Introduction to neo4j

Slide Source: neo4j

https://go.neo4j.com/rs/710-RRC-335/images/Graph_Databases_for_Beginners.pdf

Neo4j is the creator of

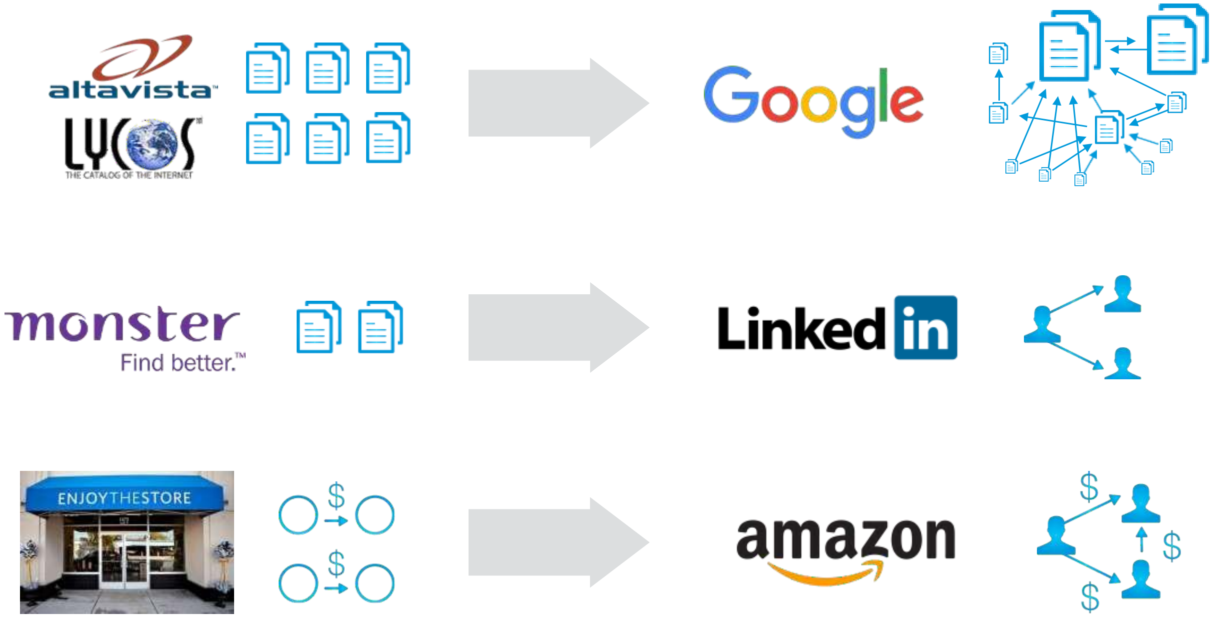
a highly scalable, ***native*** graph database.

Neo4j gives any organization the ability to ***leverage connections in data — in real-time***
to create value

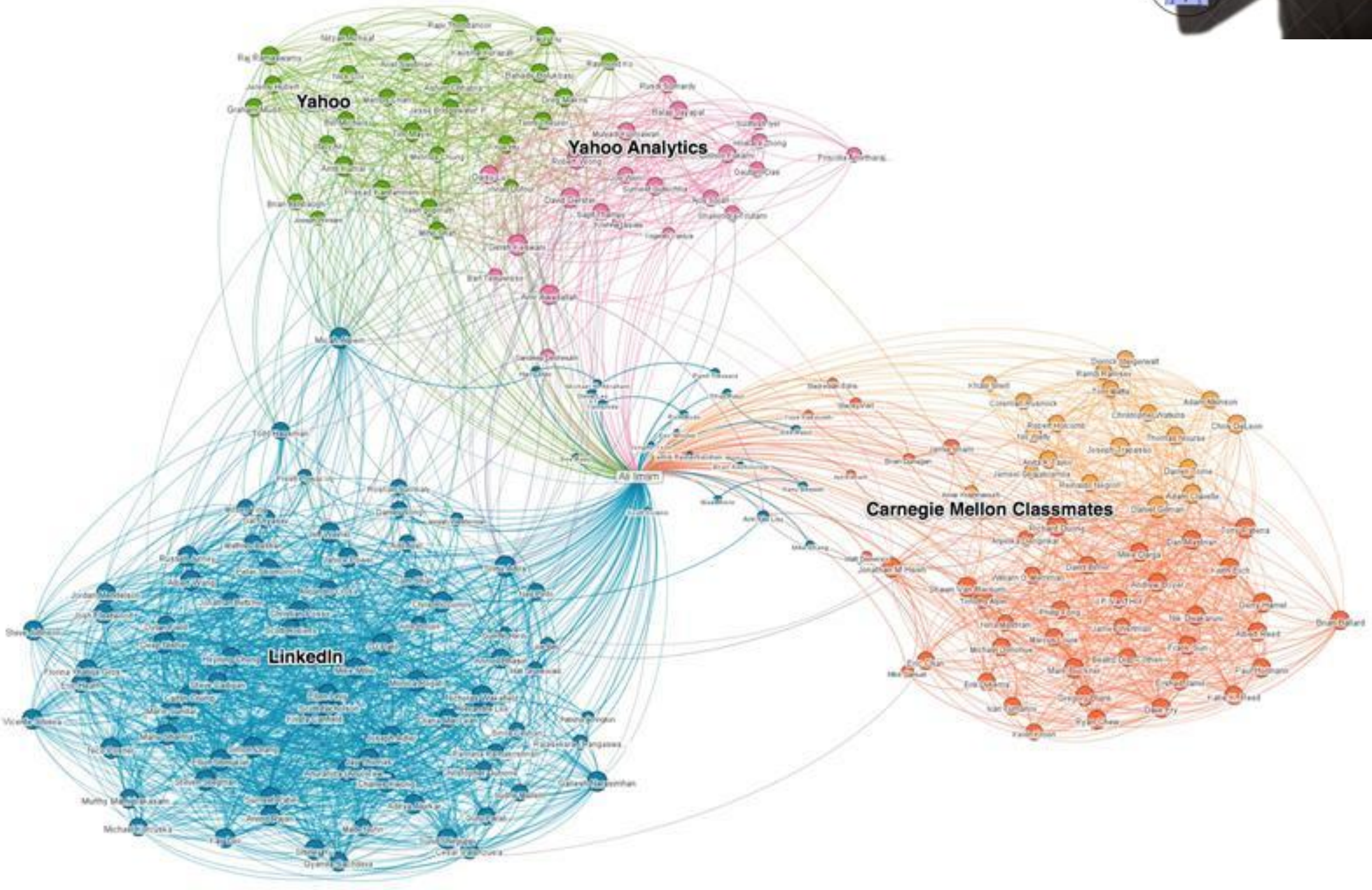
Our core belief is — *connections between data are as important as the data itself*

neo4j

Use of data connections has created industry leaders



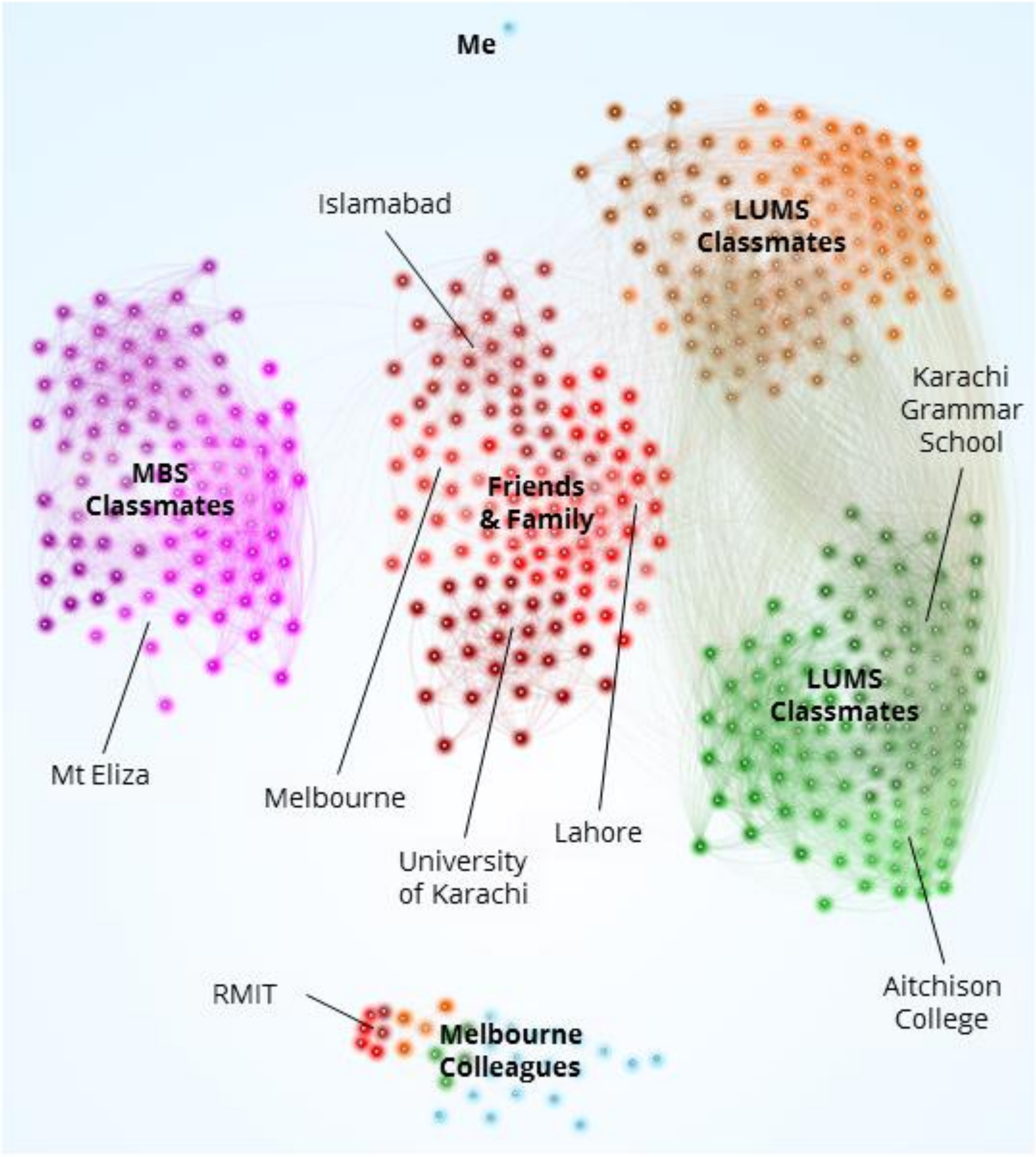
Linked Connections



Today, as processes get digitized and interconnected - we see the emergence of a *“connected enterprise”*

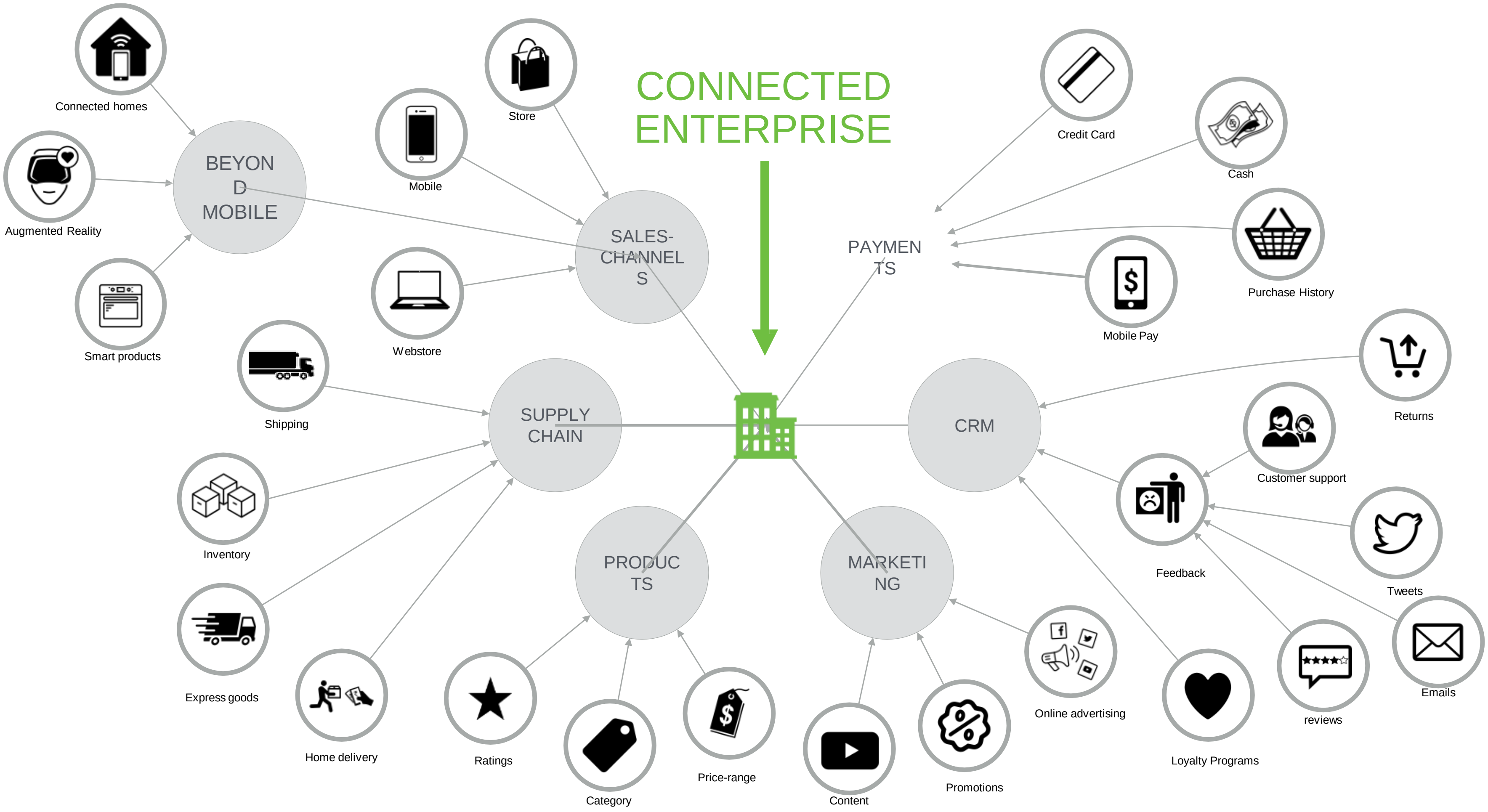
My Facebook Friends Network

18 July 2013



Source: **Friends Visual Map on Facebook**

CONNECTED ENTERPRISE



BEYOND MOBILE

Connected homes

Augmented Reality

Smart products

SALES-CHANNELS

Store

Mobile

Webstore

PAYMENTS

Credit Card

Cash

Purchase History

Mobile Pay

SUPPLY CHAIN

Shipping

Inventory

Express goods

Home delivery

CRM

Returns

Customer support

Feedback

Tweets

Emails

reviews

Loyalty Programs

PRODUCTS

Ratings

Category

Price-range

MARKETING

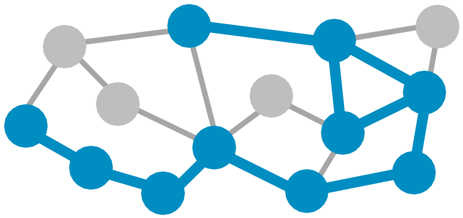
Content

Promotions

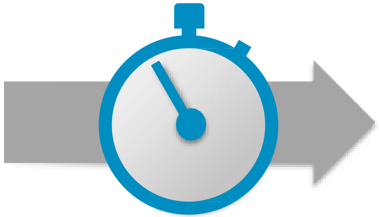
Online advertising

Relationship Queries Strain Traditional Databases

A single query can
touch a ***lot of data***



Queries can take non-sequential,
arbitrary paths through data

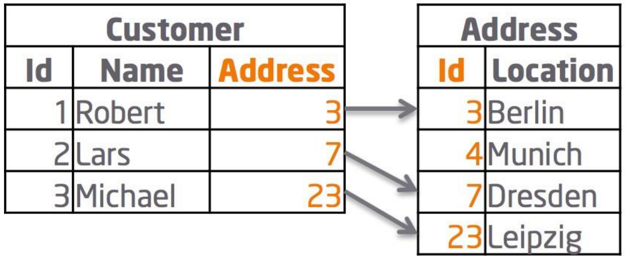


Real-time queries need speed and
consistent response times

Queries must ***run reliably***
with ***consistent results***

What you (probably) already know:

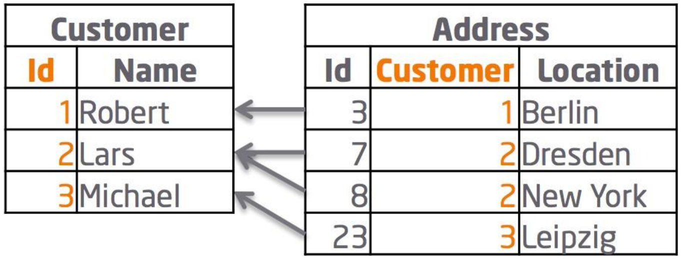
SQL Join Hell (1)



Customer		
Id	Name	Address
1	Robert	3
2	Lars	7
3	Michael	23

Address	
Id	Location
3	Berlin
4	Munich
7	Dresden
23	Leipzig

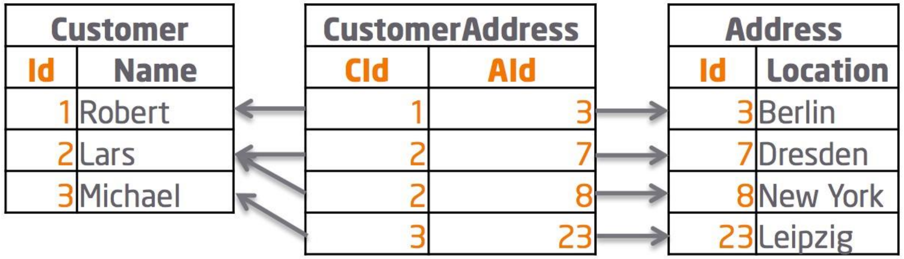
1:1 Relationship



Customer	
Id	Name
1	Robert
2	Lars
3	Michael

Address		
Id	Customer	Location
3	1	Berlin
7	2	Dresden
8	2	New York
23	3	Leipzig

1:n Relationship



Customer	
Id	Name
1	Robert
2	Lars
3	Michael

CustomerAddress	
CId	AId
1	3
2	7
2	8
3	23

Address	
Id	Location
3	Berlin
7	Dresden
8	New York
23	Leipzig

m:n Relationship

The Problem

1 Joins are executed **every time** you query the relationship

2 Executing a Join means to **search** for a key

3 **B-Tree Index: O(log(n))**
Your data grows by 10x, your time goes up by one step on each Join

4 More Data = More Searches
Slower Performance

Relational Databases can’t handle Relationships

1 **Wrong Model**

They cannot model or store relationships without complexity

2 **Degraded Performance**

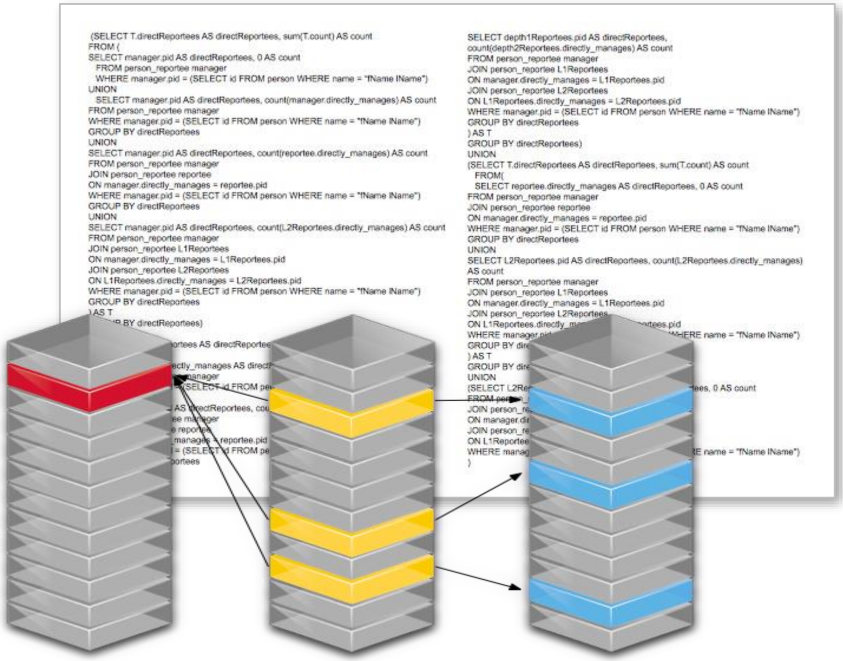
Speed plummets as data grows and as the number of joins grows

3 **Wrong Language**

SQL was built with Set Theory in mind, not Graph Theory

4 **Not Flexible**

New types of data and relationships require schema redesign

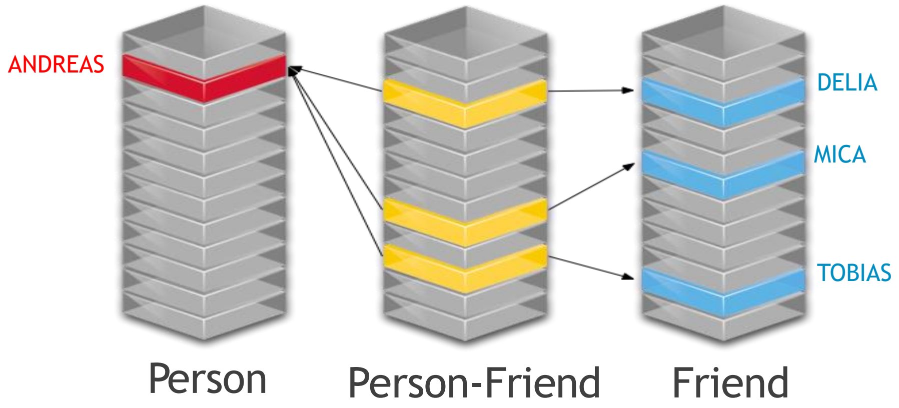


```
 (SELECT T.directReportees AS directReportees, sum(T.count) AS count
FROM (
SELECT manager.pid AS directReportees, 0 AS count
  FROM person_reportee manager
  WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
UNION
  SELECT manager.pid AS directReportees, count(manager.directly_manages) AS count
FROM person_reportee manager
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
SELECT manager.pid AS directReportees, count(reportee.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee reportee
ON manager.directly_manages = reportee.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
SELECT manager.pid AS directReportees, count(L2Reportees.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON manager.directly_manages = L1Reportees.pid
JOIN person_reportee L2Reportees
ON L1Reportees.directly_manages = L2Reportees.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
) AS T
 P BY directReportees)
            ortees AS directReportee
            ectly_manages AS direct
              nanager
            (SELECT id FROM pe
             AS directReportees, cou
            ee manager
            e reportee
            _manages = reportee.pid
            = (SELECT id FROM pe
            portees
```

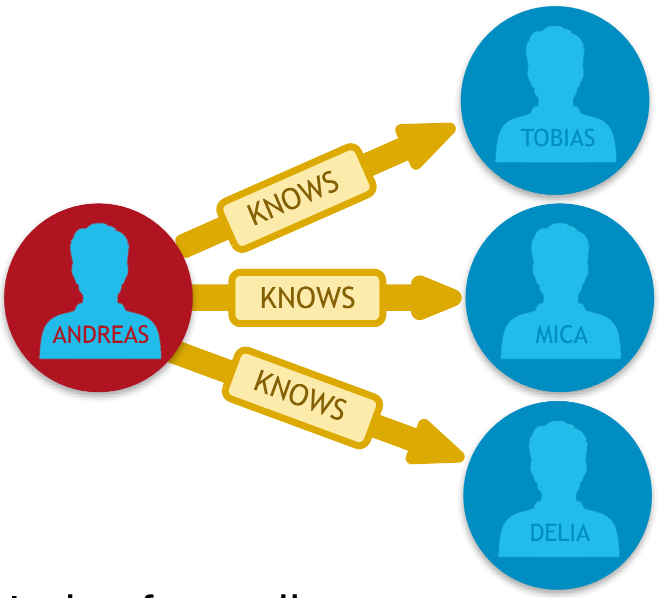
```
SELECT depth1Reportees.pid AS directReportees,
count(depth2Reportees.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON manager.directly_manages = L1Reportees.pid
JOIN person_reportee L2Reportees
ON L1Reportees.directly_manages = L2Reportees.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
) AS T
GROUP BY directReportees)
UNION
(SELECT T.directReportees AS directReportees, sum(T.count) AS count
  FROM(
  SELECT reportee.directly_manages AS directReportees, 0 AS count
FROM person_reportee manager
JOIN person_reportee reportee
ON manager.directly_manages = reportee.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
SELECT L2Reportees.pid AS directReportees, count(L2Reportees.directly_manages)
AS count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON manager.directly_manages = L1Reportees.pid
JOIN person_reportee L2Reportees
ON L1Reportees.directly_m          ortees.pid
WHERE manager.pi                 HERE name = "fName lName")
GROUP BY dire
) AS T
GROUP BY dire
UNION
(SELECT L2Re                       tees, 0 AS count
FROM person_
JOIN person_re
ON manager.di
JOIN person_re
ON L1Reportee
WHERE manag                        E name = "fName lName")
)
```

Relational Versus Graph Models

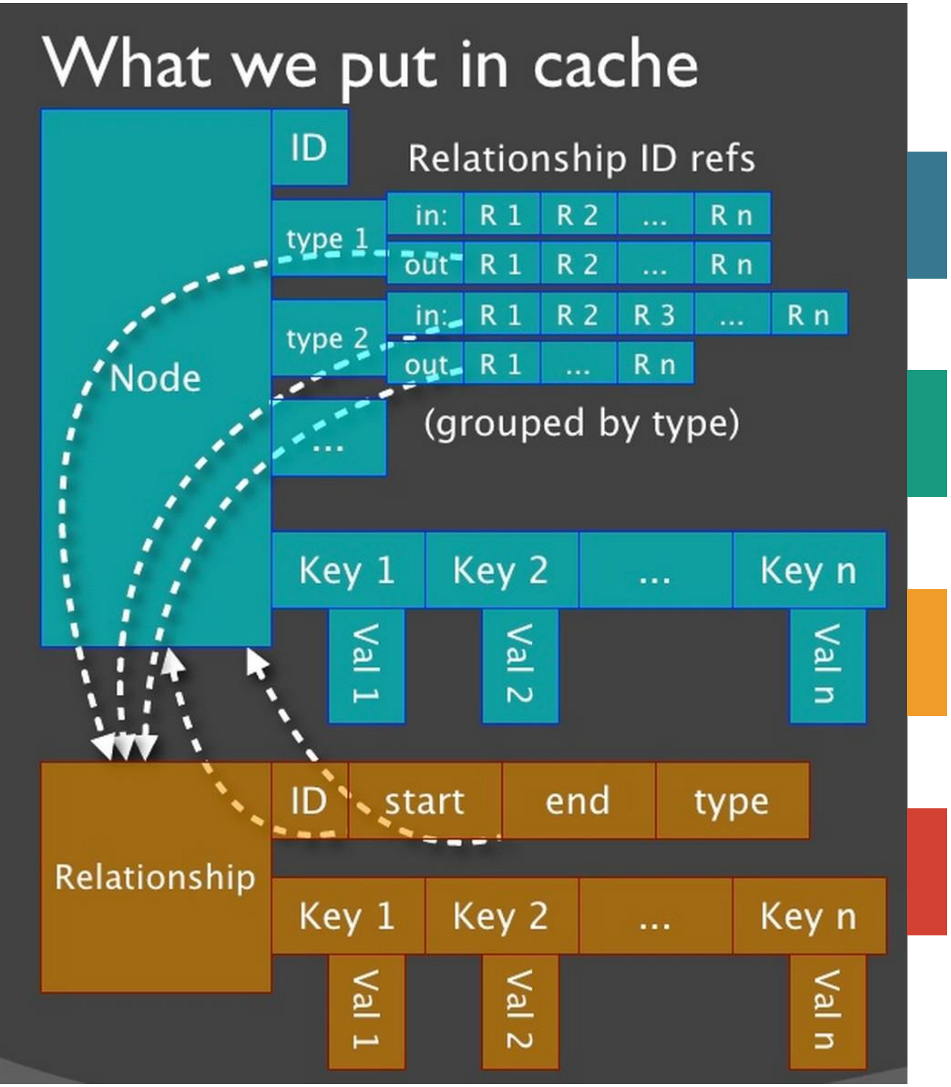
Relational Model



Graph Model



Index free adjacency



1 **Pointers instead of Lookups**

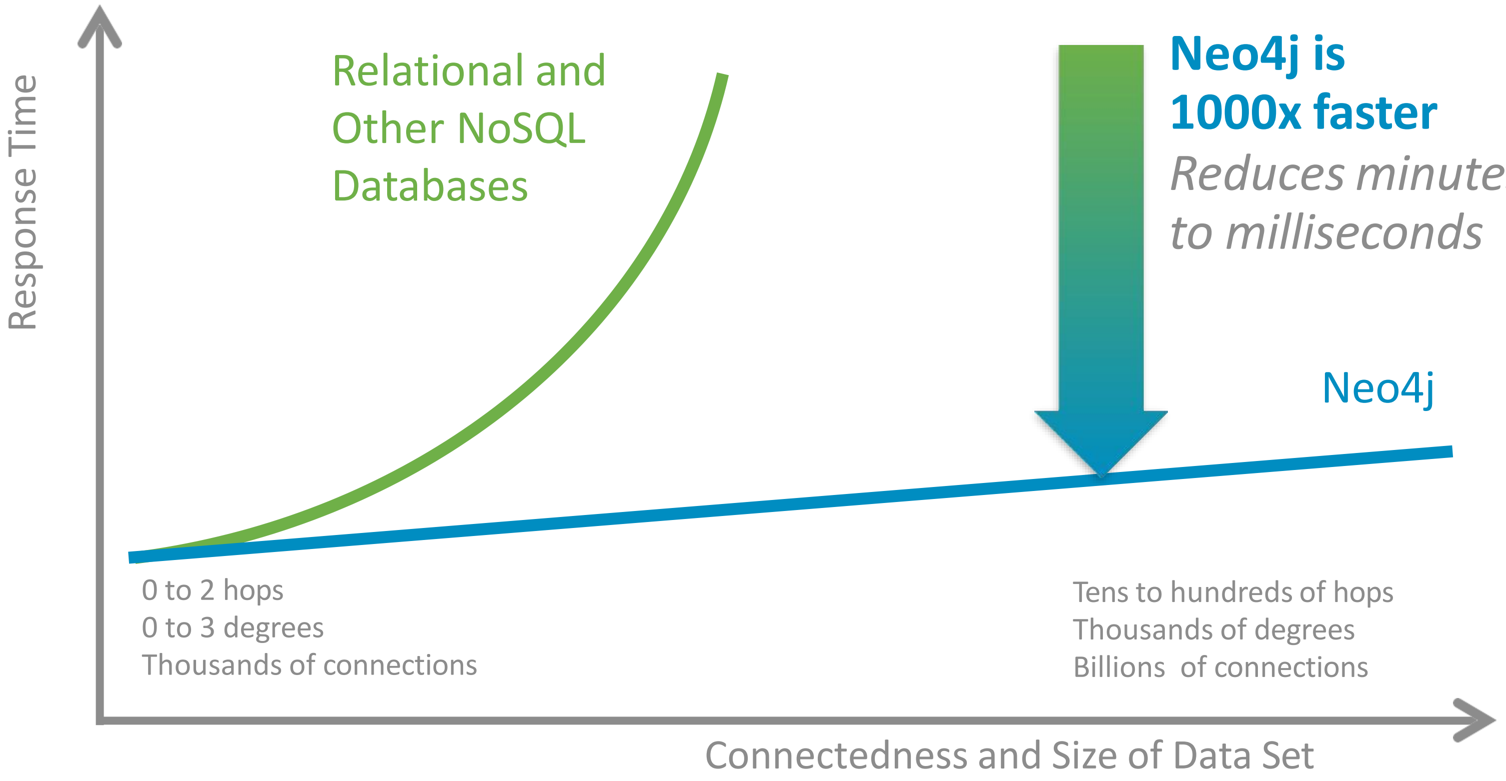
2 **Fixed Sized Records**

3 **“Joins” on Creation**

4 **Spin through this data structure**

Real Time Query Performance

Remains steady as database grows



Neo4j is 1000x faster

Reduces minutes to milliseconds

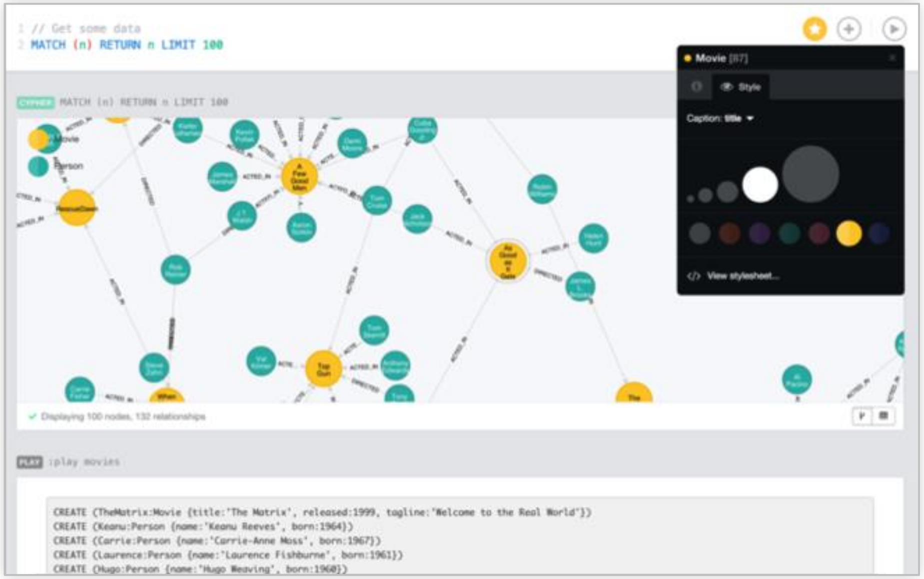
Reimagine your Data as a Graph

1 **Right Model**
Graphs simplify how you think

2 **Better Performance**
Query relationships in real time

3 **Right Language**
Cypher was purpose built for Graphs

4 **Flexible and Consistent**
Evolve your schema seamlessly while keeping transactions



```
// Get some data
MATCH (n) RETURN n LIMIT 100
```

CYPHER `MATCH (n) RETURN n LIMIT 100`

✓ Displaying 100 nodes, 132 relationships

PLAY `:play movies`

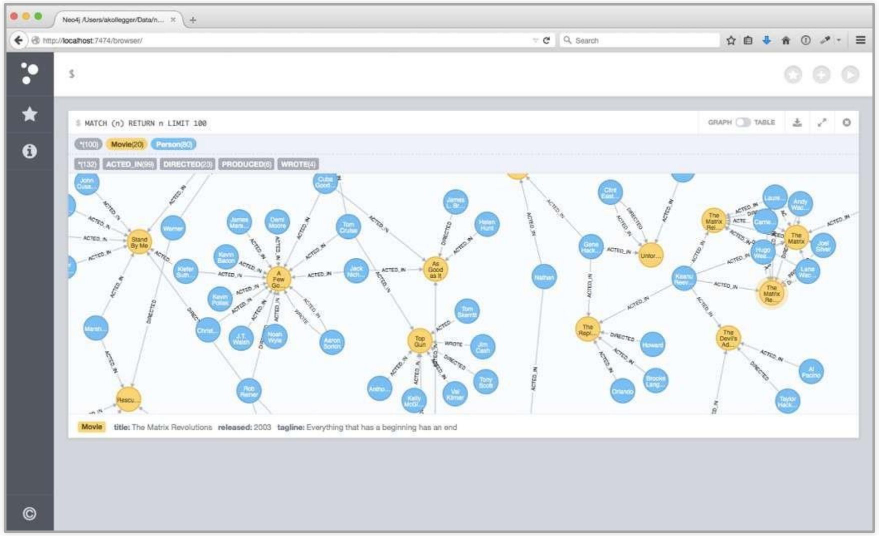
```
CREATE (TheMatrix:Movie {title:'The Matrix', released:1999, tagline:'Welcome to the Real World'})
CREATE (Keanu:Person {name:'Keanu Reeves', born:1964})
CREATE (Carrie:Person {name:'Carrie-Anne Moss', born:1967})
CREATE (Laurence:Person {name:'Laurence Fishburne', born:1961})
CREATE (Hugo:Person {name:'Hugo Weaving', born:1960})
```

Agile, High Performance and Scalable without Sacrifice

Neo4j - The #1 Database for Connected Data

Neo4j is an ***enterprise-grade native graph database*** that enables you to:

- **Store and query** data relationships
- **Traverse** any levels of depth on real-time
- **Add and connect** new data on the fly

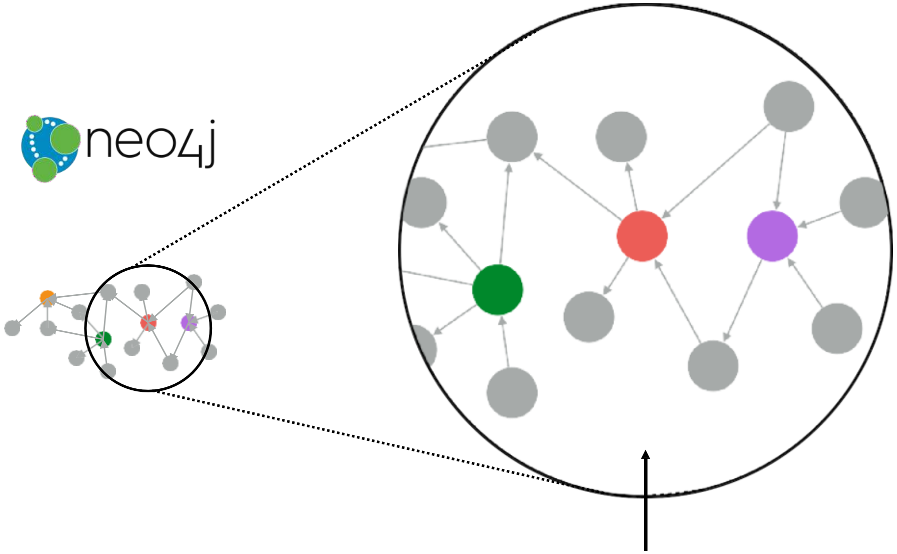


```
$ MATCH (n) RETURN n LIMIT 100
```

Movie **title:** The Matrix Revolutions **released:** 2003 **tagline:** Everything that has a beginning has an end

Designed, built and tested natively for graphs from the start to ensure:

- Performance
- ACID Transactions
- Agility
- Developer Productivity
- Hardware Efficiency



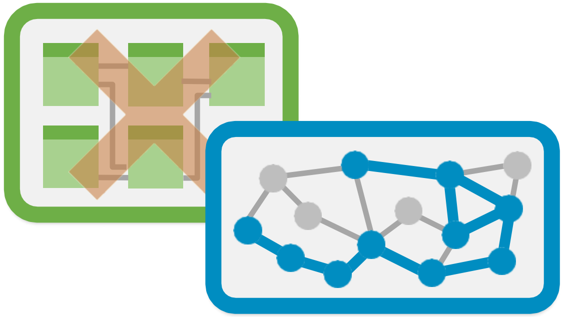
Index-free adjacency ensures lightning-fast retrieval of data and relationships

Index free adjacency:

Unlike other database models Neo4j ***connects*** data as it stores it

Neo4j Advantage - Performance

Neo4j: Native Graph from the Start



Native graph storage

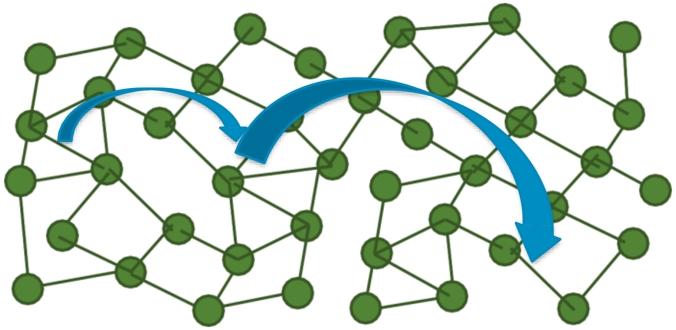
Optimized for real-time reads and ACID writes

- **Relationships stored as physical objects, eliminating need for joins and join tables**
- Nodes connected at write time, enabling scale-independent response times

Native graph querying

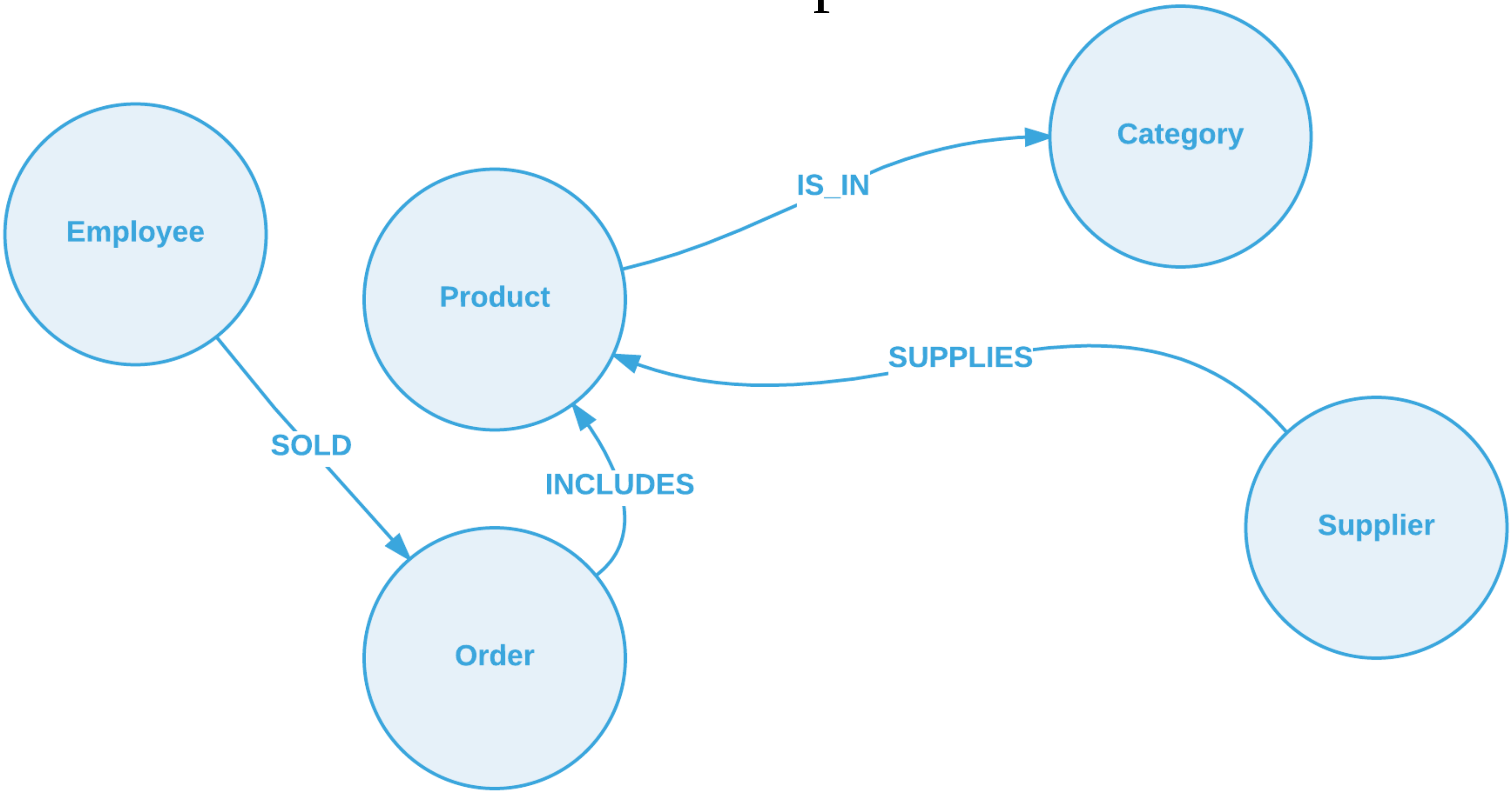
Memory structures and algorithms optimized for graphs

- Index-free adjacency enables 1M+ hops per second via in-memory pointer chasing
- Off-heap page cache improves operational robustness and scaling compared with JVM-based caches
- “Minutes to milliseconds” performance improvement



Neo4j Advantage - Performance

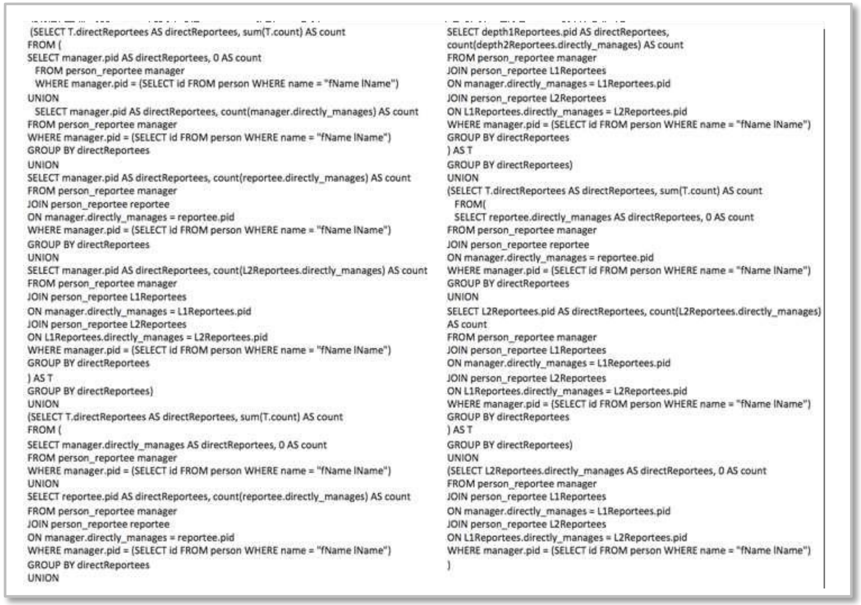
As a Graph


Category
IS_IN
Employee
Product
SUPPLIES
SOLD
INCLUDES
Supplier
Order

Express Complex Relationship Queries Easily with Cypher

Find all direct reports and how
many people they manage,
up to three levels down

SQL Query



```
 (SELECT T.directReportees AS directReportees, sum(T.count) AS count
FROM (
SELECT manager.pid AS directReportees, 0 AS count
  FROM person_reportee manager
  WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
UNION
  SELECT manager.pid AS directReportees, count(manager.directly_manages) AS count
FROM person_reportee manager
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
SELECT manager.pid AS directReportees, count(reportee.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee reportee
ON manager.directly_manages = reportee.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
SELECT manager.pid AS directReportees, count(L2Reportees.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON manager.directly_manages = L1Reportees.pid
JOIN person_reportee L2Reportees
ON L1Reportees.directly_manages = L2Reportees.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
) AS T
GROUP BY directReportees)
UNION
(SELECT T.directReportees AS directReportees, sum(T.count) AS count
FROM (
SELECT manager.directly_manages AS directReportees, 0 AS count
FROM person_reportee manager
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
UNION
SELECT reportee.pid AS directReportees, count(reportee.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee reportee
ON manager.directly_manages = reportee.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
SELECT depth1Reportees.pid AS directReportees,
count(depth2Reportees.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON manager.directly_manages = L1Reportees.pid
JOIN person_reportee L2Reportees
ON L1Reportees.directly_manages = L2Reportees.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
) AS T
GROUP BY directReportees)
UNION
(SELECT T.directReportees AS directReportees, sum(T.count) AS count
  FROM(
  SELECT reportee.directly_manages AS directReportees, 0 AS count
FROM person_reportee manager
JOIN person_reportee reportee
ON manager.directly_manages = reportee.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
SELECT L2Reportees.pid AS directReportees, count(L2Reportees.directly_manages)
AS count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON manager.directly_manages = L1Reportees.pid
JOIN person_reportee L2Reportees
ON L1Reportees.directly_manages = L2Reportees.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
) AS T
GROUP BY directReportees)
UNION
(SELECT L2Reportees.directly_manages AS directReportees, 0 AS count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON manager.directly_manages = L1Reportees.pid
JOIN person_reportee L2Reportees
ON L1Reportees.directly_manages = L2Reportees.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
)
```

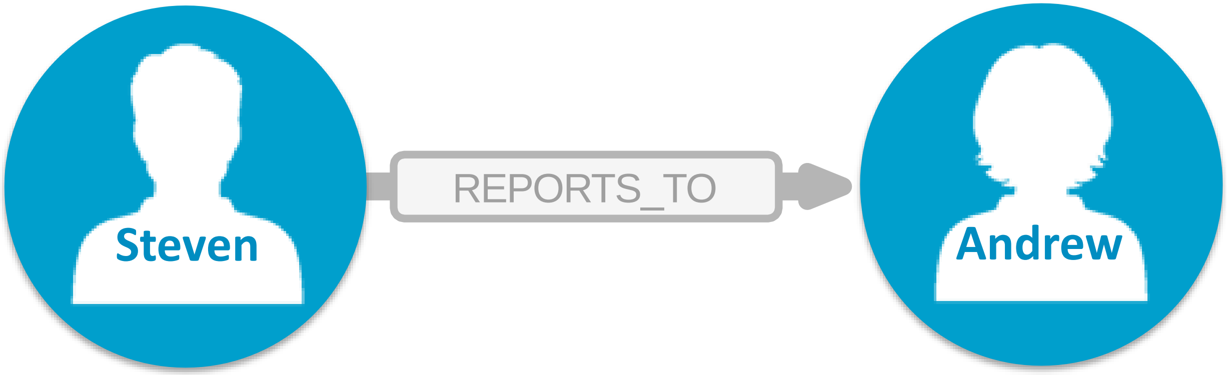
Cypher Query

```
MATCH (boss)-[:MANAGES*0..3]->(sub),
      (sub)-[:MANAGES*1..3]->(report)
WHERE boss.name = “John Doe”
RETURN sub.name AS Subordinate,
  count(report) AS Total
```

Neo4j Advantage - Developer Productivity

Graph_Databases_for_Beginners.pdf (neo4j.com)

Property Graph Model



NODE NODE

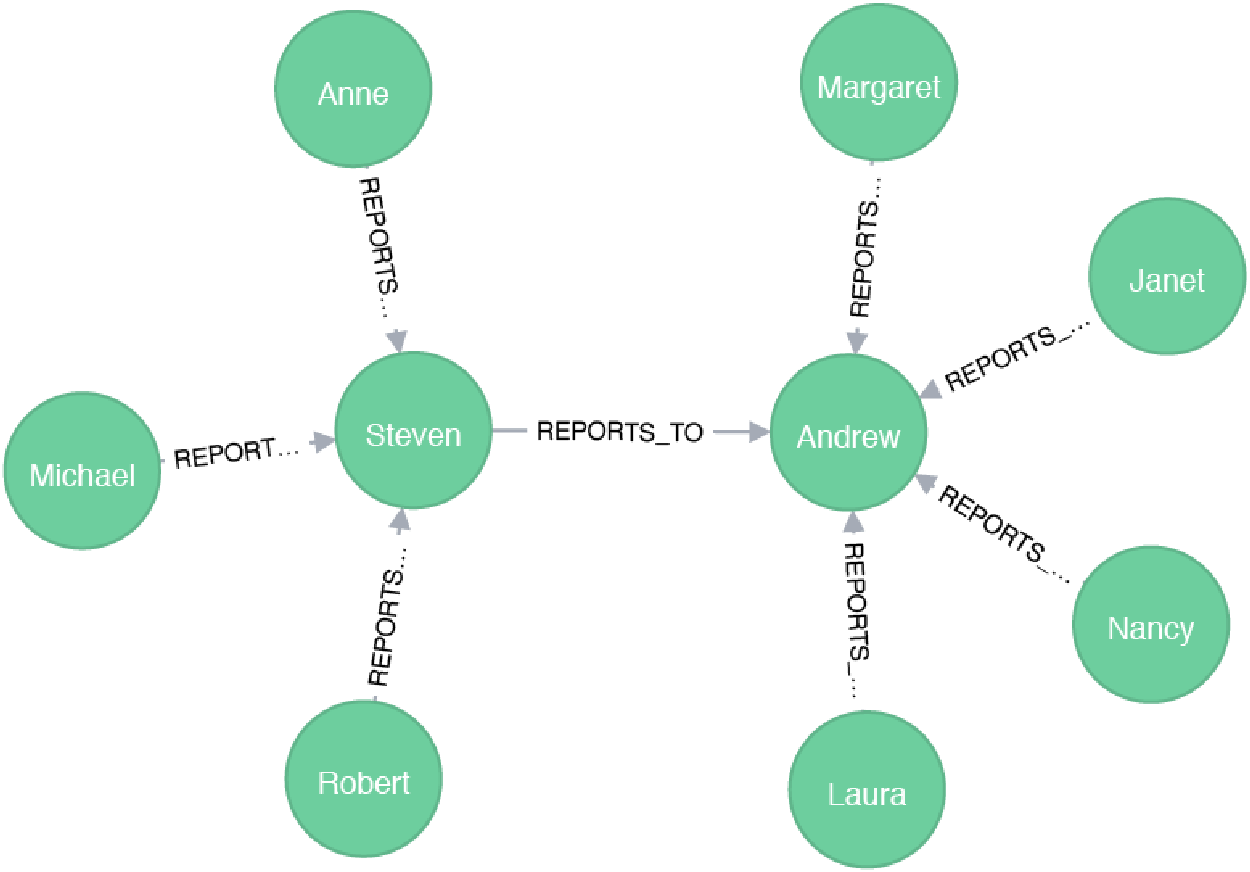
```
CREATE (:Employee{ firstName:“Steven”} ) -[:REPORTS_TO]-> (:Employee{ firstName:“Andrew”} )
```

LABEL PROPERTY LABEL PROPERTY

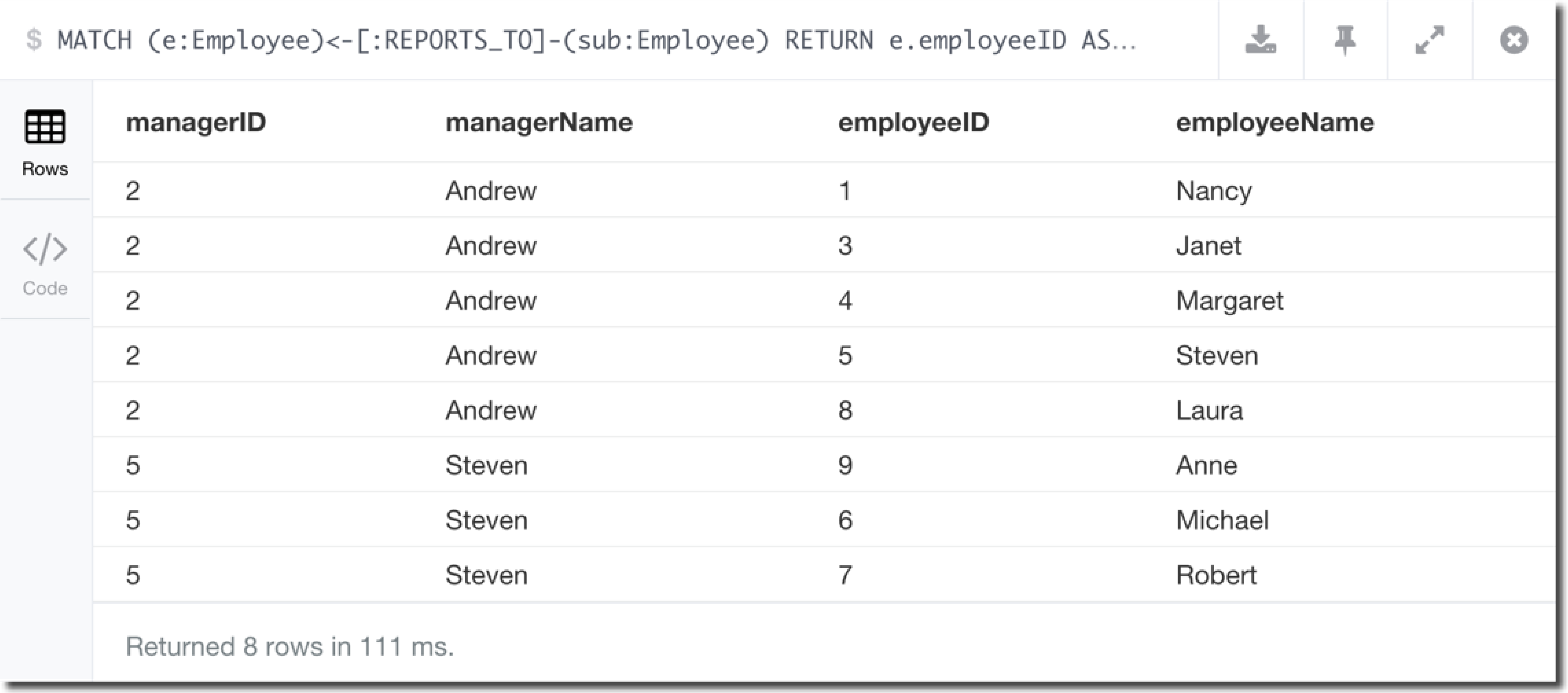
Who do people report to?

```
MATCH
 (e:Employee)<-[:REPORTS_TO]-(sub:Employee)
RETURN
   *
```

Who do people report to?



Who do people report to?



```
$ MATCH (e:Employee)<-[:REPORTS_TO]-(sub:Employee) RETURN e.employeeID AS...
```

managerID	managerName	employeeID	employeeName
2	Andrew	1	Nancy
2	Andrew	3	Janet
2	Andrew	4	Margaret
2	Andrew	5	Steven
2	Andrew	8	Laura
5	Steven	9	Anne
5	Steven	6	Michael
5	Steven	7	Robert

Returned 8 rows in 111 ms.

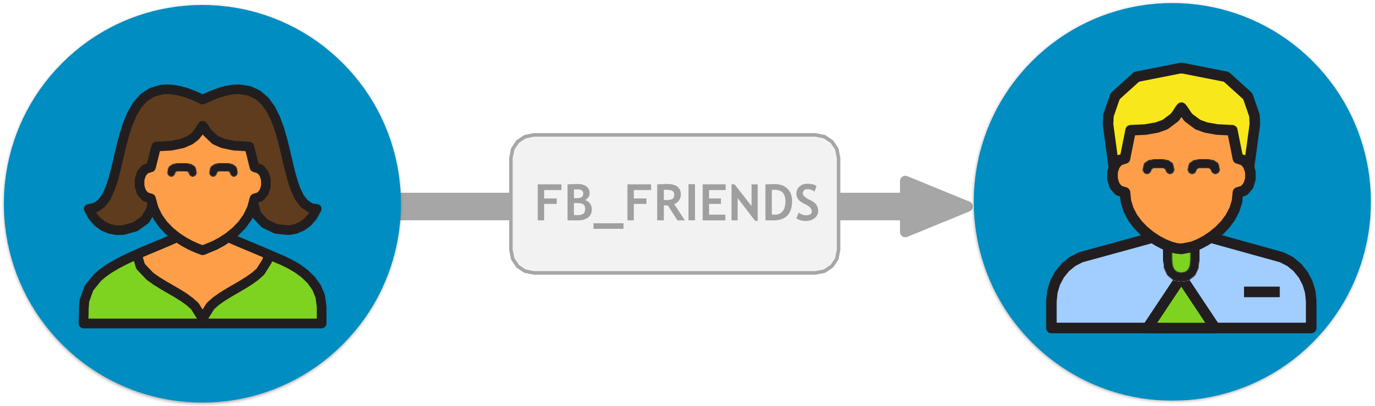
Who does Robert report to?

```
MATCH
 p=(e:Employee)<-[:REPORTS_TO]-(sub:Employee)
WHERE
  sub.firstName = ‘Robert’
RETURN
  p
```

Who do people report to?

```
MATCH
 (e:Employee)<-[:REPORTS_TO]-(sub:Employee)
RETURN
  e.employeeID AS managerID,
  e.firstName AS managerName,
  sub.employeeID AS employeeID,
  sub.firstName AS employeeName;
```

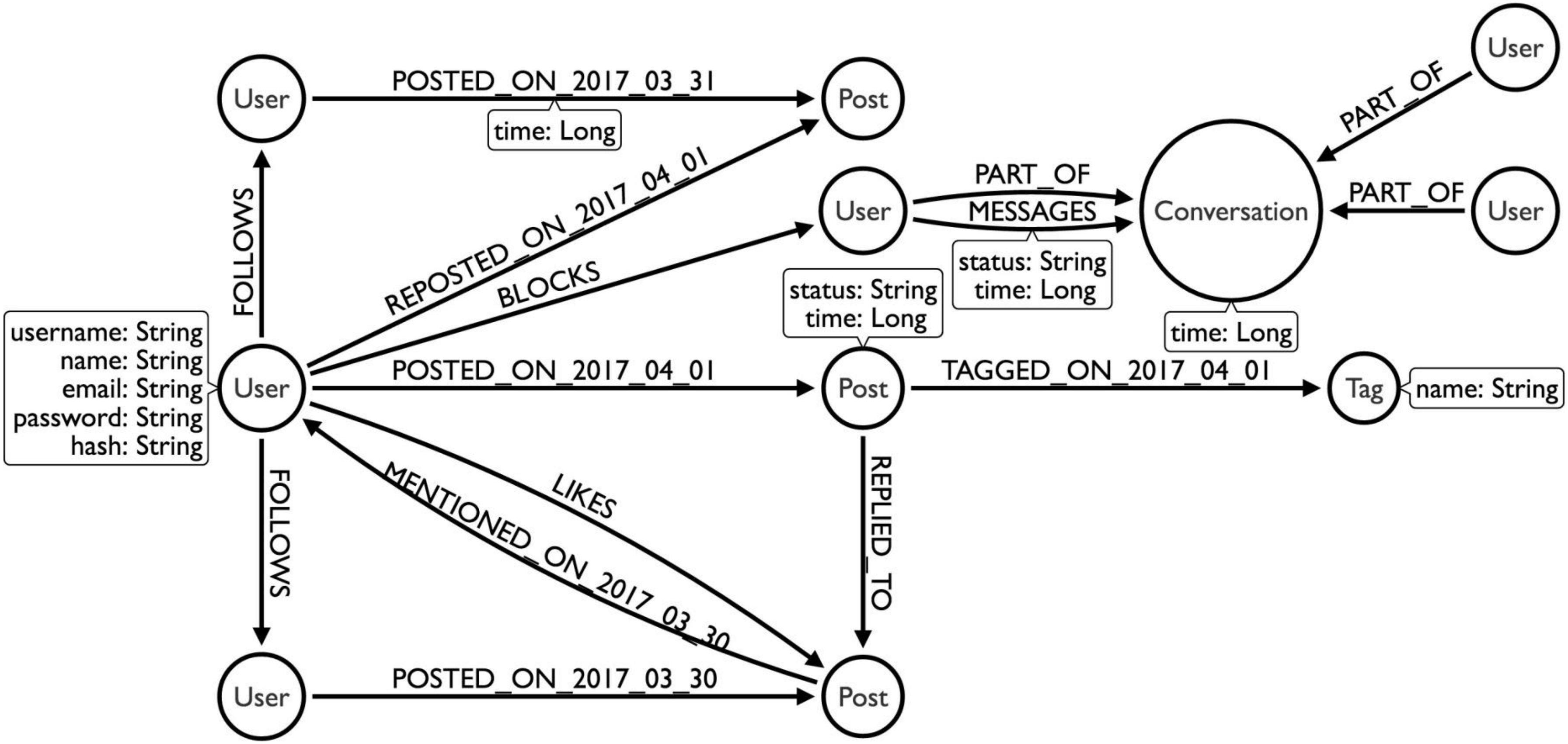
Representing Bi-Directionality



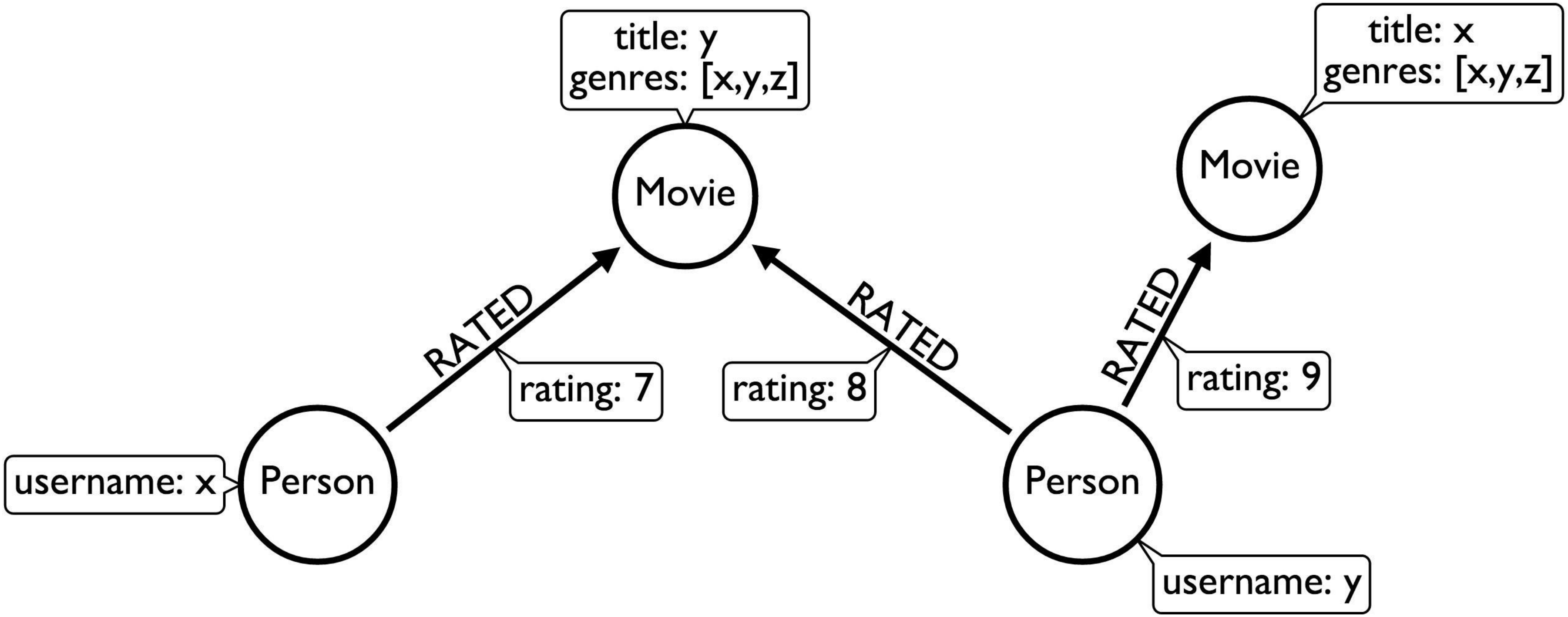
```
MATCH (:Person { name:“Ann”} ) - [:FB_FRIENDS] -> (:Person { name:“Dan”} )

MATCH (:Person { name:“Ann”} ) <- [:FB_FRIENDS] -   (:Person { name:“Dan”} )
```

Modeling a Twitter Feed

Bigger Model



Movie Data Model



Cypher Query: Movie Recommendation

What are the Top 25 Movies

- that I haven't seen
- with the same genres as Toy Story
- given high ratings
- by women under 35 who liked Toy Story

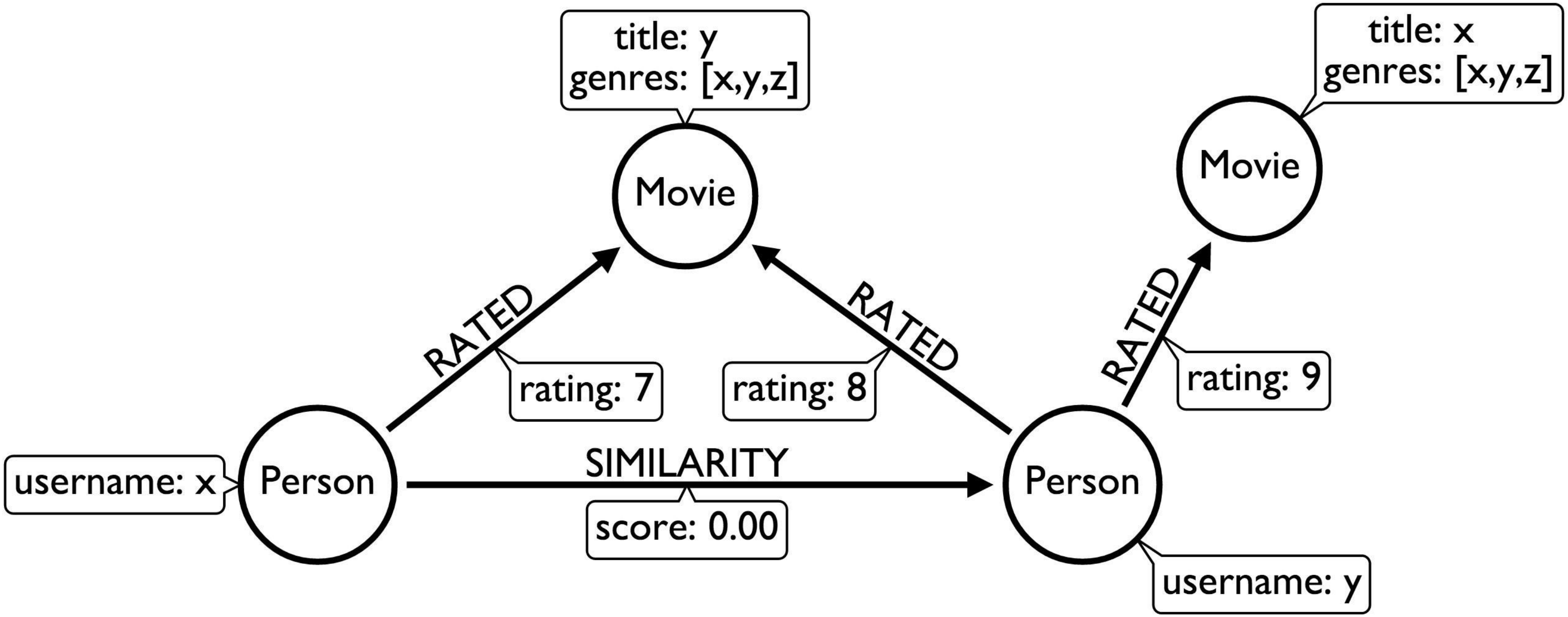
```
MATCH (watched:Movie {title:"Toy Story”}) <-[r1:RATED]- (p2) -[r2:RATED]-> (unseen:Movie)
WHERE r1.rating > 7 AND r2.rating > 7 AND p2.gender = “female” AND p2.age < 35
AND watched.genres = unseen.genres
AND NOT( (p:Person) -[:RATED|WATCHED]-> (unseen) )
AND p.username in [“maxdemarzi”,”janedoe”,”jamesdean”]
RETURN unseen.title, COUNT(*)
ORDER BY COUNT(*) DESC
LIMIT 25
```

Cypher Query: Ratings of Two Users

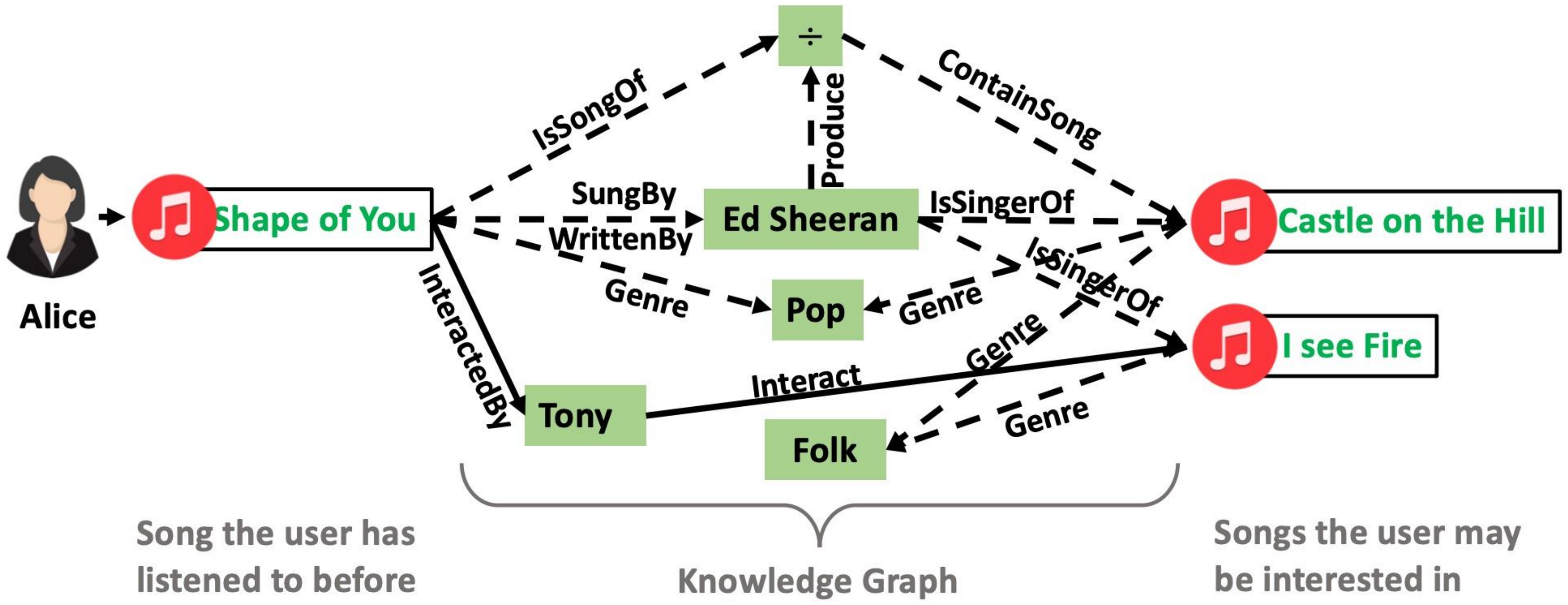
What are the Movies these 2 users have both rated

```
MATCH  (p1:Person {name:'Michael Sherman’}) -[r1:RATED]-> (m:Movie),
            (p2:Person {name:'Michael Hunger’}) -[r2:RATED]-> (m:Movie)
RETURN m.name AS Movie,
            r1.rating AS `M. Sherman's Rating`,
            r2.rating AS `M. Hunger's Rating`
```

Movie Data Model

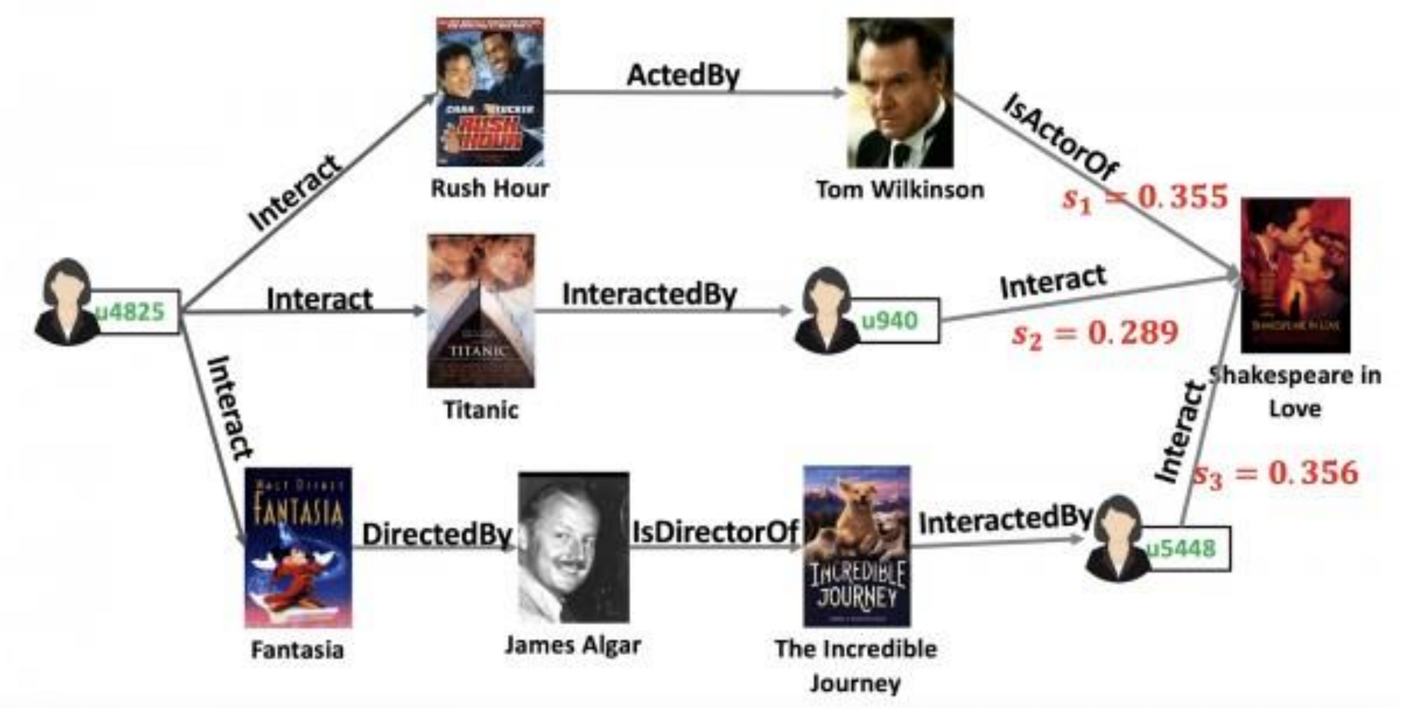


Knowledge Graphs



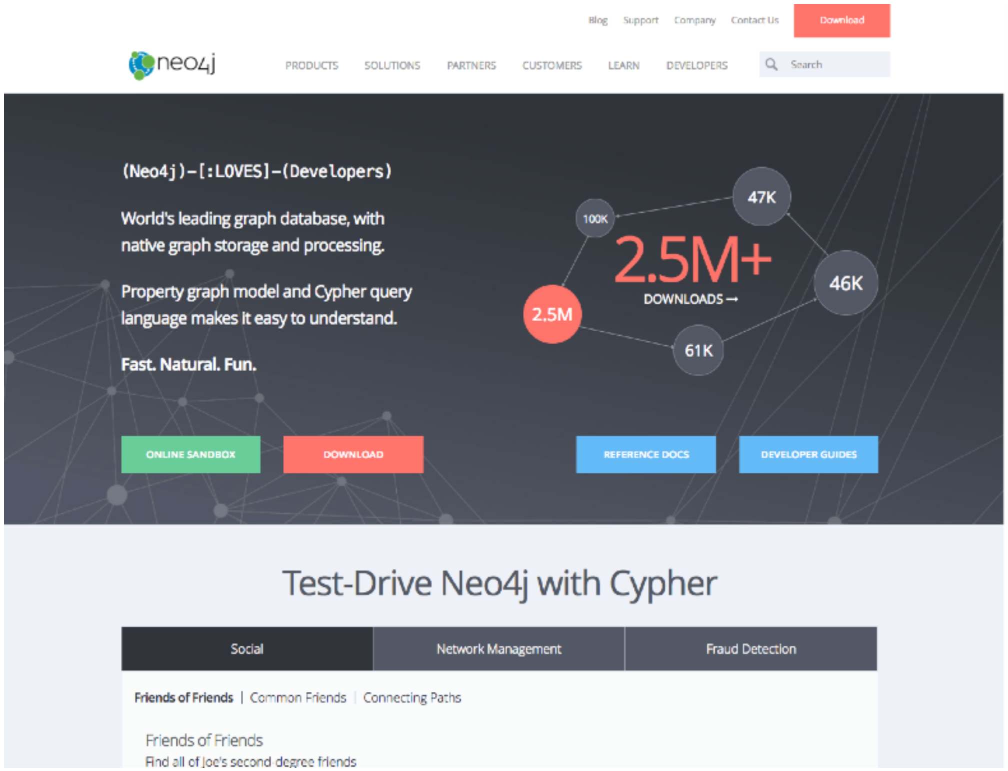
Explainable Reasoning over Knowledge Graphs for Recommendation

Knowledge Graphs



Explainable Reasoning over Knowledge Graphs for Recommendation

Neo4j Developer Page



(Neo4j)-[:LOVES]-(Developers)

World's leading graph database, with native graph storage and processing.

Property graph model and Cypher query language makes it easy to understand.

Fast. Natural. Fun.

Test-Drive Neo4j with Cypher

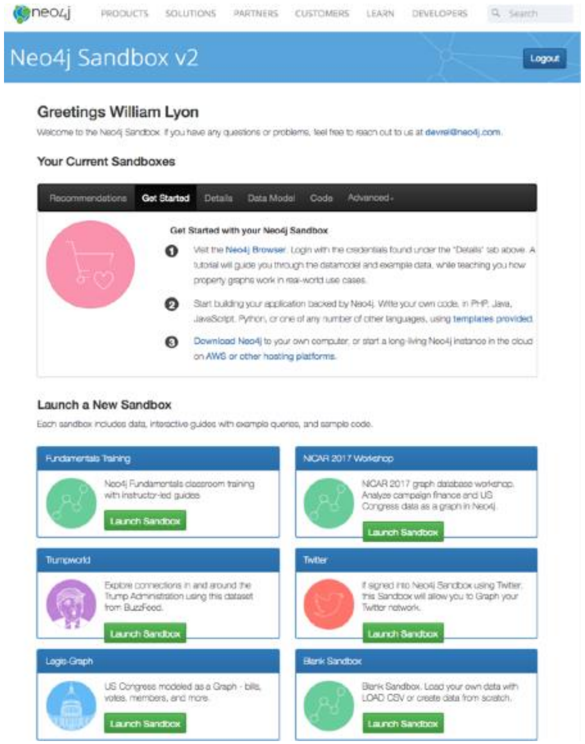
Friends of Friends

Find all of Joe's second degree friends

neo4j.com/developer

https://neo4j.com/blog/graph-of-thrones/

Neo4j Sandbox



Neo4j Sandbox v2

Greetings William Lyon

Welcome to the Neo4j Sandbox. If you have any questions or problems, feel free to reach out to us at devrel@neo4j.com.

Your Current Sandboxes

Get Started with your Neo4j Sandbox

1. Visit the Neo4j Browser. Login with the credentials found under the "Details" tab above. A tutorial will guide you through the datamodel and example data, while teaching you how property graphs work in real-world use cases.
2. Start building your application backed by Neo4j. Write your own code, in PHP, Java, JavaScript, Python, or one of any number of other languages, using templates provided.
3. Download Neo4j to your own computer, or start a long-living Neo4j instance in the cloud on AWS or other hosting platforms.

Launch a New Sandbox

Each sandbox includes data, interactive guides with example queries, and sample code.

Fundamentals Training

Neo4j Fundamentals classroom training with instructor-led guides.

NICAR 2017 Workshop

NICAR 2017 graph database workshop. Analyze campaign finance and US Congress data as a graph in Neo4j.

Trumpworld

Explore connections in and around the Trump Administration using this dataset from BuzzFeed.

Twitter

If signed into Neo4j Sandbox using Twitter, this Sandbox will allow you to Graph your Twitter network.

Legis-Graph

US Congress modeled as a Graph - bills, votes, members, and more.

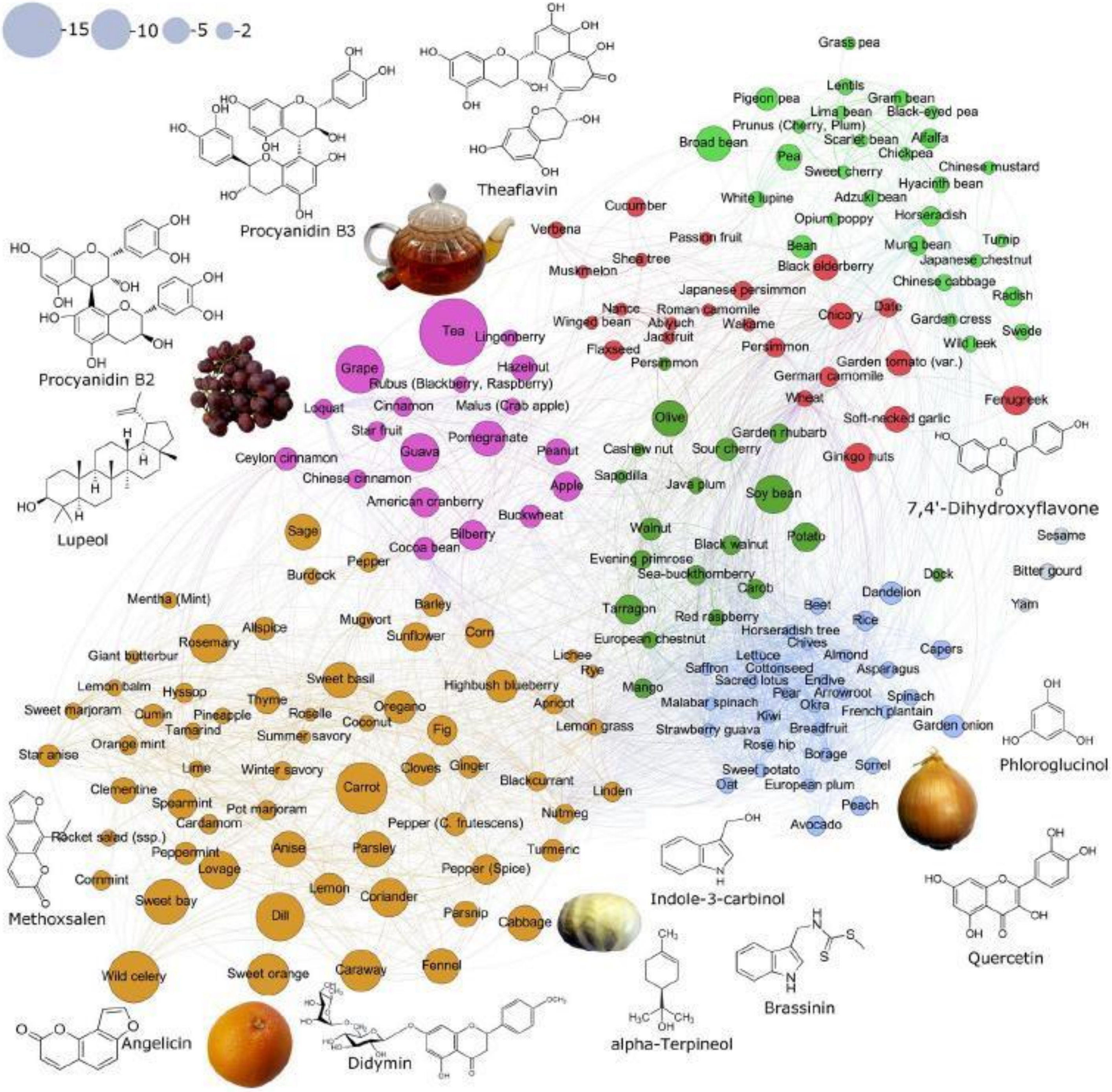
Blank Sandbox

Blank Sandbox. Load your own data with LOAD CSV or create data from scratch.

neo4jsandbox.com

Graph Machine Learning

- Machine learning on graphs is an important and ubiquitous task with applications ranging from drug design.
- The primary challenge in this domain is **finding a way to represent, or encode, graph structure so that it can be easily exploited by machine learning models**.
- https://ai.googleblog.com/2016/10/graph-powered-machine-learning-at-google.html
- GNNs (Graph Neural Networks)



15
10
5
2
Procyanidin B3
Theaflavin
Procyanidin B2
Lupeol
7,4'-Dihydroxyflavone
Phloroglucinol
Quercetin
Methoxsalen
Angelicin
Didymin
alpha-Terpineol
Brassinin
Indole-3-carbinol
Grass pea
Lentils
Pigeon pea
Gram bean
Lima bean
Black-eyed pea
Prunus (Cherry, Plum)
Broad bean
Scarlet bean
Alfalfa
Chickpea
Pea
Sweet cherry
Chinese mustard
Hyacinth bean
White lupine
Adzuki bean
Opium poppy
Horseradish
Bean
Mung bean
Turnip
Japanese chestnut
Black elderberry
Chinese cabbage
Radish
Garden cress
Swede
Wild leek
Cucumber
Verbena
Passion fruit
Shea tree
Muskmelon
Japanese persimmon
Nance
Roman camomile
Winged bean
Abiyuch
Wakame
Jackfruit
Chicory
Date
Flaxseed
Persimmon
Persimmon
Garden tomato (var.)
German camomile
Wheat
Fenugreek
Soft-necked garlic
Ginkgo nuts
Tea
Lingonberry
Grape
Hazelnut
Rubus (Blackberry, Raspberry)
Loquat
Cinnamon
Malus (Crab apple)
Star fruit
Pomegranate
Guava
Peanut
Ceylon cinnamon
Chinese cinnamon
Apple
American cranberry
Buckwheat
Bilberry
Cocoa bean
Olive
Garden rhubarb
Cashew nut
Sour cherry
Sapodilla
Java plum
Soy bean
Walnut
Potato
Black walnut
Evening primrose
Sea-buckthornberry
Carob
Tarragon
Red raspberry
European chestnut
Mango
Sesame
Bitter gourd
Dock
Yam
Dandelion
Beet
Rice
Horseradish tree
Chives
Capers
Lettuce
Almond
Saffron
Cottonseed
Asparagus
Sacred lotus
Endive
Pear
Arrowroot
Spinach
Malabar spinach
Okra
French plantain
Kiwi
Garden onion
Strawberry guava
Breadfruit
Rose hip
Borage
Sorrel
Sweet potato
Oat
European plum
Peach
Avocado
Sage
Pepper
Burdock
Mentha (Mint)
Barley
Allspice
Mugwort
Sunflower
Corn
Rosemary
Giant butterbur
Lichee
Rye
Sweet basil
Highbush blueberry
Lemon balm
Hyssop
Thyme
Apricot
Sweet marjoram
Cumin
Pineapple
Roselle
Oregano
Coconut
Tamarind
Summer savory
Fig
Lemon grass
Star anise
Orange mint
Lime
Winter savory
Cloves
Ginger
Blackcurrant
Carrot
Linden
Clementine
Spearmint
Pot marjoram
Nutmeg
Cardamom
Pepper (C. frutescens)
Rocket salad (ssp.)
Peppermint
Anise
Parsley
Turmeric
Lovage
Pepper (Spice)
Cornmint
Lemon
Coriander
Sweet bay
Dill
Parsnip
Cabbage
Wild celery
Sweet orange
Caraway
Fennel