# Welcome!

1. Find a seat. Say hi!

2. Grab a USB key, and plug it in.

3. Run the "`Install Neo4j`" application (Mac or Windows).

   When you see the README and the command prompt, you're ready.

4. Grab a coffee and get ready to begin!



neotechnology

# Neo4j Tutorial

```
(you)-[:know]->(Neo4j)
```

with Michael Hunger
@mesirii

michael@neotechnology.com

# Thanks for beta-testing

- New Neo4j Milestone 2.0.0-M06
- Shiny new Neo4j Browser
- New Windows Installer
- Please help us test it
- And provide feedback



neotechnology


# Why did you come?

# Answer five questions

- What is Neo4j?
- Why would I use it?
- How do I model data?
- How do I enter data?
- How do I query data?

and...



neo technology

provide the confidence to start building a graph enabled application



neo technology

# Schedule

8.30am - Registration, installation and breakfast
9.00am - Session 1: Graphs and Modeling
10.30am - Morning coffee break
10.45am - Session 2: Neo4j and Cypher
12.00pm - Lunch
1.00pm - Session 3: Advanced queries using Cypher
2.30pm - Afternoon coffee break
2.45pm - Session 4: Next steps and Q&A
4.00pm - End of training



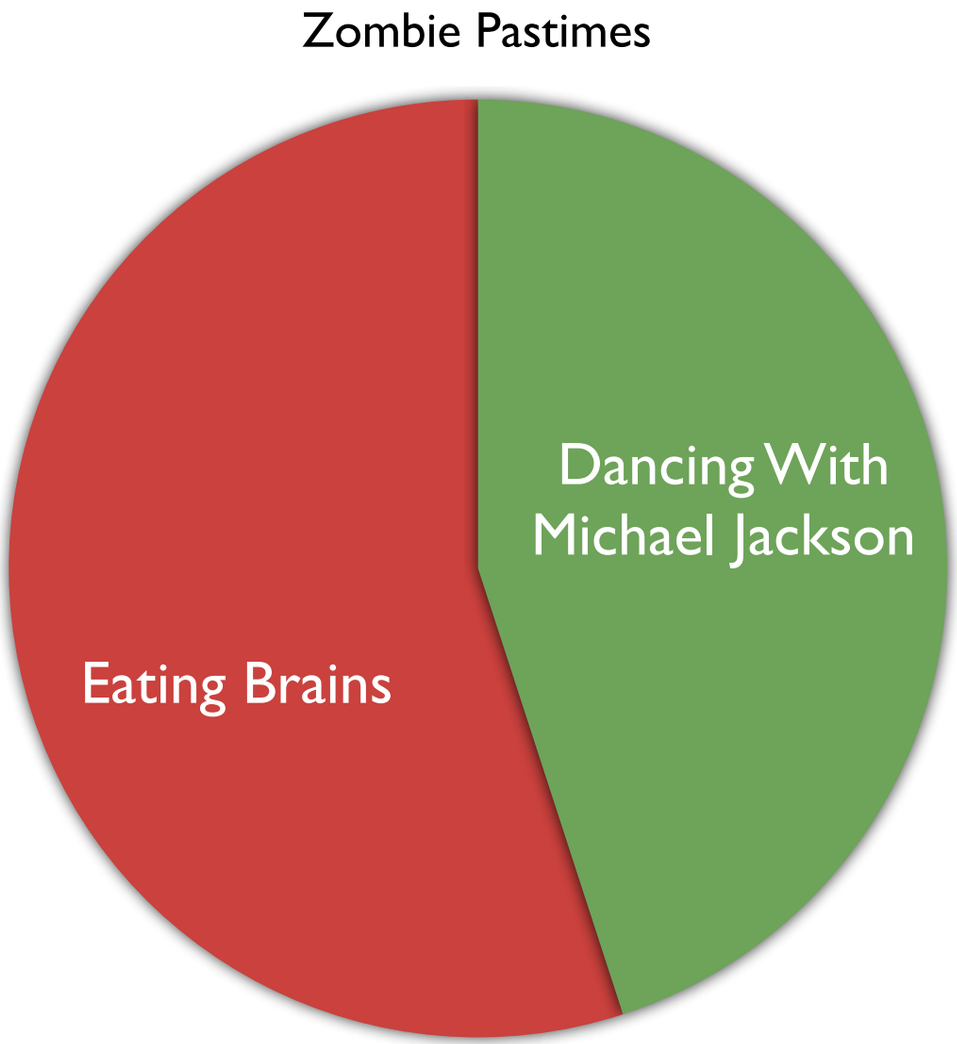
neotechnology

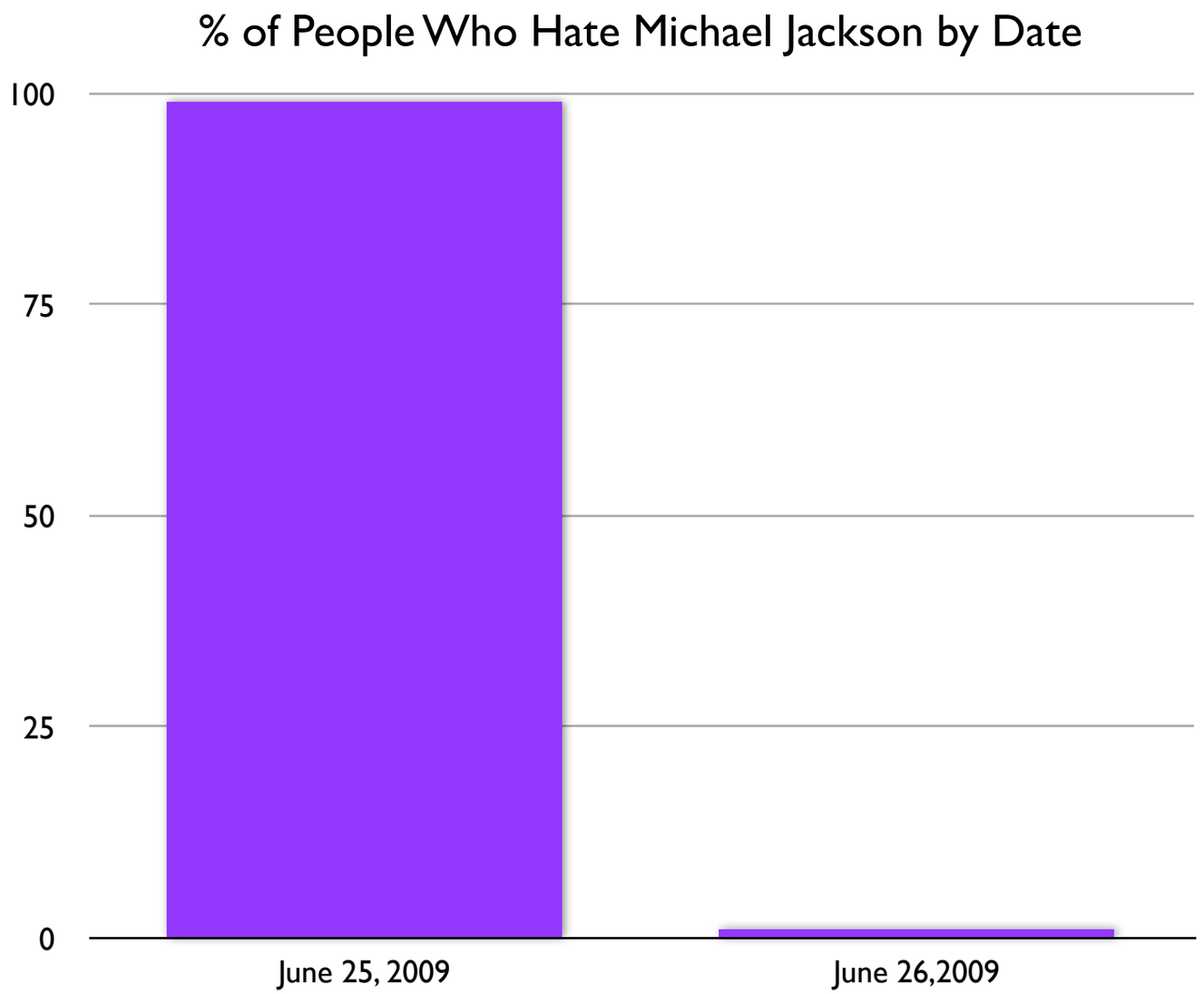

# (Introduction to Graphs)

# But first...
# an important message...



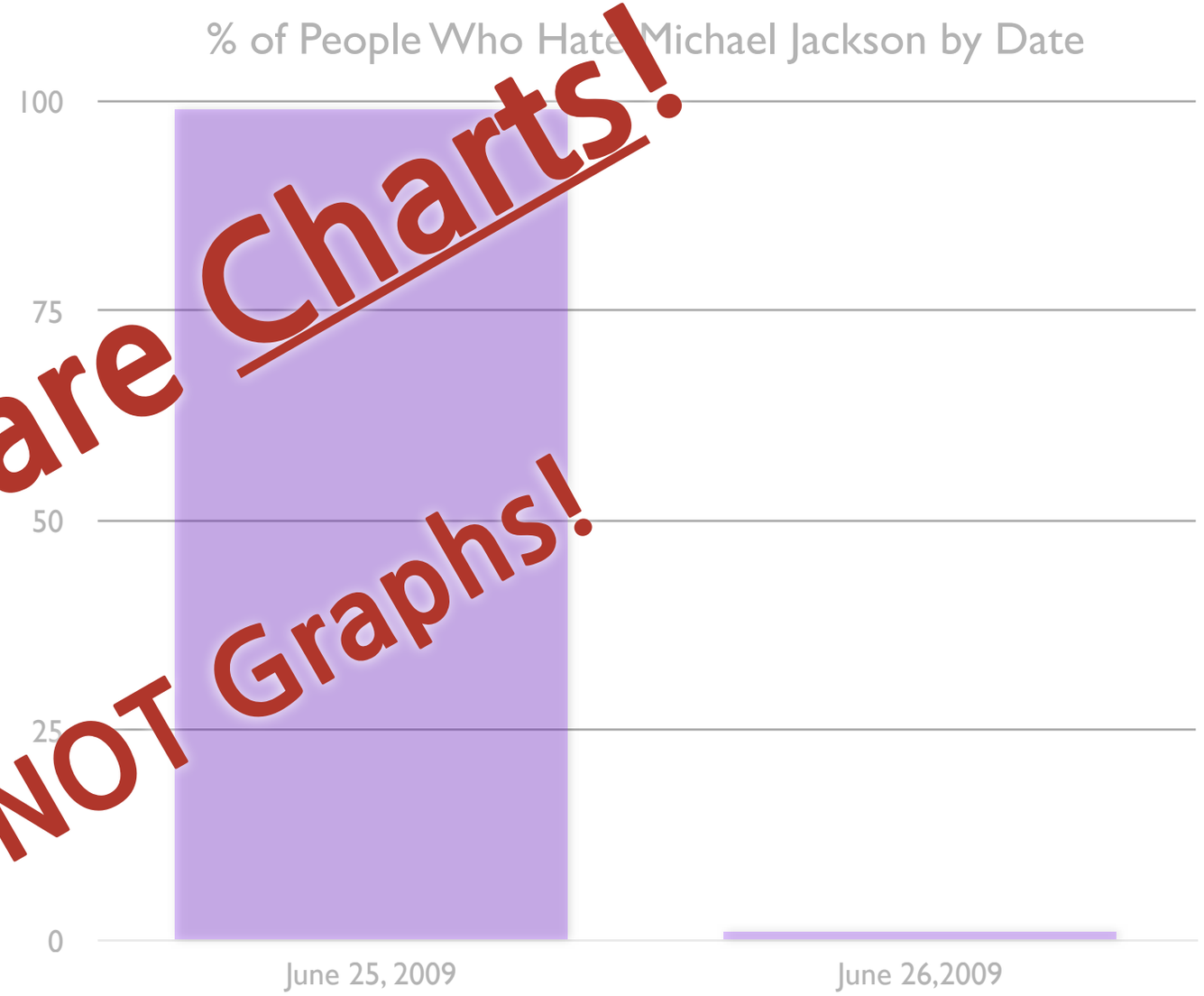
neotechnology






neotechnology


These are Charts!

NOT Graphs!



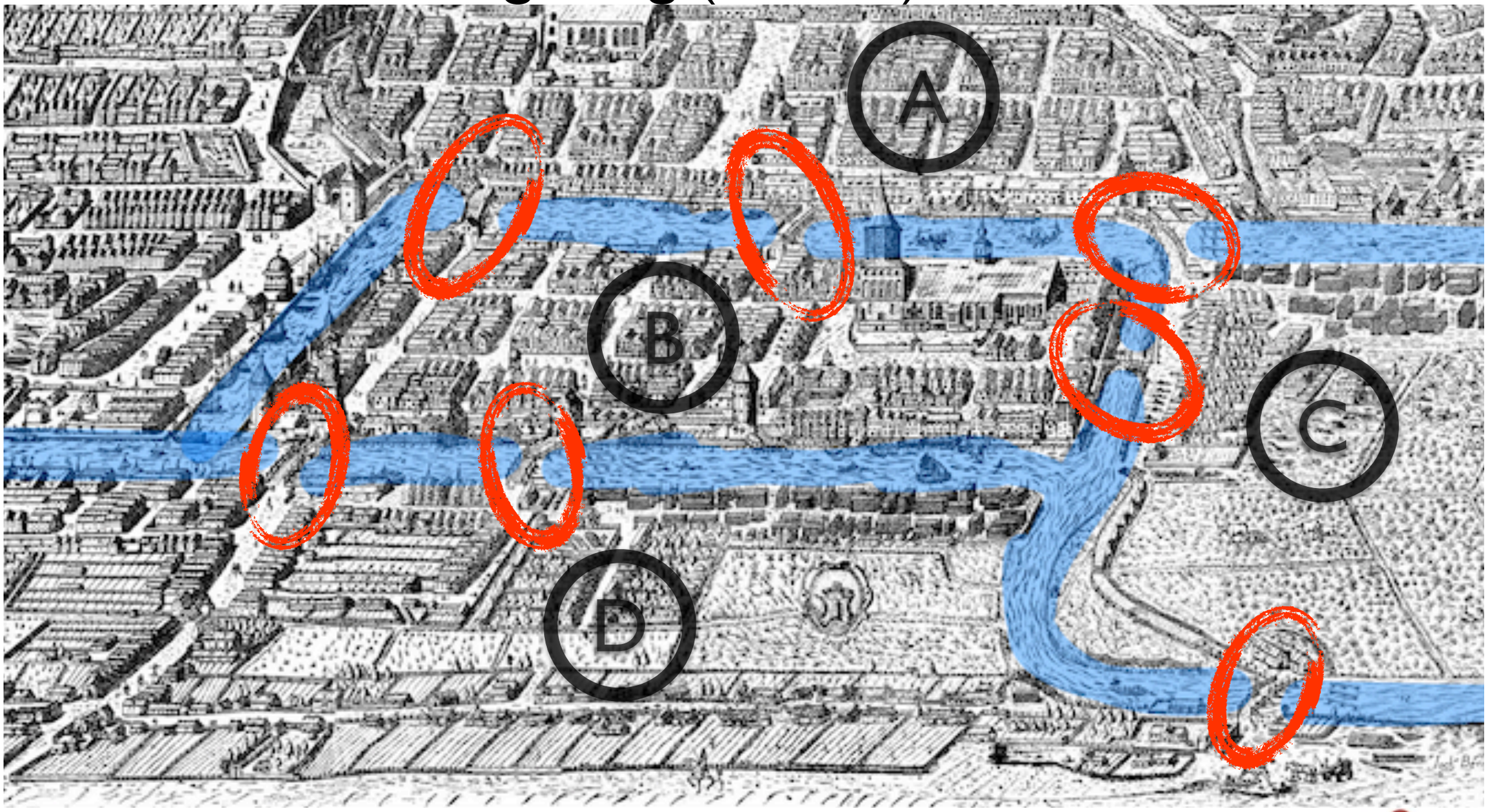
neo technology

# Leonhard Euler 1707-1783



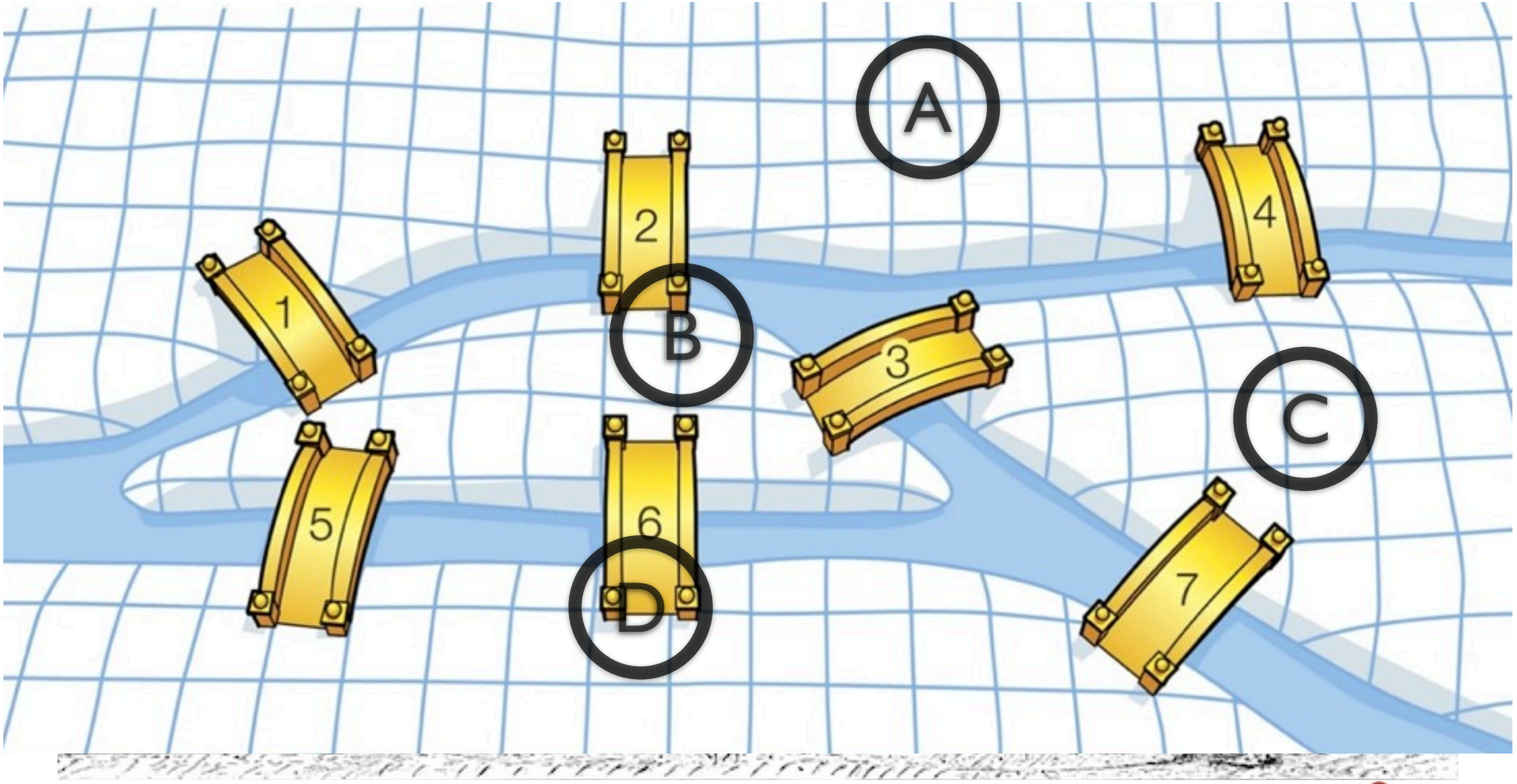
neotechnology

# Königsberg (Prussia) - 1736



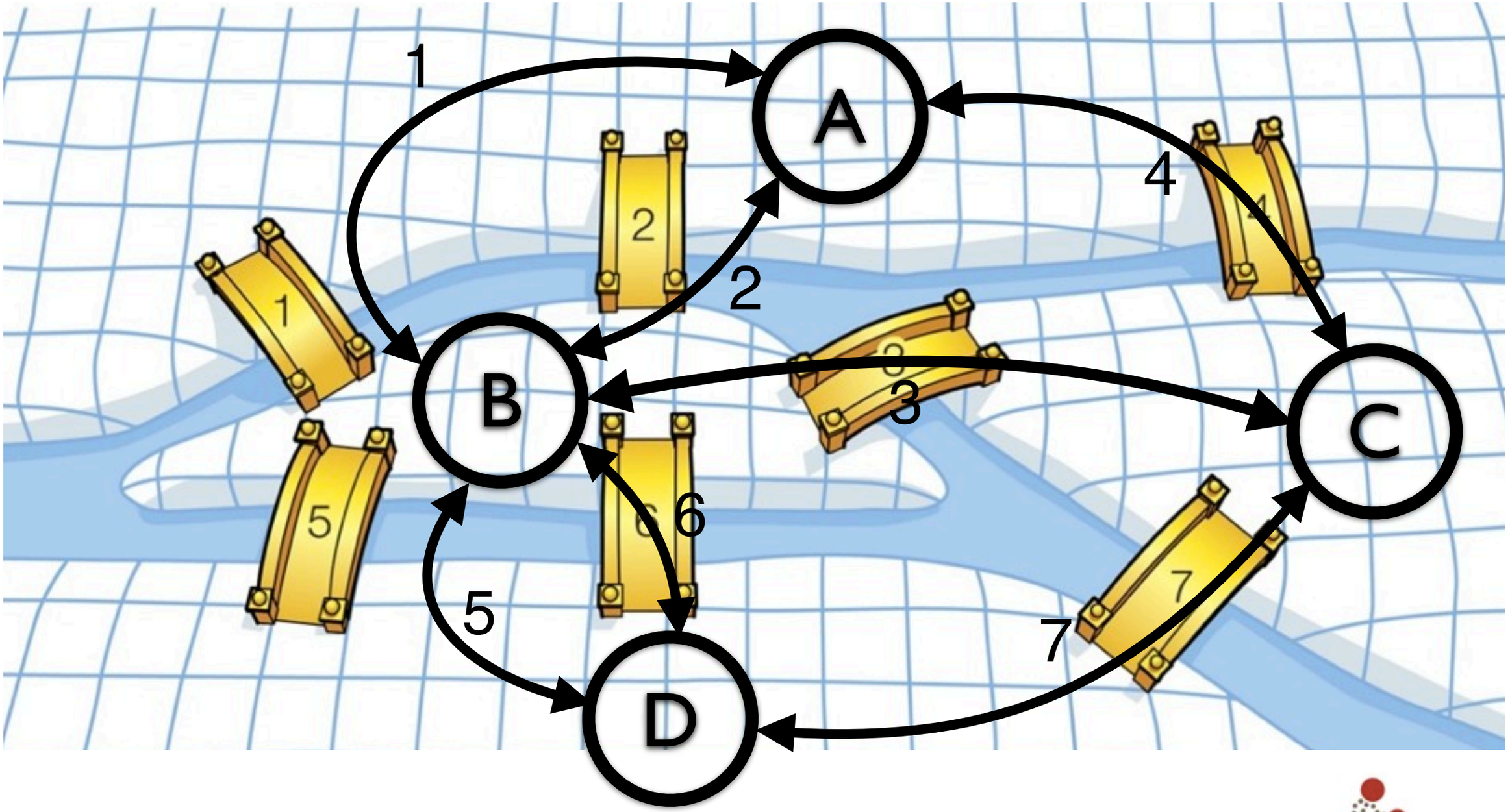



neotechnology





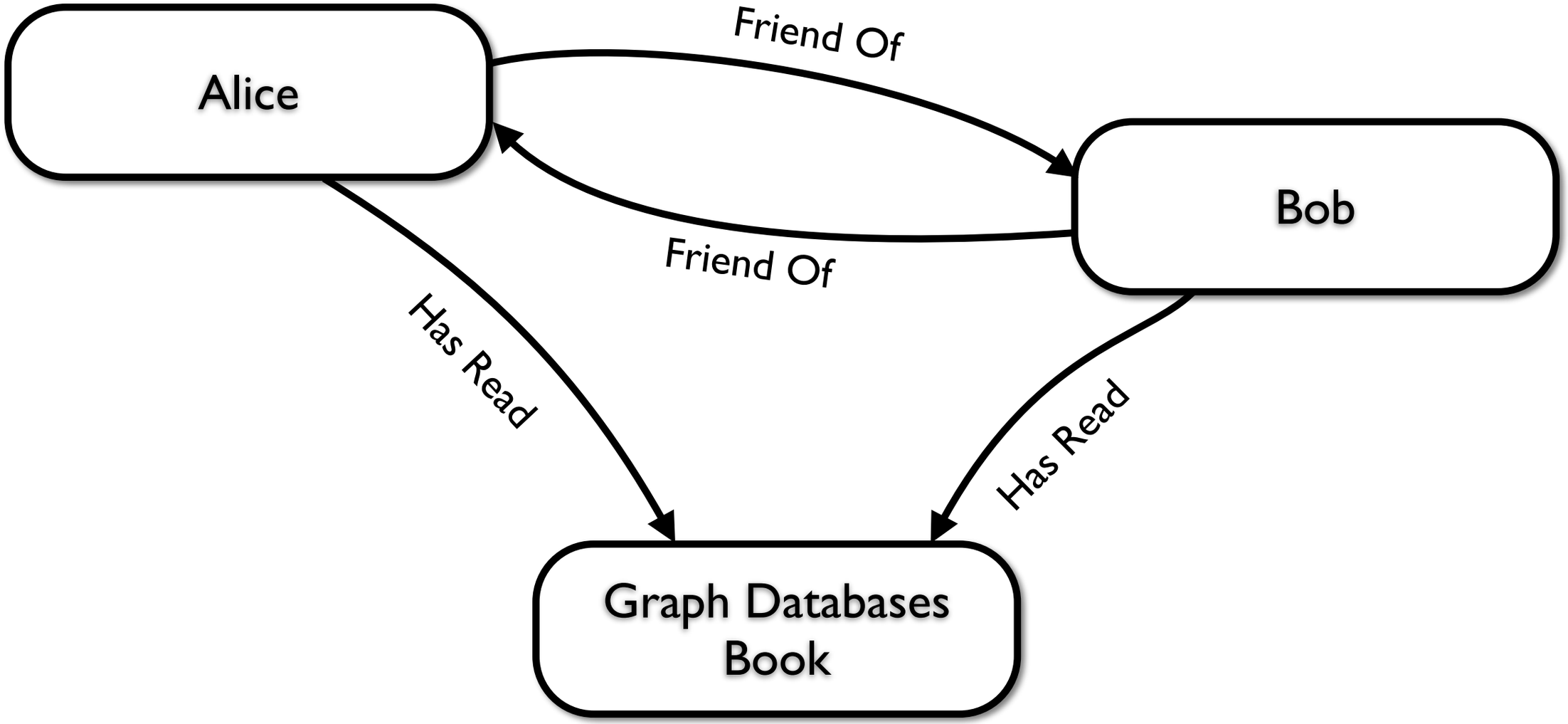

neotechnology





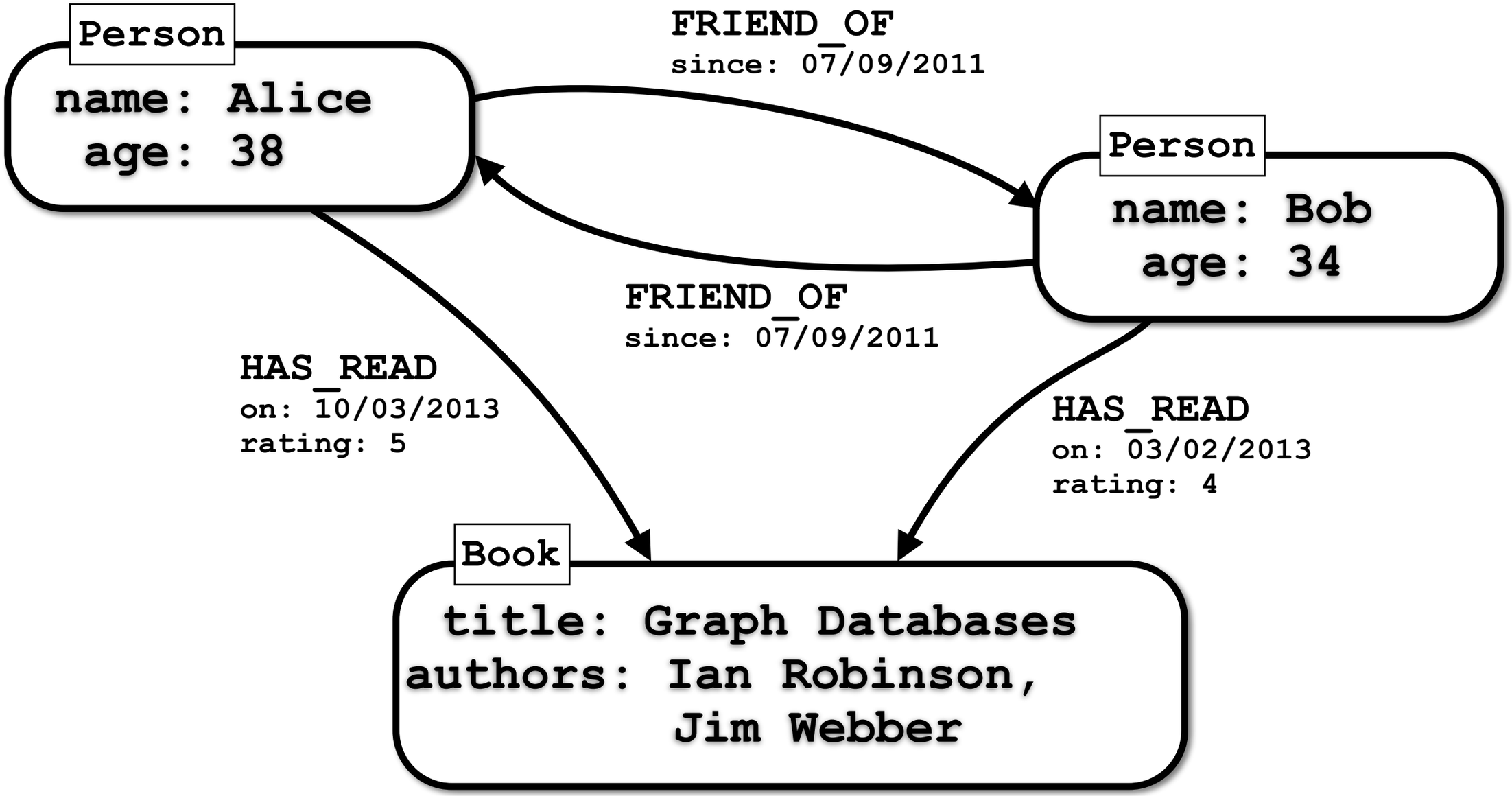

neotechnology

# So what about data?





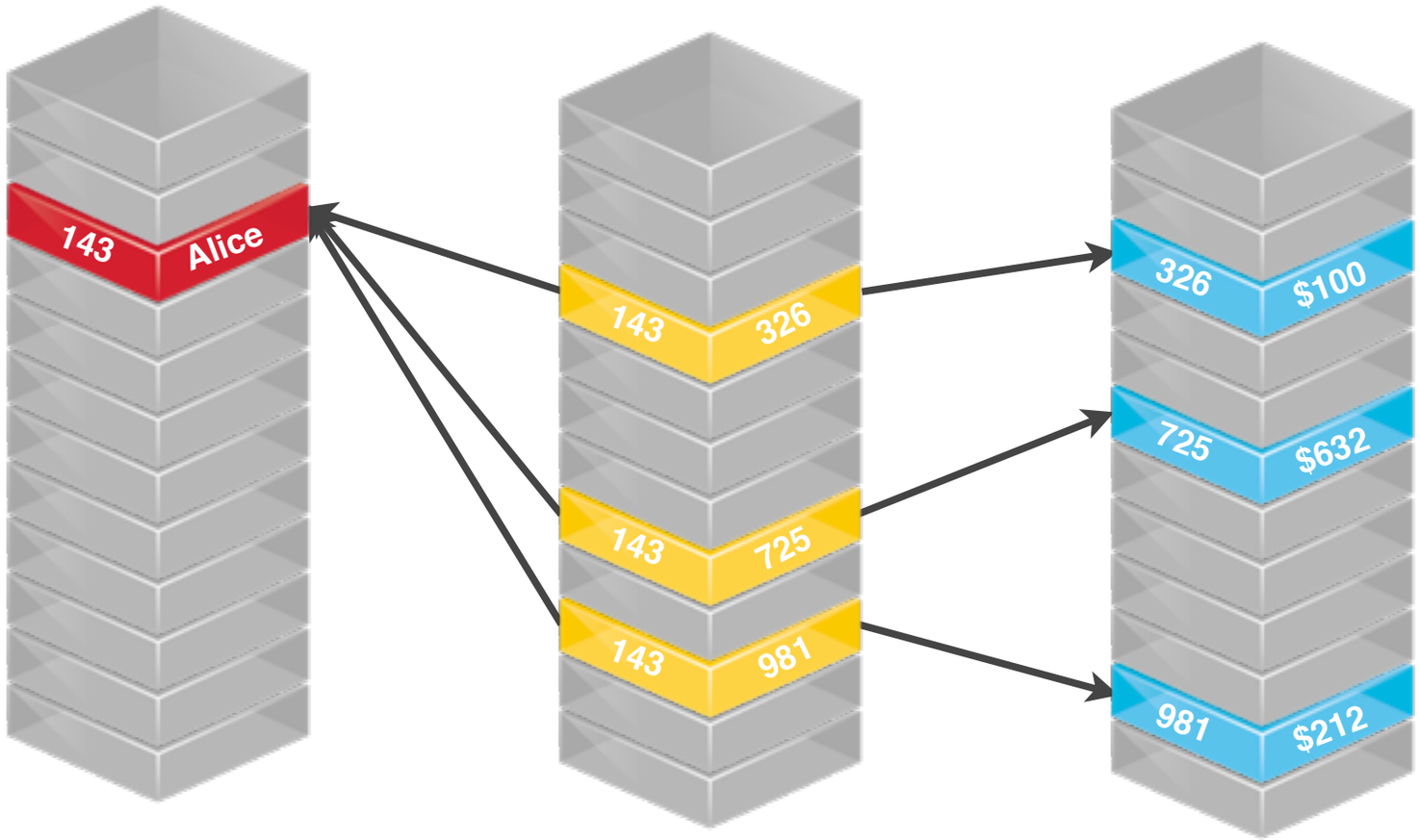

neotechnology





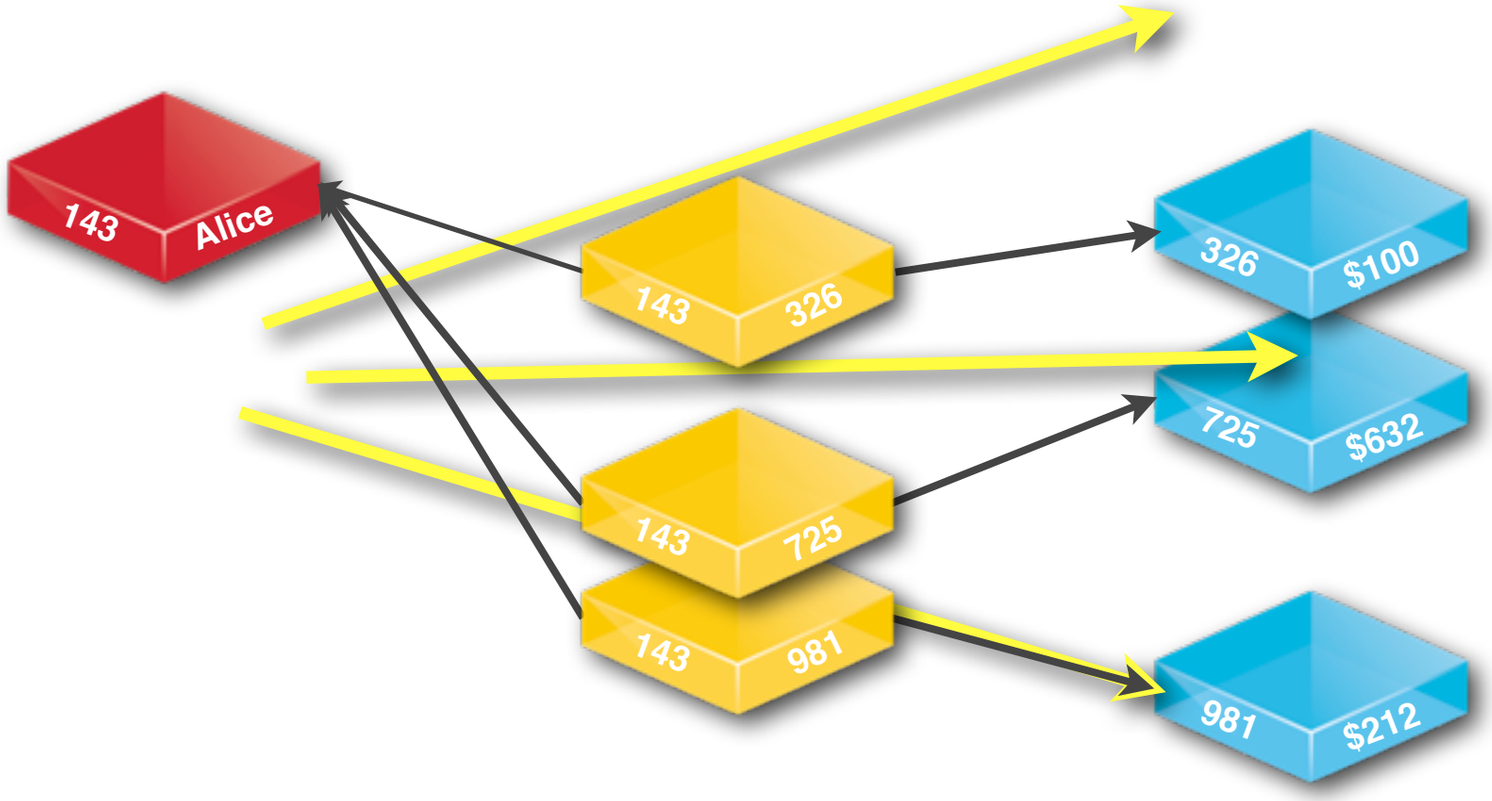

neotechnology


# Segue: Graphs in a Relational World



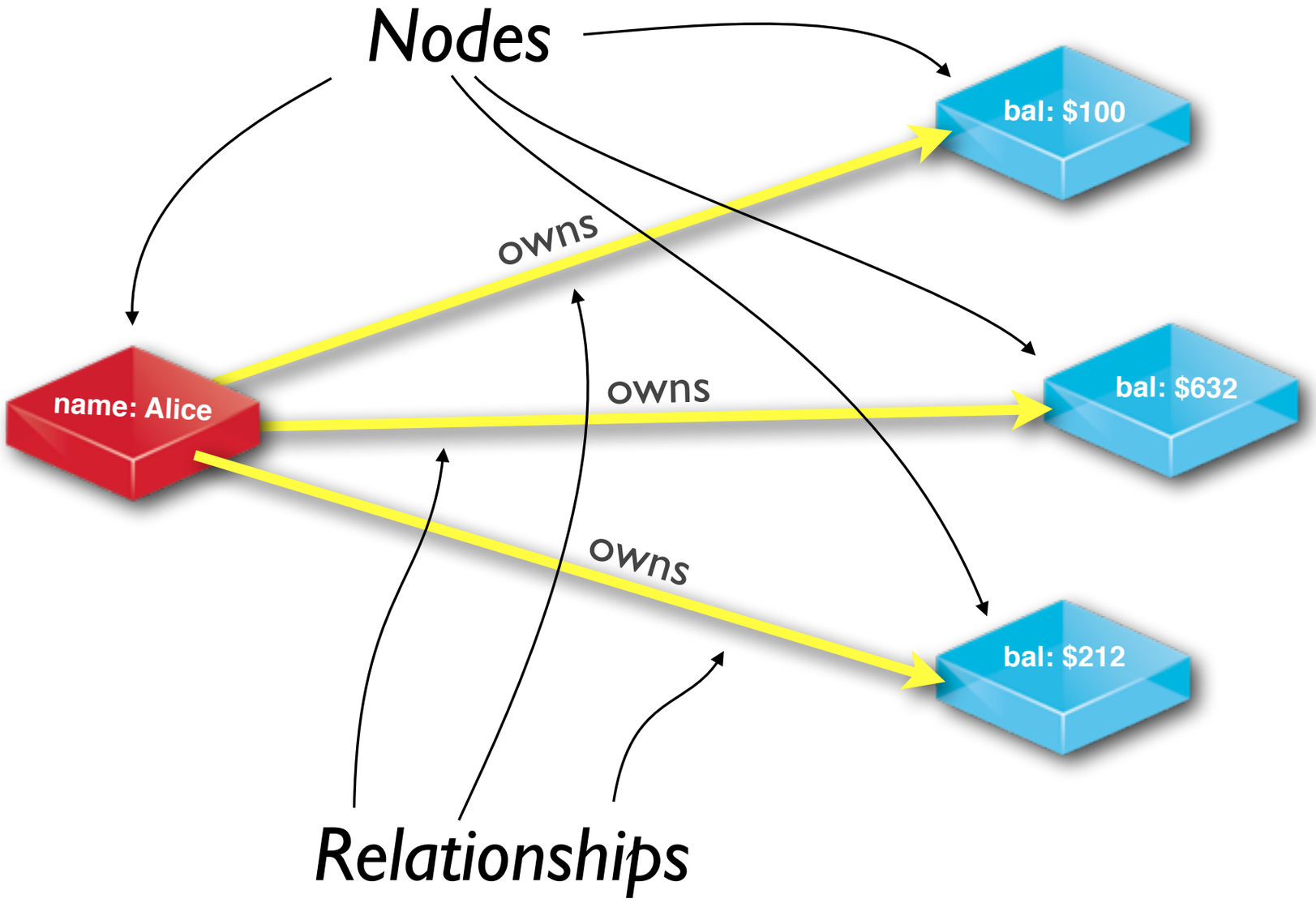



neotechnology






neotechnology





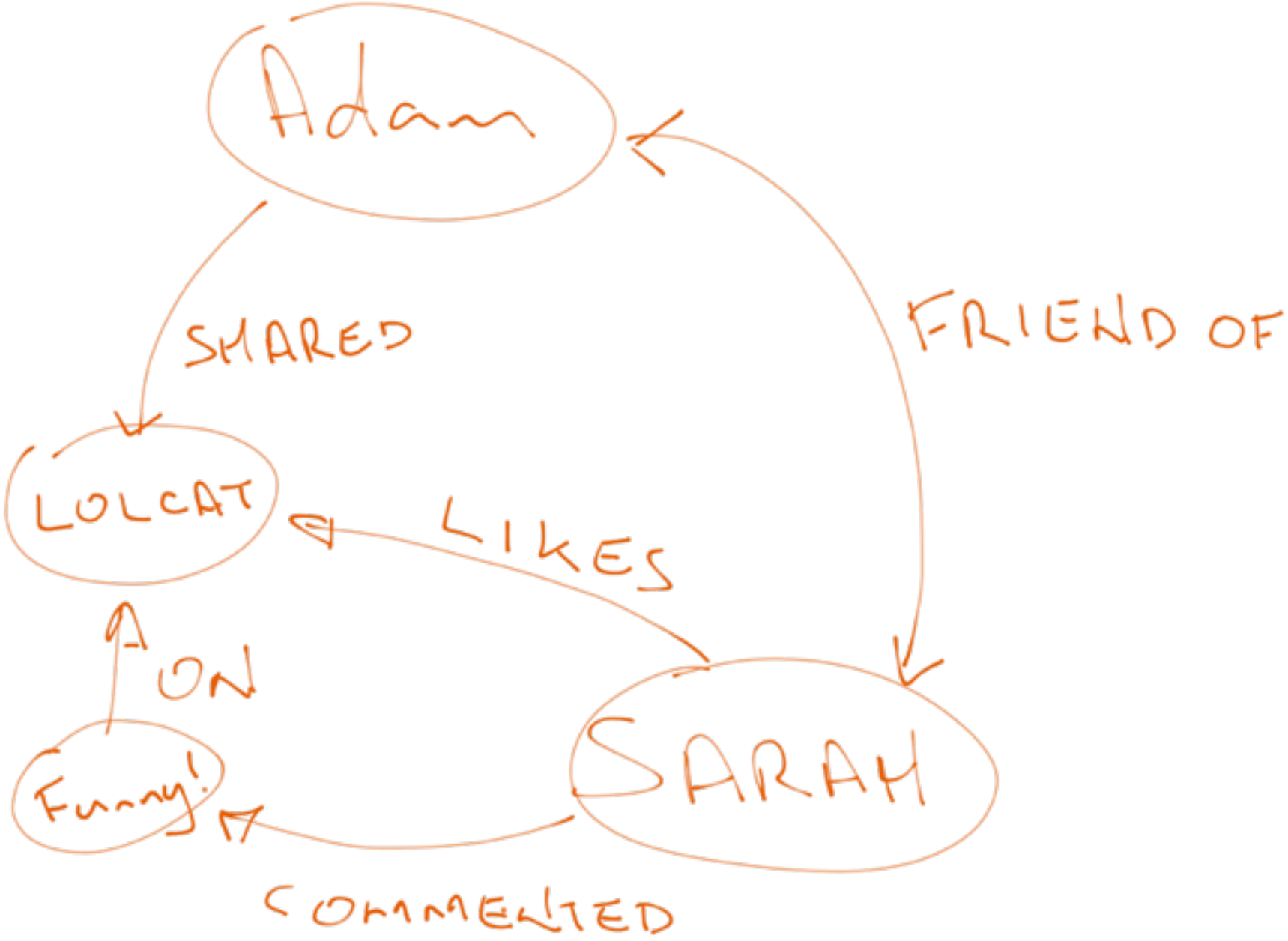

neotechnology


# (Modeling with Graphs)

# Start with a whiteboard

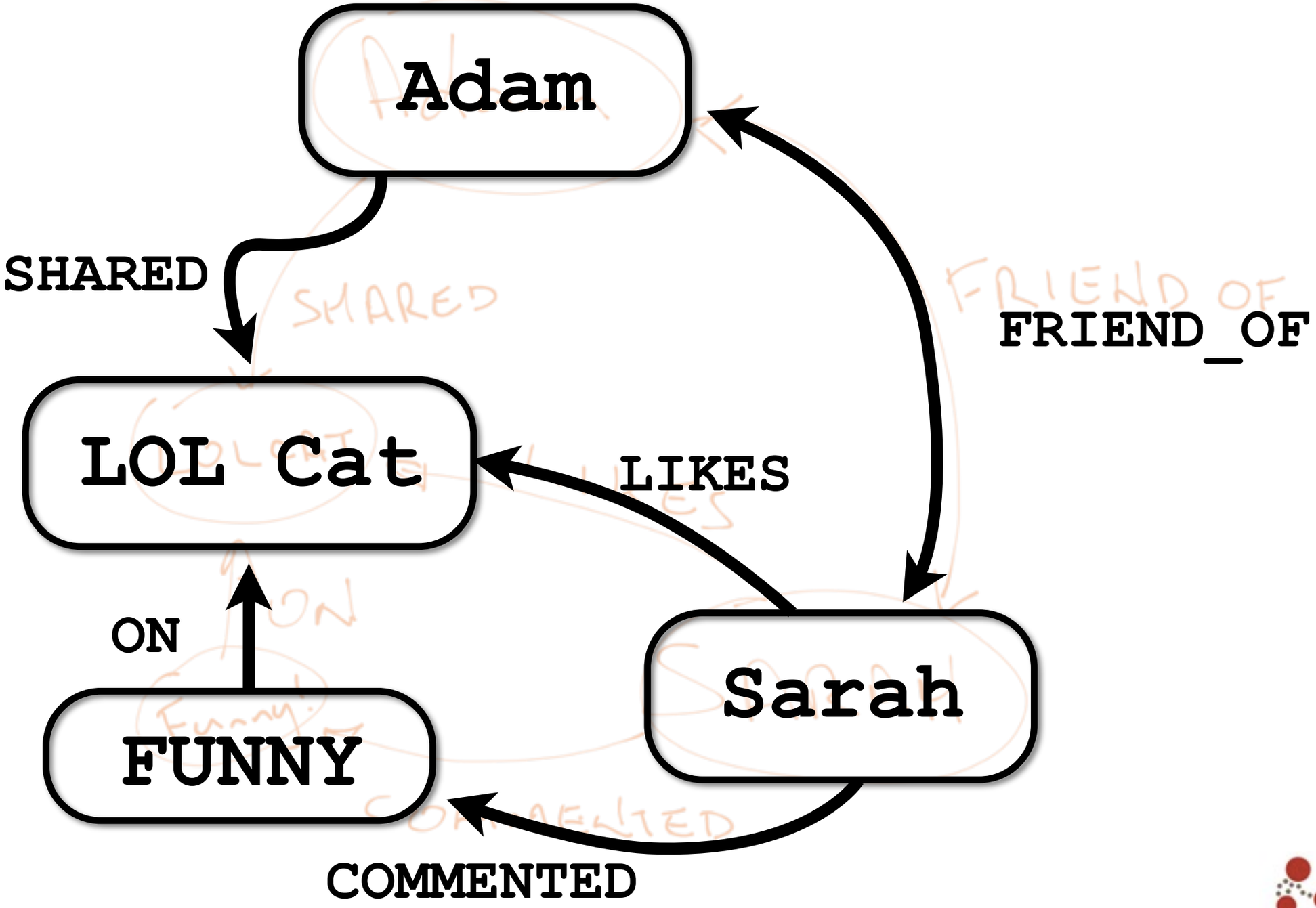


neo technology



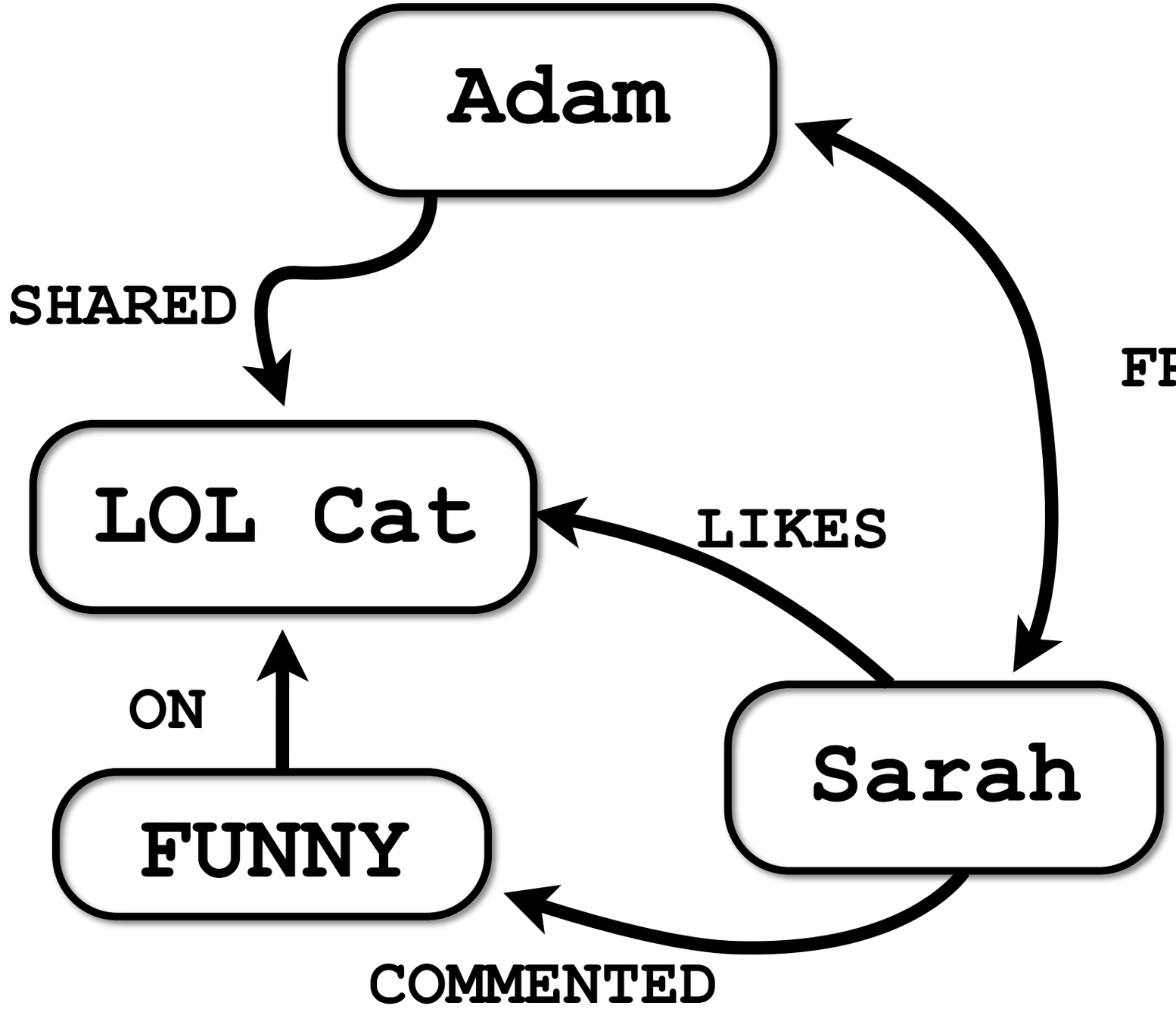



neotechnology



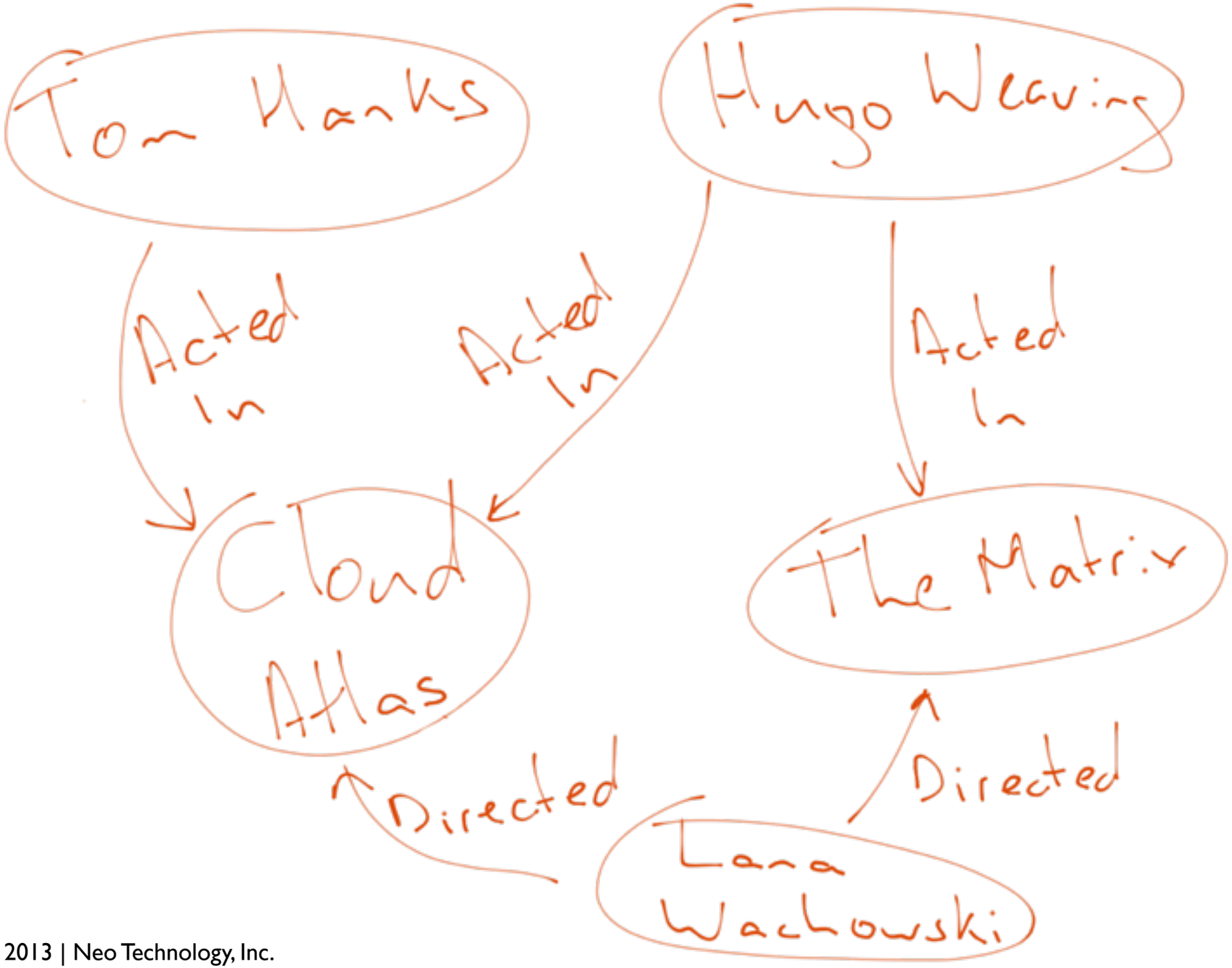



neotechnology



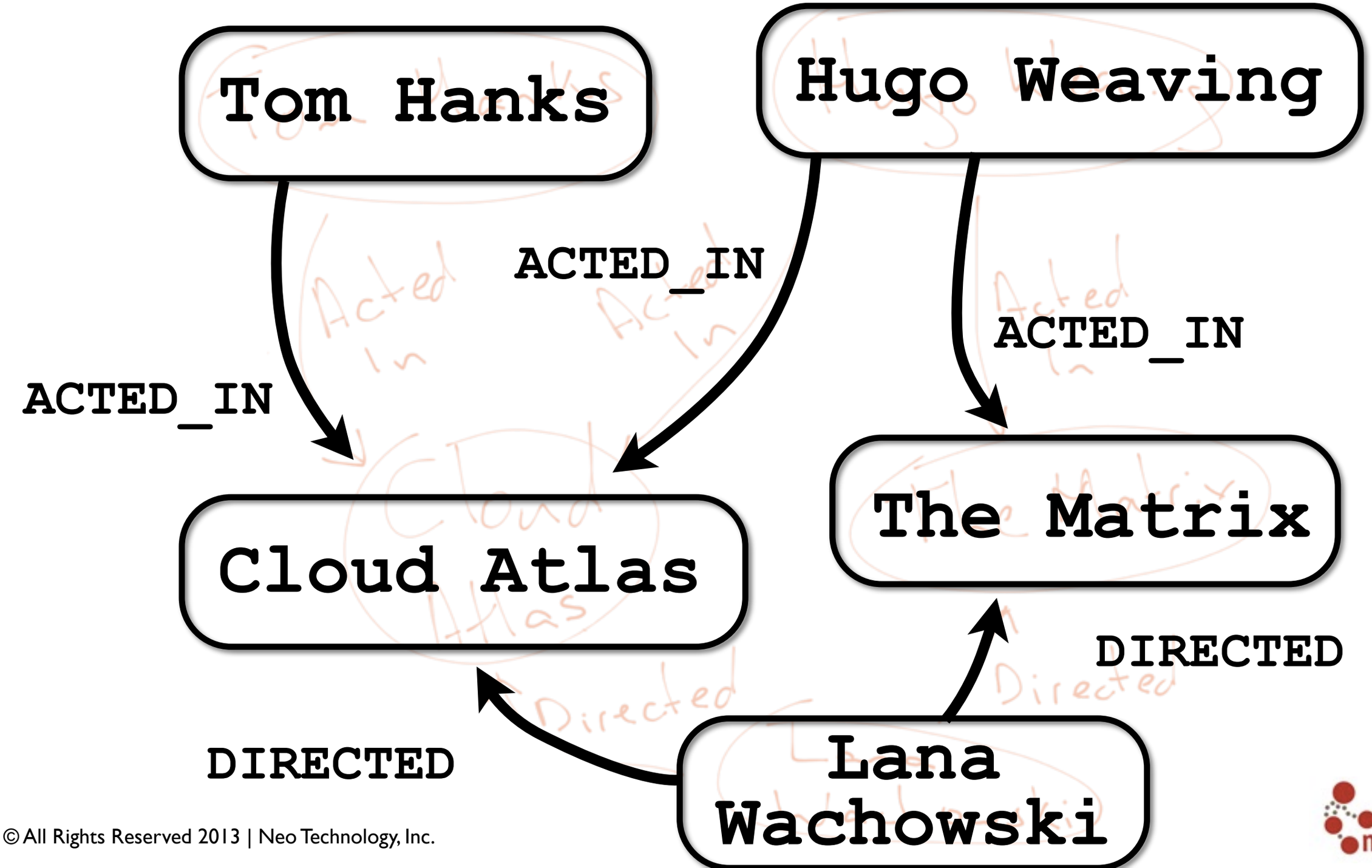



neotechnology





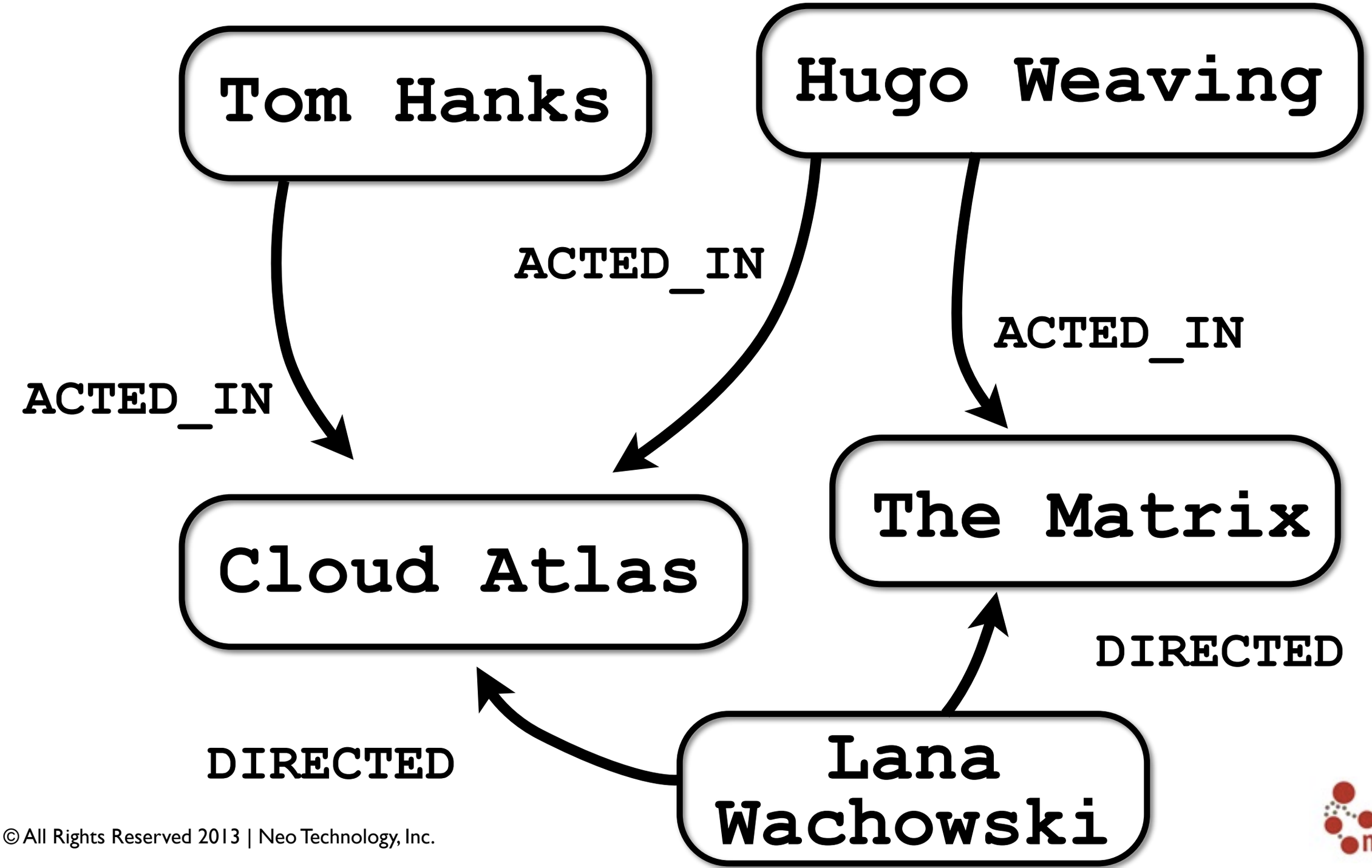

neotechnology





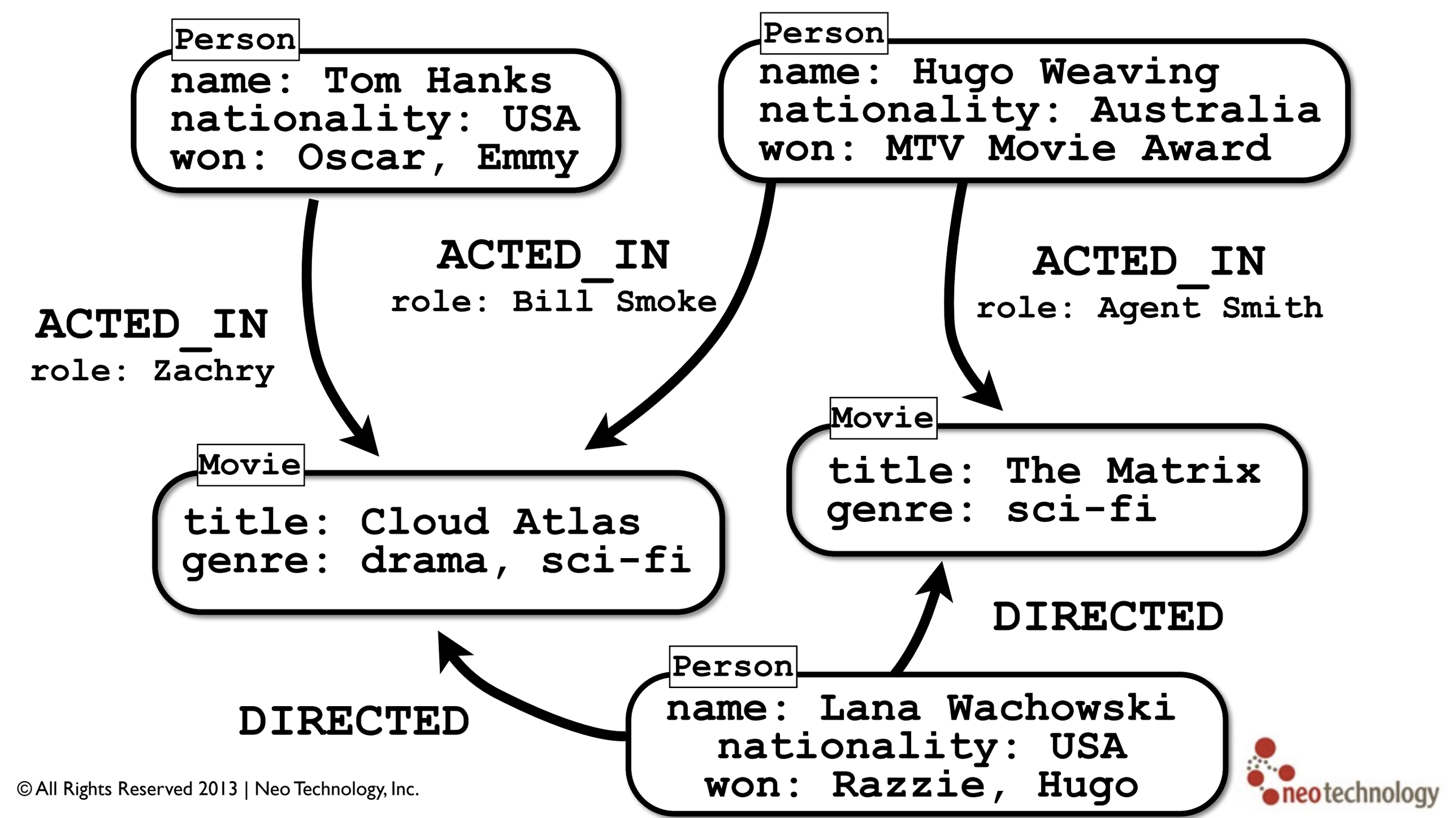

neotechnology

# Model Incrementally






neotechnology



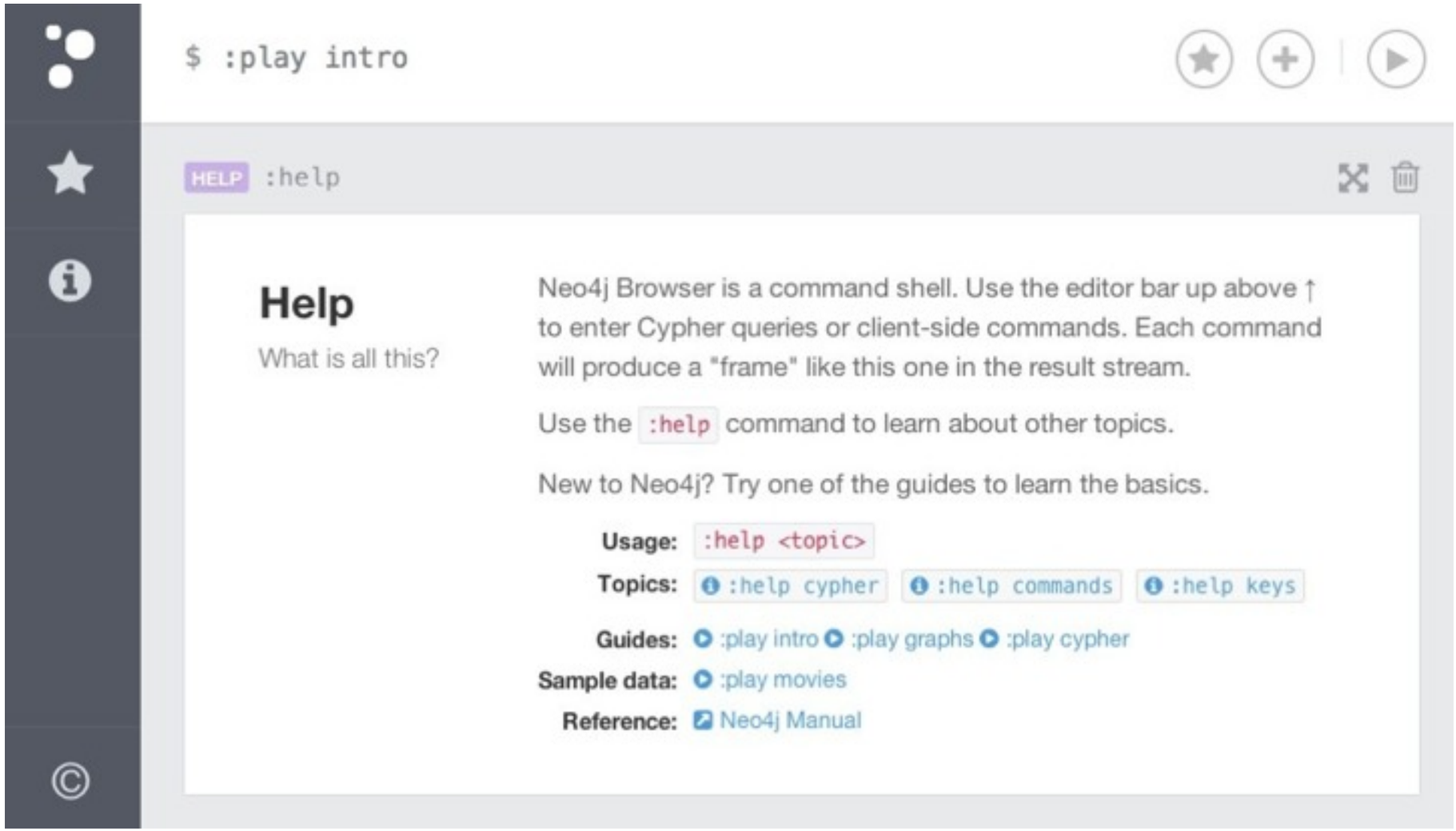



neotechnology


# Coffee?


# (Brief tour of Neo4j)

# Neo4j Browser - finding help

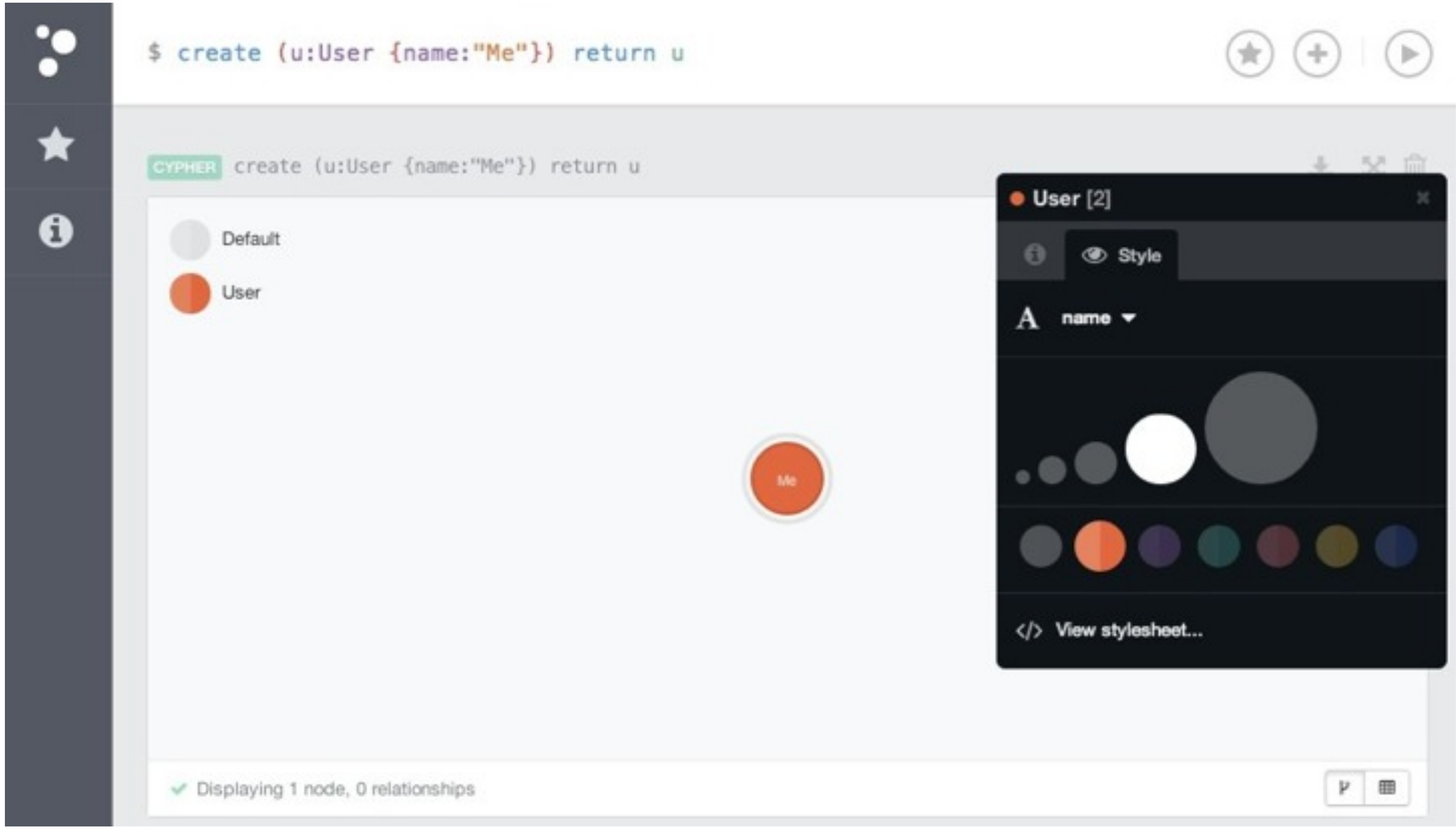



http://localhost:7474/



neotechnology

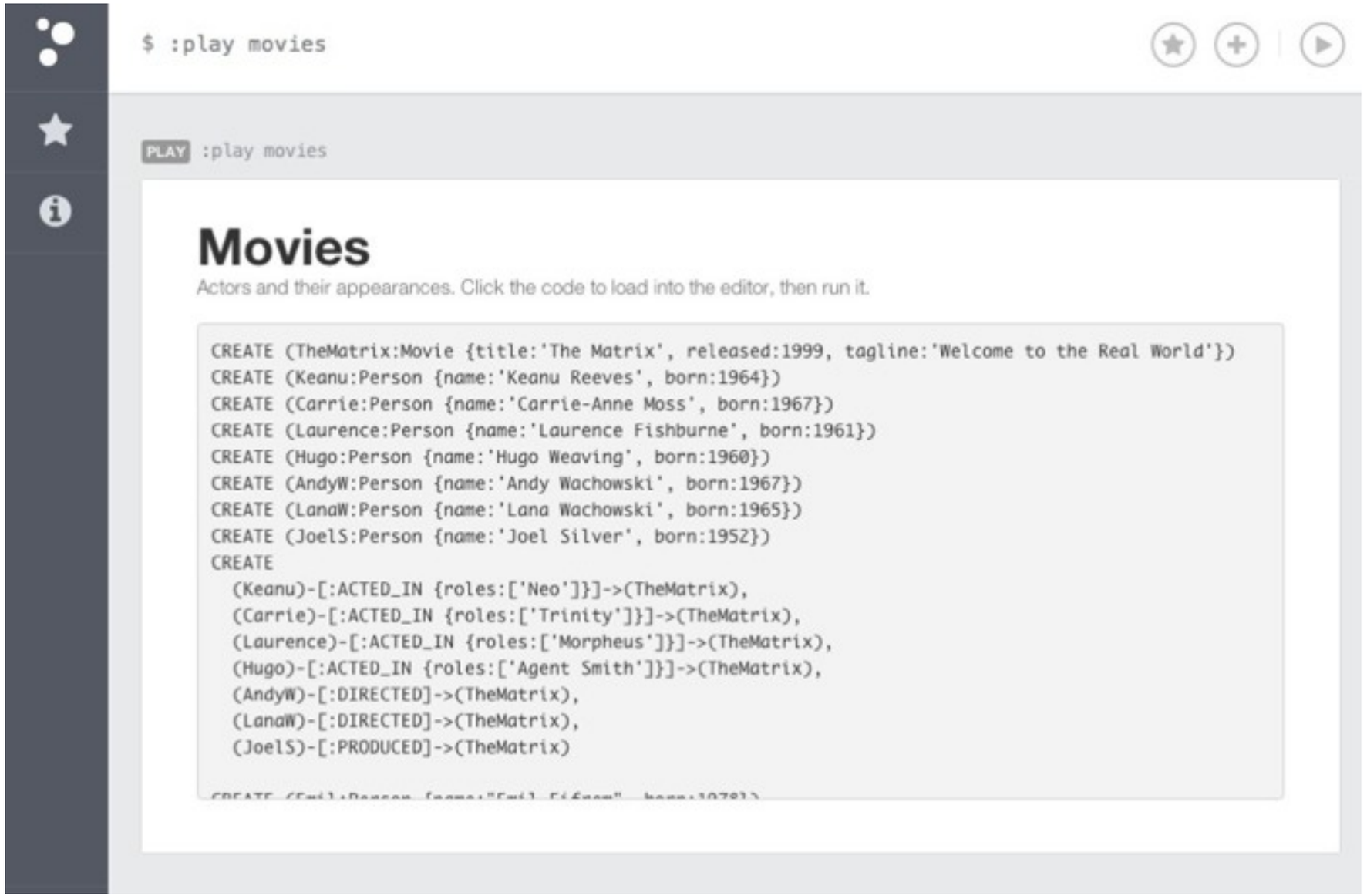
# Neo4j Browser - execute Cypher, visualize






neotechnology

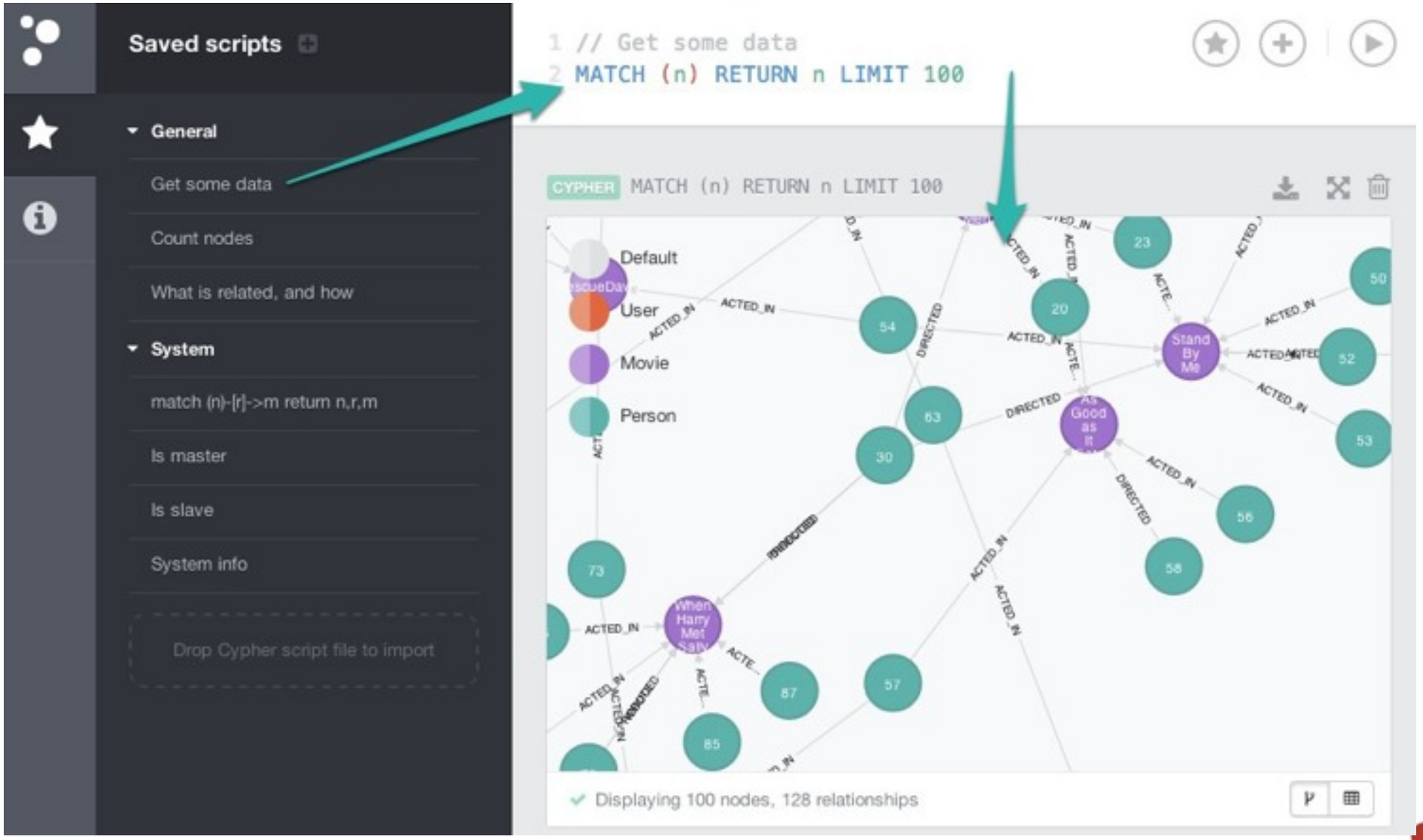
# Neo4j Browser - importing sample data






neotechnology

# Neo4j Data browser - display more data






neotechnology


# (Introduction to Cypher)

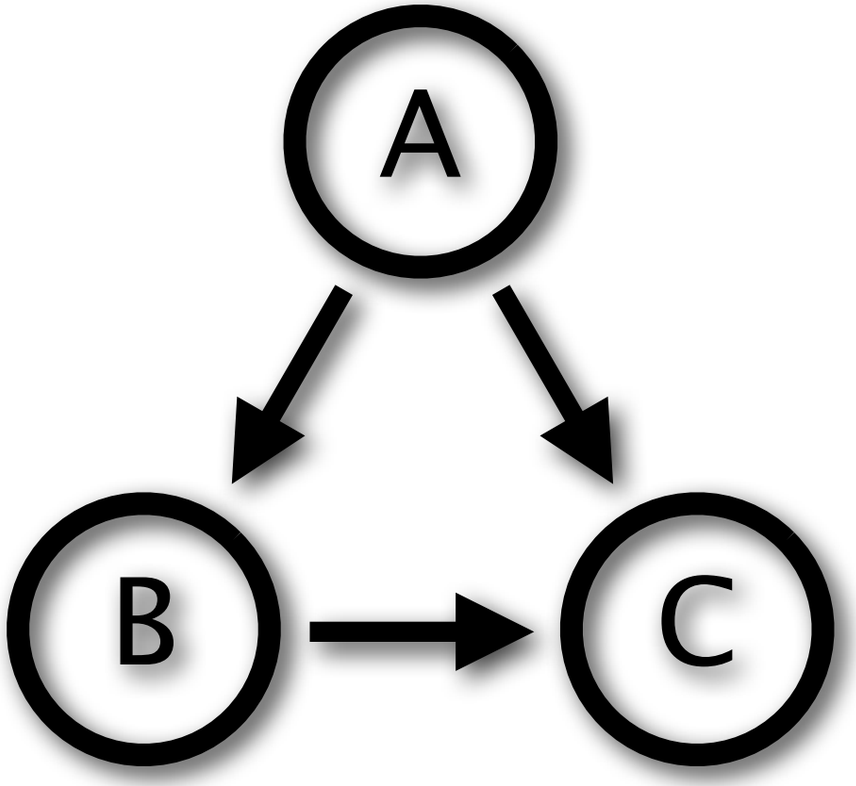
# Cypher is Neo4j's graph query language



neo technology

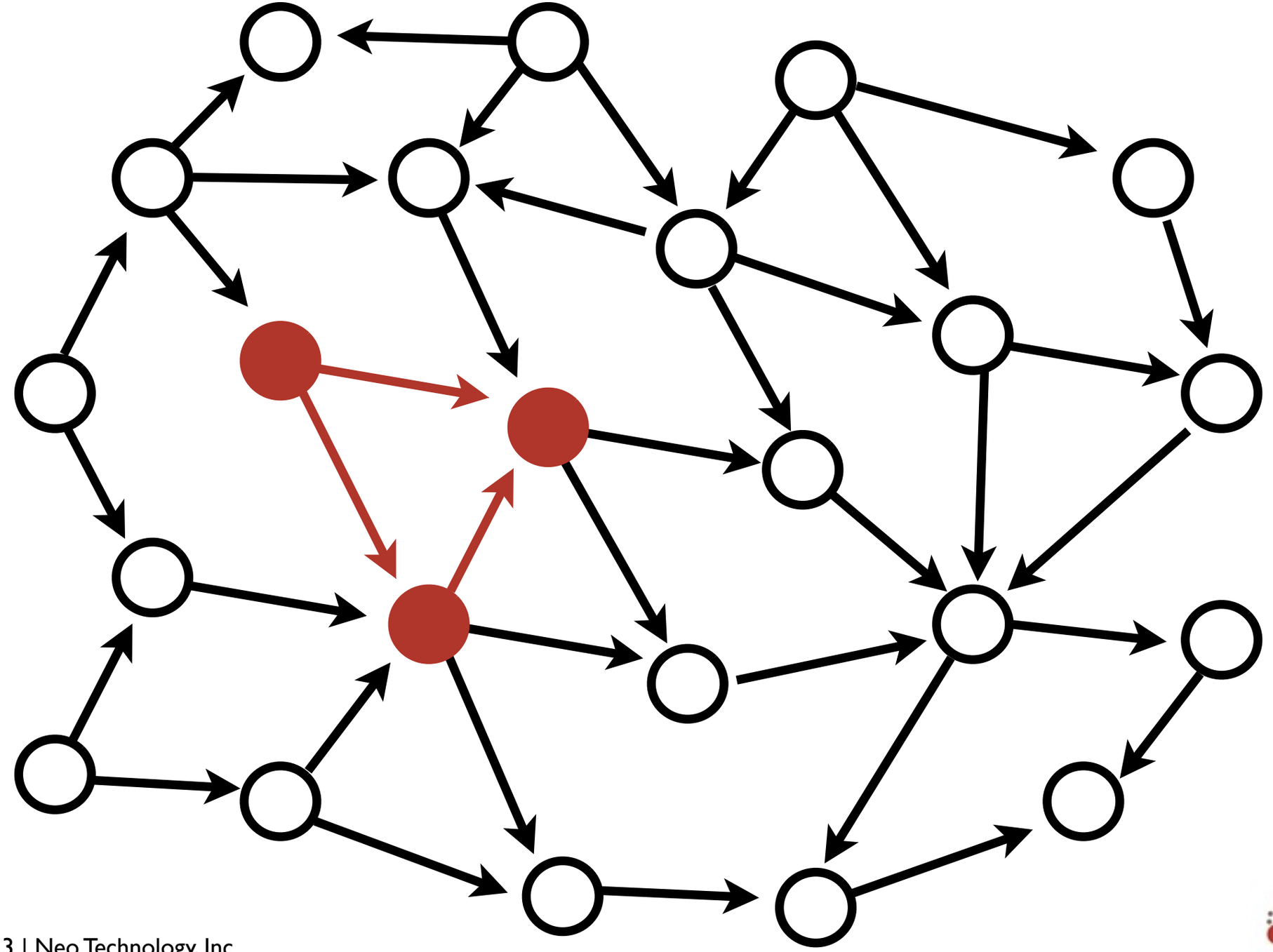
- Declarative Pattern-Matching language
- SQL-like syntax
- Designed for graphs

# It's all about Patterns



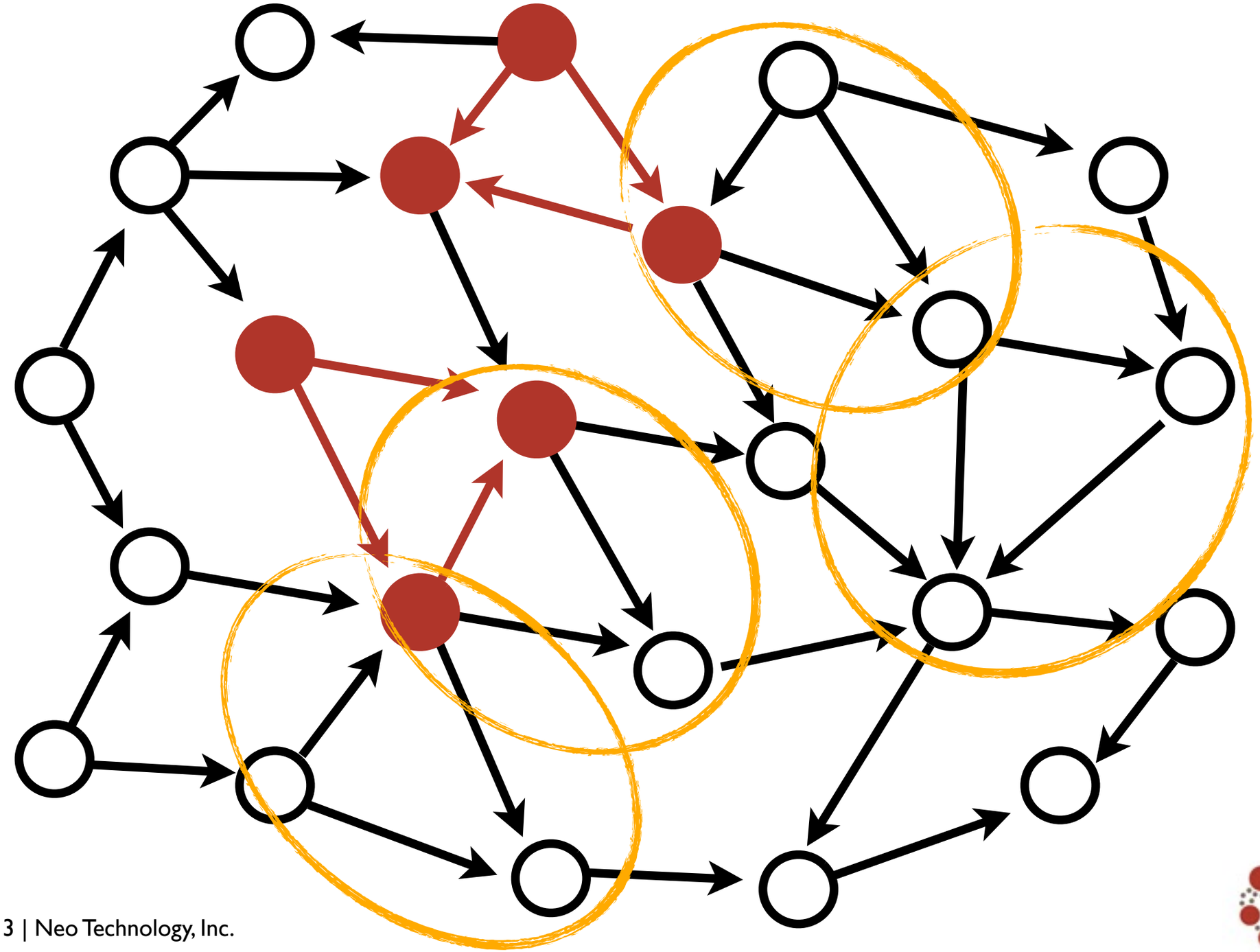



neotechnology





neotechnology





neotechnology

# How?



neo technology

# Two nodes, one relationship


```
(a) --> (b)
```



neotechnology

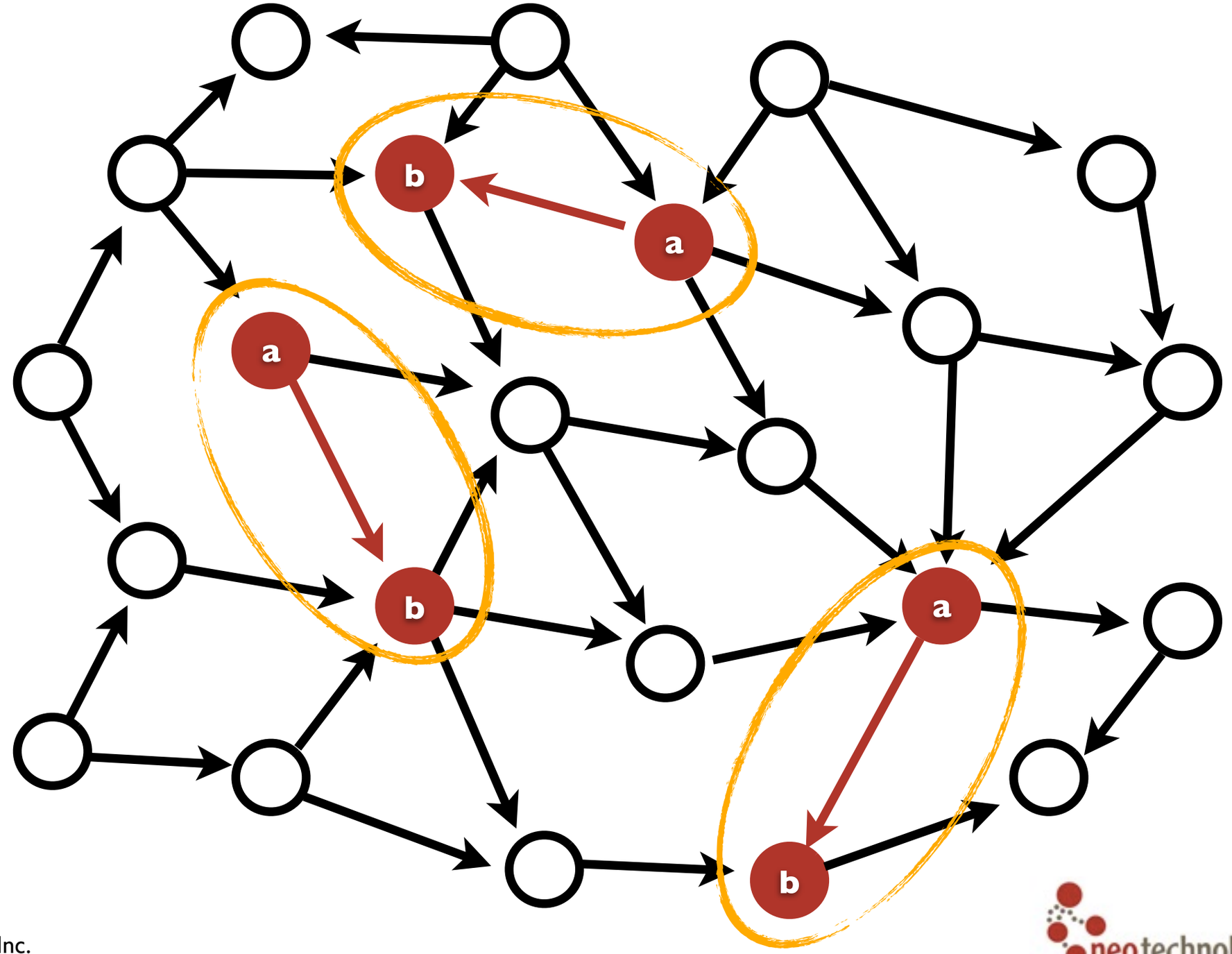
# Two nodes, one relationship

```
MATCH (a)-->(b)
RETURN a, b;
```




neotechnology

```
MATCH (a)-->(b)
RETURN a, b;
```






neotechnology

# Two nodes, one relationship


```
(a) --> ()
```



neotechnology

# Two nodes, one relationship

```
MATCH (a)-->()
RETURN a;
```




neotechnology

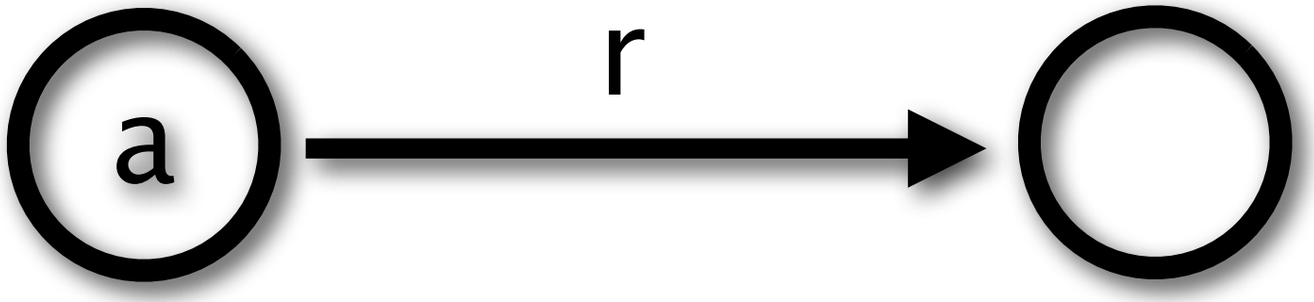
# Two nodes, one relationship

```
MATCH (a)-->()
RETURN a.name;
```




neotechnology

# Two nodes, one relationship



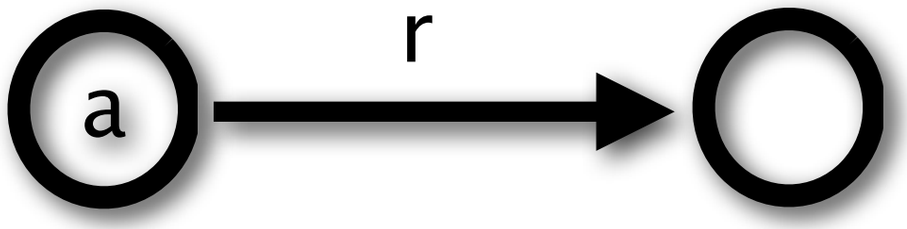

```
(a) -[r]-> ()
```



neotechnology

# Two nodes, one relationship

```
MATCH (a)-[r]->()
RETURN a.name, type(r);
```






neotechnology

# Two nodes, one relationship


```
(a) -[:ACTED_IN]-> (m)
```



neotechnology

# Two nodes, one relationship

```
MATCH (a)-[:ACTED_IN]->(m)
RETURN a.name, m.title;
```




neotechnology

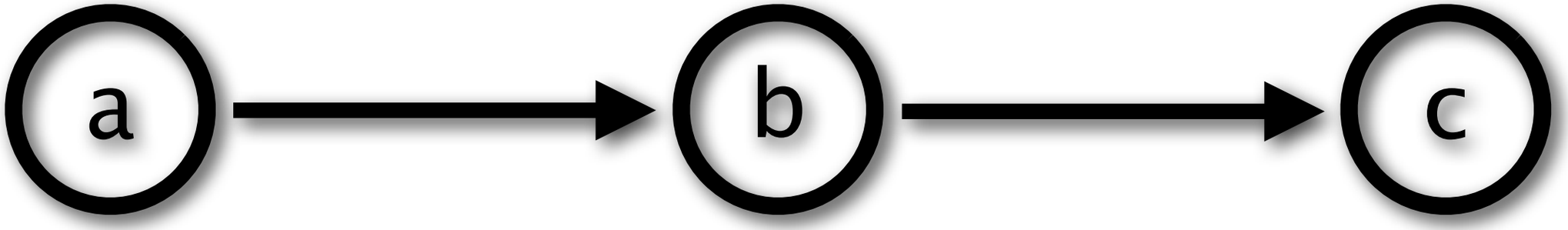
# Two nodes, one relationship

```
MATCH (a)-[r:ACTED_IN]->(m)
RETURN a.name, r.roles, m.title;
```




neotechnology

# Paths

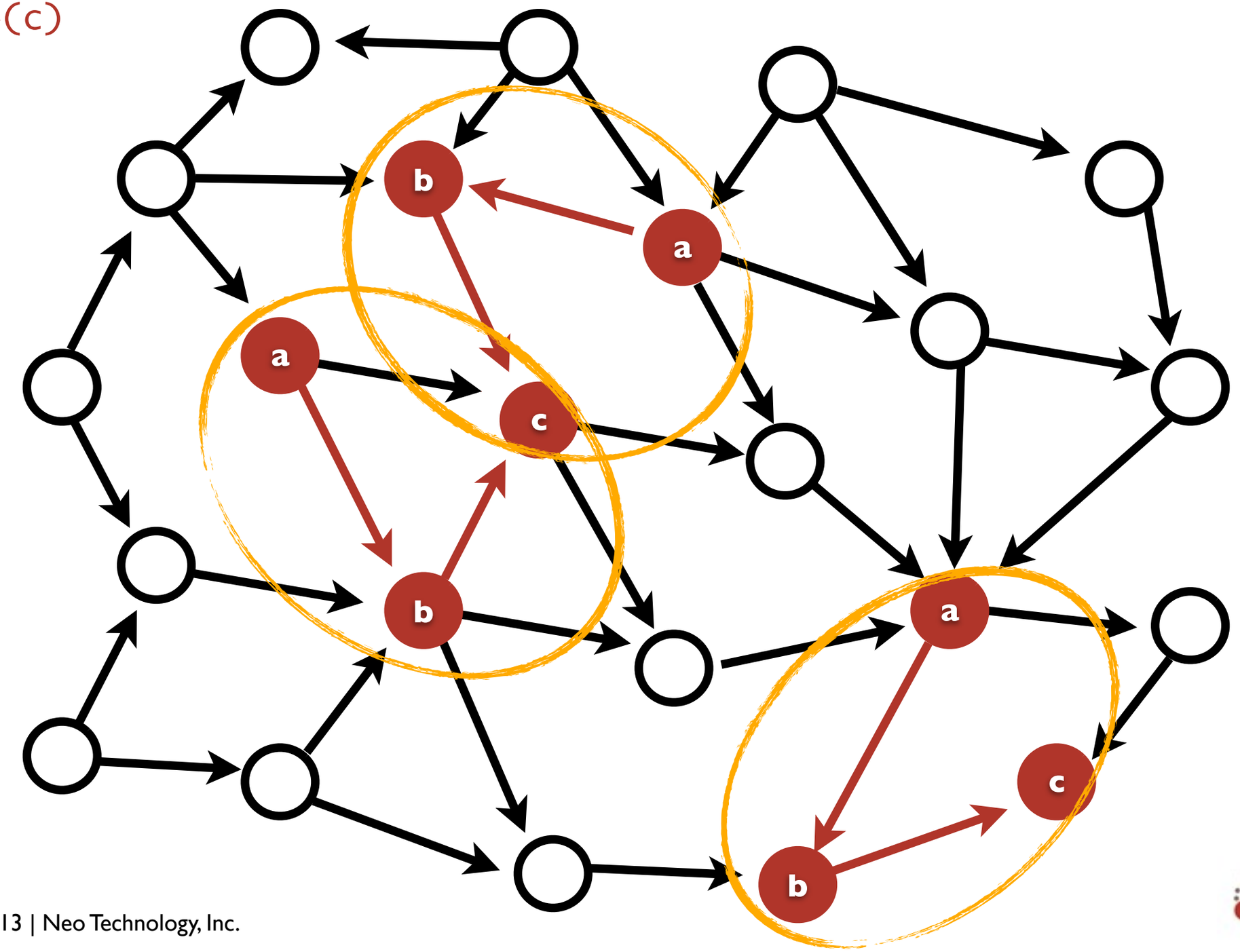



```
(a)-->(b)-->(c)
```



neotechnology

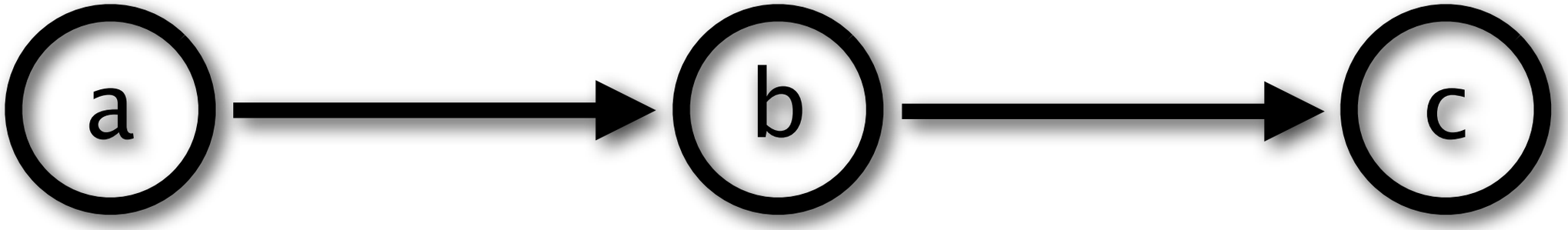
(a)-->(b)-->(c)






neotechnology

# Paths

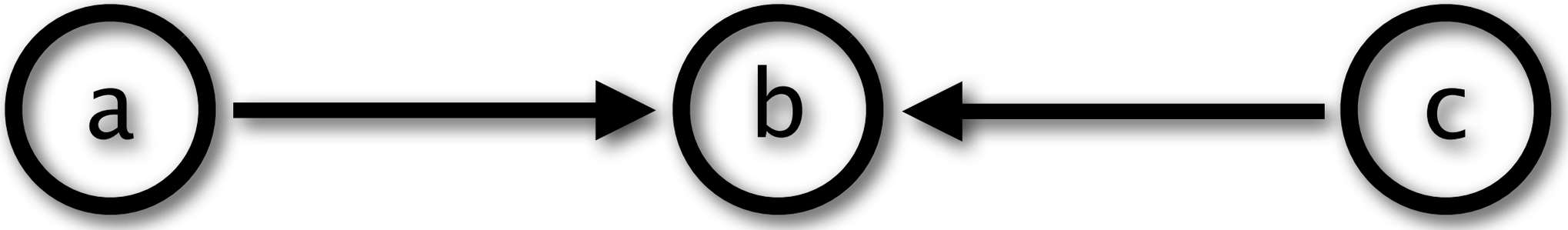



```
(a)-->(b)-->(c)
```



neotechnology

# Paths

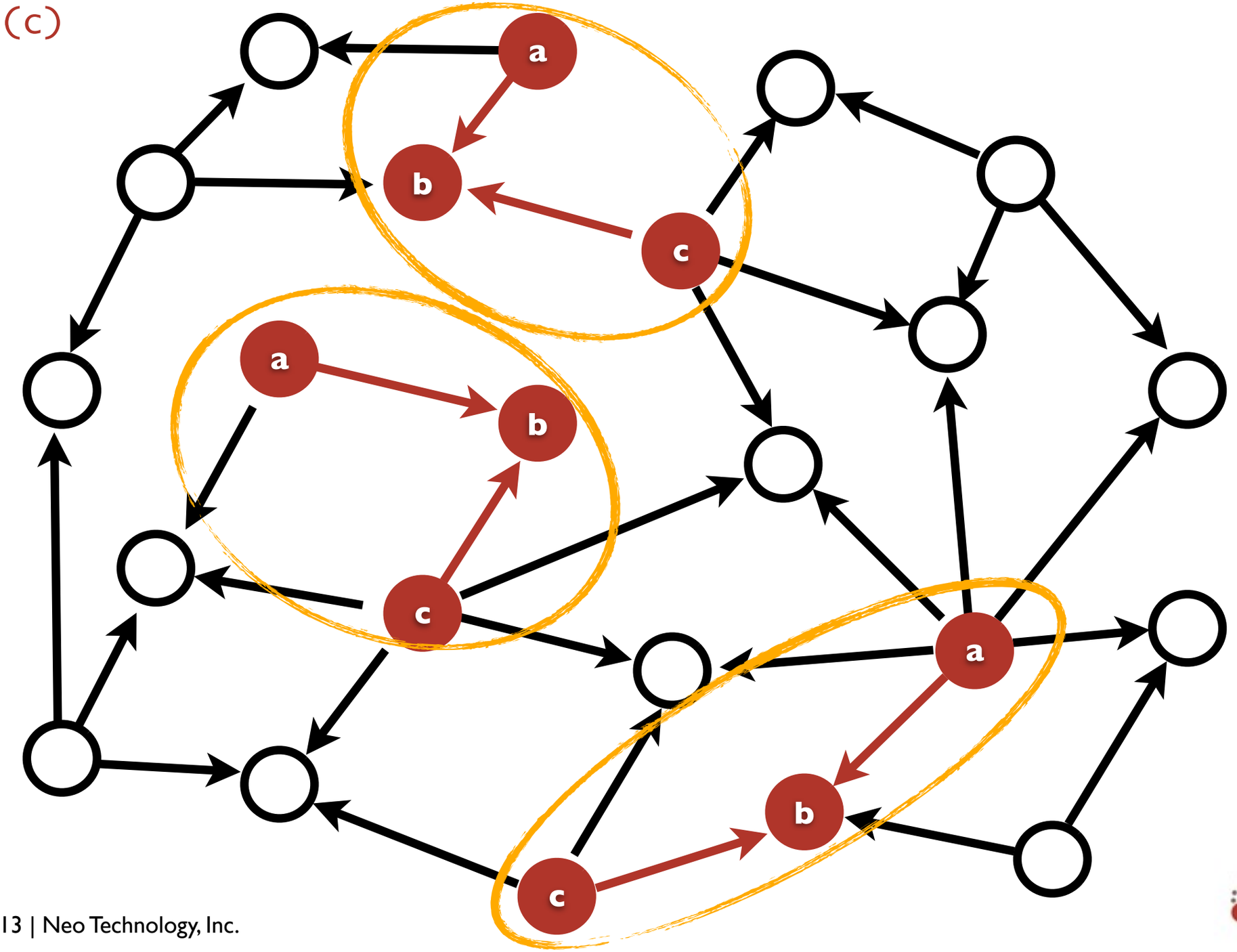



```
(a)-->(b)<--(c)
```



neotechnology

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

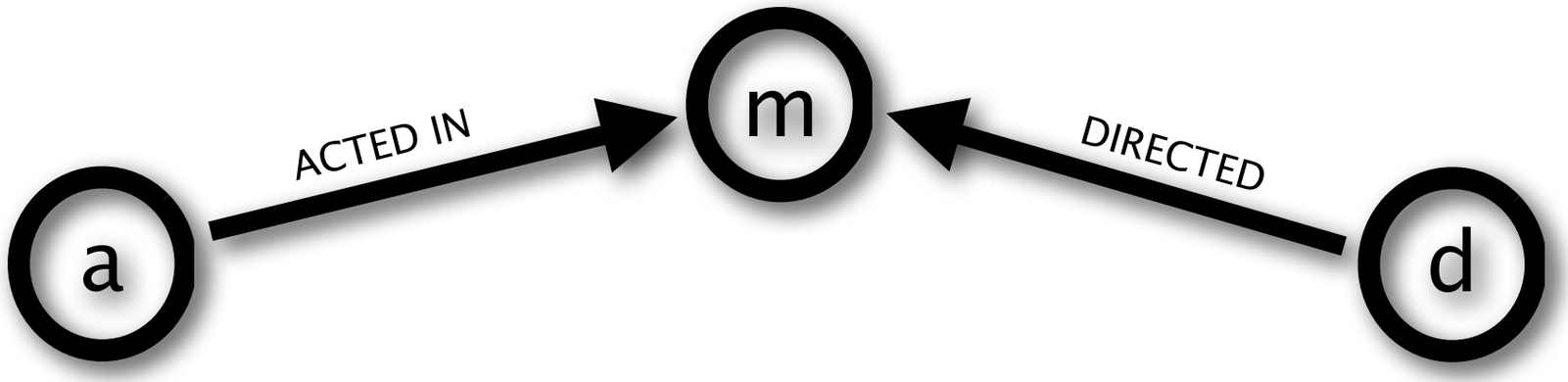




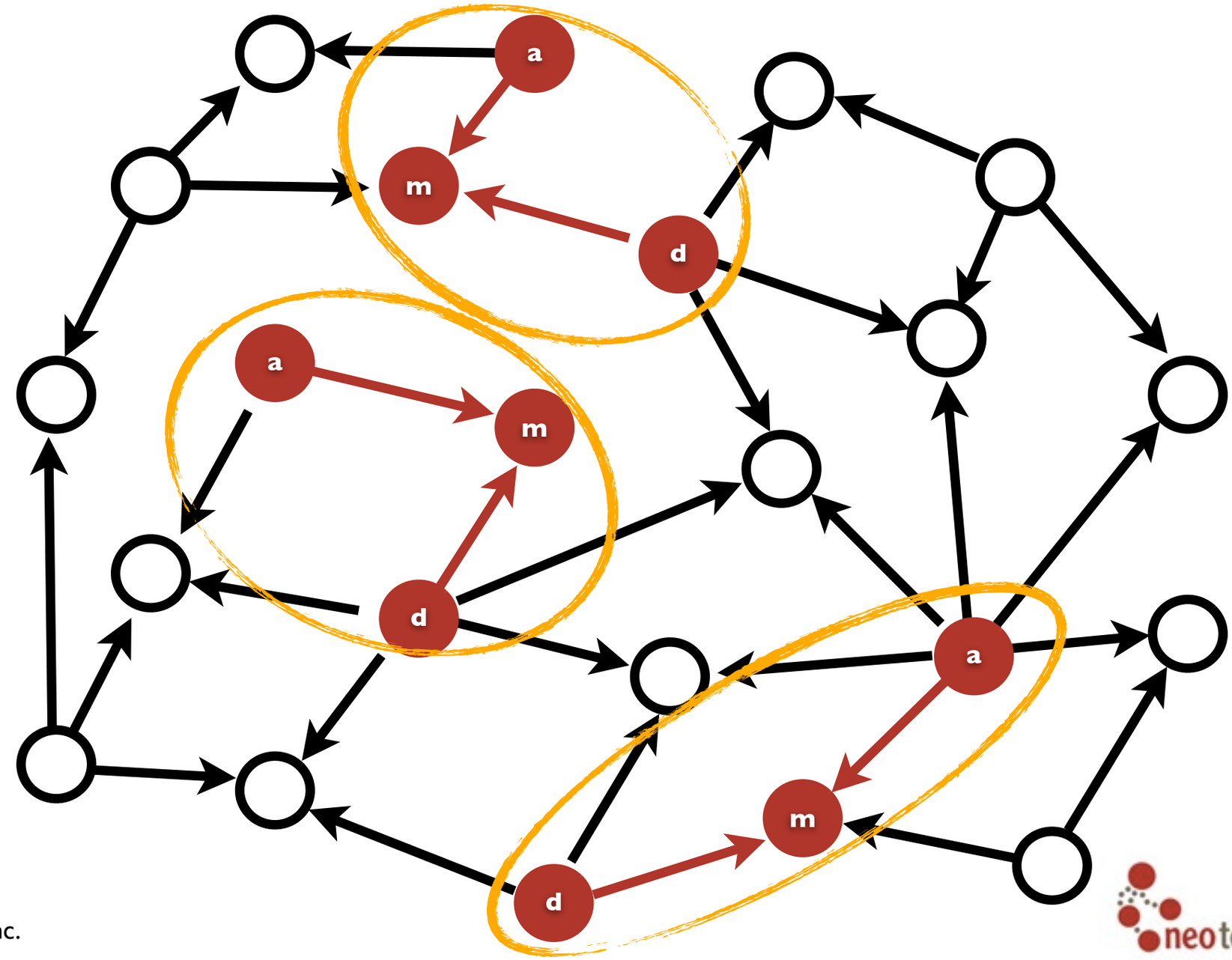

neotechnology

# Paths

```
MATCH (a)-[:ACTED_IN]->(m)<-[:DIRECTED]-(d)
RETURN a.name, m.title, d.name;
```



```
MATCH (a)-[:ACTED_IN]->(m)<-[:DIRECTED]-(d)
RETURN a.name, m.title, d.name;
```






neotechnology

# Paths

```
MATCH (a)-[:ACTED_IN]->(m)<-[:DIRECTED]-(d)
RETURN a.name, m.title, d.name;
```

| a.name | m.title | d.name |
|---|---|---|
| “Keanu Reeves” | “The Matrix” | “Andy Wachowski” |
| “Keanu Reeves” | “The Matrix Reloaded” | “Andy Wachowski” |
| “Noah Wyle” | “A Few Good Men” | “Rob Reiner” |
| “Tom Hanks” | “Cloud Atlas” | “Andy Wachowski” |
| ... | ... | ... |

# Paths

```
MATCH (a)-[:ACTED_IN]->(m)<-[:DIRECTED]-(d)
RETURN a.name AS actor, m.title AS movie,
       d.name AS director;
```

| actor | movie | director |
|---|---|---|
| “Keanu Reeves” | “The Matrix” | “Andy Wachowski” |
| “Keanu Reeves” | “The Matrix Reloaded” | “Andy Wachowski” |
| “Noah Wyle” | “A Few Good Men” | “Rob Reiner” |
| “Tom Hanks” | “Cloud Atlas” | “Andy Wachowski” |
| ... | ... | ... |

# Paths

```
MATCH (a)-[:ACTED_IN]->(m), (m)<-[:DIRECTED]-(d)
RETURN a.name, m.title, d.name;
```




neotechnology

# Paths

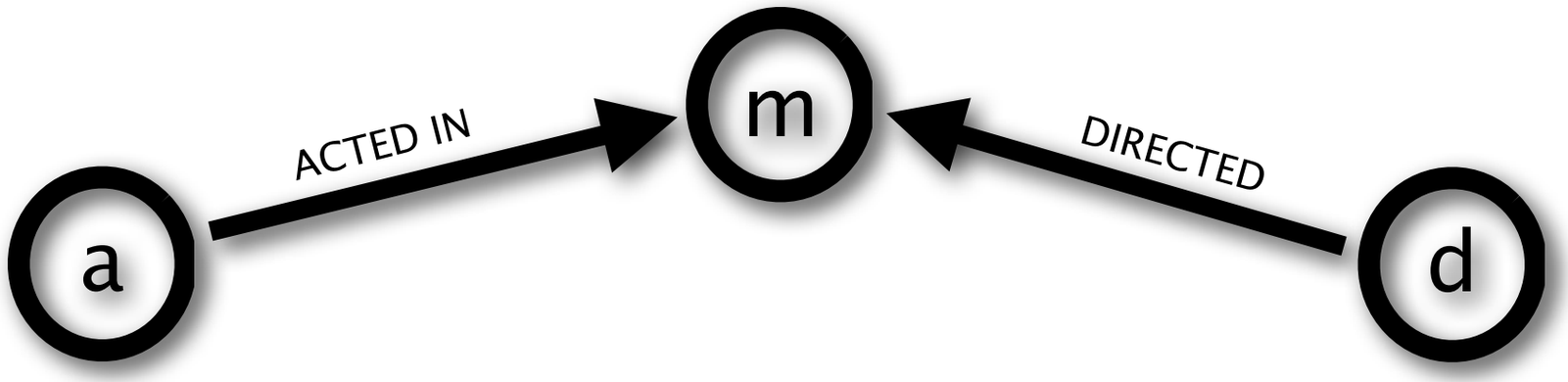
```
MATCH (a)-[:ACTED_IN]->(m), (d)-[:DIRECTED]->(m)
RETURN a.name, m.title, d.name;
```




neotechnology

# Paths

```
MATCH p=(a)-[:ACTED_IN]->(m)<-[:DIRECTED]-(d)
RETURN p;
```

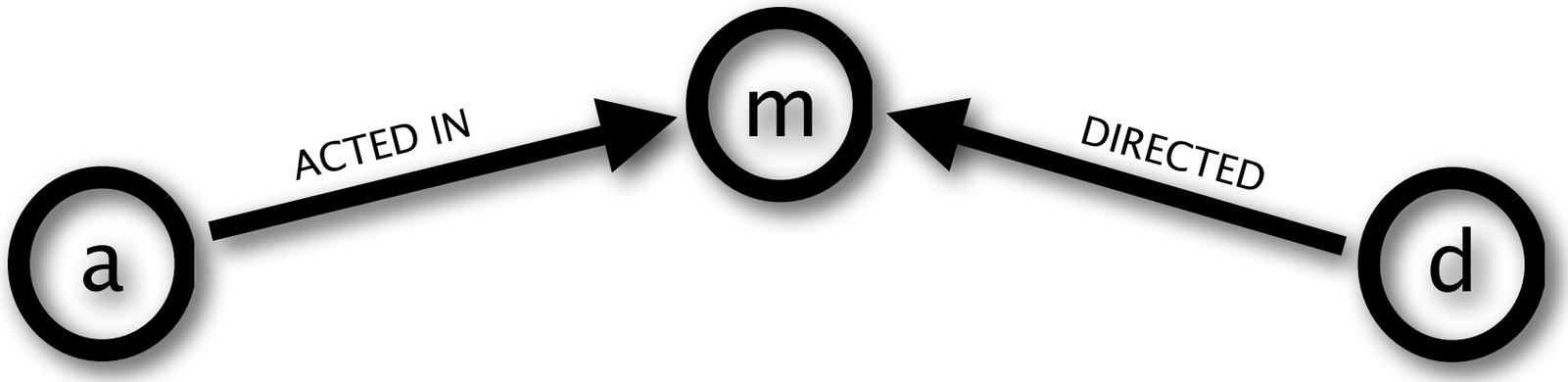





neotechnology

# Paths

```
MATCH p=(a)-[:ACTED_IN]->(m)<-[:DIRECTED]-(d)
RETURN nodes(p);
```






neotechnology

# Paths

```
MATCH p1=(a)-[:ACTED_IN]->(m), p2=(d)-[:DIRECTED]->(m)
RETURN p1, p2;
```




neotechnology

# Aggregation

```
MATCH (a)-[:ACTED_IN]->(m)<-[:DIRECTED]-(d)
RETURN a.name, m.title, d.name;
```

| a.name | m.title | d.name |
|---|---|---|
| "Keanu Reeves" | "The Matrix" | "Andy Wachowski" |
| "Keanu Reeves" | "The Matrix Reloaded" | "Andy Wachowski" |
| "Noah Wyle" | "A Few Good Men" | "Rob Reiner" |
| "Tom Hanks" | "Cloud Atlas" | "Andy Wachowski" |
| ... | ... | ... |

# Aggregation

```
MATCH (a)-[:ACTED_IN]->(m)<-[:DIRECTED]-(d)
RETURN a.name, d.name, count(*);
```

| a.name | d.name | count(*) |
|---|---|---|
| “Aaron Sorkin" | “Rob Reiner” | 2 |
| “Keanu Reeves” | “Andy Wachowski” | 3 |
| “Hugo Weaving” | “Tom Tykwer” | 1 |
| ... | ... | ... |

# Aggregation

```
MATCH (a)-[:ACTED_IN]->(m)<-[:DIRECTED]-(d)
RETURN a.name, d.name, count(m);
```

| a.name | d.name | count(m) |
|---|---|---|
| “Aaron Sorkin" | “Rob Reiner” | 2 |
| “Keanu Reeves” | “Andy Wachowski” | 3 |
| “Hugo Weaving” | “Tom Tykwer” | 1 |
| ... | ... | ... |

# LAB

Which directors also acted in their movie?

```
MATCH (d)-[:DIRECTED]->(m)<-[:ACTED_IN]-(d)
RETURN d.name, m.title;
```

# Unique relationships in paths

```
MATCH (a)-[:ACTED_IN]->(m)<-[:ACTED_IN]-(a)
RETURN a.name, m.title;
```

# Sort & Limit

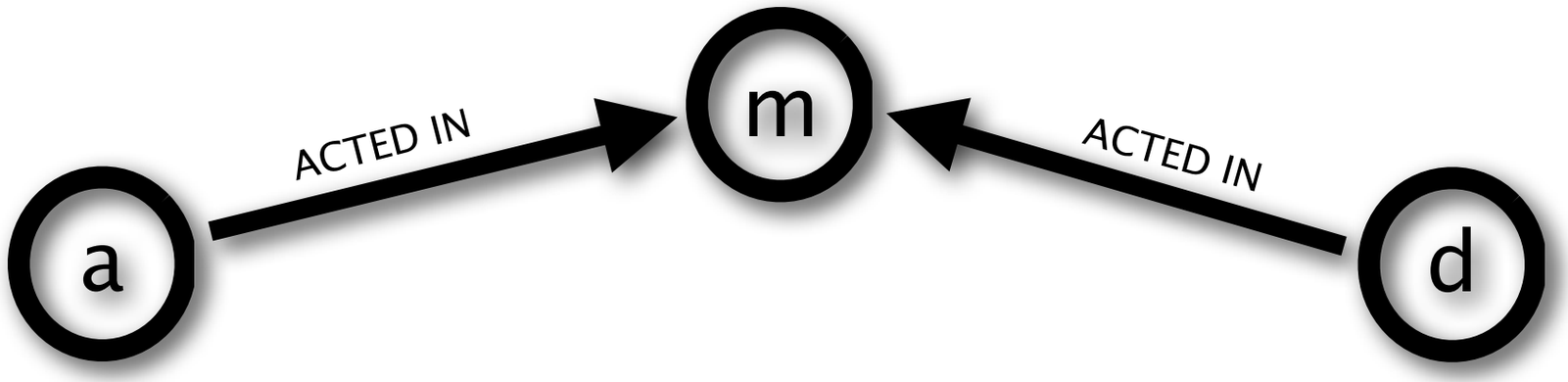
```
MATCH (a)-[:ACTED_IN]->(m)<-[:DIRECTED]-(d)
RETURN a.name, d.name, count(*) AS count
ORDER BY count DESC
LIMIT 5;
```

# Aggregation

| | |
|---|---|
| `count(x)` | *- add up the number of occurrences* |
| `min(x)` | *- get the lowest value* |
| `max(x)` | *- get the highest value* |
| `avg(x)` | *- get the average of a numeric value* |
| `sum(x)` | *- add up all values* |
| `collect(x)` | *- collected all the occurrences into an array* |

# Aggregation

```
MATCH (a)-[:ACTED_IN]->(m)<-[:DIRECTED]-(d)
RETURN a.name, d.name, collect(m.title);
```






neotechnology

# Starting somewhere

# All-nodes Query

```
MATCH (n)
RETURN n;
```

Matches every node in the graph



neotechnology

# Find a specific node (within all-nodes)

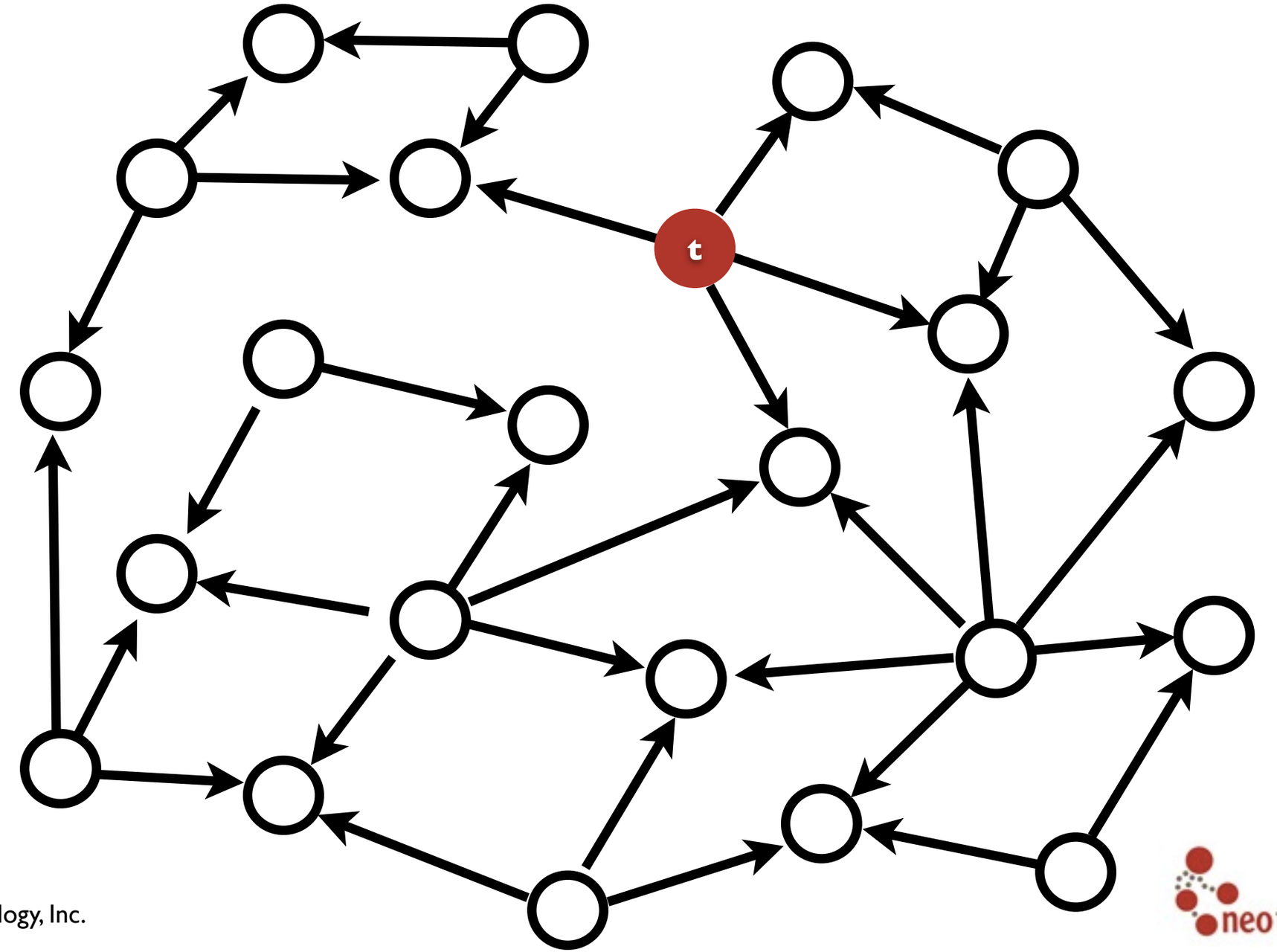
```
MATCH (n)
WHERE has(n.name) AND n.name = "Tom Hanks"
RETURN n;
```

`WHERE` - filter the results

`has(n.name)` - the name property must exist

`n.name = "Tom Hanks"` - and have that value

```
MATCH (n)
WHERE has(n.name) AND n.name = "Tom Hanks"
RETURN n;
```





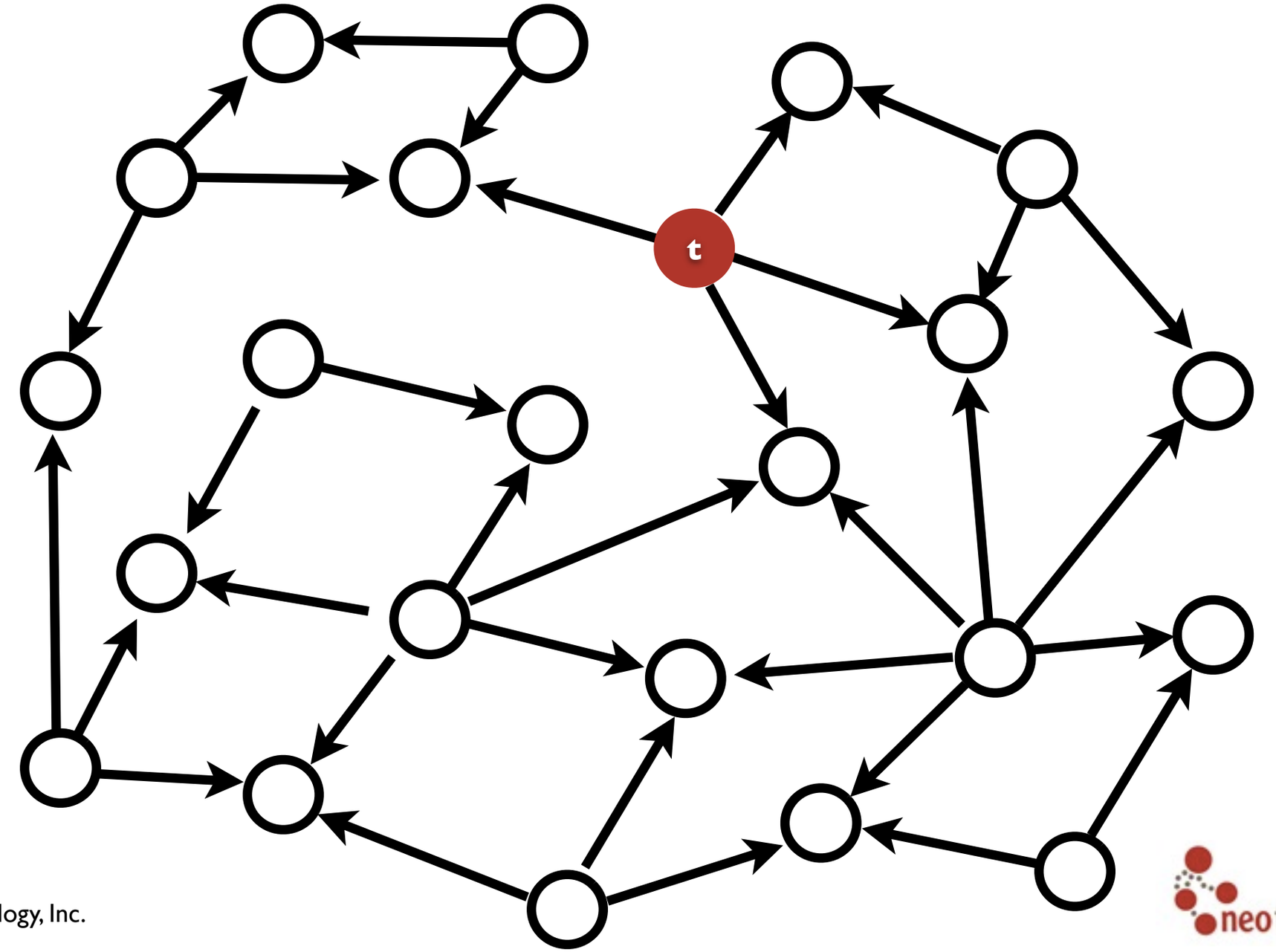

neotechnology

# Find a specific node (with a label)

```
MATCH (tom:Person)
WHERE tom.name="Tom Hanks"
RETURN tom;
```

`:Person` - Matches only nodes labeled as Person

```
MATCH (tom:Person)
WHERE tom.name="Tom Hanks"
RETURN tom;
```






neotechnology

# Start with a specific (labeled) node

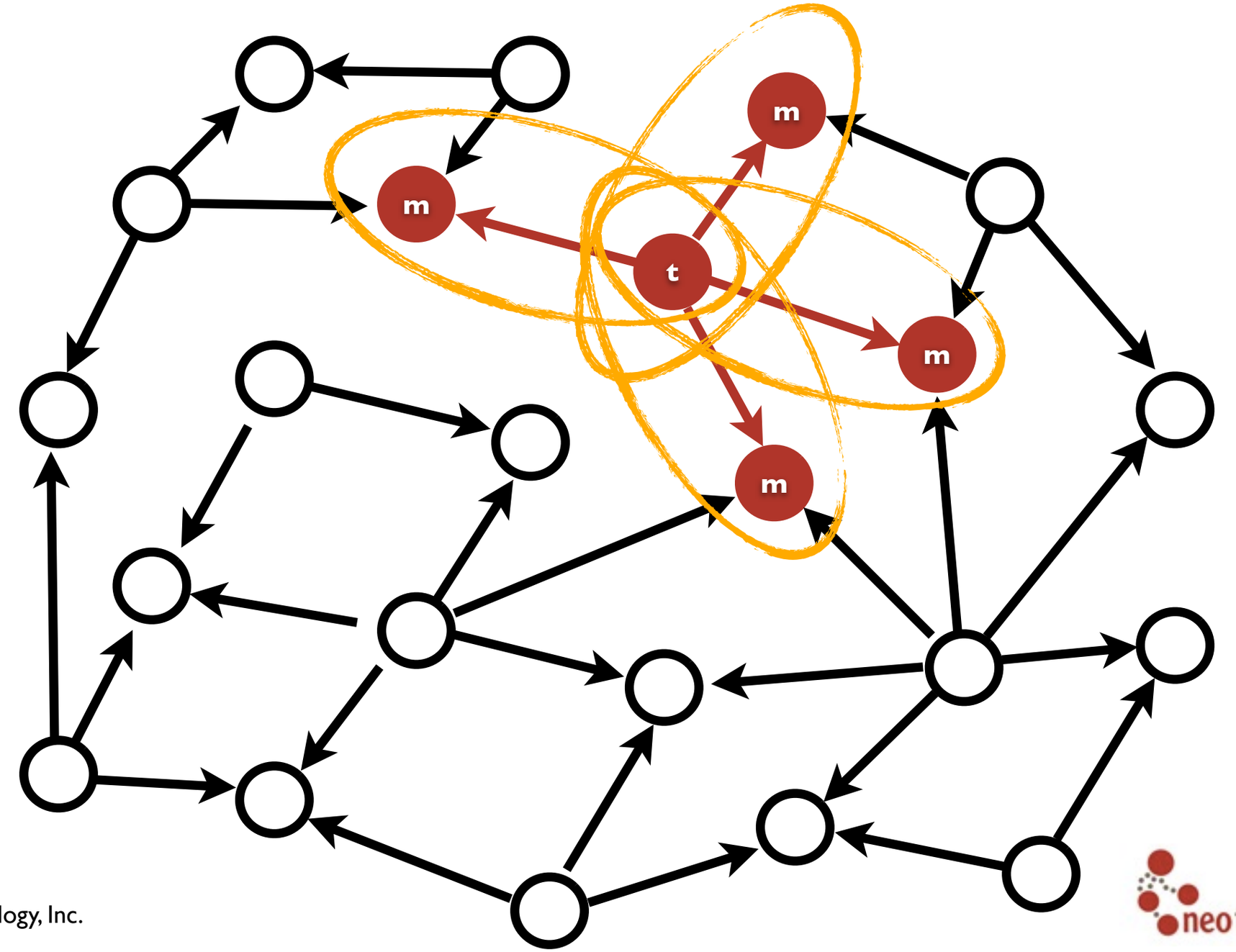
```
MATCH (tom:Person)-[:ACTED_IN]->(movie:Movie)
WHERE tom.name="Tom Hanks"
RETURN movie.title;
```

*(Movies featuring Tom Hanks)*



neotechnology

```
MATCH (tom:Person)-[:ACTED_IN]->(movie:Movie)
WHERE tom.name="Tom Hanks"
RETURN movie.title;
```





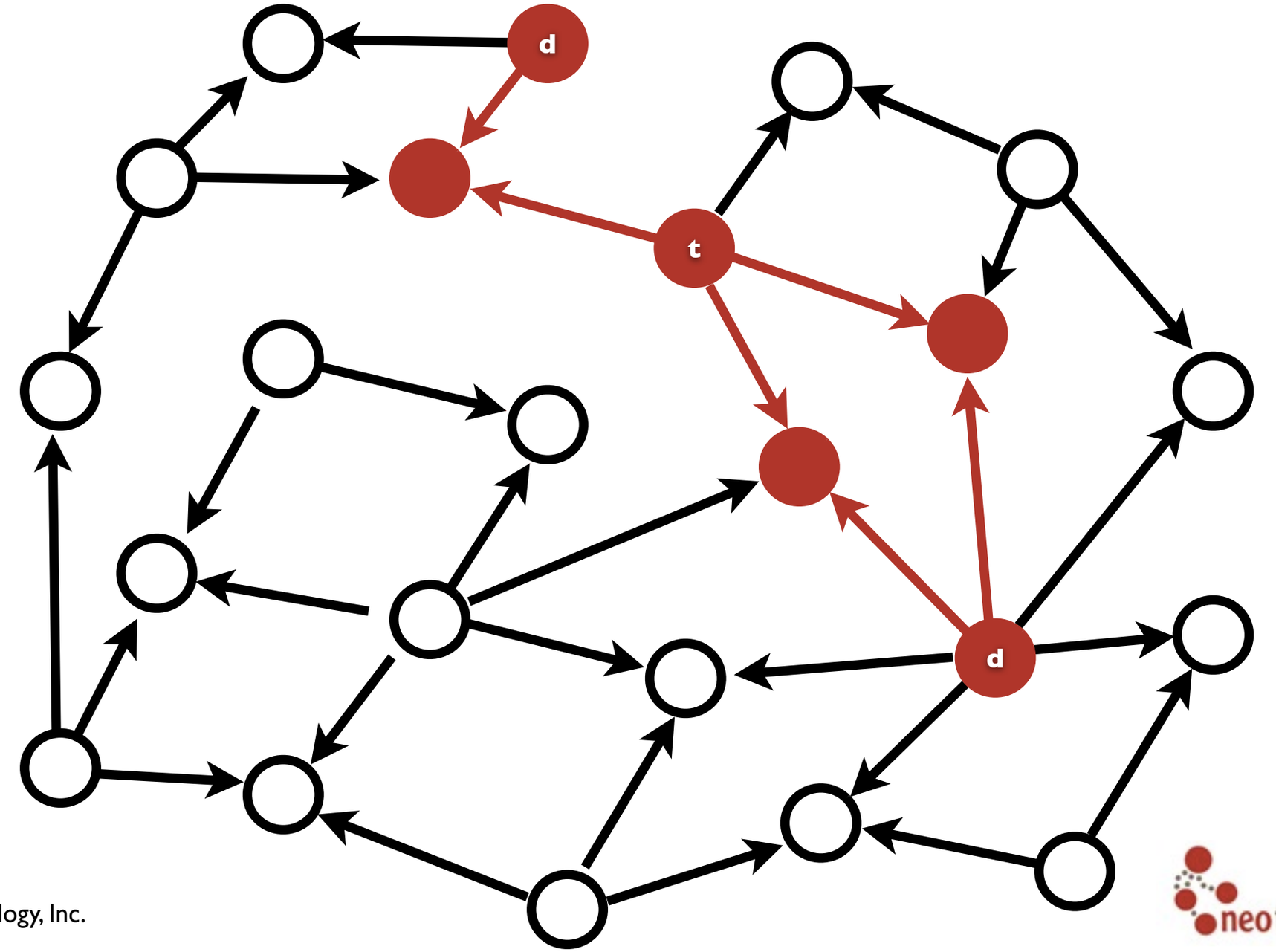

neotechnology

# Start with a specific labeled node

```
MATCH (tom:Person)-[:ACTED_IN]->(movie:Movie),
   (director:Person)-[:DIRECTED]->(movie:Movie)
WHERE tom.name="Tom Hanks"
RETURN director.name;
```

*(Directors who worked with Tom Hanks)*

```
MATCH (tom:Person)-[:ACTED_IN]->(movie:Movie)
  (director:Person)-[:DIRECTED]->(movie)
WHERE tom.name="Tom Hanks"
RETURN director.name;
```






neotechnology

# Start with a specific labeled node

```
MATCH (tom:Person)-[:ACTED_IN]->()<-[:DIRECTED]-(director)
WHERE tom.name="Tom Hanks"
RETURN DISTINCT director.name;
```

*(Directors who worked with Tom Hanks)*

# Create label-specific index

```
CREATE INDEX ON :Person(name)
CREATE INDEX ON :Movie(title)
```

Create two indexes.

- nodes labeled as Person, indexed by name property
- nodes labeled as Movie, indexed by title property



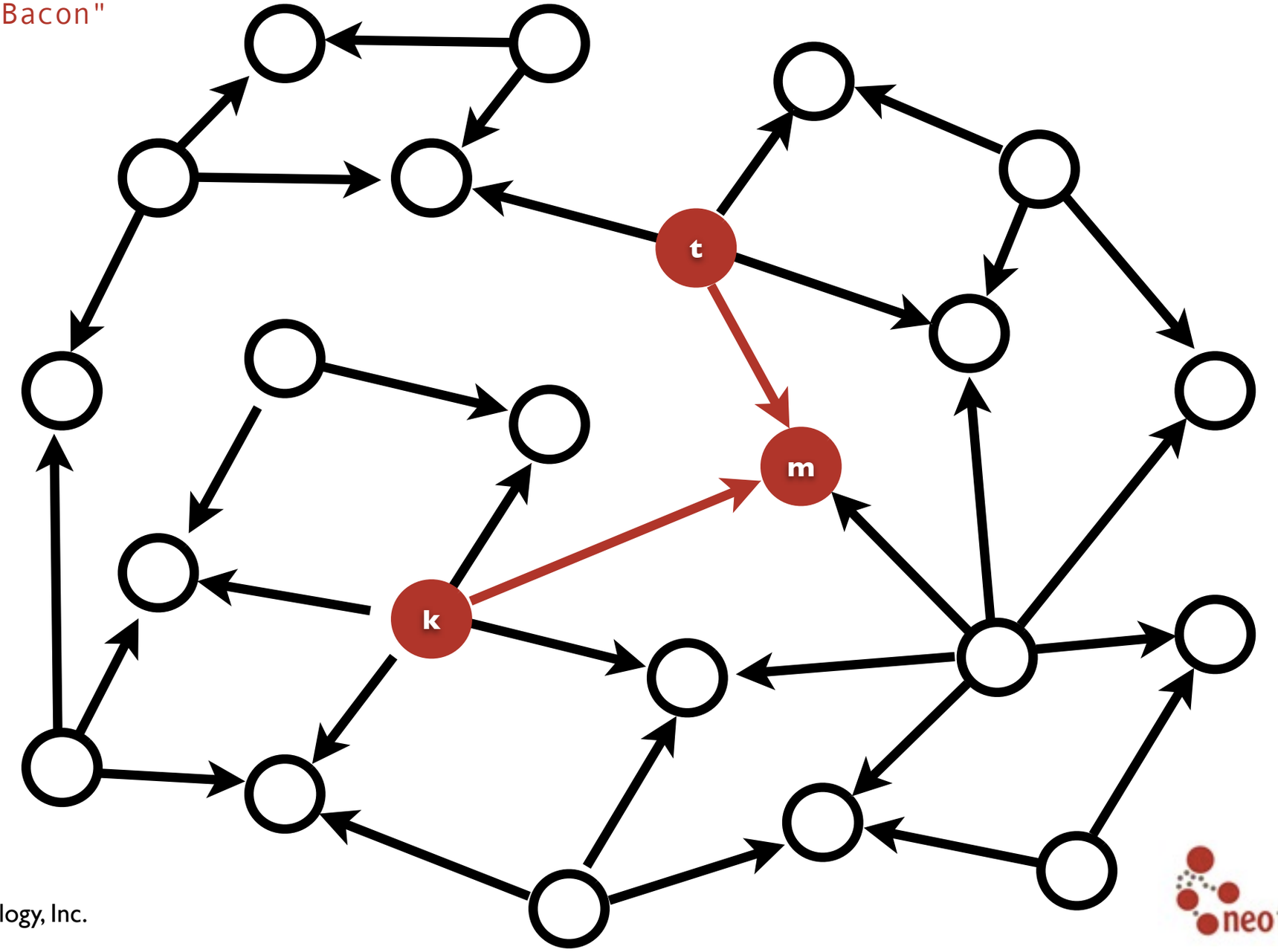
neo technology

# Labels on pattern "anchors"

```
MATCH (tom:Person)-[:ACTED_IN]->(movie),
  (kevin:Person)-[:ACTED_IN]->(movie)
WHERE tom.name="Tom Hanks" AND
      kevin.name="Kevin Bacon"
RETURN DISTINCT movie.title;
```

*(Movies featuring both Tom Hanks and Kevin Bacon)*

```
MATCH (tom:Person)-[:ACTED_IN]->(movie),
  (kevin:Person)-[:ACTED_IN]->(movie)
WHERE tom.name="Tom Hanks" AND
      kevin.name="Kevin Bacon"
RETURN movie.title;
```






neotechnology

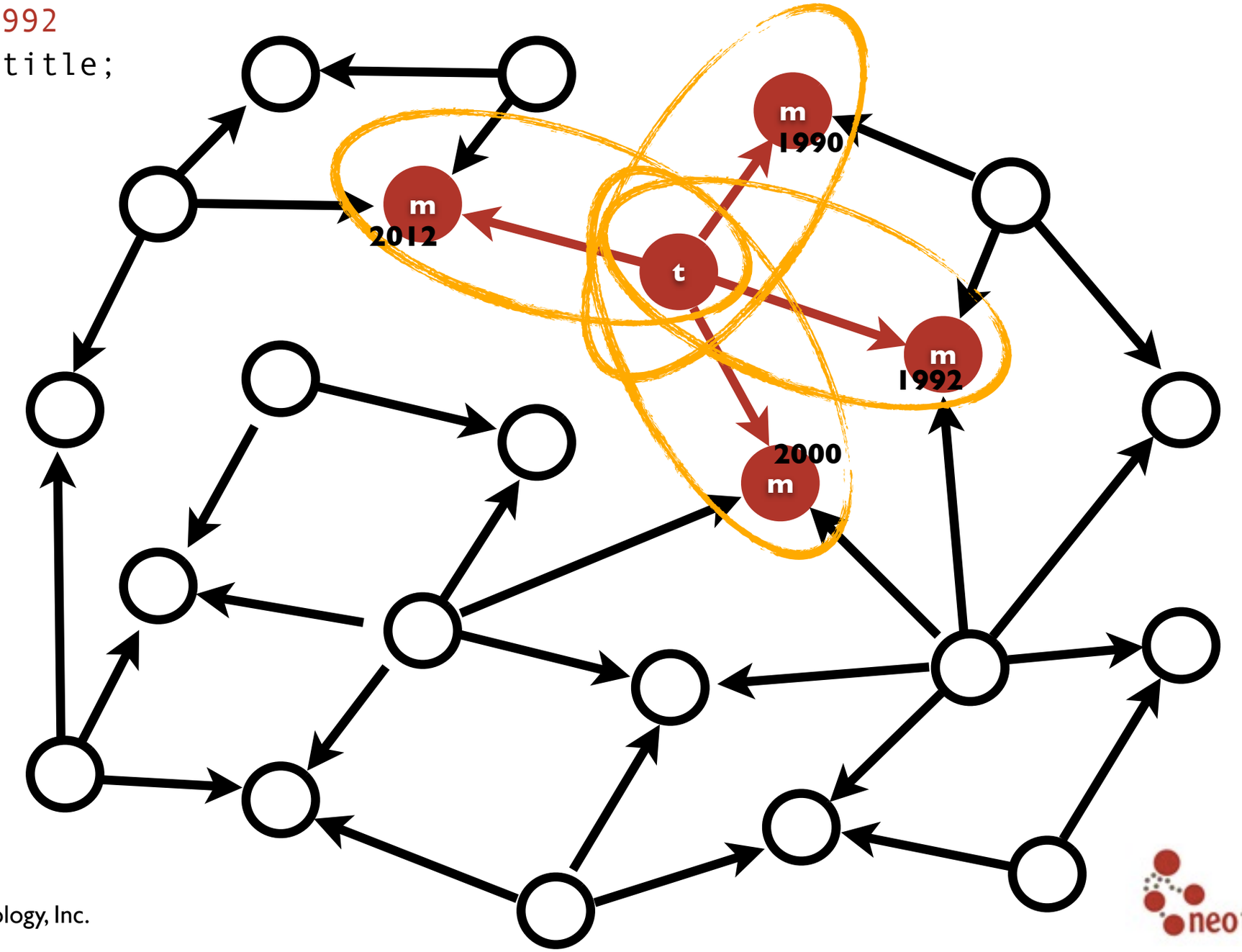
# Conditions

# Constraints on properties

```
MATCH (tom:Person)-[:ACTED_IN]->(movie:Movie)
WHERE tom.name="Tom Hanks"
AND movie.released < 1992
RETURN DISTINCT movie.title;
```

*(Movies in which Tom Hanks acted, that were released before 1992)*

```
MATCH (tom:Person)-[:ACTED_IN]->(movie:Movie)
WHERE tom.name="Tom Hanks"
AND movie.released < 1992
RETURN DISTINCT movie.title;
```






neotechnology

# Constraints on properties

```
MATCH (actor:Person)-[r:ACTED_IN]->(movie)
WHERE actor.name="Keanu Reeves"
AND "Neo" IN r.roles
RETURN DISTINCT movie.title;
```

*(Doesn‘t work in M06)*

*(Movies in which Keanu Reeves played Neo)*

# Constraints on properties

```
MATCH (actor:Person)-[r:ACTED_IN]->(movie)
WHERE actor.name="Keanu Reeves"
AND ANY( x IN r.roles WHERE x = "Neo")
RETURN DISTINCT movie.title;
```

*(Movies in which Keanu Reeves played Neo)*

# Constraints based on comparisons

```
MATCH (tom:Person)-[:ACTED_IN]->(movie),
  (a:Person)-[:ACTED_IN]->(movie)
WHERE tom.name="Tom Hanks"
AND a.born < tom.born
RETURN DISTINCT a.name;
```

*(Actors who worked with Tom and are older than he was)*

# Constraints based on comparisons

```
MATCH (tom:Person)-[:ACTED_IN]->(movie),
  (a:Person)-[:ACTED_IN]->(movie)
WHERE tom.name="Tom Hanks"
AND a.born < tom.born
RETURN DISTINCT a.name, (tom.born - a.born) AS diff;
```

*(Actors who worked with Tom and are older than he was)*

# Constraints based on patterns

```
MATCH (gene:Person)-[:ACTED_IN]->(movie),
  (n)-[:ACTED_IN]->(movie)
WHERE gene.name="Gene Hackman"
RETURN DISTINCT n.name;
```

*(Actors who worked with Gene Hackman)*

# Constraints based on patterns

```
MATCH (gene:Person)-[:ACTED_IN]->(movie),
  (n)-[:ACTED_IN]->(movie)
WHERE gene.name="Gene Hackman"
AND (n)-[:DIRECTED]->()
RETURN DISTINCT n.name;
```

*(Actors who worked with Gene and were directors of their own films)*

# Constraints based on patterns

```
MATCH (keanu:Person)-[:ACTED_IN]->(movie),
  (n)-[:ACTED_IN]->(movie),
 (hugo:Person)
WHERE keanu.name="Keanu Reeves" AND
      hugo.name="Hugo Weaving"
AND NOT (hugo)-[:ACTED_IN]->(movie)
RETURN DISTINCT n.name;
```

*(Actors who worked with Keanu, but not when he was also working with Hugo)*

# LAB

Who are the five busiest actors?

```
MATCH (a:Person)-[:ACTED_IN]->()
RETURN a.name, count(*) AS count
ORDER BY count DESC
LIMIT 5;
```



neo technology

# ADVANCED LAB

Recommend 3 actors that Keanu Reeves should work with (but hasn't).

```
MATCH (keanu:Person)-[:ACTED_IN]->()<-[:ACTED_IN]-(c),
      (c)-[:ACTED_IN]->()<-[:ACTED_IN]-(coc)
WHERE keanu.name="Keanu Reeves"
  AND NOT((keanu)-[:ACTED_IN]->()<-[:ACTED_IN]-(coc))
  AND coc <> keanu
RETURN coc.name, count(coc)
ORDER BY count(coc) DESC
LIMIT 3;
```



neo technology


# Coffee? Lunch?

```
MATCH (kevin:Person)-[:ACTED_IN]->(movie)
WHERE kevin.name="Kevin Bacon"
RETURN DISTINCT movie.title;
```

*(Movies featuring Kevin Bacon)*


# (Updating Graphs with Cypher)

# Creating nodes

```
CREATE (m:Movie {title:"Mystic River",
  released:1993});
```

```
MATCH (movie:Movie)
WHERE movie.title="Mystic River"
RETURN movie;
```



neotechnology

# Adding properties

```
MATCH (movie:Movie)
WHERE movie.title="Mystic River"
SET movie.tagline = "We bury our sins here, Dave. We wash them clean."
RETURN movie;
```

# Changing properties

```
MATCH (movie:Movie)
WHERE movie.title="Mystic River"
SET movie.released = 2003
RETURN movie;
```

# Creating Relationships

```
MATCH (movie:Movie),(kevin:Person)
WHERE movie.title="Mystic River" AND
      kevin.name="Kevin Bacon"
CREATE UNIQUE (kevin)-[:ACTED_IN {roles:["Sean"]}]->(movie)
```

```
MATCH (kevin)-[:ACTED_IN]->(movie)
WHERE kevin.name="Kevin Bacon"
RETURN movie.title;
```



neotechnology

# LAB

Change Kevin Bacon's role in Mystic River
from "Sean" to "Sean Devine"

```
MATCH (kevin:Person)-[r:ACTED_IN]->(movie:Movie)
WHERE movie.title="Mystic River" AND
      kevin.name="Kevin Bacon"
SET r.roles = ["Sean Devine"]
RETURN r.roles;
```



neo technology

# LAB

Change Kevin Bacon's role in Mystic River
from "Sean" to "Sean Devine"

```
MATCH (kevin:Person)-[r:ACTED_IN]->(movie:Movie)
WHERE movie.title="Mystic River" AND
      kevin.name="Kevin Bacon"
SET r.roles = [n in r.roles WHERE n <> "Sean"] + "Sean Devine"
RETURN r.roles;
```



neo technology

# LAB

Add Clint Eastwood as the director of Mystic River

```
MATCH (movie:Movie),(clint:Person)
WHERE movie.title="Mystic River" AND
      clint.name="Clint Eastwood"
CREATE UNIQUE (clint)-[:DIRECTED]->(movie);
```



neo technology

# LAB

List all the characters in the movie “The Matrix”

```
MATCH (matrix:Movie)<-[r:ACTED_IN]-()
WHERE matrix.title="The Matrix"
RETURN r.roles;
```

```
MATCH (matrix:Movie)<-[r:ACTED_IN]-(a)
WHERE matrix.title="The Matrix"
AND a.name =~ ".*Emil.*"
RETURN a;
```

# Deleting nodes

```
MATCH (emil:Person)
WHERE emil.name="Emil Eifrem"
DELETE emil;
```

*(Wrong Error Message)*



neotechnology

# Deleting relationships

```
MATCH (emil:Person)-[r]-()
WHERE emil.name="Emil Eifrem"
DELETE r;
```



neotechnology

# Deleting nodes and relationships

# Deleting nodes and relationships

```
START emil=node(*)
MATCH (emil)-[r?]->()
WHERE emil.name="Emil Eifrem"
DELETE r, emil;
```

START - used to scope starting points

node(*) - scope to all possible nodes



neotechnology

# LAB

Add KNOWS relationships between all actors who were in the same movie

```
MATCH (a:Person)-[:ACTED_IN]->()<-[:ACTED_IN]-(b:Person)
CREATE UNIQUE (a)-[:KNOWS]->(b);
```

# Match or Create is MERGE

```
MERGE (p:Person {name:"Clint Eastwood"})
RETURN p
```

Uses indexes, constraints and locks to guarantee unique lookup and creation

# Match or Create is MERGE

```
MERGE (p:Person {name:"Clint Eastwood"})
ON CREATE p SET p.created = timestamp()
ON MATCH p SET p.accessed = coalesce(p.accessed,0)+1
RETURN p
```

```
ON CREATE x SET - executed on create
ON MATCH x SET  - executed on match
```



neotechnology


# Coffee?

# (More Cypher)



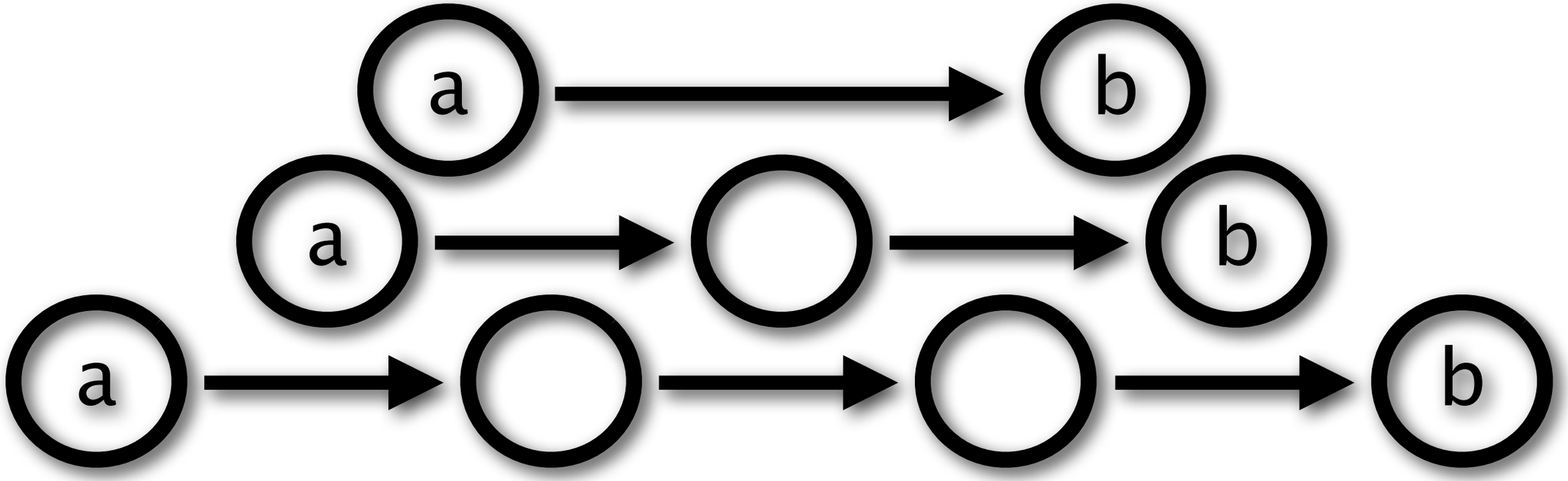
neotechnology

# Matching multiple relationships

```
MATCH (a)-[:ACTED_IN|:DIRECTED]->()<-[:ACTED_IN|:DIRECTED]-(b)
CREATE UNIQUE (a)-[:KNOWS]->(b);
```

*(Create KNOWS relationships between anyone, Actors or Directors, who worked together)*

# Variable length paths




```
(a)-[*1..3]->(b)
```



neotechnology

# Friends-of-Friends

```
MATCH (keanu:Person)-[:KNOWS*2]->(fof)
WHERE keanu.name="Keanu Reeves"
RETURN DISTINCT fof.name;
```



neotechnology

# LAB

Return Friends-of-Friends who are not immediate friends

```
MATCH (keanu:Person)-[:KNOWS*2]->(fof)
WHERE keanu.name="Keanu Reeves"
AND NOT (keanu)-[:KNOWS]-(fof)
RETURN DISTINCT fof.name;
```

# Bacon Number!

```
MATCH p=shortestPath(
   (charlize:Person)-[:KNOWS*]->(bacon:Person)
  )
WHERE charlize.name="Charlize Theron" AND
      bacon.name="Kevin Bacon"
RETURN length(rels(p));
```



neotechnology

# LAB

Return the names of the people joining Charlize to Kevin.

```
MATCH p=shortestPath((charlize:Person)-[:KNOWS*]->(bacon:Person))
WHERE charlize.name="Charlize Theron" AND
      bacon.name="Kevin Bacon"
RETURN [n in nodes(p) | n.name] AS names;
```

# (Graphs are Everywhere)

Neo4j Case Studies



neotechnology

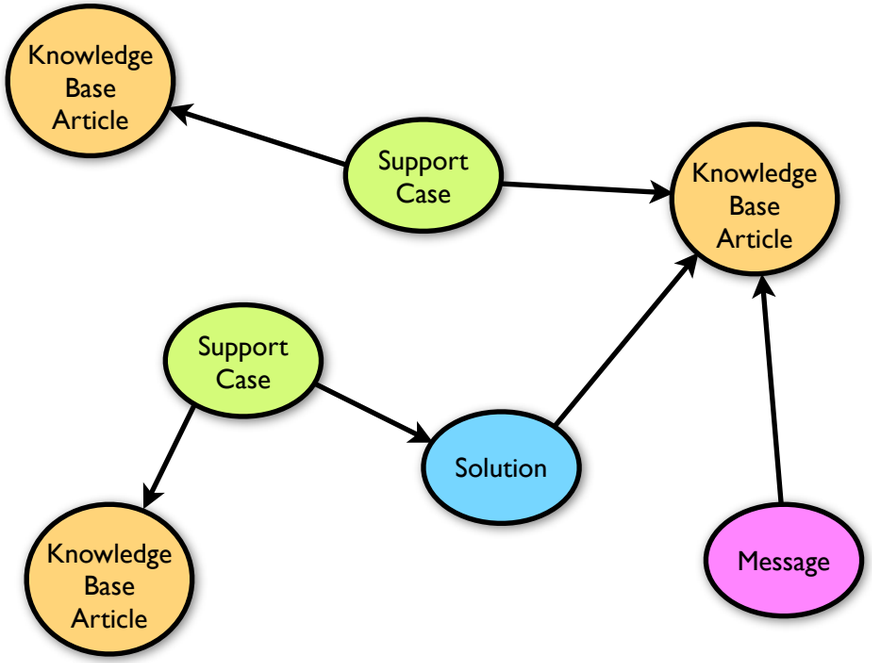



neotechnology

*Industry: Communications*
*Use case: Recommendations*
**San Jose, CA**
**Cisco.com**

## Background

- Cisco.com serves customer and business customers with Support Services
- Needed real-time recommendations, to encourage use of online knowledge base
- Cisco had been successfully using Neo4j for its internal master data management solution.
  - Identified a strong fit for online recommendations




## Business problem

- Call center volumes needed to be lowered by improving the efficacy of online self service
- Leverage large amounts of knowledge stored in service cases, solutions, articles, forums, etc.
- Problem resolution times, as well as support costs, needed to be lowered

## Solution & Benefits

- Cases, solutions, articles, etc. continuously scraped for cross-reference links, and represented in Neo4j
- Real-time reading recommendations via Neo4j
- Neo4j Enterprise with HA cluster
- The result: customers obtain help faster, with decreased reliance on customer support

*Industry: Communications*
*Use case: Master Data Management*
**San Jose, CA**

# Cisco HMP

## Background

- One of the world's largest communications equipment manufacturers
- #91 Global 2000. $44B in annual sales.
- Needed a system that could accommodate its master data hierarchies in a performant way
- HMP is a Master Data Management system at whose heart is Neo4j. Data access services available 24x7 to applications companywide


## Business problem

- Sales compensation system had become unable to meet Cisco's needs
- Existing Oracle RAC system had reached its limits:
  - Insufficient flexibility for handling complex organizational hierarchies and mappings
  - "Real-time" queries were taking > 1 minute!
- Business-critical "P1" system needs to be continually available, with zero downtime

## Solution & Benefits

- Cisco created a new system: the Hierarchy Management Platform (HMP)
- Allows Cisco to manage master data centrally, and centralize data access and business rules
- Neo4j provided "Minutes to Milliseconds" performance over Oracle RAC, serving master data in real time
- The graph database model provided exactly the flexibility needed to support Cisco's business rules
- HMP so successful that it has expanded to include product hierarchy

*Industry: Logistics*
*Use case: Parcel Routing*

accenture

## Background

- One of the world's largest logistics carriers
- Projected to outgrow capacity of old system
- New parcel routing system
  - Single source of truth for entire network
  - B2C & B2B parcel tracking
  - Real-time routing: up to 5M parcels per day

## Business problem

- 24x7 availability, year round
- Peak loads of 2500+ parcels per second
- Complex and diverse software stack
- Need predictable performance & linear scalability
- Daily changes to logistics network: route from any point, to any point

## Solution & Benefits

- Neo4j provides the ideal domain fit:
  - a logistics network is a graph
- Extreme availability & performance with Neo4j clustering
- Hugely simplified queries, vs. relational for complex routing
- Flexible data model can reflect real-world data variance much better than relational
- "Whiteboard friendly" model easy to understand





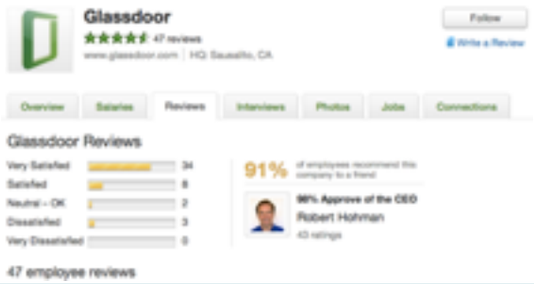
neotechnology

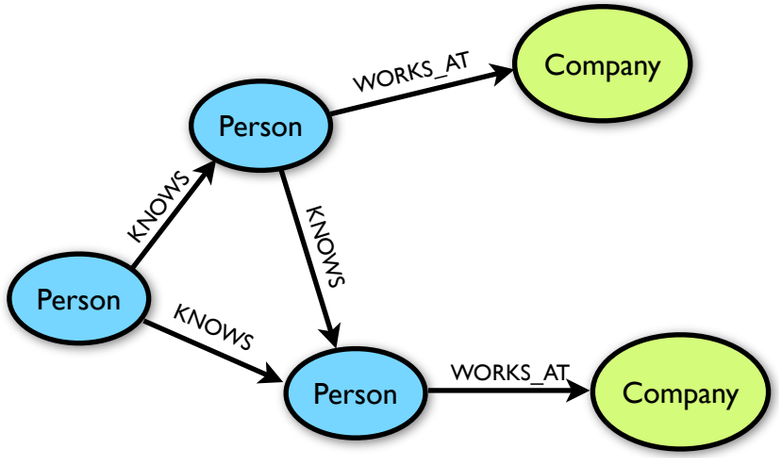
*Industry: Online Job Search*
*Use case: Social / Recommendations*
**Sausalito, CA**

# GlassDoor

## Background

- Online jobs and career community, providing anonymized inside information to job seekers




## Business problem

- Wanted to leverage known fact that most jobs are found through personal & professional connections
- Needed to rely on an existing source of social network data. Facebook was the ideal choice.
- End users needed to get instant gratification
- Aiming to have the best job search service, in a very competitive market

## Solution & Benefits

- First-to-market with a product that let users find jobs through their network of Facebook friends
- Job recommendations served real-time from Neo4j
- Individual Facebook graphs imported real-time into Neo4j
- Glassdoor now stores > 50% of the entire Facebook social graph
- Neo4j cluster has grown seamlessly, with new instances being brought online as graph size and load have increased

*Industry: Web/ISV*
*Use case: Content Management, Social, Access Control*
**San Jose, CA**

# Adobe

## Background

- One of the ten largest software companies globally
- $4B+ in revenue. Over 11,000 employees.
- Launched Creative Cloud in 2012, allowing its Creative Suite users to collaborate via the Cloud


## User-Content-Access Graph


## Business problem

- Adobe needed a highly robust and available, 24x7 distributed global system, supporting collaboration for users of its highest revenue product line
- Storing creative artifacts in the cloud meant managing access rights for (eventually) millions of users, groups, collections, and pieces of content
- Complex access control rules controlling who was connected to whom, and who could see or edit what, proved a significant technical challenge

## Solution & Benefits

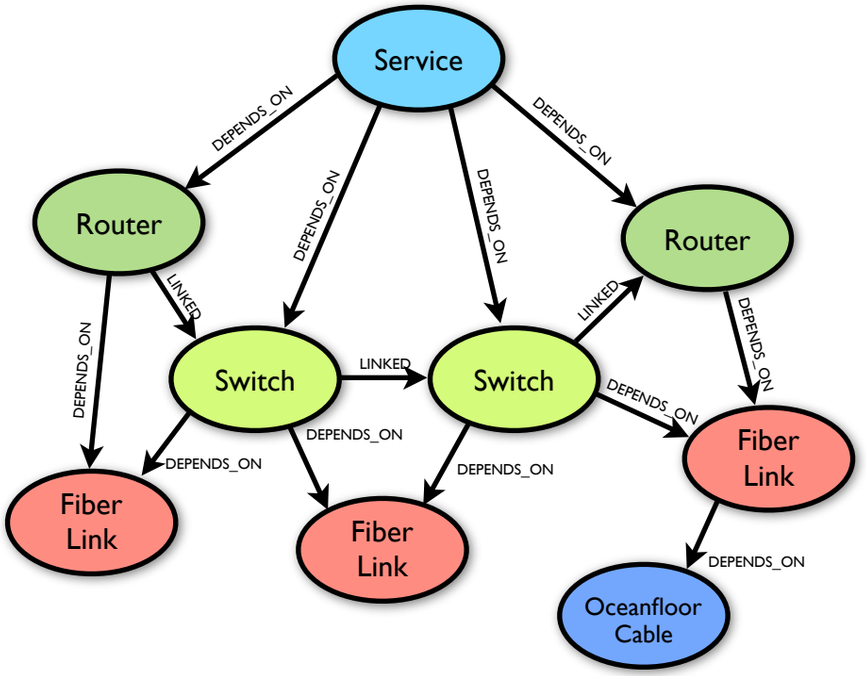
- Selected Neo4j to meet very aggressive project deadlines. The flexibility of the graph model, and performance, were the two major selection factors.
- Easily evolve the system to meet tomorrow's needs
- Extremely high availability and transactional performance requirements. 24x7 with no downtime.
- Neo4j allows consistently fast response times with complex queries, even as the system grows
- First (and possibly still only) database cluster to run across three Amazon EC2 regions: U.S., Europe, Asia

*Industry: Communications*
*Use case: Network Management*
**Paris, France**

**SFR**

# Background

- Second largest communications company in France
- Part of Vivendi Group, partnering with Vodafone




# Business problem

- Infrastructure maintenance took one full week to plan, because of the need to model network impacts
- Needed rapid, automated "what if" analysis to ensure resilience during unplanned network outages
- Identify weaknesses in the network to uncover the need for additional redundancy
- Network information spread across > 30 systems, with daily changes to network infrastructure
- Business needs sometimes changed very rapidly

# Solution & Benefits

- Flexible network inventory management system, to support modeling, aggregation & troubleshooting
- Single source of truth (Neo4j) representing the entire network
- Dynamic system loads data from 30+ systems, and allows new applications to access network data
- Modeling efforts greatly reduced because of the near 1:1 mapping between the real world and the graph
- Flexible schema highly adaptable to changing business requirements

*Industry: Communications*
*Use case: Social gaming*
**Frankfurt, Germany**

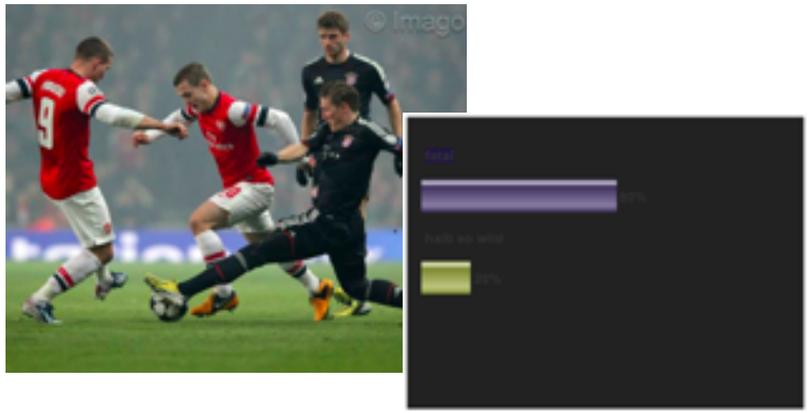
# Deutsche Telecom

## Background

- Europe's largest communications company
- Provider of mobile & land telephone lines to consumers and businesses, as well as internet services, television, and other services

| > 236,000 | 50 | > 58 bn. € |
|---|---|---|
| Employees worldwide in 2011 | Countries | Revenue in 2011 |

## Interactive Television Programming



## Business problem

- The Fanorakel application allows fans to have an interactive experience while watching sports
- Fans can vote for referee decisions and interact with other fans watching the game
- Highly connected dataset with real-time updates
- Queries need to be served real-time on rapidly changing data
- One technical challenge is to handle the very high spikes of activity during popular games

## Solution & Benefits

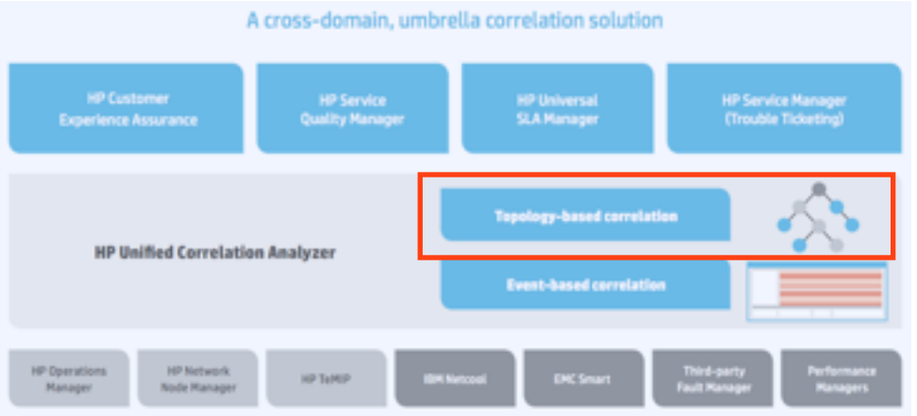
- Interactive, social offering gives fans a way to experience the game more closely
- Increased customer stickiness for Deutsche Telekom
- A completely new channel for reaching customers with information, promotions, and ads
- Clear competitive advantage

*Industry: Web/ISV, Communications*
*Use case: Network Management*
**Global (U.S., France)**

**Hewlett Packard**

# Background

- World's largest provider of IT infrastructure, software & services
- HP's Unified Correlation Analyzer (UCA) application is a key application inside HP's OSS Assurance portfolio
- Carrier-class resource & service management, problem determination, root cause & service impact analysis
- Helps communications operators manage large, complex and fast changing networks




# Business problem

- Use network topology information to identify root problems causes on the network
- Simplify alarm handling by human operators
- Automate handling of certain types of alarms Help operators respond rapidly to network issues
- Filter/group/eliminate redundant Network Management System alarms by event correlation

# Solution & Benefits

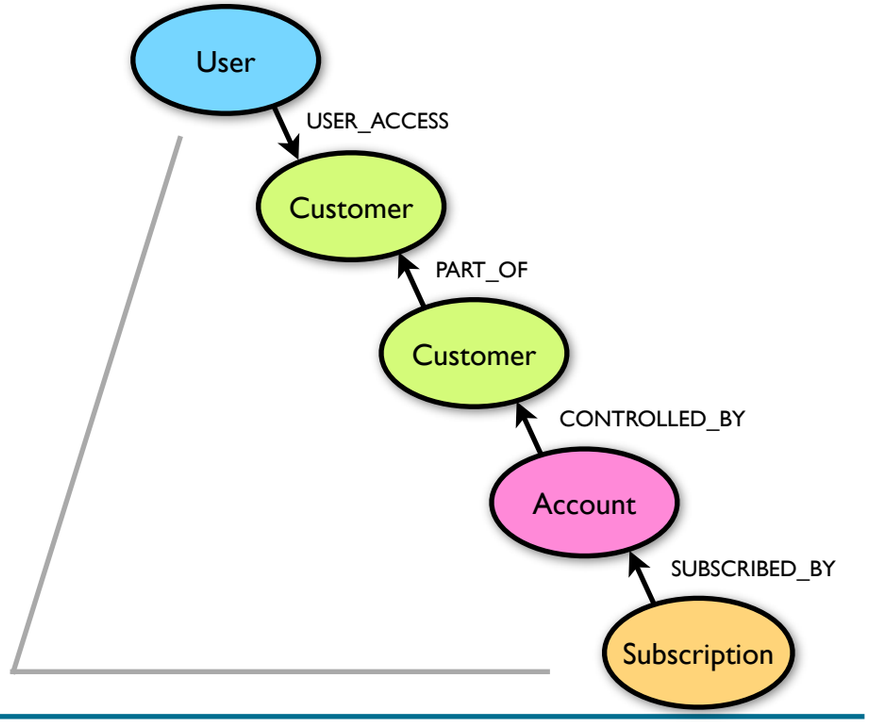
- Accelerated product development time
- Extremely fast querying of network topology
- Graph representation a perfect domain fit
- 24x7 carrier-grade reliability with Neo4j HA clustering
- Met objective in under 6 months

*Industry: Communications*
*Use case: Resource Authorization & Access Control*
**Oslo, Norway**

# Telenor

## Background

- 10th largest Telco provider in the world, leading in the Nordics
- Online self-serve system where large business admins manage employee subscriptions and plans
- Mission-critical system whose availability and responsiveness is critical to customer satisfaction




## Business problem

- Degrading relational performance. User login taking minutes while system retrieved access rights
- Millions of plans, customers, admins, groups.
  Highly interconnected data set w/massive joins
- Nightly batch workaround solved the performance problem, but meant data was no longer current
- Primary system was Sybase. Batch pre-compute workaround projected to reach 9 hours by 2014: longer than the nightly batch window

## Solution & Benefits

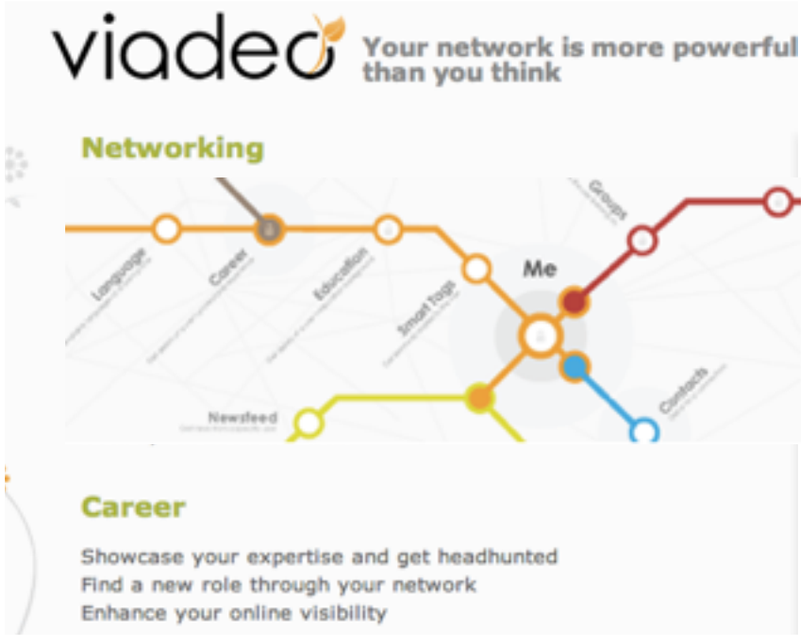
- Moved authorization functionality from Sybase to Neo4j
- Modeling the resource graph in Neo4j was straightforward, as the domain is inherently a graph
- Able to retire the batch process, and move to real-time responses: measured in milliseconds
- Users able to see fresh data, not yesterday's snapshot
- Customer retention risks fully mitigated

*Industry: Professional Social Network*
*Use case: Social, Recommendations*
**Silicon Valley & France**

# Viadeo

## Background

- World's second-largest professional network (after LinkedIn)
- 50M members. 30K+ new members daily.
- Over 400 staff with offices in 12 countries




## Business problem

- Business imperative for real-time recommendations: to attract new users and retain existing ones
- Key differentiator: show members how they are connected to any other member
- Real-time traversals of social graph not feasible with MySQL cluster. Batch precompute meant stale data.
- Process taking longer & longer: > 1 week!

## Solution & Benefits

- Neo4j solution implemented in 8 weeks with 3 part-time programmers
- Able to move from batch to real-time: improved responsiveness with up-to-date data.
- Viadeo (at the time) had 8M members and 35M relationships.
- Neo4j cluster now sits at the heart of Viadeo's professional network, connecting 50M + professionals

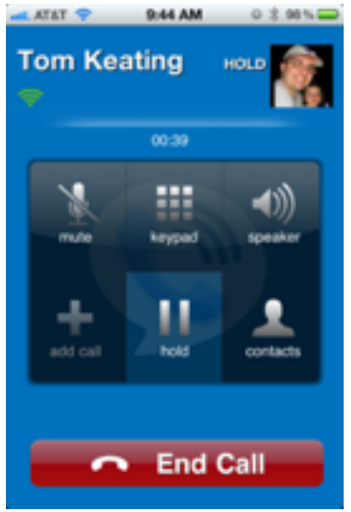
*Industry: Communications*
*Use case: Social, Mobile*

**Hong Kong**

# Maaii

## Background

- Hong Kong based telephony infrastructure provider (aka M800 aka Pop Media)
- Exclusive China Mobile partner for international toll-free services. SMS Hub & other offerings
- 2012 Red Herring Top 100 Global Winner


## Business problem

- Launched a new mobile communication app "Maaii" allowing consumers to communicate by voice & text
  (Similar to Line, Viber, Rebtel, VoxOx...)
- Needed to store & relate devices, users, and contacts
- Import phone numbers from users' address books. Rapidly serve up contacts from central database
  to the mobile app
- Currently around 3M users w/200M nodes in the graph

## Solution & Benefits

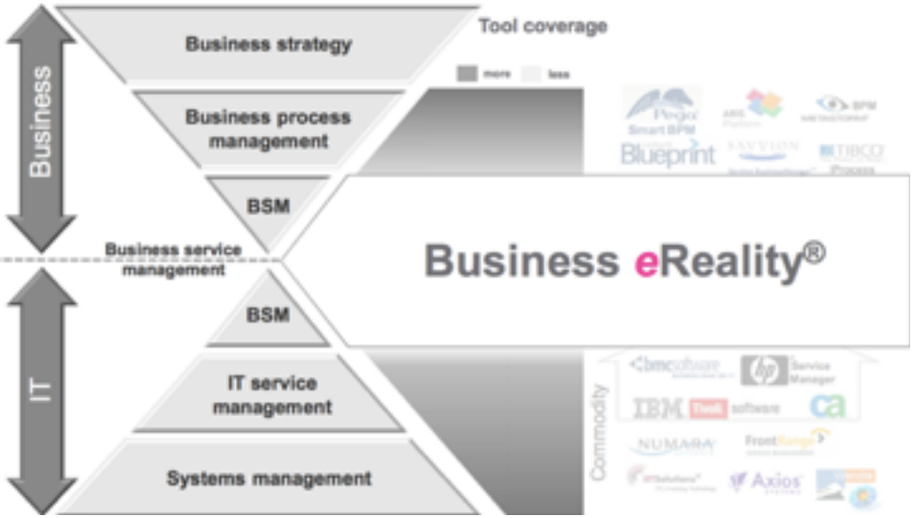
- Quick transactional performance for key operations:
  - friend suggestions ("friend of friend")
  - updating contacts, blocking calls, etc.
  - etc.
- High availability telephony app uses Neo4j clustering
- Strong architecture fit: Scala w/Neo4j embedded

*Industry: Web/ISV, Communications*
*Use case: Data Center Management*
**Zürich, Switzerland**

# Junisphere

## Background

- Junisphere AG is a Zurich-based IT solutions provider
- Founded in 2001. Profitable. Self funded.
- Software & services.
- Novel approach to infrastructure monitoring:
  Starts with the end user, mapped to business processes and services, and dependent infrastructure




## Business problem

- "Business Service Management" requires mapping of complex graph, covering: business processes--> business services--> IT infrastructure
- Embed capability of storing and retrieving this information into OEM application
- Re-architecting outdated C++ application based on relational database, with Java

## Solution & Benefits

- Actively sought out a Java-based solution that could store data as a graph
- Domain model is reflected directly in the database:
  - "No time lost in translation"
  - "Our business and enterprise consultants now speak the same language, and can model the domain with the database on a 1:1 ratio."
- Spring Data Neo4j strong fit for Java architecture

*Industry: Education*
*Use case: Resource Authorization & Access Control*
**San Francisco, CA**

# Teachscape

## Background

- Teachscape, Inc. develops online learning tools for K-12 teachers, school principals, and other instructional leaders.
- Teachscape evaluated relational as an option, considering MySQL and Oracle.
- Neo4j was selected because the graph data model provides a more natural fit for managing organizational hierarchy and access to assets.


## Business problem

- Neo4j was selected to be at the heart of a new architecture.
- The user management system, centered around Neo4j, will be used to support single sign-on, user management, contract management, and end-user access to their subscription entitlements.

## Solution & Benefits

- ***Domain and technology fit***
  - simple domain model where the relationships are relatively complex. Secondary factors included support for transactions, strong Java support, and well-implemented Lucene indexing integration
- ***Speed and Flexibility***
  - The business depends on being able to do complex walks quickly and efficiently. This was a major factor in the decision to use Neo4j.
- ***Ease of Use***
  - accommodate efficient access for home-grown and commercial off-the-shelf applications, as well as ad-hoc use.
  - Extreme availability & performance with Neo4j clustering
  - Hugely simplified queries, vs. relational for complex routing
  - Flexible data model can reflect real-world data variance much better than relational
  - "Whiteboard friendly" model easy to understand

*Industry: Life Sciences*
*Use case: Content Management*
**Cambridge, Massachusetts**

# SevenBridges Genomics

## Background

- Bioinformatics company offering gene sequencing "as a service" (over the web)
- Provider of genomic information services
- Needed a new platform to support storage & retrieval of sequenced genomes in the cloud


## Business problem

- Neo4j is used to store metadata about each sequenced genome (including a pointer to the sequenced genome itself, which is a binary file stored on Amazon S3), and to support search and other forms of information processing against the genomic data.
- graph database was chosen because "Our specific domain maps naturally onto graph paradigm".

## Solution & Benefits

- ***Domain fit***
  - Domain naturally lends itself to a graph representation.
  - Graph model determined to be a perfect fit.
- ***Agility & Performance***
  - Saved time with Neo4j as compared to the alternatives.
  - Queries "practically write themselves."
- ***Solution Completeness***
  - "Neo4j is incomparably better than other graph databases."


# (More about Neo4j)

# Neo4j APIs

Neo4j offers three APIs:

1. Cypher for most work
2. REST for management
3. Plugin API for special cases

# Language Drivers

Friends of Neo4j speak many languages, and work in many frameworks.

### Neo4j REST API

**Neo4j Team**

Discoverable, language-neutral data access from anything that can send HTTP requests. You could write a whole application with just bash scripts and curl.

### Spring Data Neo4j

**Neo4j Team**

Familiar POJO-based development, enabling object-to-graph mapping using annotations. Amazingly simple, with full graph power just a traversal query away.

### Java API

**Neo4j Team**

For intimate access, talk directly to Neo4j's graph engine directly in your JVM based application. Full feature parity with Neo4j Server, including HA clustering.


### neo4j.rb

**Andreas Ronge**

Ruby on Rails? Try coasting along graph paths with Neo4j. Everything you know and love, wrapped with graph glory.


### Neography

**Max de Marzi**

For native Ruby access to Neo4j, Neography provides a thin, elegant wrapper around the REST API.


### Neo4jPHP

**Josh Adell**

Neo4jPHP provides an API that is both intuitive and flexible, and it takes advantage of 'under-the -hood' performance enhancements, such as caching and lazy-loading.

http://www.neo4j.org/develop/drivers



neotechnology

Traverses 1,000,000+ relationships / second on commodity hardware



neo technology

# Scale: 34 Bn nodes/relationships



neo technology

# Licensing options


# (Next steps)

# The rise of the graph



neo technology

# It's not just social



neo technology

# Find the graphs in your domain, and model them



neo technology

# Get involved in the community



neo technology


http://stackoverflow.com/questions/tagged/neo4j



neotechnology

## Neo4j Google Group

Share your experiences and expertise with fellow graphistas.

Join now »

http://groups.google.com/group/neo4j



neotechnology

## GitHub Issues

Encountered an issue with Neo4j?
Submit it here.

https://github.com/neo4j/neo4j/issues



neotechnology

# Meetups / User Groups


Neo4j meetups are worldwide. Make a connection or start a new group.

**Join a Meetup »**

http://neo4j.meetup.com/



neotechnology

# Graphistas World Map

Add yourself to the graphistas world map and let it become a smaller place.

**Add yourself »**

http://www.neo4j.org/participate/contributors#map



neotechnology


# (Thank You)

http://www.neo4j.org

http://groups.google.com/group/neo4j

http://www.neo4j.org/develop