
# Building social network with Neo4j and Python

Andrii Soldatenko

3-4 July 2016

@a_soldatenko

# Agenda:

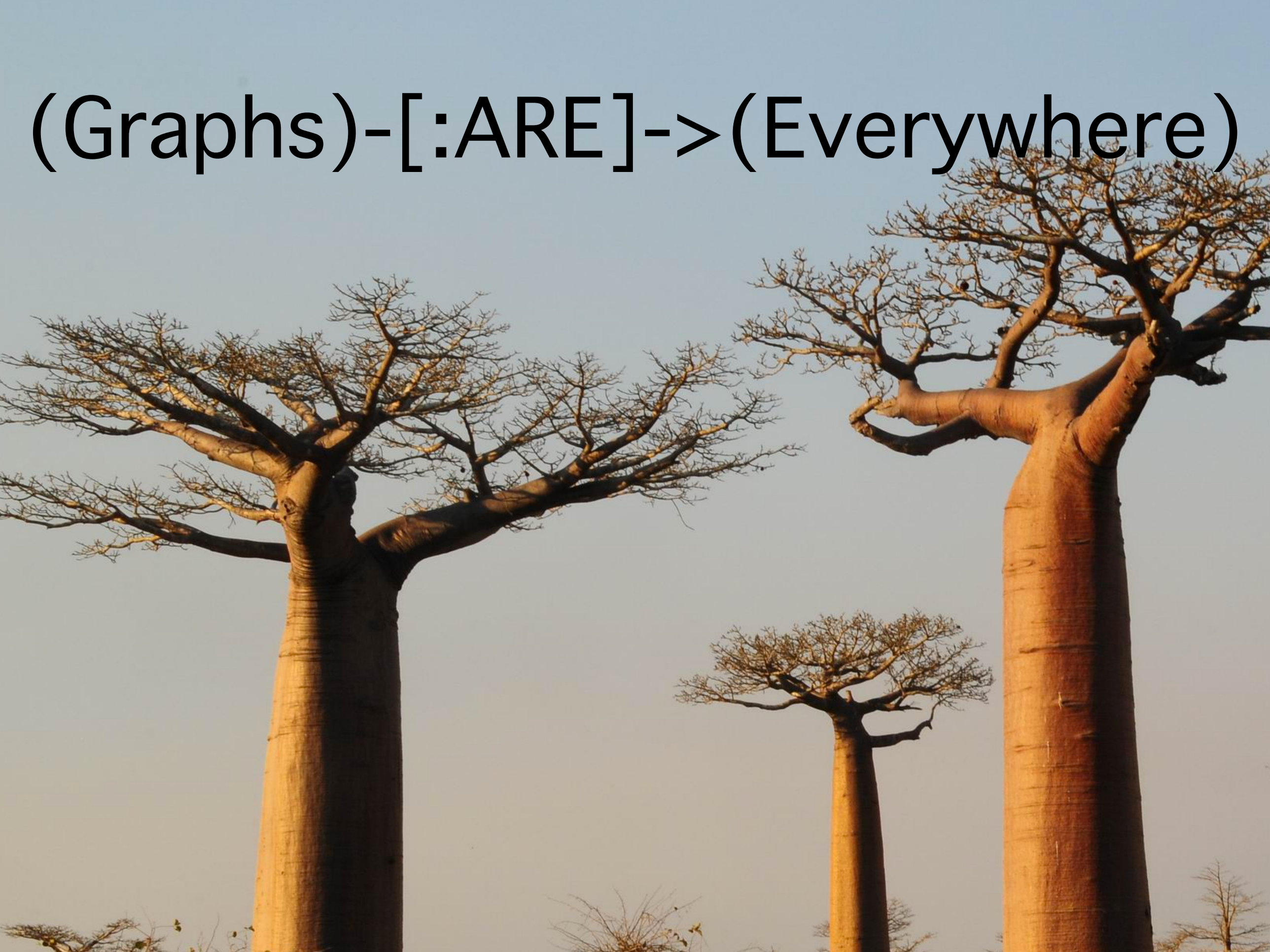
- Who am I?
- 101 Graph data structure in Python
- Graph Databases in Practise
- Neo4j overview
- PEP-249 and Neo4j
- python neo4j clients

# Andrii Soldatenko

- Backend Python Developer at 

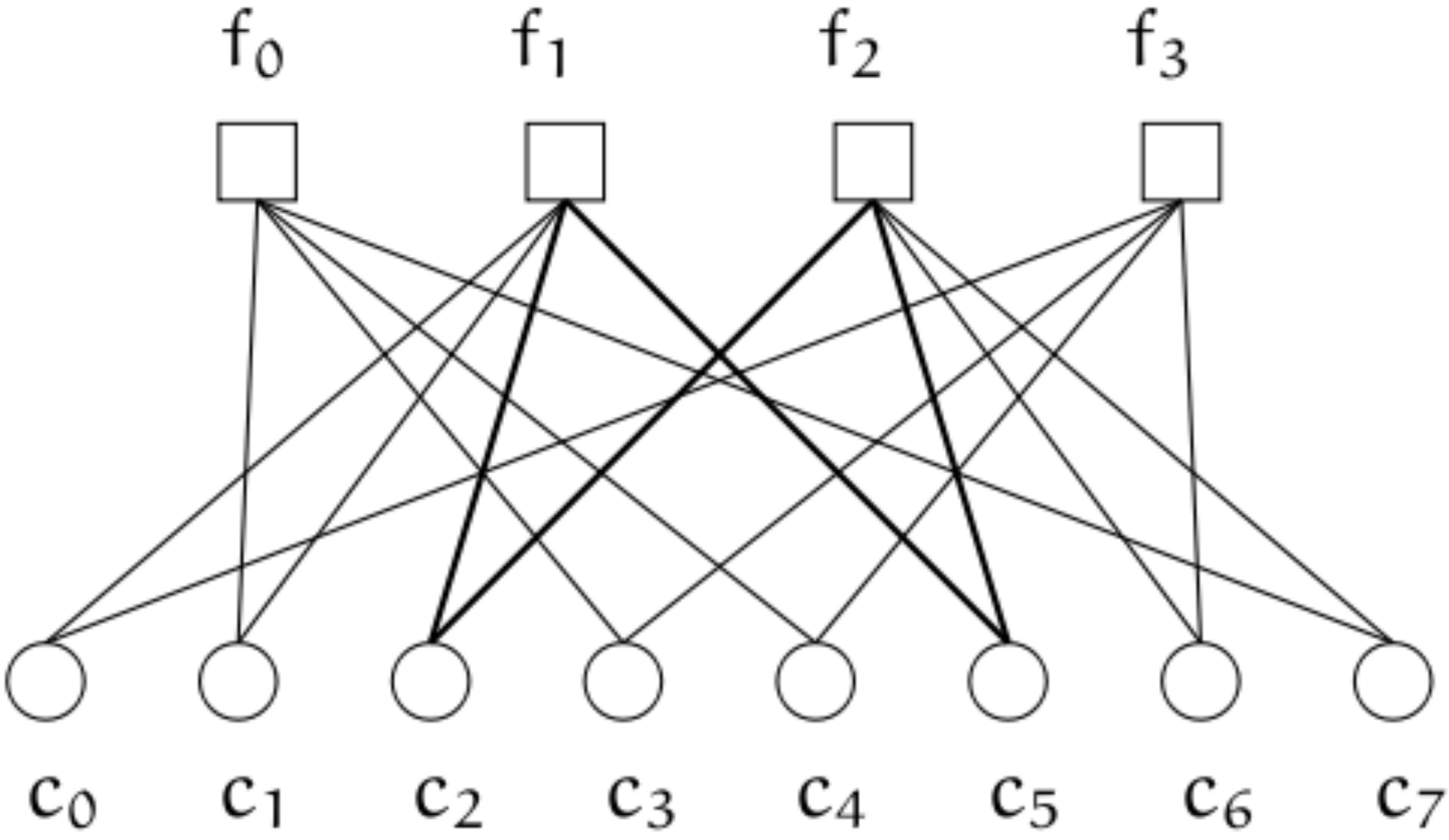
- CTO in Persollo.com
- Speaker at many PyCons and Python meetups
- blogger at https://asoldatenko.com

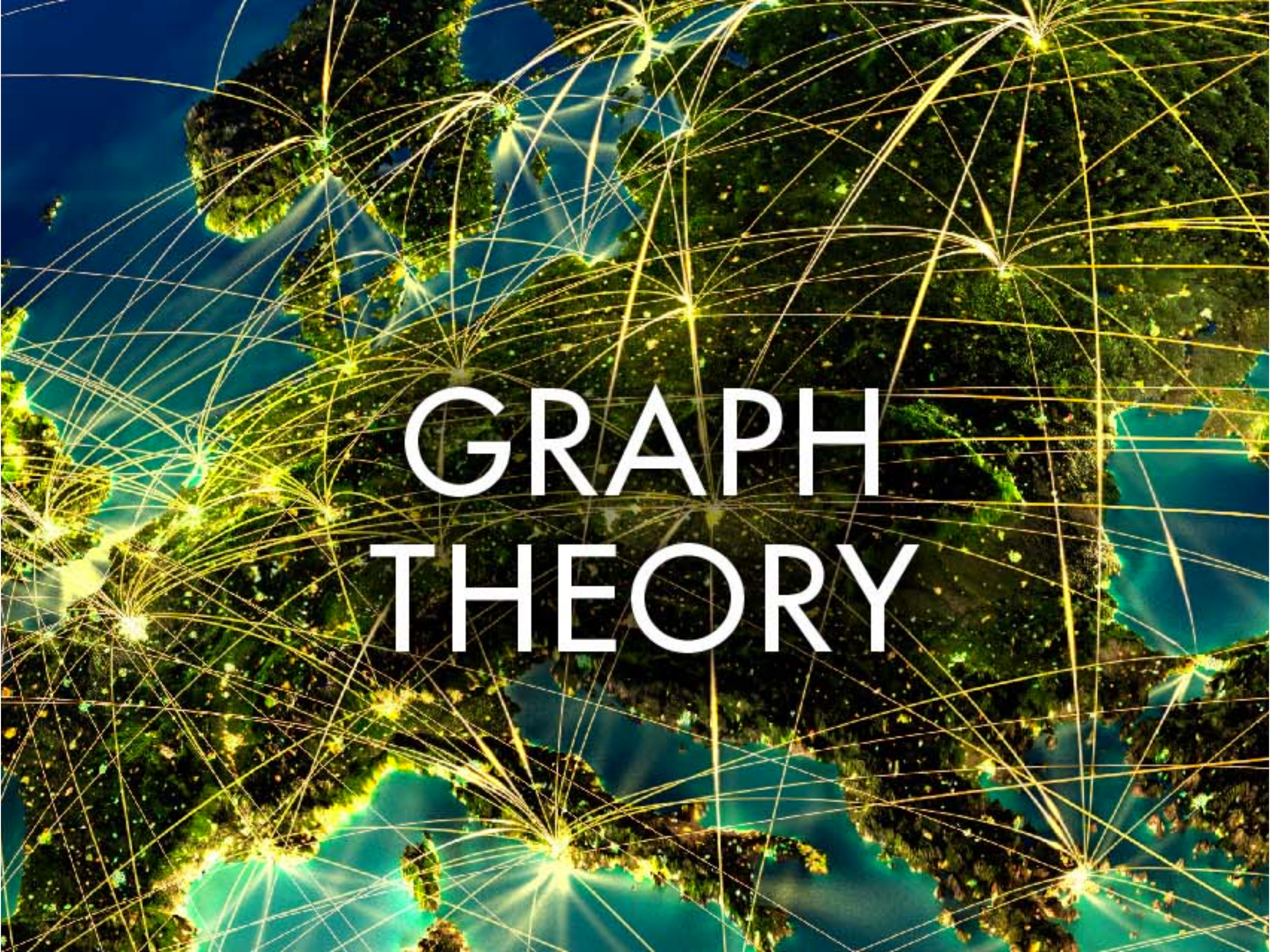
# (Graphs)-[:ARE]->(Everywhere)



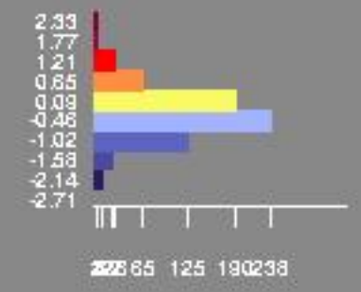
# Graph Usage

# Preface

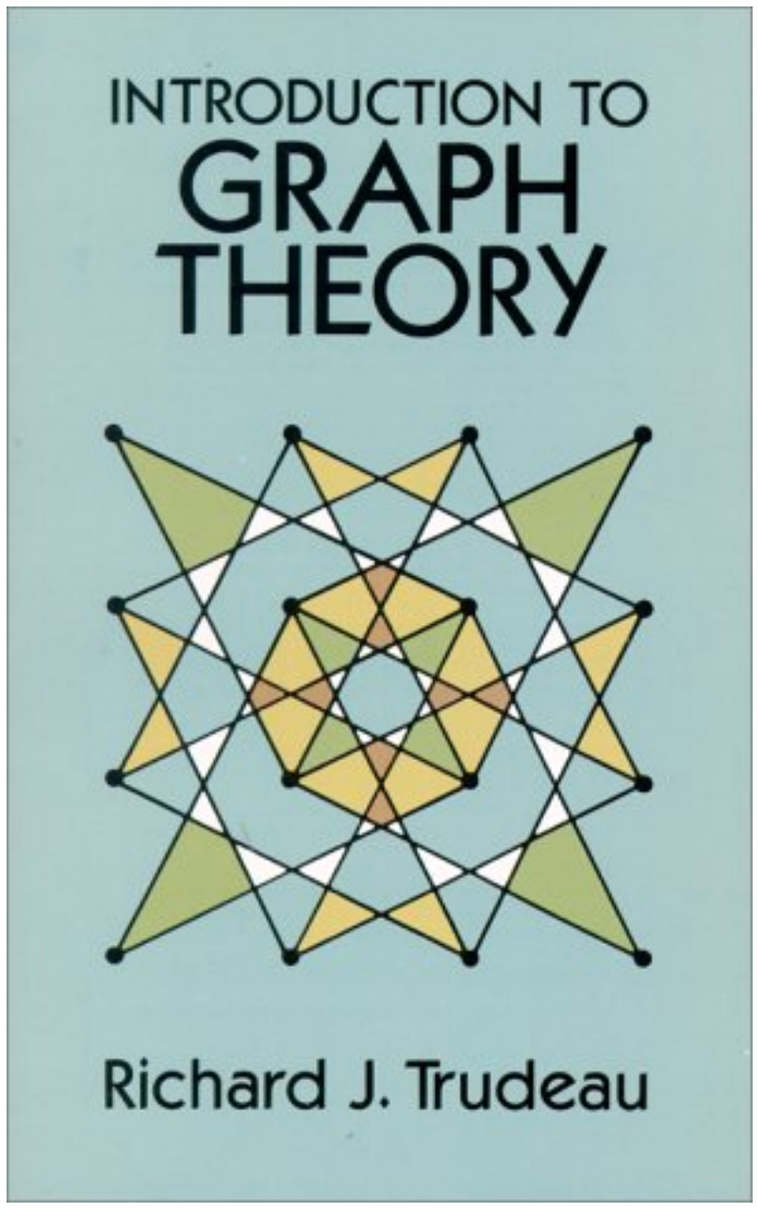



GRAPH
THEORY

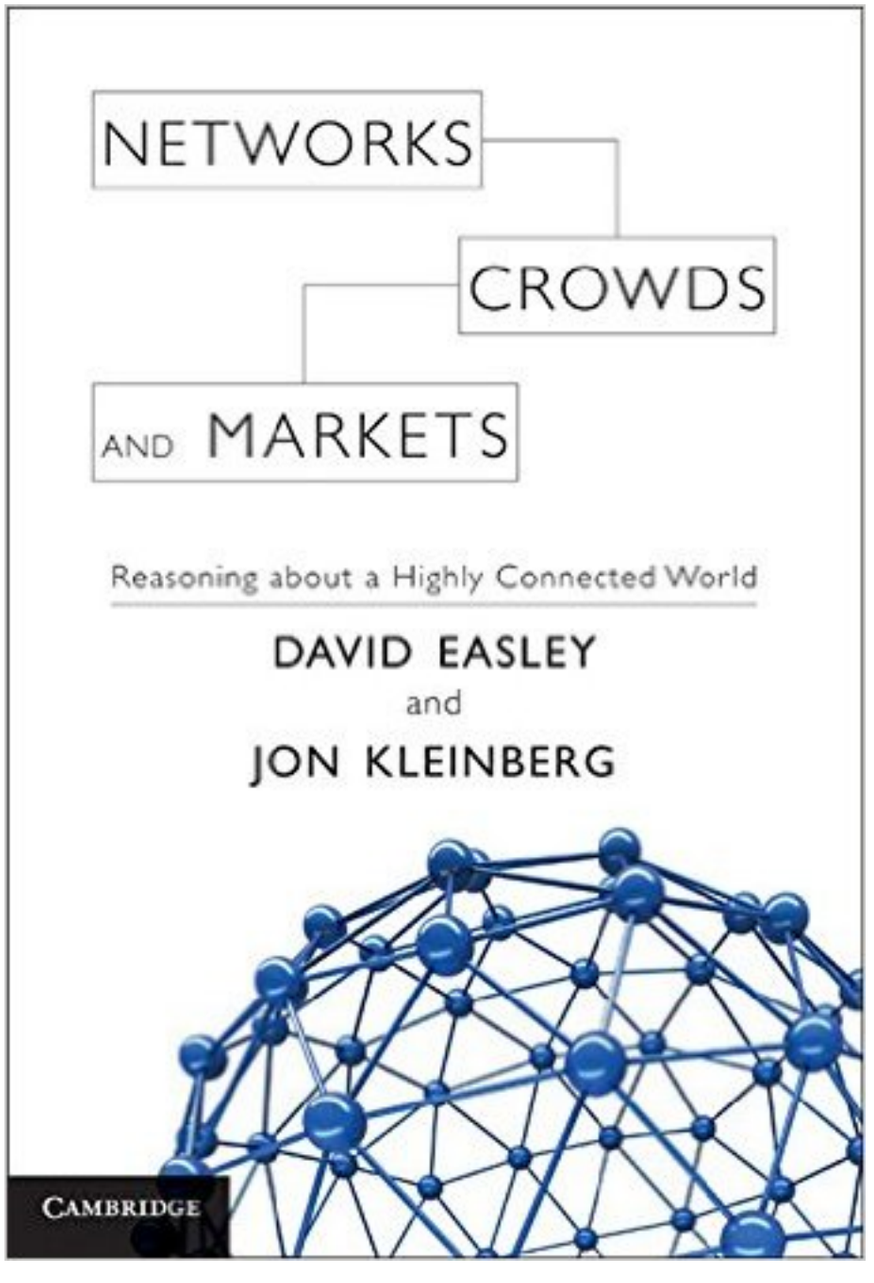
# World Trade 1994
# Residuals Model 1



# Books




http://bit.ly/29gkuz3




http://bit.ly/29btHYt

# 101 Graph data structure

## Adjacency matrix:

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 1 |
| 2 | 1 | 0 | 1 | 1 | 1 |
| 3 | 0 | 1 | 0 | 1 | 0 |
| 4 | 0 | 1 | 1 | 0 | 1 |
| 5 | 1 | 1 | 0 | 1 | 0 |

# 101 Graph data structure

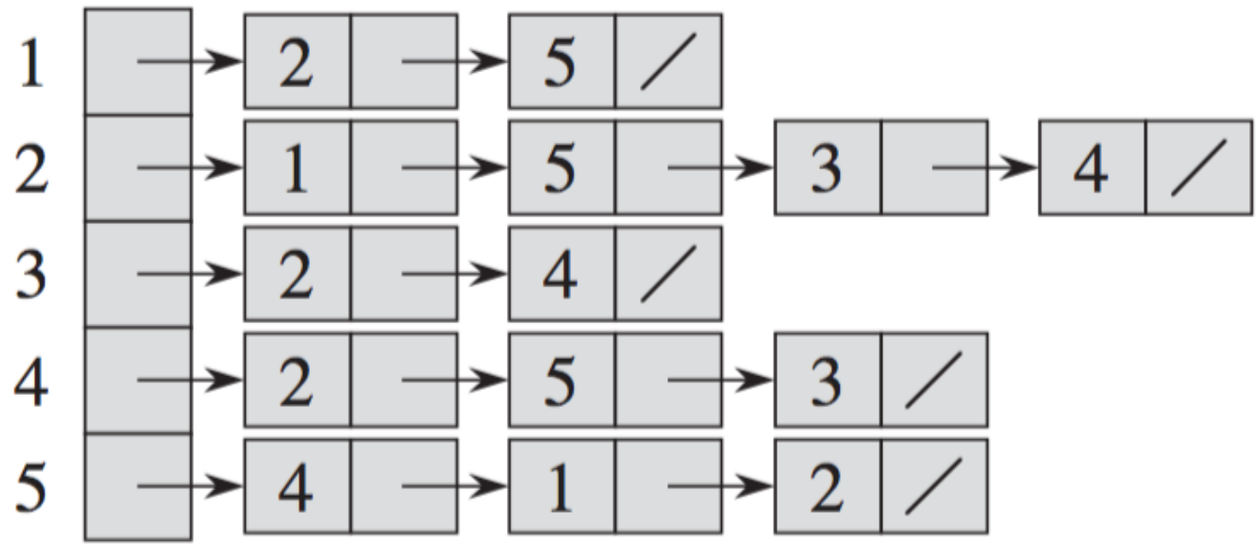
## Adjacency list:

```
graph = {'A': ['B', 'C'],
         'B': ['C', 'D'],
         'C': ['D'],
         'D': ['C'],
         'E': ['F'],
         'F': ['C']}
```

from https://www.python.org/doc/essays/graphs/

# BDFL

## Великодушный пожизненный диктатор



# Find path

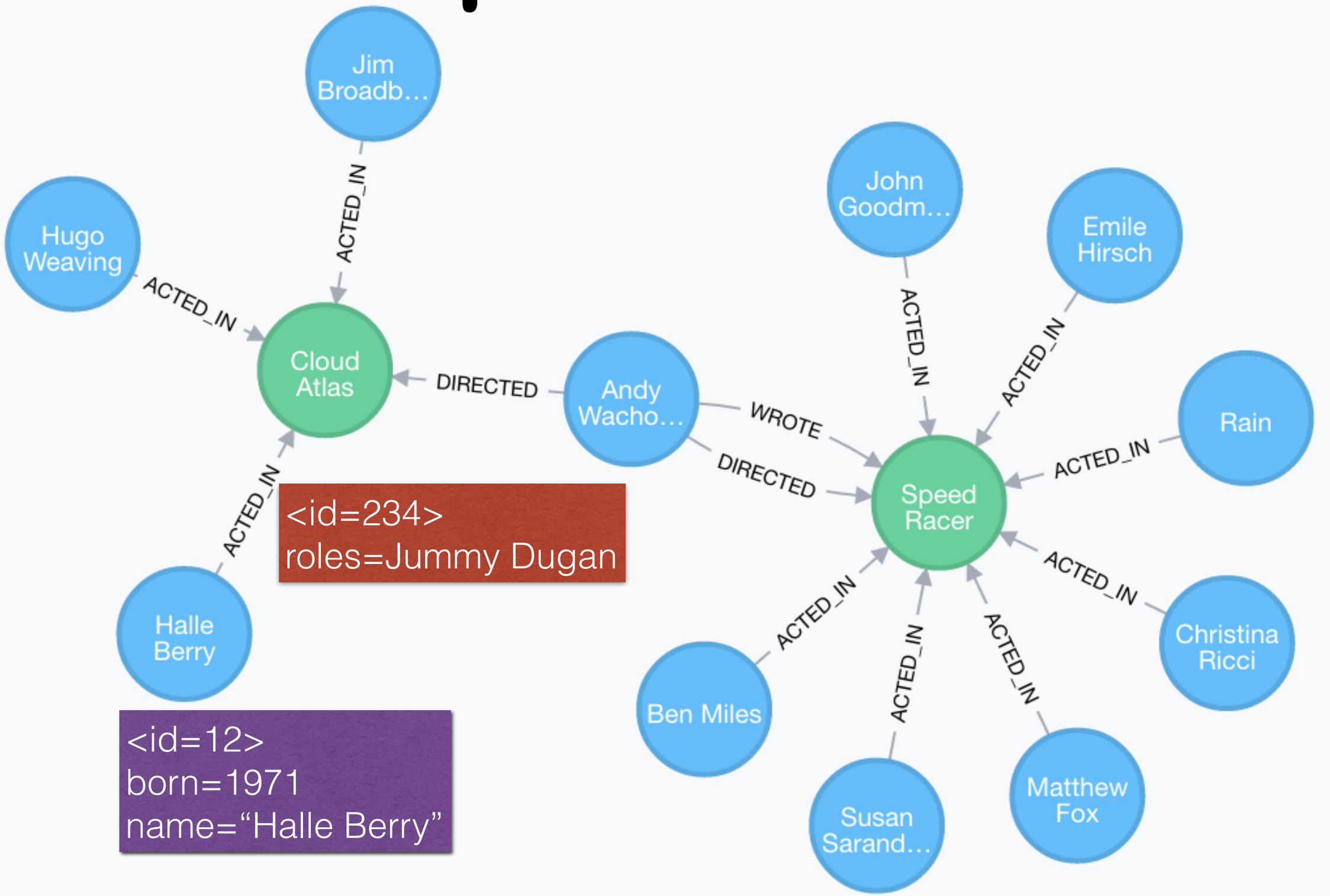
```
def f_p(g, s, e, path=[]):
    """Find path in graph by Guido"""
    path = path + [s]
    if s == e:
        return path
    if not s in g:
        return None
    for node in g[s]:
        if node not in path:
            n = f_p(g, node, e, path)
            if n: return newpath
    return None
```

# Example:

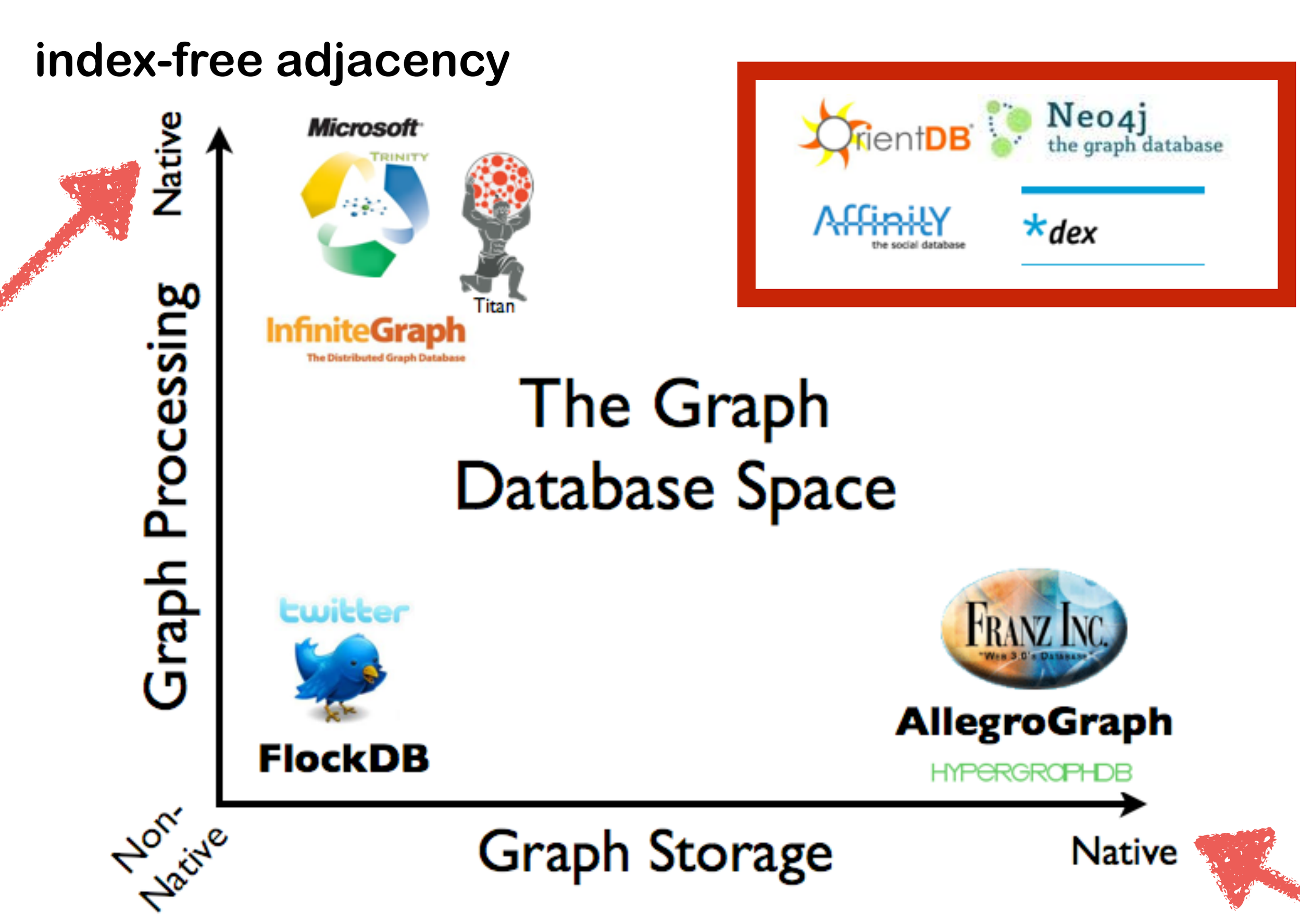
```
>>> graph = {'A': ['B', 'C'],
             'B': ['C', 'D'],
             'C': ['D'],
             'D': ['C'],
             'E': ['F'],
             'F': ['C']}

>>> find_path(graph, 'A', 'D')
['A', 'B', 'C', 'D']
>>>
```

# Graph database

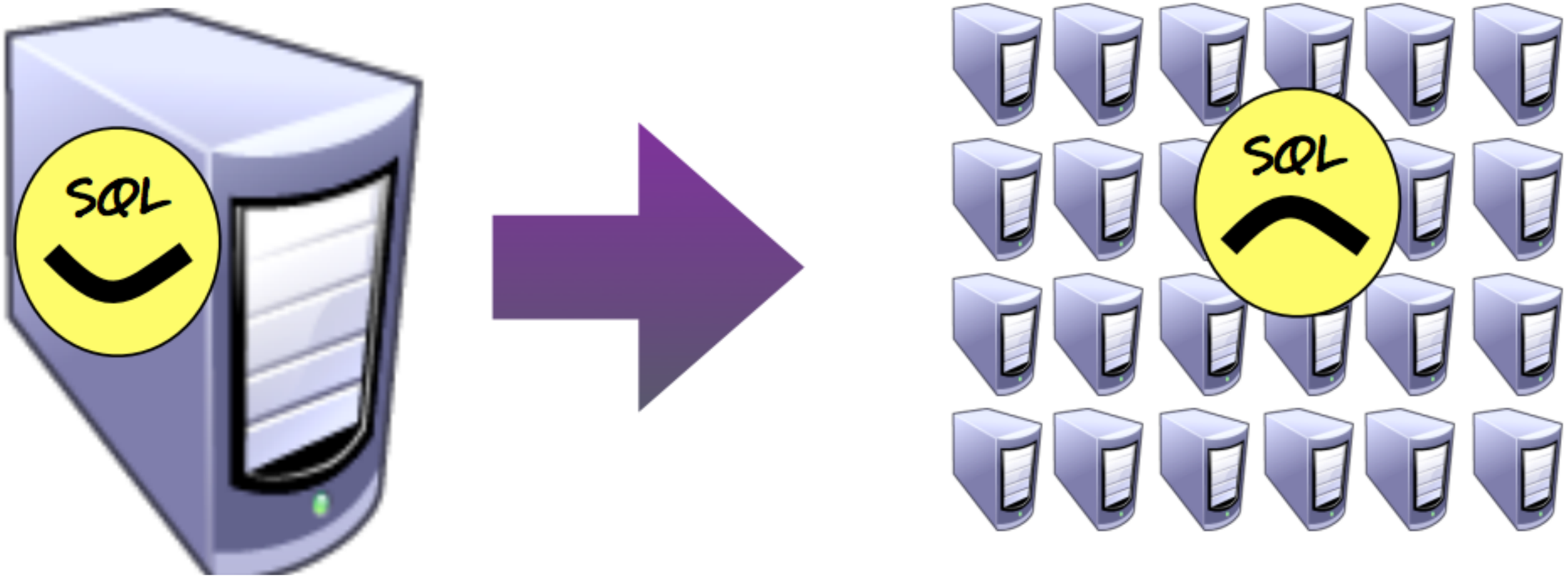


# index-free adjacency

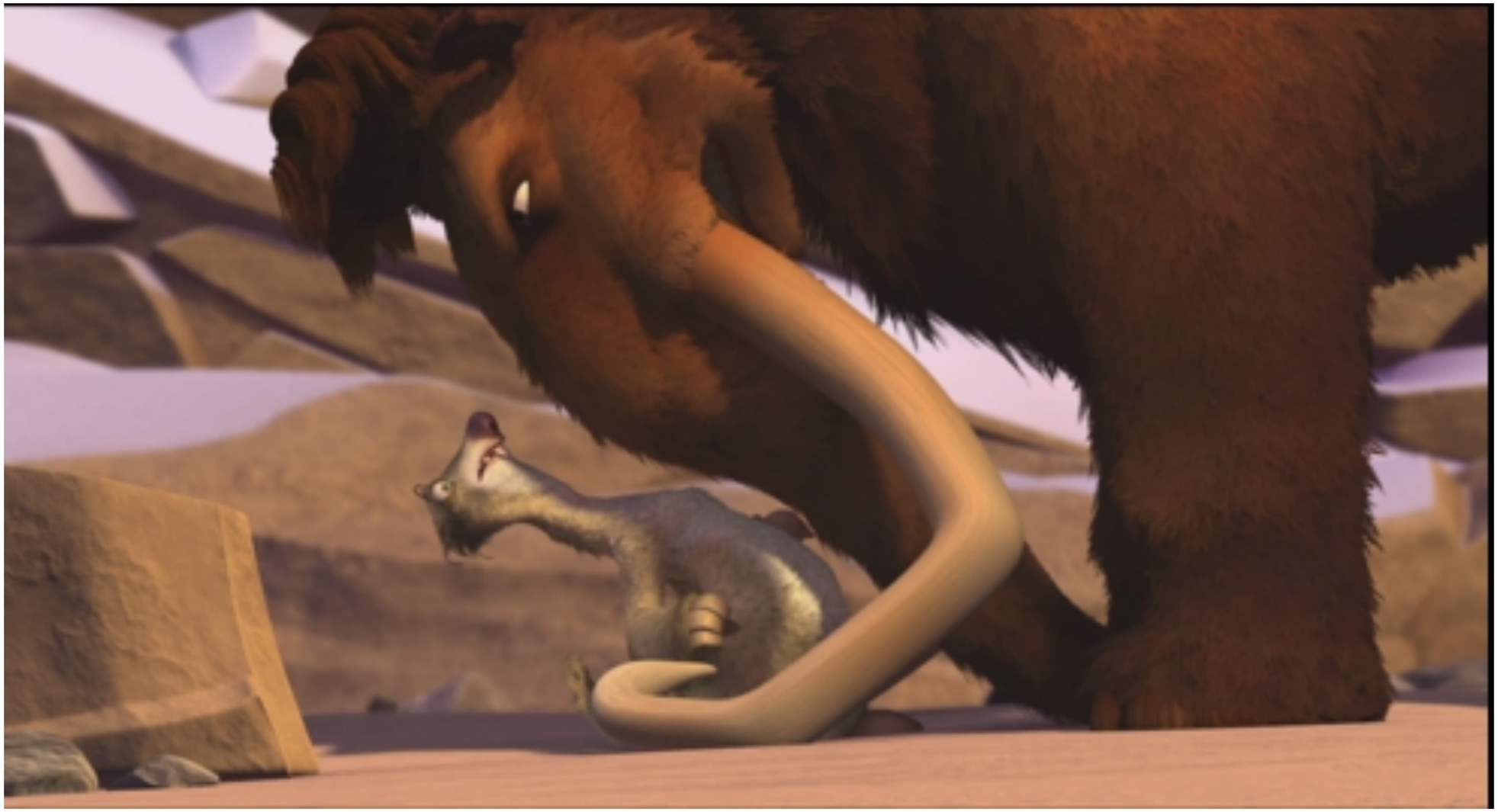



http://bit.ly/298VpUR

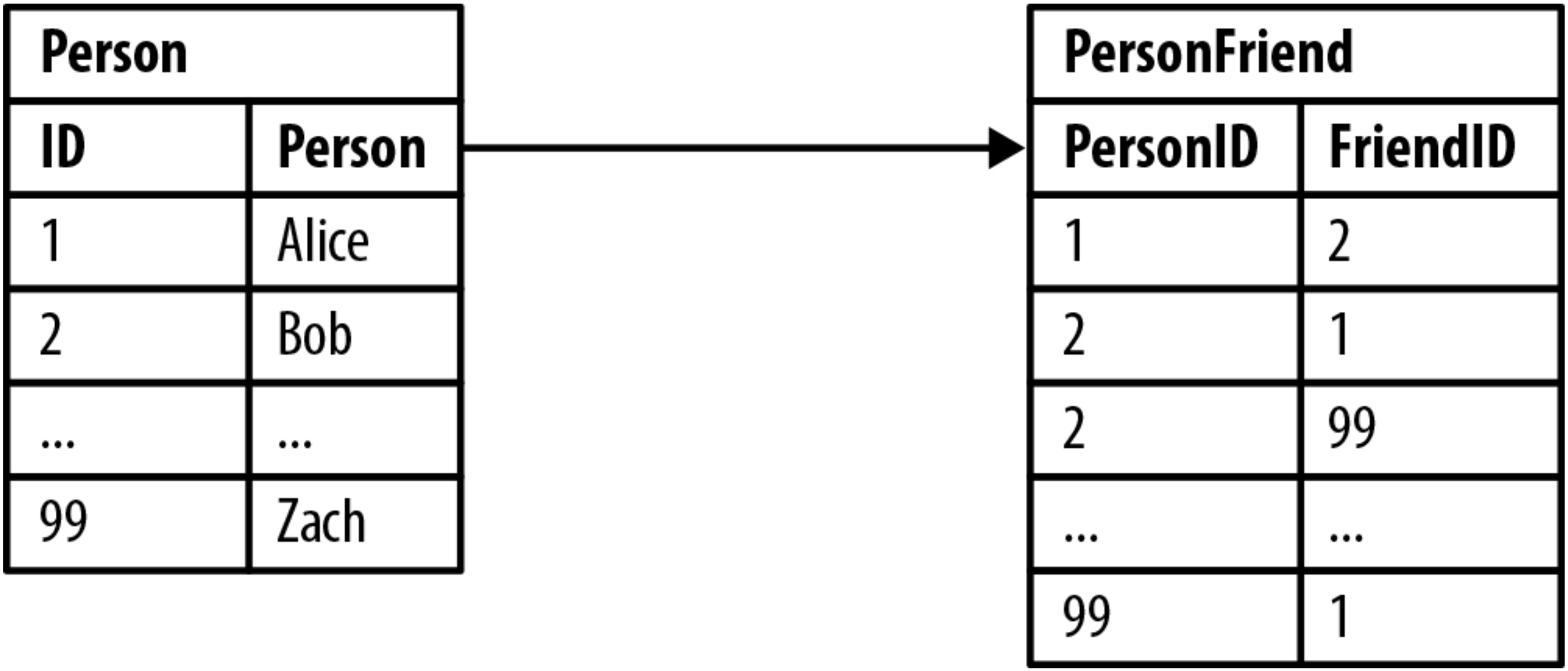
# SQL vs NoSQL



# Who is my boss?



# Friends problem: SQL



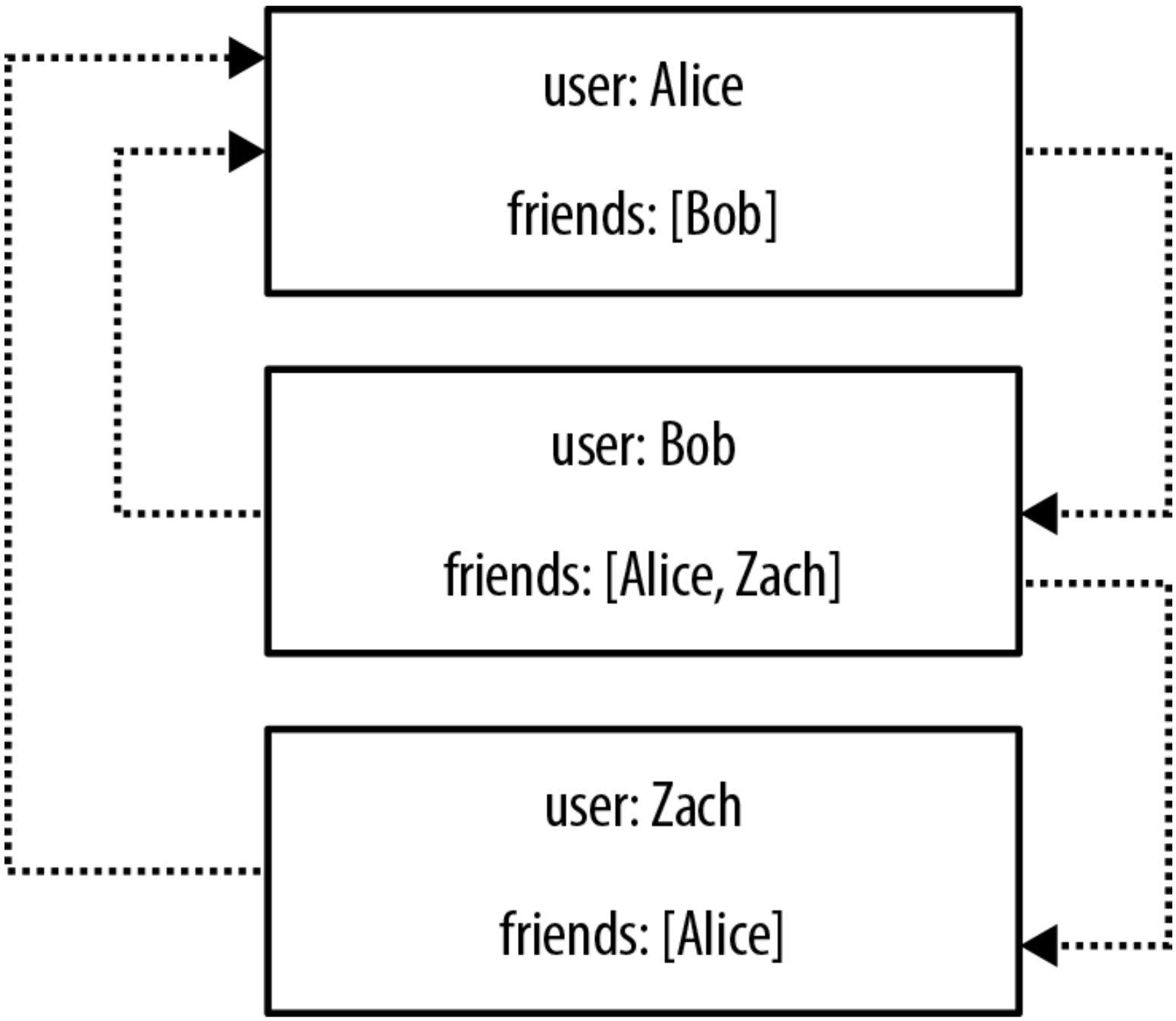
# “Bob” friends

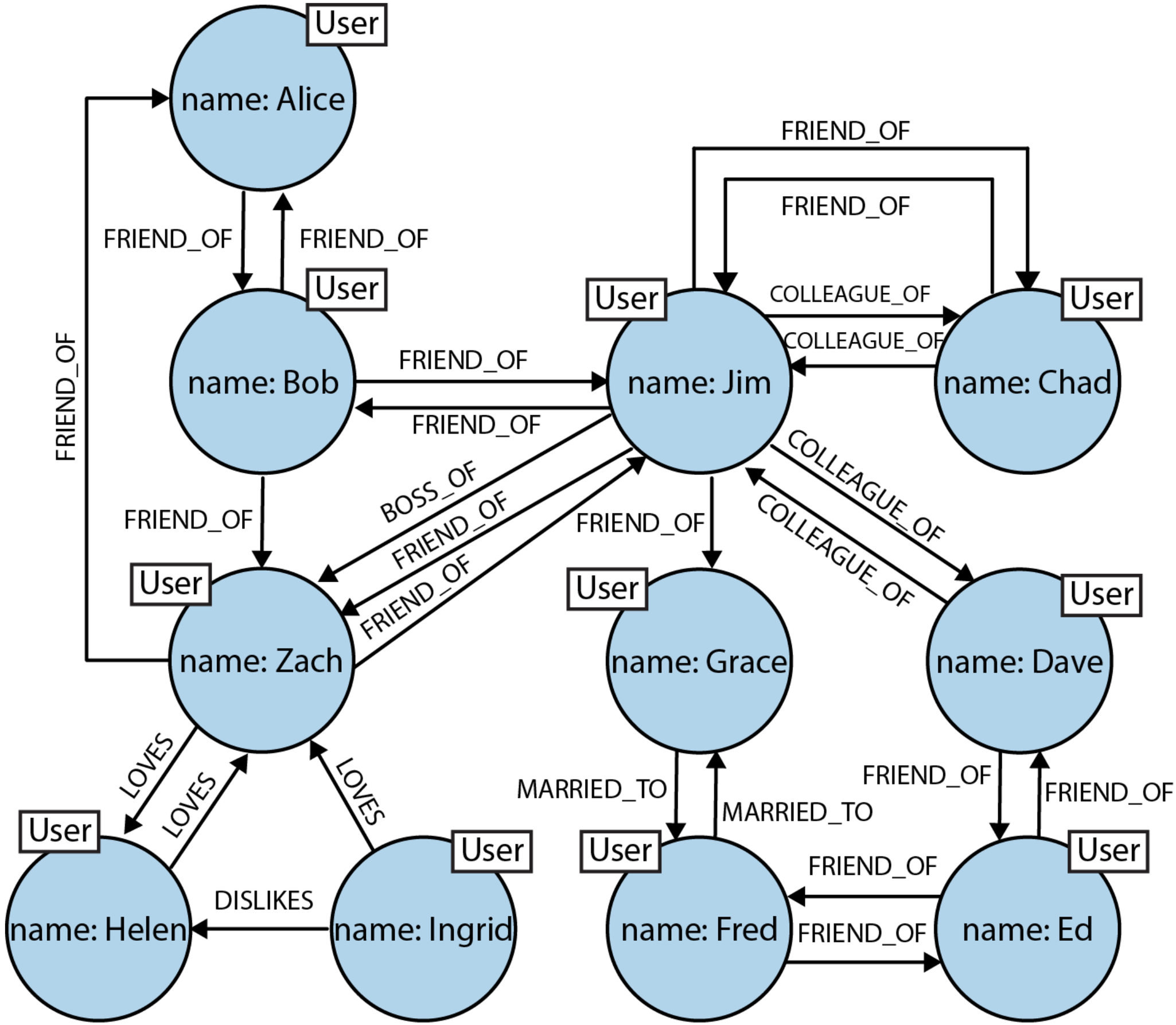
```
SELECT p1.Person
FROM Person p1 JOIN
PersonFriend
ON PersonFriend.FriendID =
p1.ID JOIN Person p2
ON PersonFriend.PersonID =
p2.ID WHERE p2.Person = 'Bob'
```

# “Bob” friends-of-friends

```
SELECT p1.Person AS PERSON, p2.Person
AS FRIEND_OF_FRIEND
FROM PersonFriend pf1 JOIN Person p1
ON pf1.PersonID = p1.ID JOIN
PersonFriend pf2
ON pf2.PersonID = pf1.FriendID JOIN
Person p2
ON pf2.FriendID = p2.ID
WHERE p1.Person = 'Alice' AND
pf2.FriendID <> p1.ID
```

# Friends problem: Not only SQL




User
name: Alice
FRIEND_OF
FRIEND_OF
FRIEND_OF
User
name: Bob
FRIEND_OF
FRIEND_OF
FRIEND_OF
FRIEND_OF
User
name: Jim
COLLEAGUE_OF
COLLEAGUE_OF
User
name: Chad
BOSS_OF
FRIEND_OF
FRIEND_OF
FRIEND_OF
COLLEAGUE_OF
COLLEAGUE_OF
User
name: Zach
User
name: Grace
User
name: Dave
LOVES
LOVES
LOVES
MARRIED_TO
MARRIED_TO
FRIEND_OF
FRIEND_OF
User
name: Helen
DISLIKES
User
name: Ingrid
User
name: Fred
FRIEND_OF
FRIEND_OF
User
name: Ed

# Experiment

MySQL

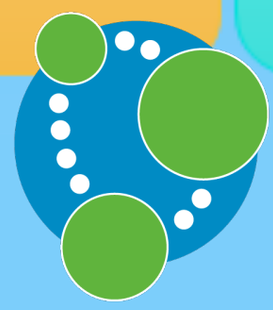
1 000 000 X 50 = 50 000 000

| Depths | MySQL execution time(s) | Neo4j execution time (s) | Number off return records |
|---|---|---|---|
| 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 |

A C I D

Atomicity Consistency Isolation Durability

*Safe Operation*



neo4j

# Availability
# only enterprise :(


neo4j

# Neo4j Search

# Graph Query Languages


**Cypher Query Language (CQL)**

**SPARQL**

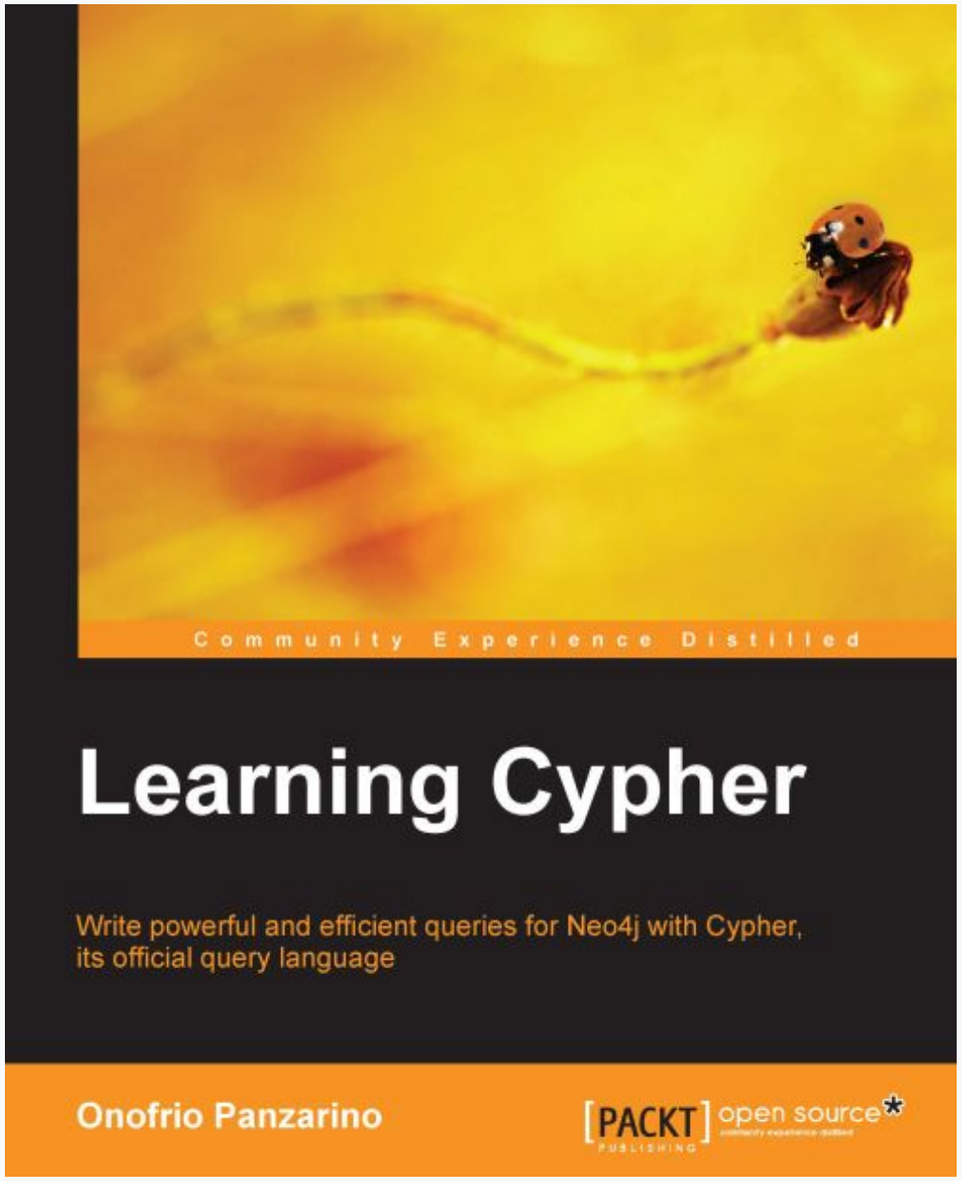
https://github.com/tinkerpop/gremlin

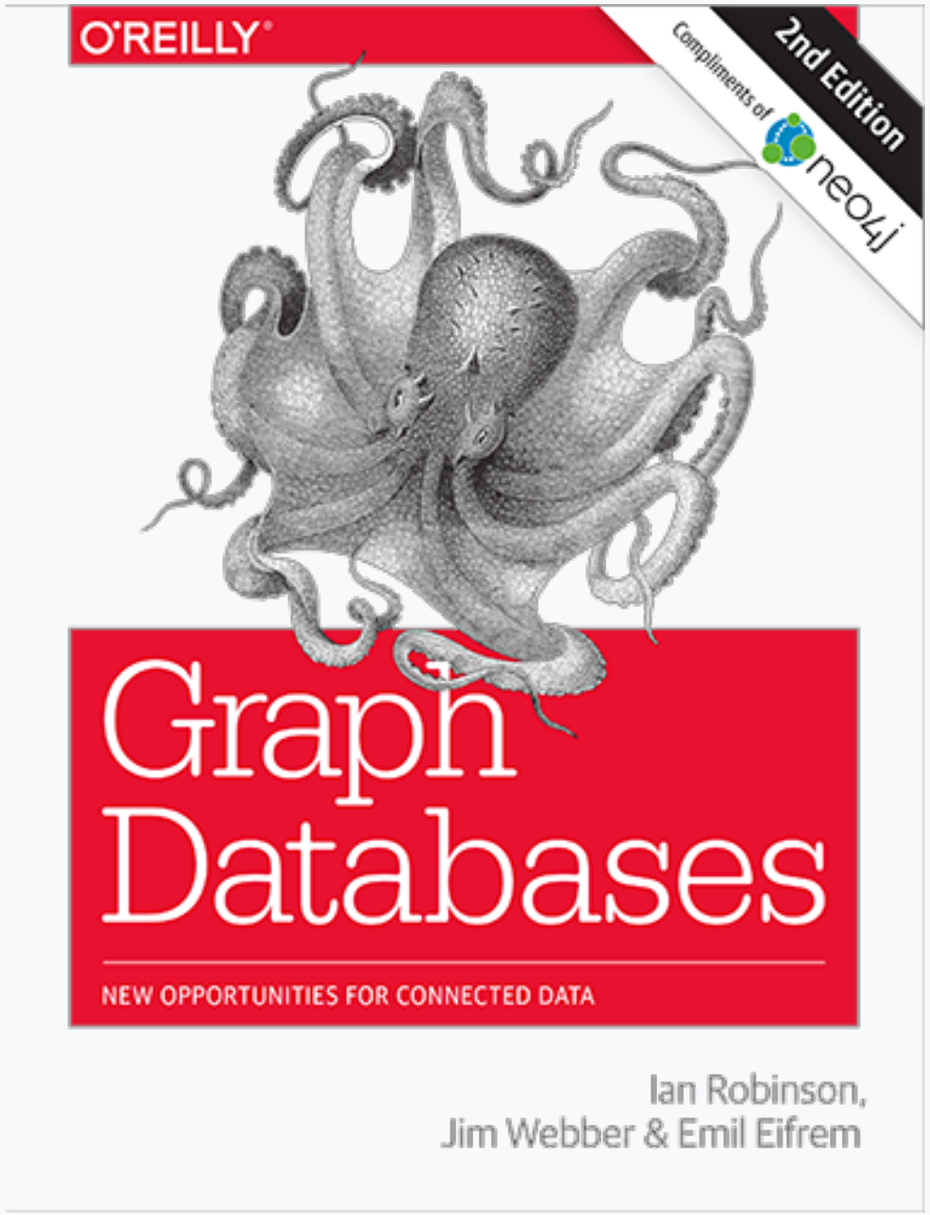
https://www.w3.org/TR/rdf-sparql-query/

# Cypher Query Language

```
MATCH (relationship, pattern matches)
WHERE (filter condition)
RETURN (what to return, nodes, rels,
properties)
```

# Advanced Cypher




http://neo4j.com/docs/cypher-refcard/current/

# Cypher: Example

```
MATCH (n:Person {name: "Tom Hanks”})
      -[:ACTED_IN]->(m)
RETURN n,m
```

# Cypher: Example

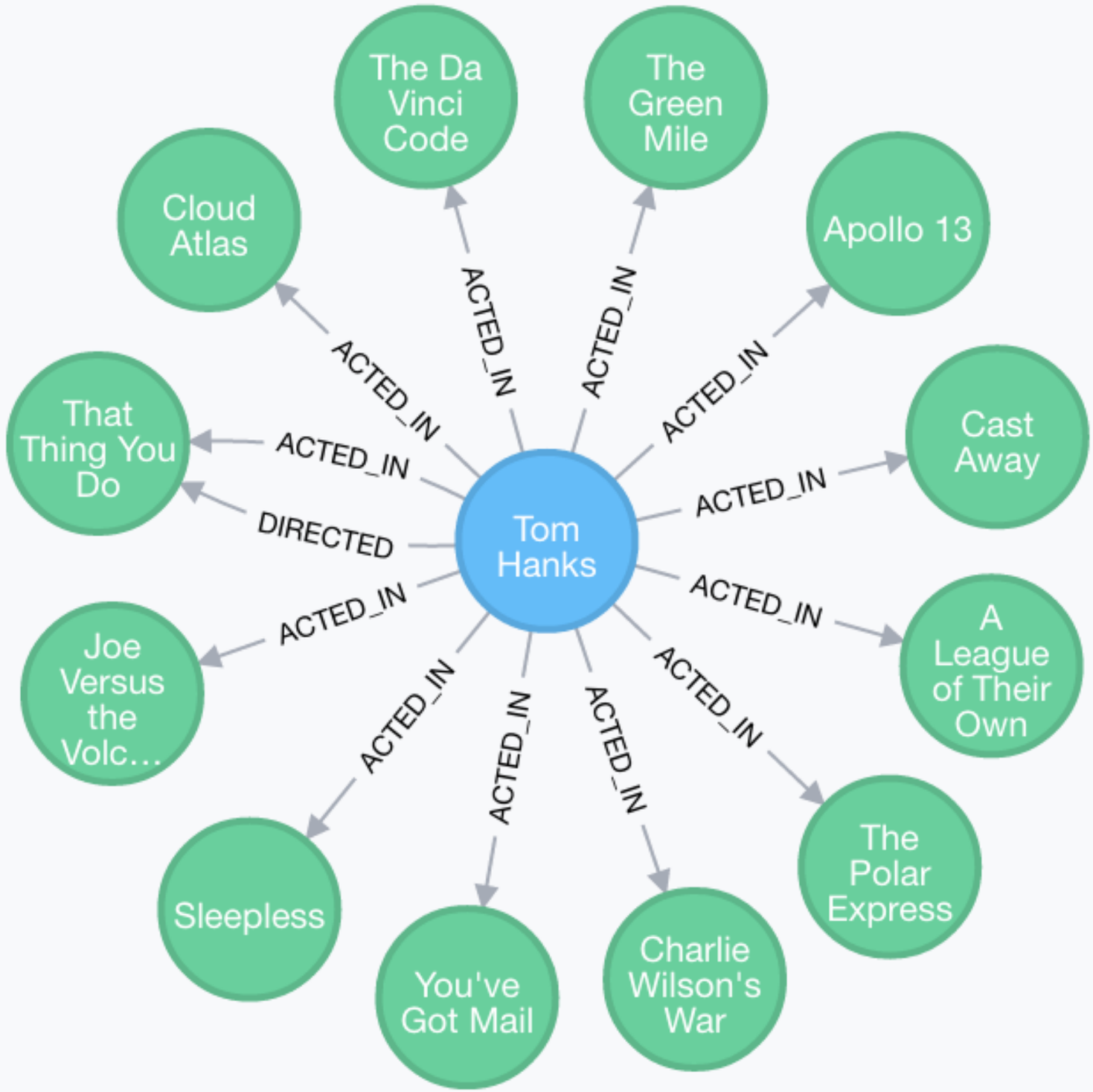
```
MATCH (n:Person {name: "Tom Hanks"})
      -[:ACTED_IN]->(m)
RETURN n,m
```

```
SELECT n,m
FROM persons;
```

# Cypher: Example

```
SELECT n,m
FROM persons;
```

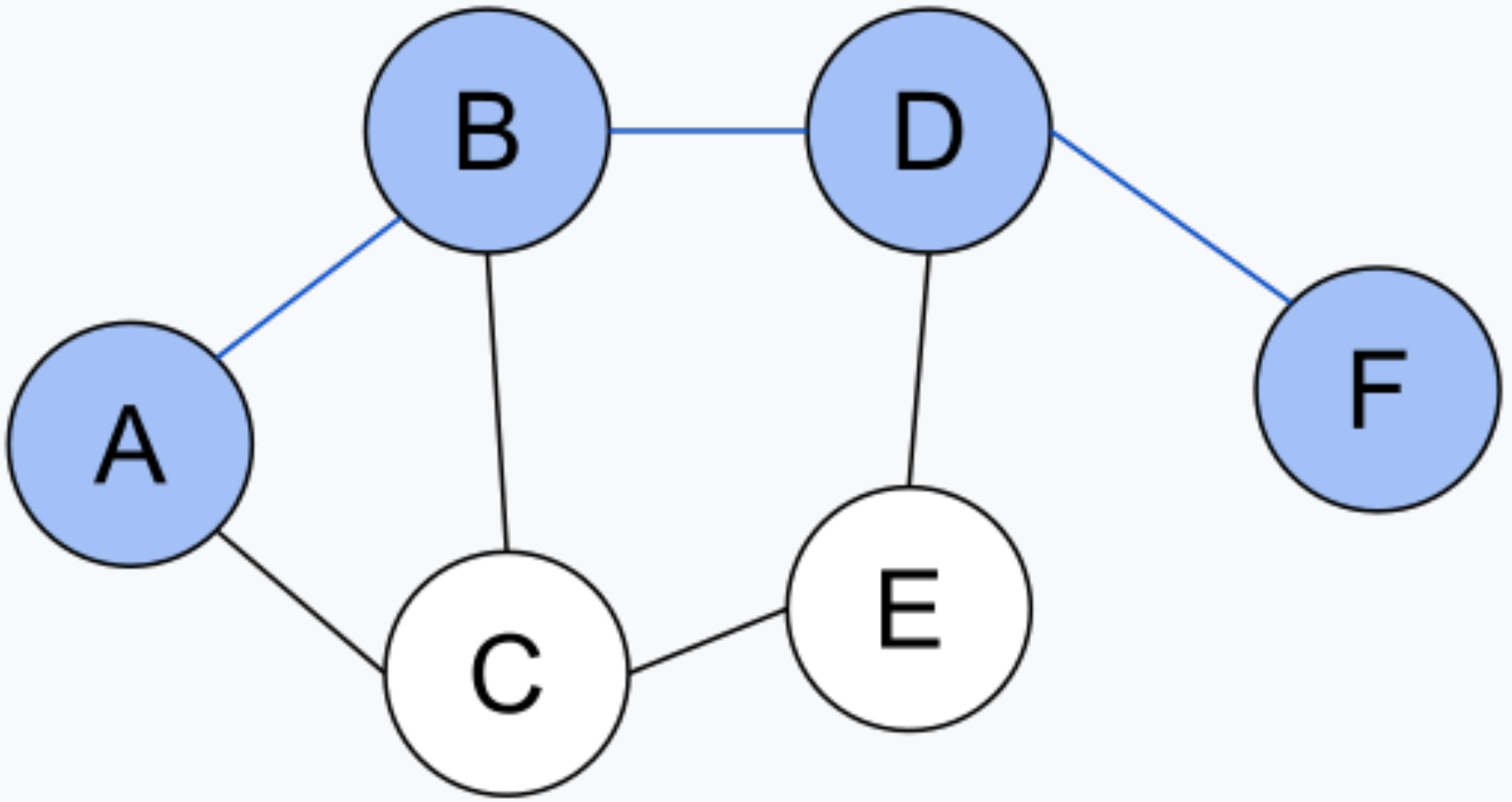
# Cypher: Example



# Cypher: Example

| tomHanksMovies.title |
| --- |
| Charlie Wilson's War |
| The Polar Express |
| A League of Their Own |
| Apollo 13 |
| The Green Mile |

# Pros



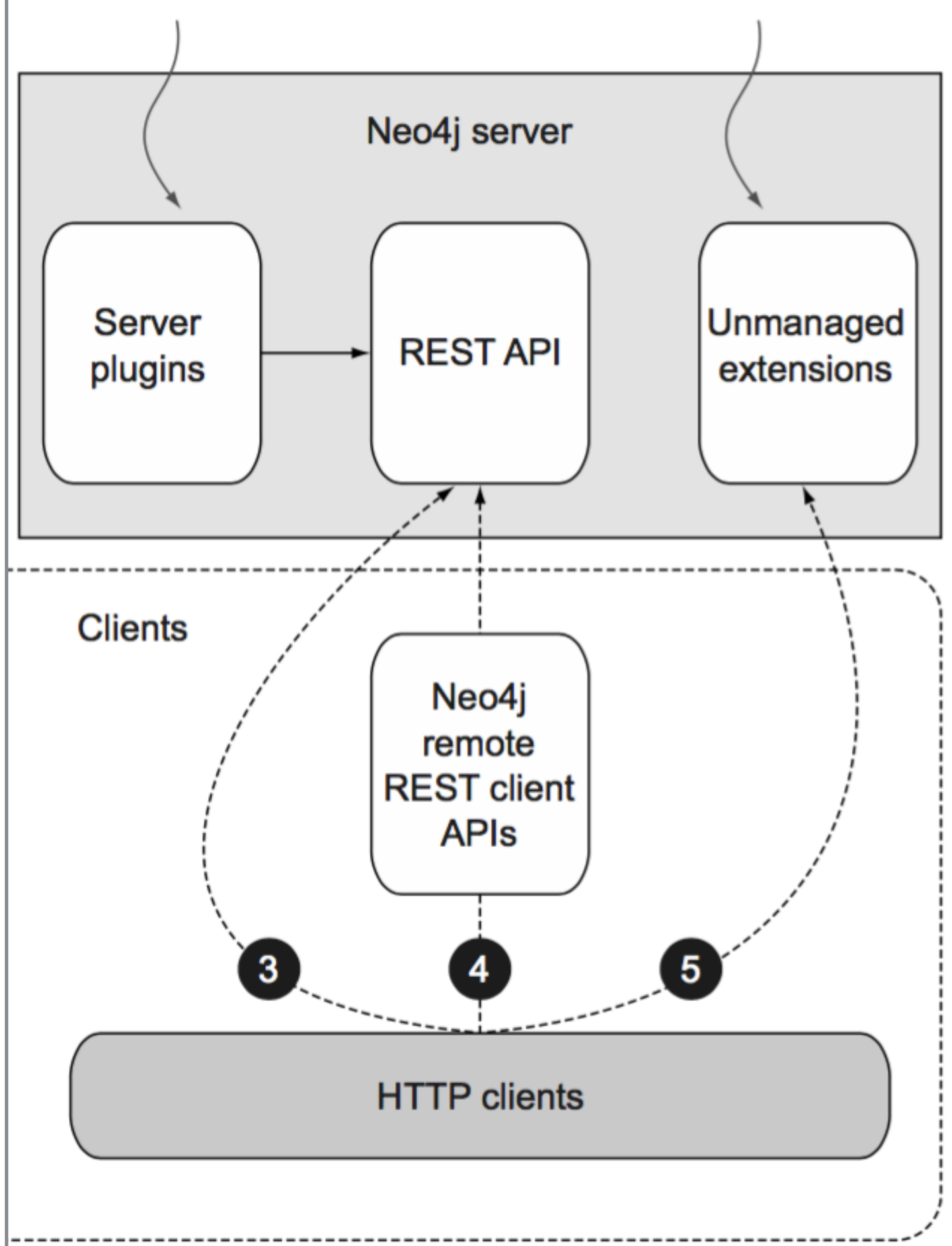

neo4j

# Contra

- when you want to update all or a subset of entities

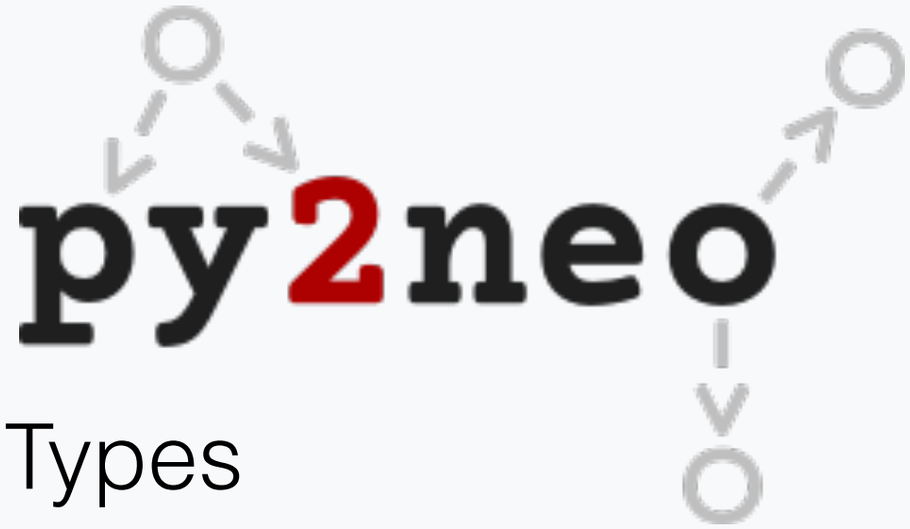
- unable to handle lots of data

# Python


Neo4j server
Server plugins
REST API
Unmanaged extensions
Clients
Neo4j remote REST client APIs
3
4
5
HTTP clients

# Python


200 OK




Graph Data Types

Graph Databases

Batch & Manual Indexing

Calendar subgraph

neokit – Command Line Toolkit for Neo4

Object-Graph Mapping

https://github.com/nigelsmall/py2neo

# PEP 249

# Python Database API Specification v2.0

https://www.python.org/dev/peps/pep-0249/

# Python DB API 2.0 for Neo4j

```
class Connection(object):
    def __init__(self, db_uri):
        self.errorhandler = default_error_handler
        self._host = urlparse(db_uri).netloc
        self._http =
http.HTTPConnection(self._host)
        self._tx = TX_ENDPOINT
        self._messages = []
        self._cursors = set()
        self._cursor_ids = 0
```

https://github.com/jakewins/neo4jdb-python

# neo4jdb-python

```
>>> from neo4j.contextmanager import
Neo4jDBConnectionManager
>>>
>>>
>>> neo_url = 'http://localhost:7474'
>>> manager = Neo4jDBConnectionManager(neo_url)
>>> with manager.write() as w:
...     w.execute("CREATE (TheMatrix:Movie
{title:'The Matrix'})")
...
```

https://github.com/jakewins/neo4jdb-python

neomodel

Object Graph Mapper

py2neo


```
class FriendRel(StructuredRel):
    since = DateTimeProperty(default=lambda:
datetime.now(pytz.utc))
    met = StringProperty()

class Person(StructuredNode):
    name = StringProperty()
    friends = RelationshipTo('Person', 'FRIEND',
model=FriendRel)

rel = jim.friend.connect(bob)
rel.since # datetime object
```

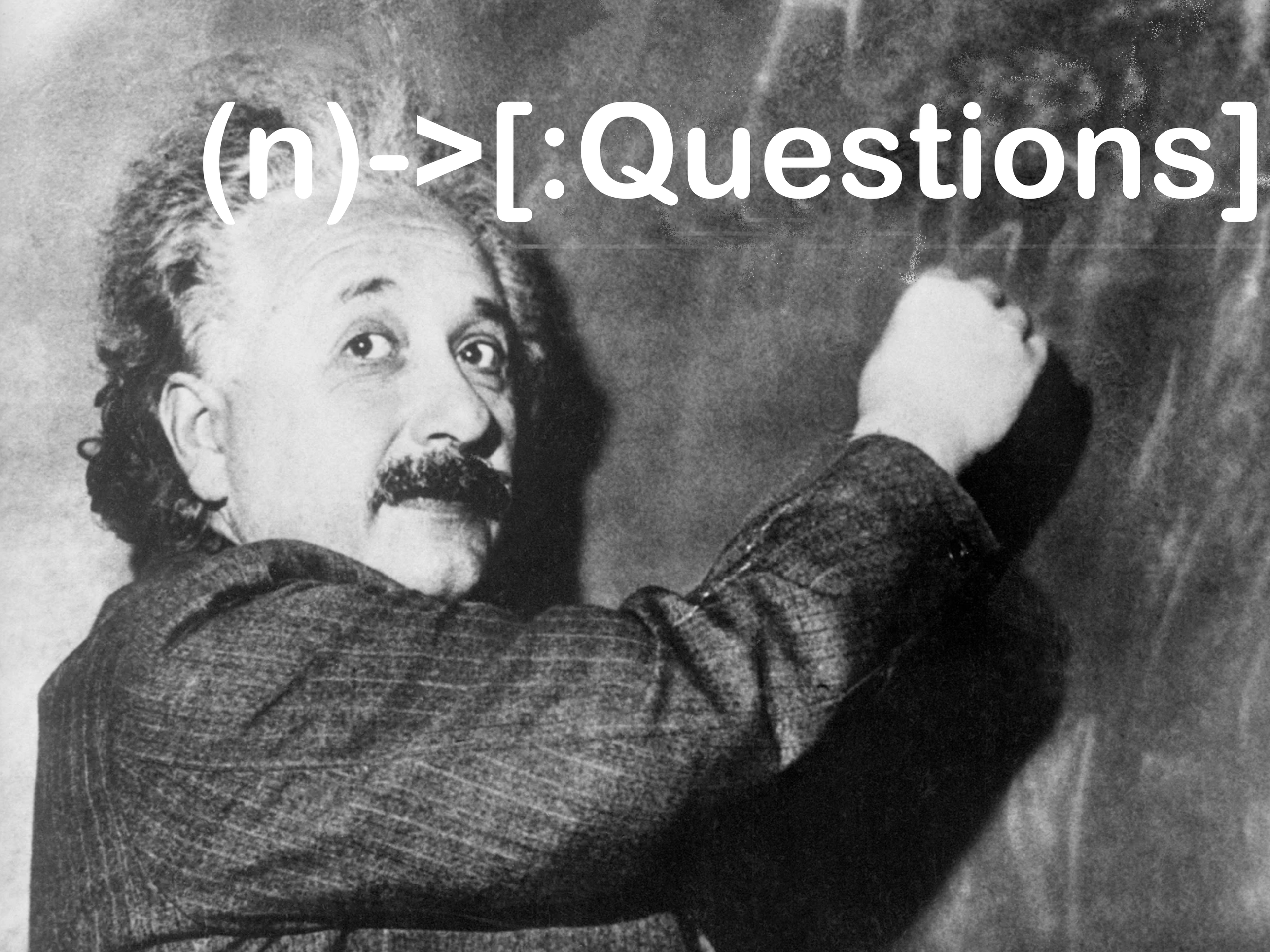
| | Async | Python 3 | OGM |
|---|---|---|---|
| neomodel | NO | NO | YES |
| py2neo | NO | YES | YES |
| neo4jdb-python | NO | NO | NO |

| | Async | Python 3 | OGM |
|---|---|---|---|
| aio-libs + py2neo | YES | YES | YES |

# Slides

https://asoldatenko.com/pyconru2016.pdf

# (n)->[:Questions]



# Thank You


@a_soldatenko

andrii.soldatenko@gmail.com