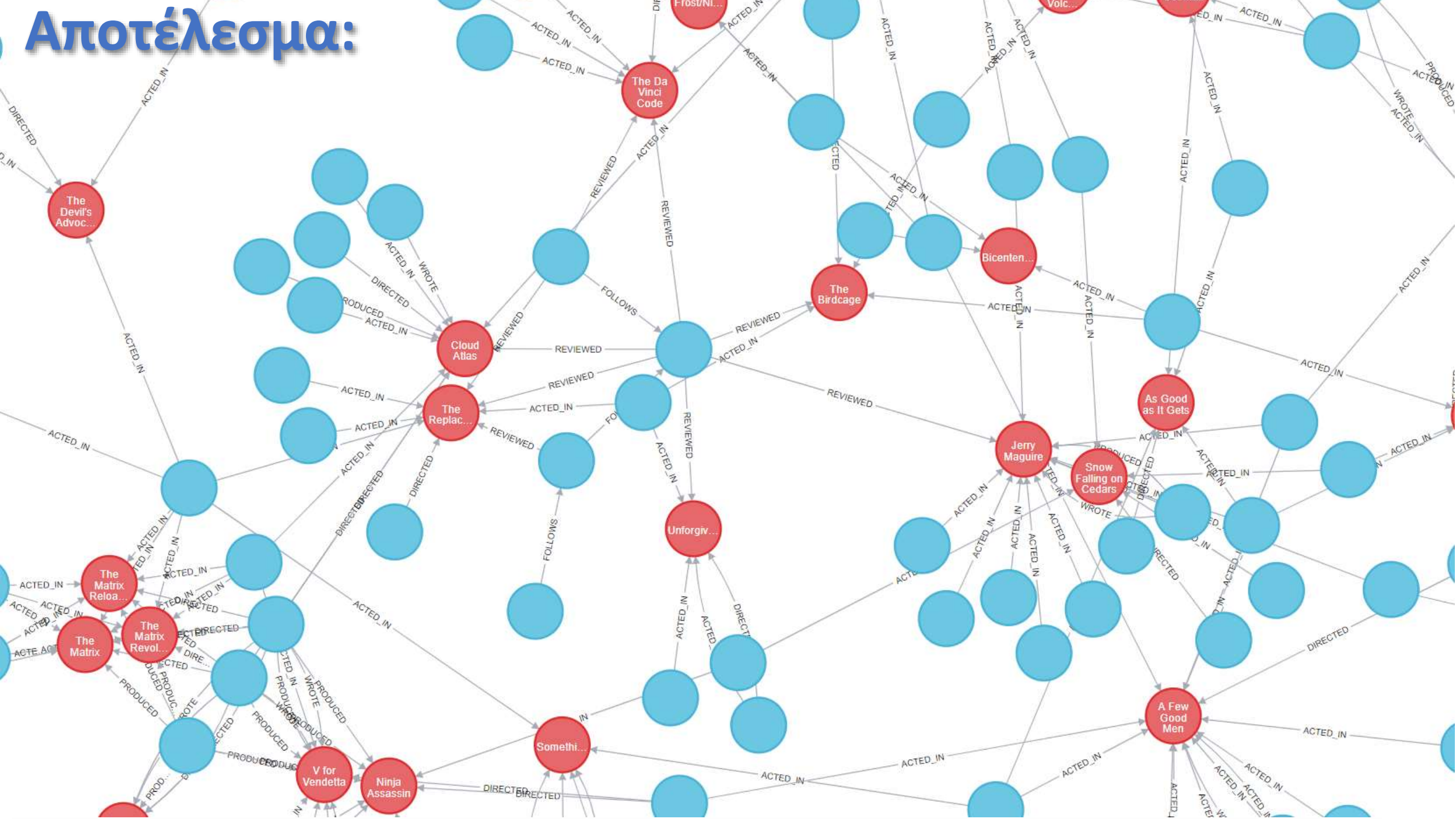
## 1. Create:

nodes

relationships



# Αποτέλεσμα:

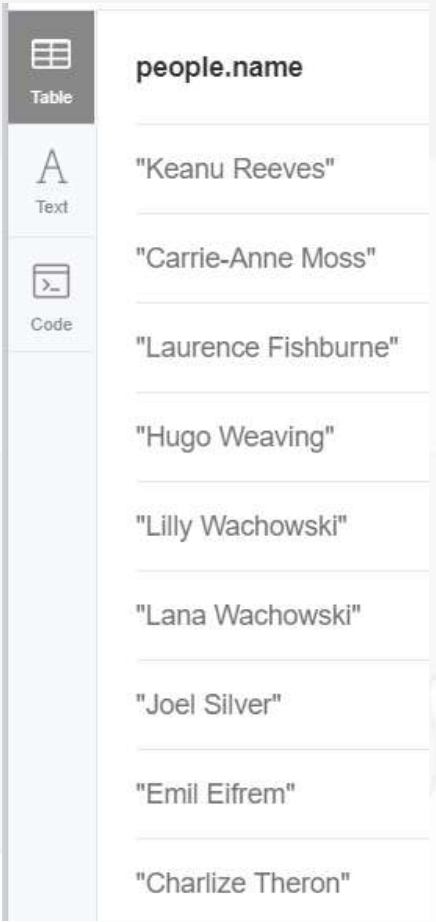

The Da Vinci Code
The Devil's Advoc...
Cloud Atlas
The Replac...
The Birdcage
Bicenten...
Jerry Maguire
As Good as It Gets
Snow Falling on Cedars
Unforgiv...
The Matrix Reloa...
The Matrix Revol...
The Matrix
V for Vendetta
Ninja Assassin
Somethi...
A Few Good Men
ACTED_IN
DIRECTED
REVIEWED
WROTE
PRODUCED
FOLLOWS

# Η Cypher query language παράδειγμα: Movie Graph

## 2. Απλή αναζήτηση

10 people in the DB

```
MATCH (people:Person) RETURN people.name LIMIT 10
```



# Η Cypher query language παράδειγμα: Movie Graph

## 3. Σύνθετη αναζήτηση

in any same movie node “(m)”

Tom Hanks' co-actors...

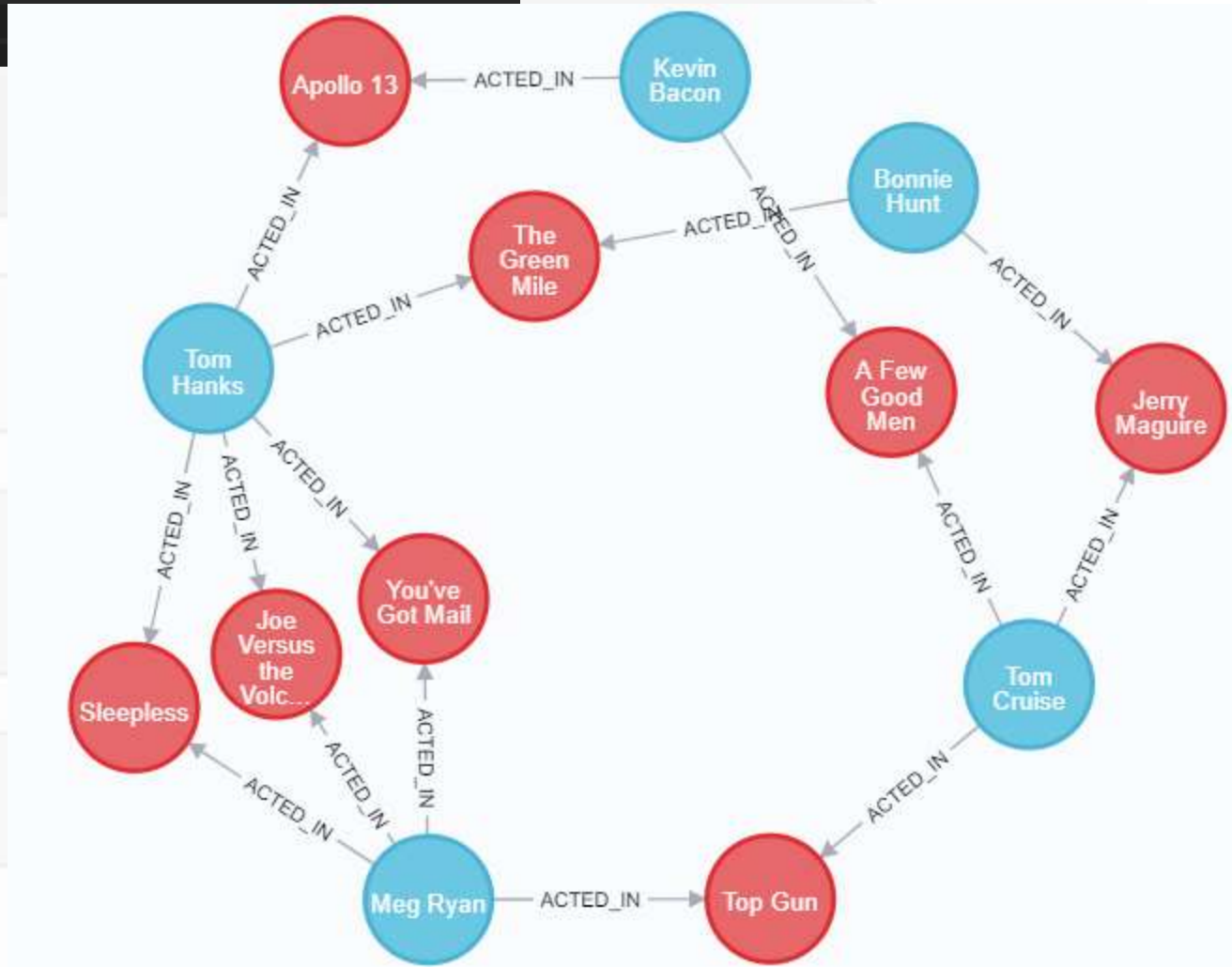
```
MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]->(m)<-[:ACTED_IN]-(coActors) RETURN coActors.name
```

| coActors.name |
| --- |
| "Rosie O'Donnell" |
| "Lori Petty" |
| "Madonna" |
| "Geena Davis" |
| "Bill Paxton" |
| "Hugo Weaving" |
| "Jim Broadbent" |
| "Halle Berry" |
| "Paul Bettany" |

# Η Cypher query language παράδειγμα: Movie Graph

Βρες κάποιον για να συστήσει τον Tom Hanks στον Tom Cruise

```
MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]->(m)<-[:ACTED_IN]-(coActors),
  (coActors)-[:ACTED_IN]->(m2)<-[:ACTED_IN]-(cruise:Person {name:"Tom Cruise"})
RETURN tom, m, coActors, m2, cruise
```

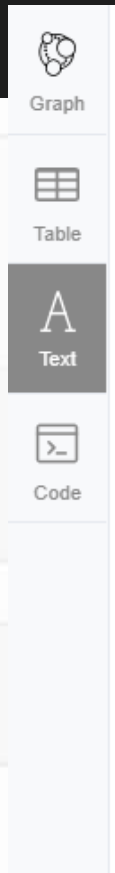


# Η Cypher query language παράδειγμα: Movie Graph

## 4. Επίλυση Bacon path

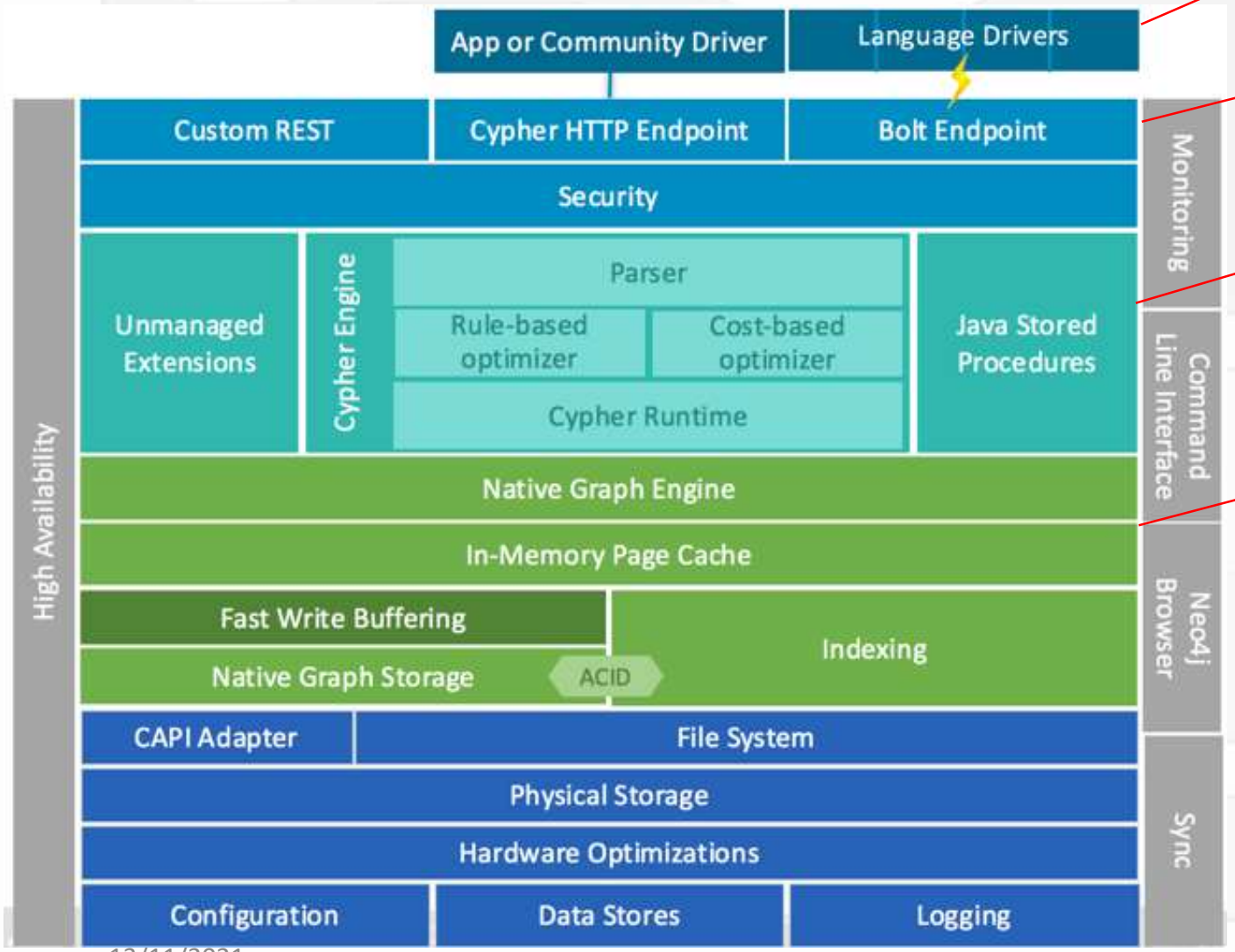
Shortest path query: Βρες τις ταινίες και τους ηθοποιούς που απέχουν το πολύ 4 σχέσεις απ' τον Kevin Bacon

```
MATCH (bacon:Person {name:"Kevin Bacon"})-[*1..4]-(hollywood)
RETURN DISTINCT hollywood
```




| "hollywood" |
| --- |
| {"name":"Frank Darabont","born":1959} |
| {"name":"Tom Hanks","born":1956} |
| {"name":"Bonnie Hunt","born":1961} |
| {"name":"James Cromwell","born":1940} |
| {"name":"Patricia Clarkson","born":1959} |
| {"name":"Gary Sinise","born":1955} |
| {"name":"David Morse","born":1953} |
| {"name":"Michael Clarke Duncan","born":1957} |
| {"title":"The Green Mile","tagline":"Walk a mile |

# Αρχιτεκτονική neo4j




Connectors built on top of the built-in APIs

Built-in APIs:
REST, Cypher queries via HTTP, Bolt

Cypher query language layer

Fast in-memory graph engine

# Απόδοση

- Benchmark dataset: 1.000.000 έγγραφές/κόμβοι με 50 σχέσεις ο καθένας
- Σκοπός: Βρες τις εγγραφές που σχετίζονται σε βάθος [depth] σχέσεις απο μια συγκεκριμένη εγγραφή.

| 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 |

Εντυπωσιακό scaling

# APIs and Connectors

Built-in APIs

- HTTP Cypher frontend
- REST
- Bolt

Connectors

- Java
- .NET
- C/C++
- JavaScript
- Python
- Go
- Ruby
- PHP
- R
- Erlang/Elixir
- Haskell
- Clojure
- Perl

# Στην πράξη: Πως μιλάω μ’ αυτό το πράγμα?

Ας υποθέσουμε οτι υλοποιούμε μια asp.net web app για να σερβίρουμε δεδομένα απο μια βάση neo4j μέσω ενός REST api:

1. Προσθέτουμε το neo4jclient nuget package στο project μας
2. Αρχικοποιούμε ένα GraphClient object και το συνδέουμε Connect() στη βάση
3. Το GraphClient παρέχει ένα GraphClient.Cypher fluent query interface που υποστηρίζει Cypher queries με τη μορφή C# lamda expressions όπως ακριβώς θα κάναμε με .net LINQ to SQL (ή streams στη java).
4. Το query επιστρέφει ένα IEnumerable απο json objects τα οποία μπορούμε να κάνουμε deserialize σε μια λιστα χρησιμοποιώντας Newtonsoft.Json
5. Τις εγγραφές της λιστας μπορούμε να τις σερβίρουμε στη συνέχει μέσω ενος Web API controller.

```
Neo4jDotNetDemo                    Neo4jDotNetDemo.WebApiConfig          Register(HttpConfig
```

```
1  using Neo4jClient;
2  using Newtonsoft.Json.Serialization;
3  using System;
4  using System.Collections.Generic;
5  using System.Configuration;
6  using System.Linq;
7  using System.Web.Http;
8
9  namespace Neo4jDotNetDemo
10 {
11     public static class WebApiConfig
12     {
13         public static void Register(HttpConfiguration config)
14         {
15             // Web API configuration and services
16
17             // Web API routes
18             config.MapHttpAttributeRoutes();
19
20             config.Formatters.JsonFormatter.SerializerSettings.ContractResolver = new CamelCasePropertyNamesContractResolver();
21             config.Formatters.JsonFormatter.SerializerSettings.NullValueHandling = Newtonsoft.Json.NullValueHandling.Ignore;
22
23             var appXmlType = config.Formatters.XmlFormatter.SupportedMediaTypes.FirstOrDefault(t => t.MediaType == "application/xml");
24             config.Formatters.XmlFormatter.SupportedMediaTypes.Remove(appXmlType);
25
26             //Use an IoC container and register as a Singleton
27             var url = ConfigurationManager.AppSettings["GraphDBUrl"];
28             var user = ConfigurationManager.AppSettings["GraphDBUser"];
29             var password = ConfigurationManager.AppSettings["GraphDBPassword"];
30             var client = new GraphClient(new Uri(url), user, password);
31             client.Connect();
32
33             GraphClient = client;
34         }
35
36         public static IGraphClient GraphClient { get; private set; }
37     }
38 }
39
```

```
100 %    No issues found
```

Neo4jDotNetDemo | Neo4jDotNetDemo.Controllers.SearchController

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Net;
5  using System.Net.Http;
6  using System.Web.Http;
7
8  namespace Neo4jDotNetDemo.Controllers
9  {
10     [RoutePrefix("search")]
11     public class SearchController : ApiController
12     {
13         [HttpGet]
14         [Route("")]
15         public IHttpActionResult SearchMoviesByTitle(string q)
16         {
17             //    query = ("MATCH (movie:Movie) "
18             //         "WHERE movie.title =~ {title} "
19             //         "RETURN movie")
20             //    params={"title": "(?i).*" + q + ".*"}
21
22             var data = WebApiConfig.GraphClient.Cypher
23                 .Match("(m:Movie)")
24                 .Where("m.title =~ {title}")
25                 .WithParam("title", "(?i).*" + q + ".*")
26                 .Return<Movie>("m")
27                 .Results.ToList();
28
29             return Ok(data.Select(c => new { movie = c}));
30         }
31     }
32 }
33
```

query στο Cypher
IEnumerable επιστρέφει
τ' αποτελέσματα απ'
neo4j σαν μια λιστα IList.

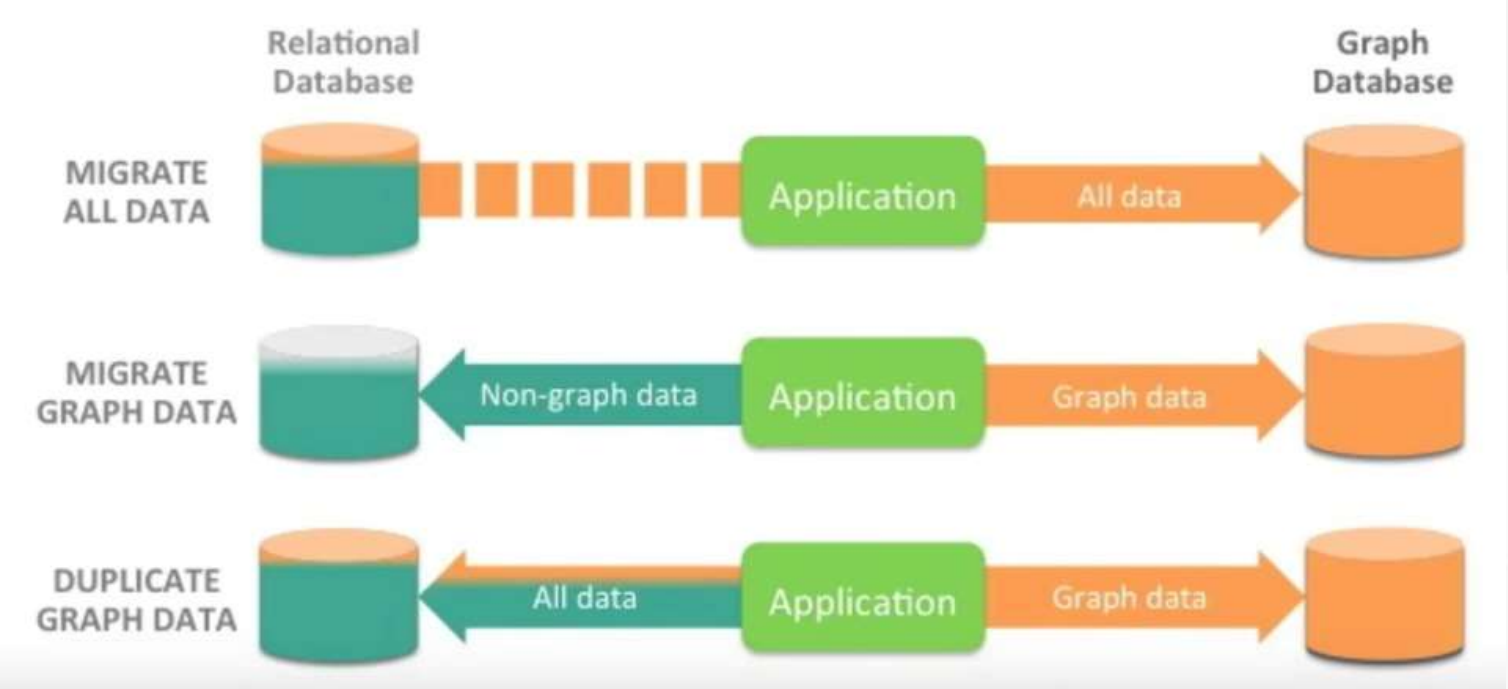
```
[RoutePrefix("movie")]
public class MovieController : ApiController
{
    [HttpGet]
    [Route("{title}")]
    public IHttpActionResult GetMovieByTitle(string title)
    {
        //query = ("MATCH (movie:Movie {title:{title}}) "
        // "OPTIONAL MATCH (movie)<-[r]-(person:Person) "
        // "RETURN movie.title as title,"
        // "collect([person.name, "
        // "         head(split(lower(type(r)), '_')), r.roles]) as cast "
        // "LIMIT 1")

        var data = WebApiConfig.GraphClient.Cypher
            .Match("(movie:Movie {title:{title}})")
            .OptionalMatch("(movie)<-[r]-(person:Person)")
            .WithParam("title", title)
            .Return((movie, a) => new
            {
                movie = movie.As<Movie>().title,
                cast = Return.As<IEnumerable<string>>("collect([person.name, head(split(lower(type(r)), '_')), r.roles])")
            })
            .Limit(1)
            .Results.FirstOrDefault();

        var result = new MovieResult();
        result.title = data.movie;

        var castresults = new List<CastResult>();
        foreach (var item in data.cast)
        {
            var tempData = JsonConvert.DeserializeObject<dynamic>(item);
            var roles = tempData[2] as JArray;
            var castResult = new CastResult
            {
                name = tempData[0],
                job = tempData[1],
            };
            if (roles != null)
            {
                castResult.role = roles.Select(c => c.Value<string>());
            }
            castresults.Add(castResult);
        }
        result.cast = castresults;
```

Πιο σύνθετο query
πανω στο Cypher IEnumerable
επιστρέφει ένα custom object
απ' τη neo4j σαν JSON array.

..το οποίο μπορούμε να κάνουμε cast
σε μια λίστα IList.

# Πρακτικά tips

1. Συνήθως ο καλύτερος τρόπος είναι να σπάσουμε το πρόβλημα σε επιμέρους σενάρια και να χρησιμοποιήσουμε και relational και document/graph DBs όπου χρειάζεται η καθεμια.



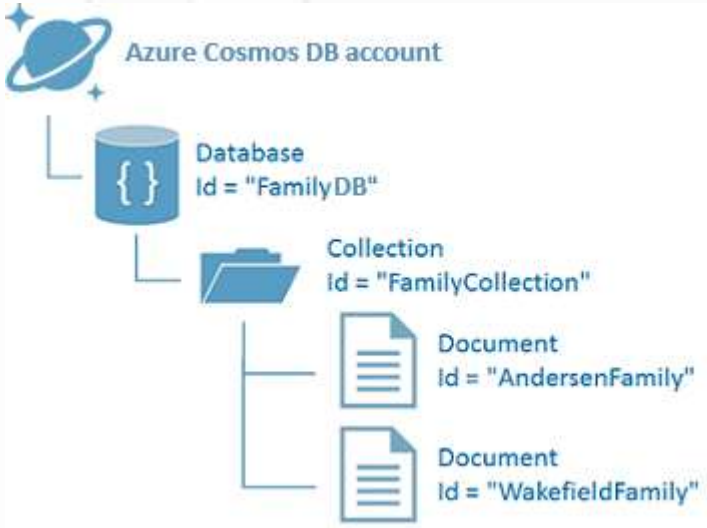

2. Βοηθάει να σκεφτούμε προκαταβολικά αν το μοντέλο θα ναι ~στατικό (=>RDBMs) or θ' αλλάζει συχνά (=>non-SQL DBs).
3. Καλό είναι να σκεφτούμε απ' την αρχή όχι μόνο transactional workloads αλλά και τυχόν απαιτήσεις για analytics.
4. Τρέχουμε τα queries σε async mode. Οι graph DBs είναι γρήγορες, αλλά όχι και ΤΟΣΟ γρήγορες!

# Παρεμπιπτόντως: Azure COSMOS DB

Ένα τυπικό παράδειγμα μιας ***multi-model DB***.

= επιτρέπει τη μοντελοποίηση διαφορετικών τμημάτων του προβλήματος/dataset χρησιμοποιώντας RDBMS, document, graph κλπ αναλόγως τι ταιριάζει καλύτερα,
όλα εντός της ίδιας βάσης και διαχειρίσιμα/αναζητήσιμα μέσω ενός generic API.

Azure Cosmos DB
Globally distributed, multi-model database service for any scale




| | Java | .NET | Node.js | Python | Gremlin | Go | Xamarin |
|---|---|---|---|---|---|---|---|
| SQL API | ↗ | ↗ | ↗ | ↗ | | | ↗ |
| Azure Cosmos DB's API for MongoDB | ↗ | ↗ | ↗ | ↗ | | ↗ | ↗ |
| Gremlin API | ↗ | ↗ | ↗ | ↗ | ↗ | | |
| Table API | ↗ | ↗ | ↗ | ↗ | | | |
| Cassandra API | ↗ | ↗ | ↗ | ↗ | | | |

# Συμπεράσματα

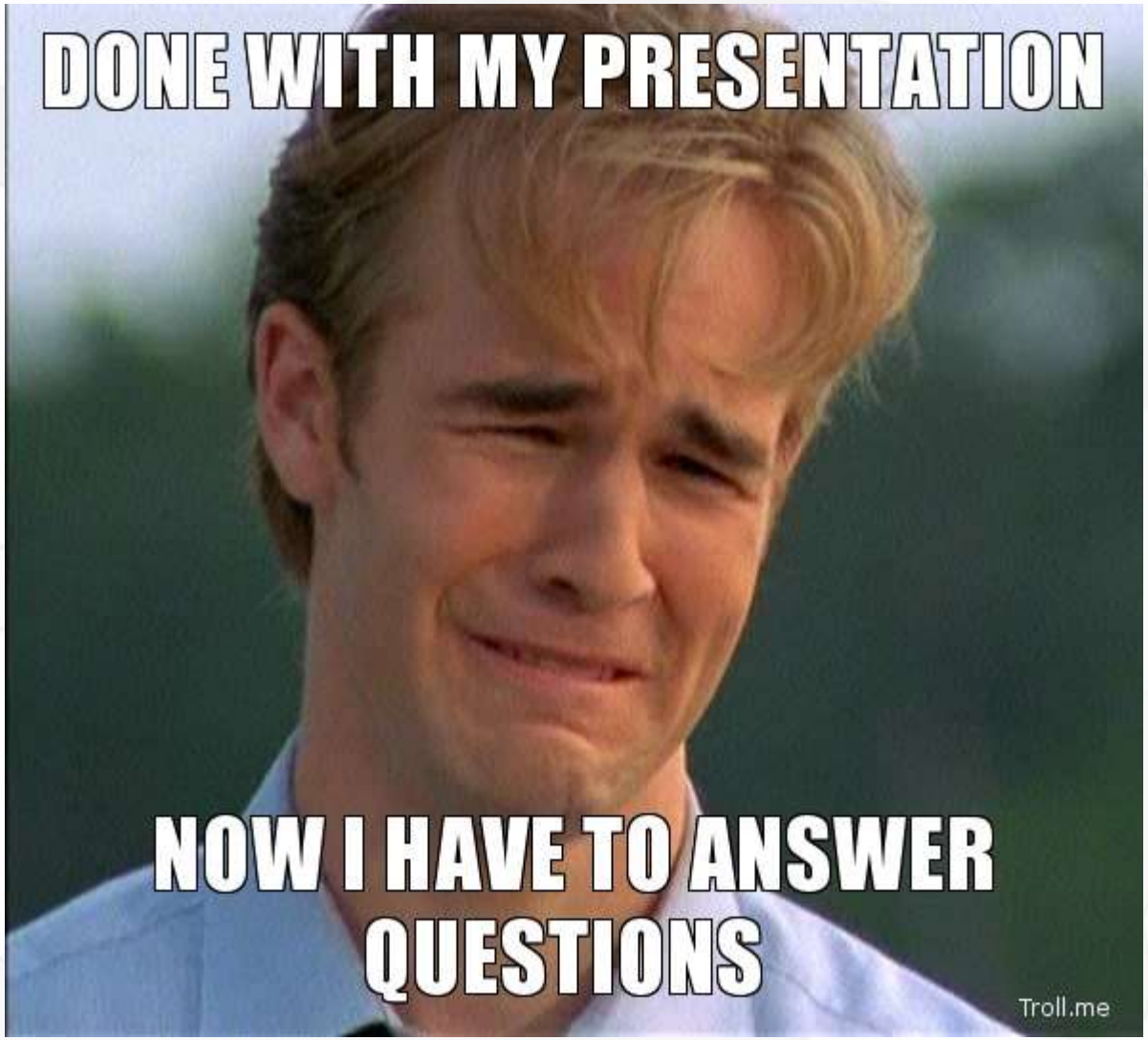
Υπάρχουν NP-hard (=μη πολυωνυμικής πολυπλοκότητας) υπολογιστικά προβλήματα (ή υποπροβλήματα) που επιδέχονται απλούστερες και πολύ ταχύτερες λύσεις αν μοντελοποιηθούν σαν γράφοι.

Παρά τη μόδα των non-SQL βάσεων, οι RDBMSs και η SQL δεν παύουν να ειναι το βασικό μας εργαλείο.

Μακροπρόθεσμα οι multi-model DBs πιθανότατα θα γίνουν το standard.

Την επόμενη φορά που θα παλεύετε να ταιριάξετε ένα σύνθετο SQL schema στον κώδικά σας και στο πρόβλημα του πελάτη, σκεφτείτε μήπως ένας διαφορετικός τρόπος μοντελοποίησης με γράφους βγάζει καλύτερα νόημα και τρέχει σφαίρα.

# Questions


DONE WITH MY PRESENTATION
NOW I HAVE TO ANSWER QUESTIONS
Troll.me