Intro to Neo4j and Graph Databases

Neo4j Webinar March 2016

ABOUT ME

- William Lyon
- Developer Relations Engineer @neo4j
- http://neo4j.com/developer

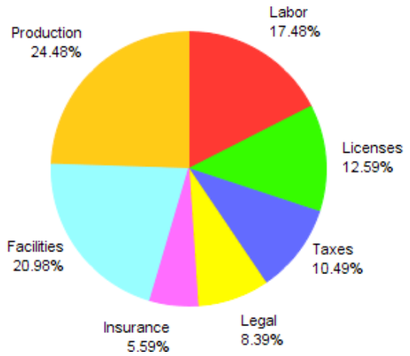
will@neo4j.com

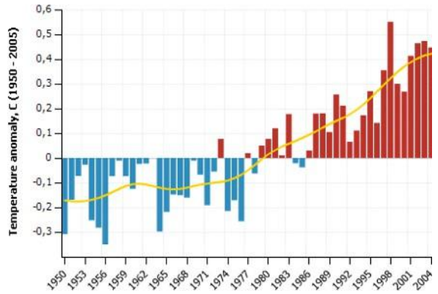
@lyonwj

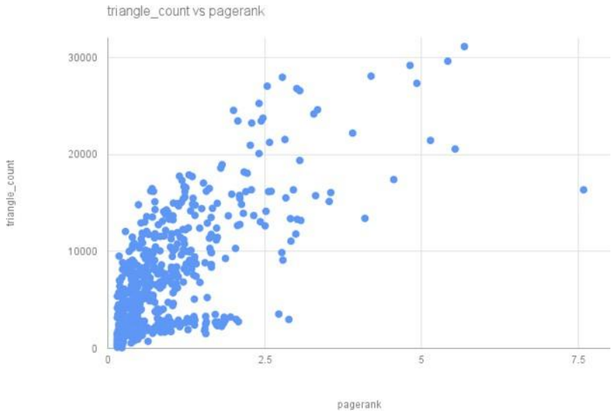
Agenda

- What is a graph [database]?
- Use cases - why graphs?
- Neo4j product overview
  - Labeled property graph data model
  - Cypher query language
- RDBMS to graph
- Resources
- Questions?

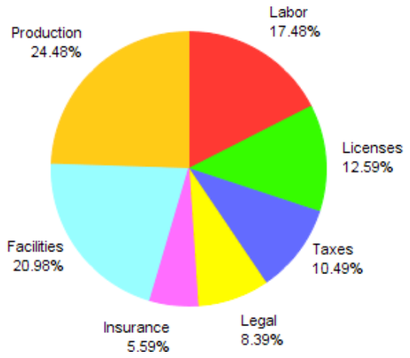
Chart

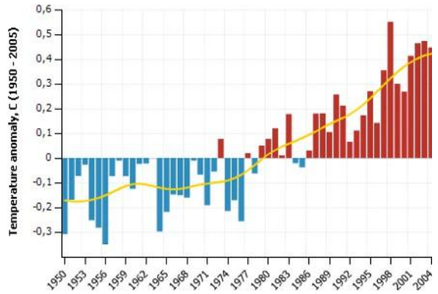

Labor
17.48%
Licenses
12.59%
Taxes
10.49%
Legal
8.39%
Insurance
5.59%
Facilities
20.98%
Production
24.48%

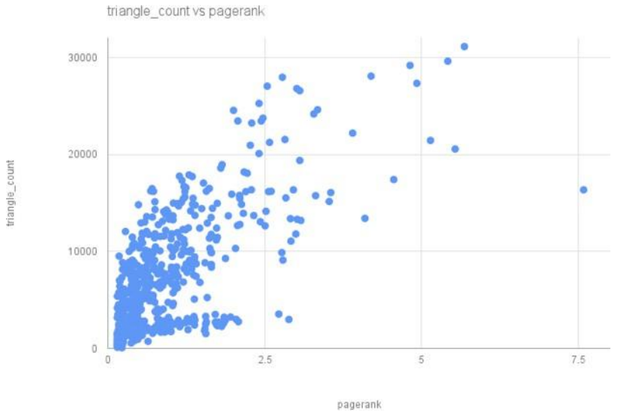

Temperature anomaly, C (1950 - 2005)
0,6
0,5
0,4
0,3
0,2
0,1
0
-0,1
-0,2
-0,3
1950
1953
1956
1959
1962
1965
1968
1971
1974
1977
1980
1983
1986
1989
1992
1995
1998
2001
2004


triangle_count vs pagerank
triangle_count
30000
20000
10000
0
0
2.5
5
7.5
pagerank

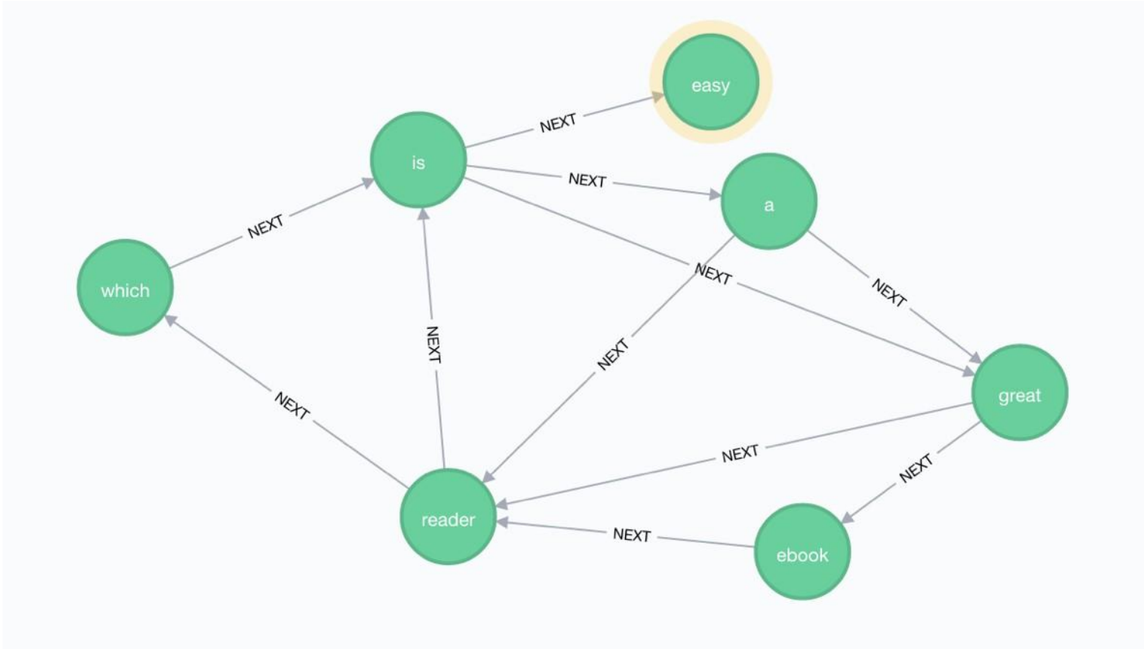
Chart

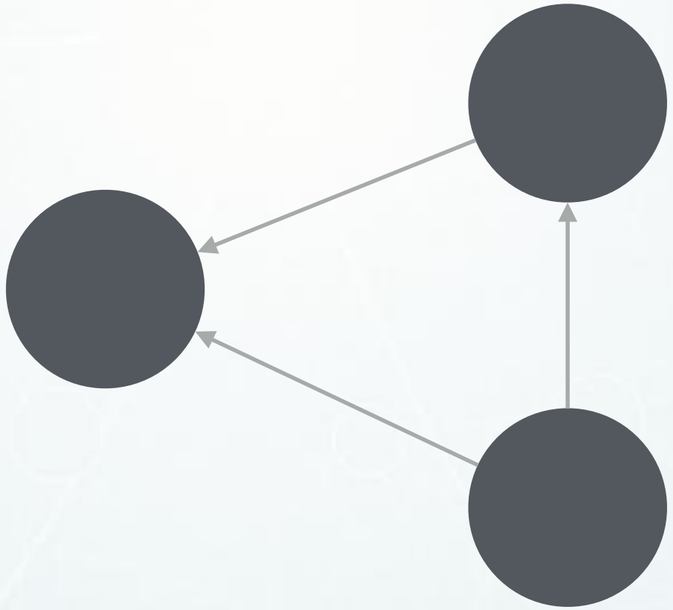

Production
24.48%
Labor
17.48%
Licenses
12.59%
Taxes
10.49%
Legal
8.39%
Insurance
5.59%
Facilities
20.98%


Temperature anomaly, C (1950 - 2005)
0,6
0,5
0,4
0,3
0,2
0,1
0
-0,1
-0,2
-0,3
1950
1953
1956
1959
1962
1965
1968
1971
1974
1977
1980
1983
1986
1989
1992
1995
1998
2001
2004

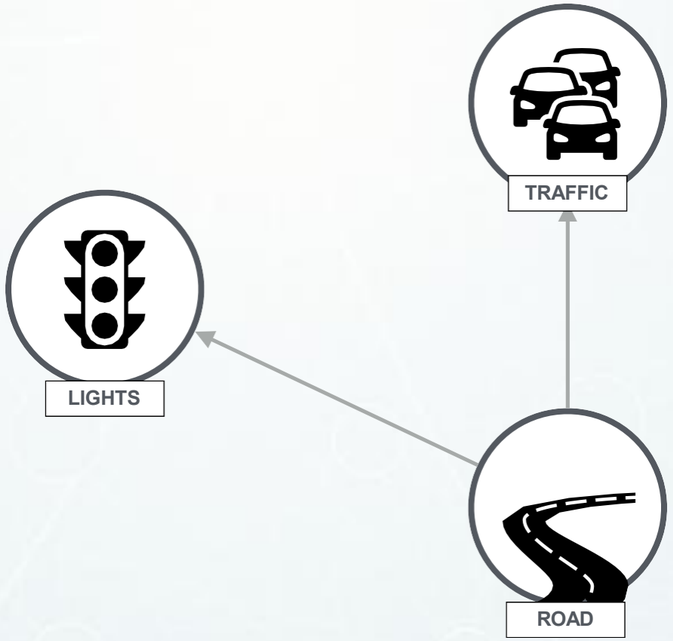

triangle_count vs pagerank
triangle_count
30000
20000
10000
0
0
2.5
5
7.5
pagerank

Graph

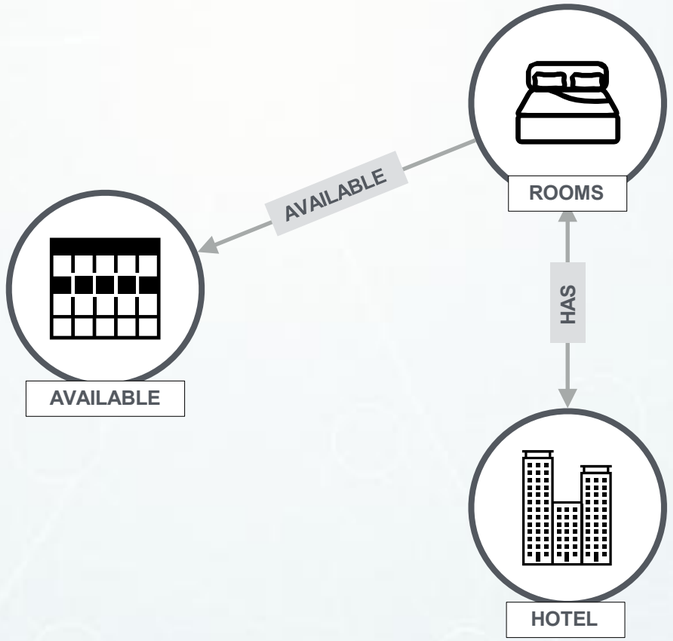

easy
is
a
which
great
reader
ebook
NEXT



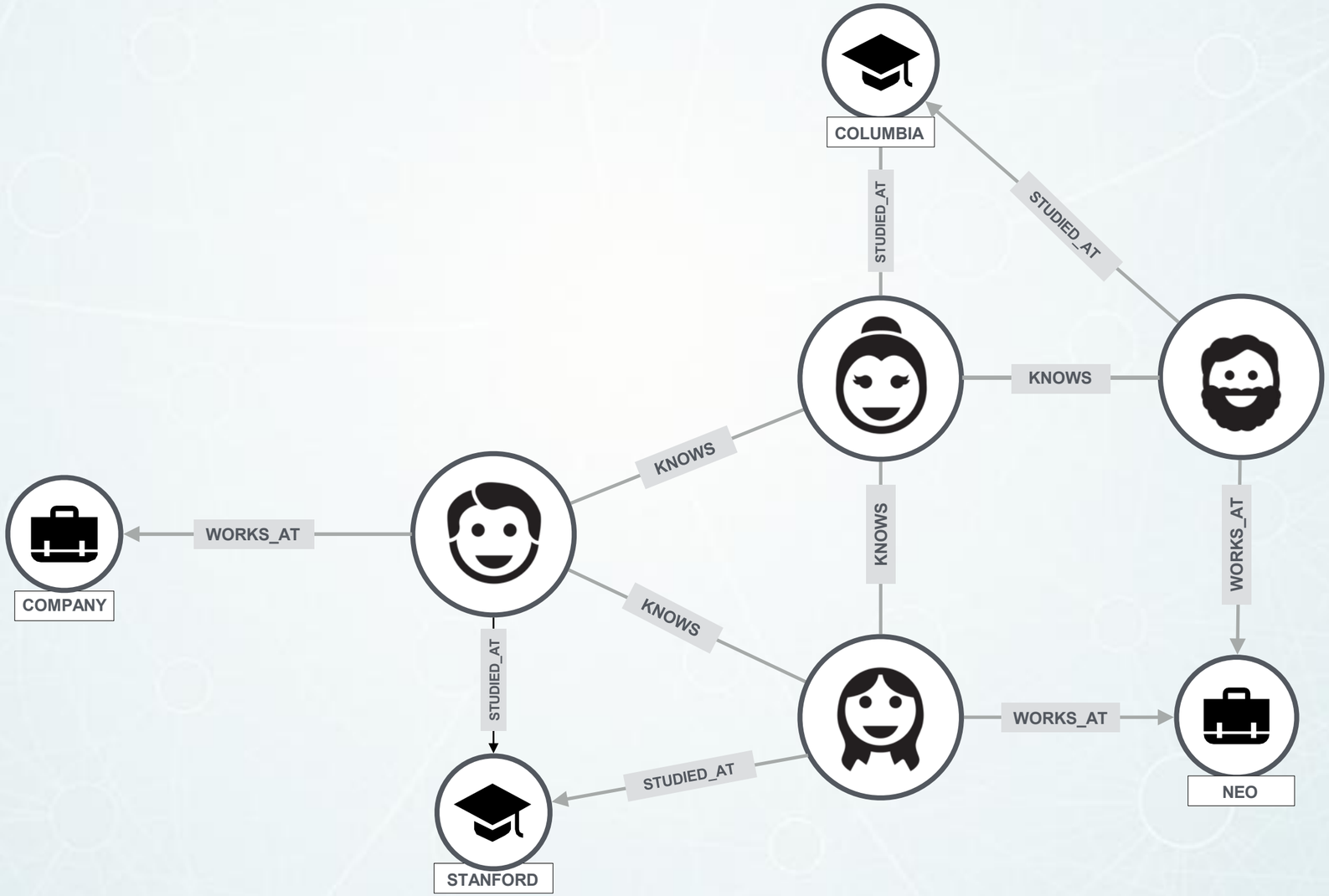
A Graph Is Connected Data



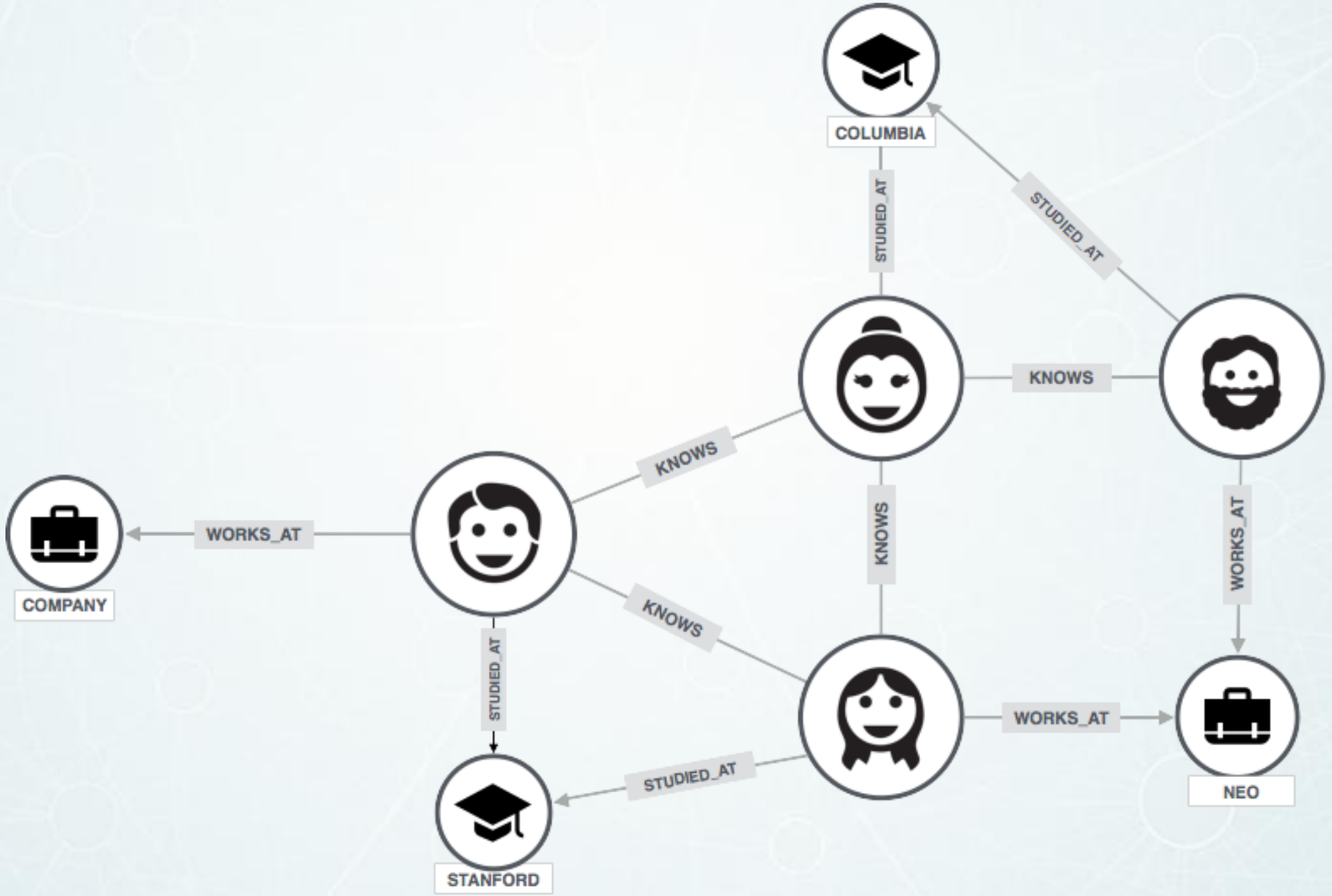
A Graph Is Connected Data



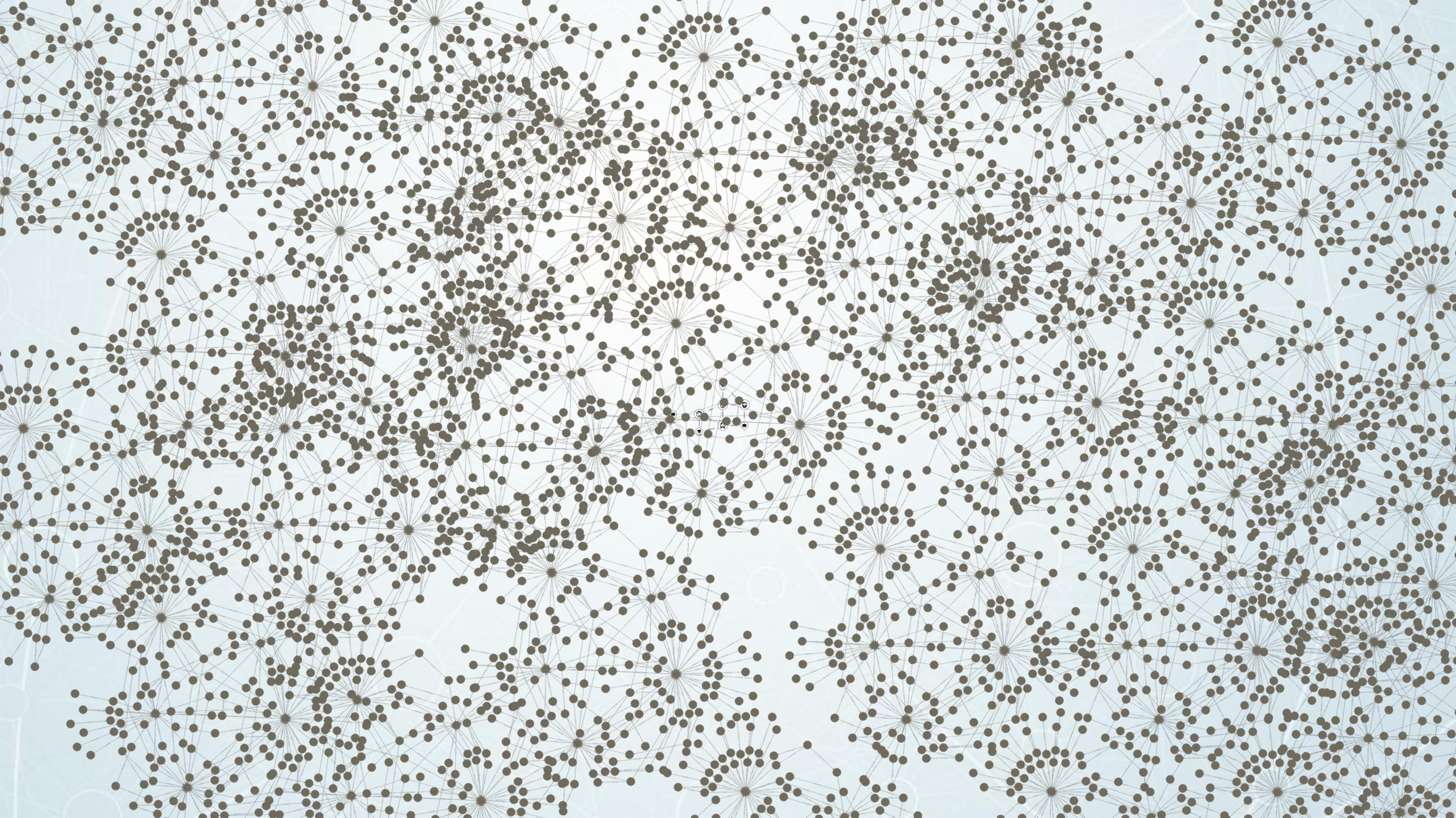
A Graph Is Connected Data



A Graph Is Connected Data

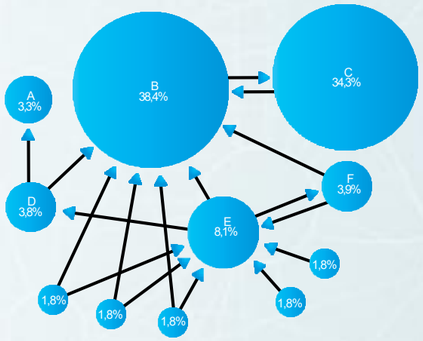


A Graph Is Connected Data

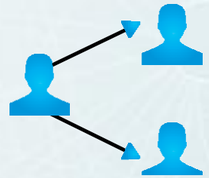


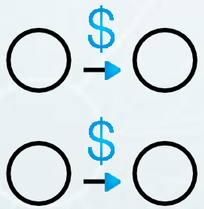
Use of Graphs has created some of the most successful companies in the world

Google



LinkedIn





PayPal

Today we see graph-projects in virtually every industry

Finance

Social networks

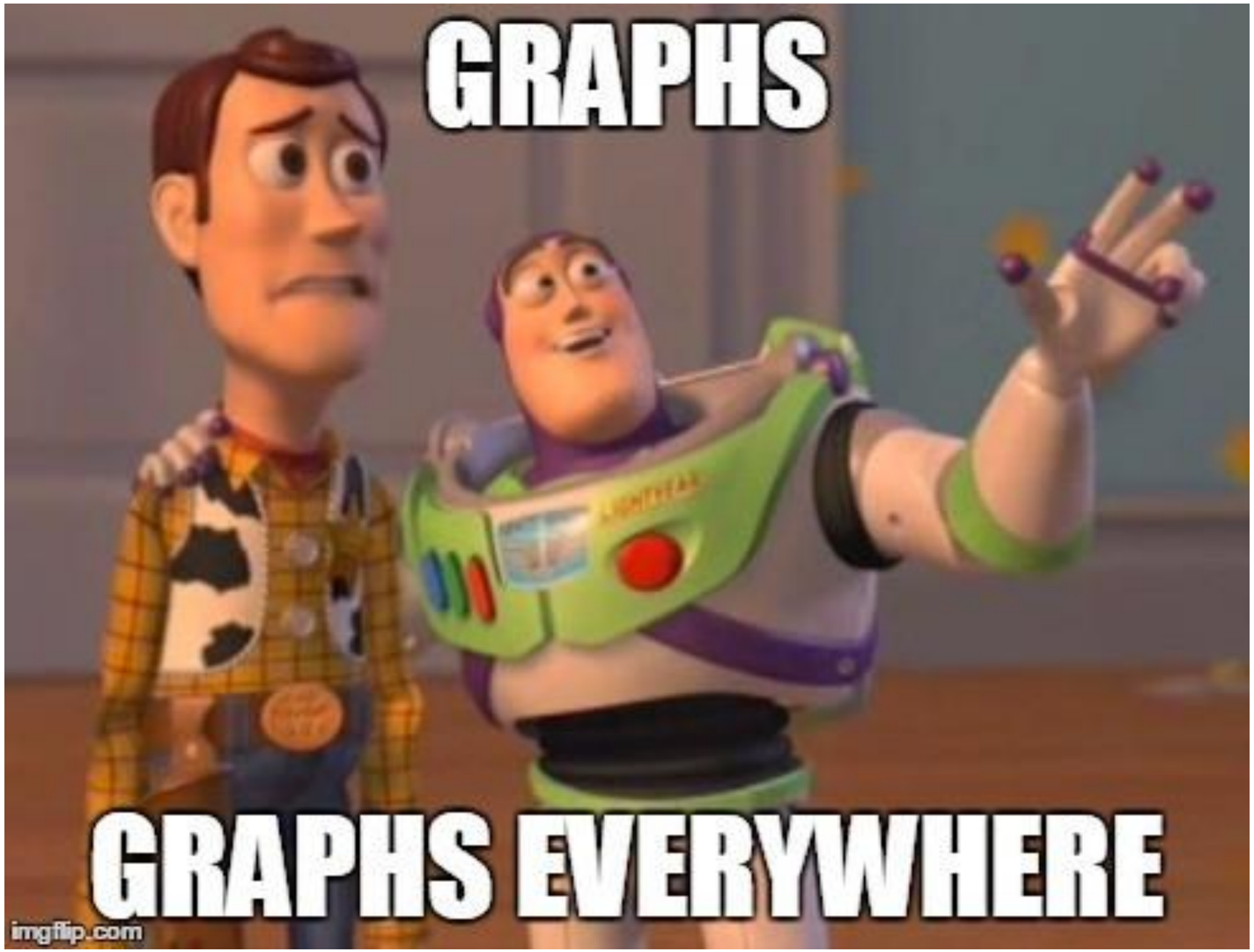
HR & Recruiting

Manufacturing & Logistics

Health Care

Telco

Retail


GRAPHS
GRAPHS EVERYWHERE
imgflip.com

NEO4j USE CASES

Real Time Recommendations

Master Data Management

Fraud Detection

Graph Based Search

Network & IT-Operations

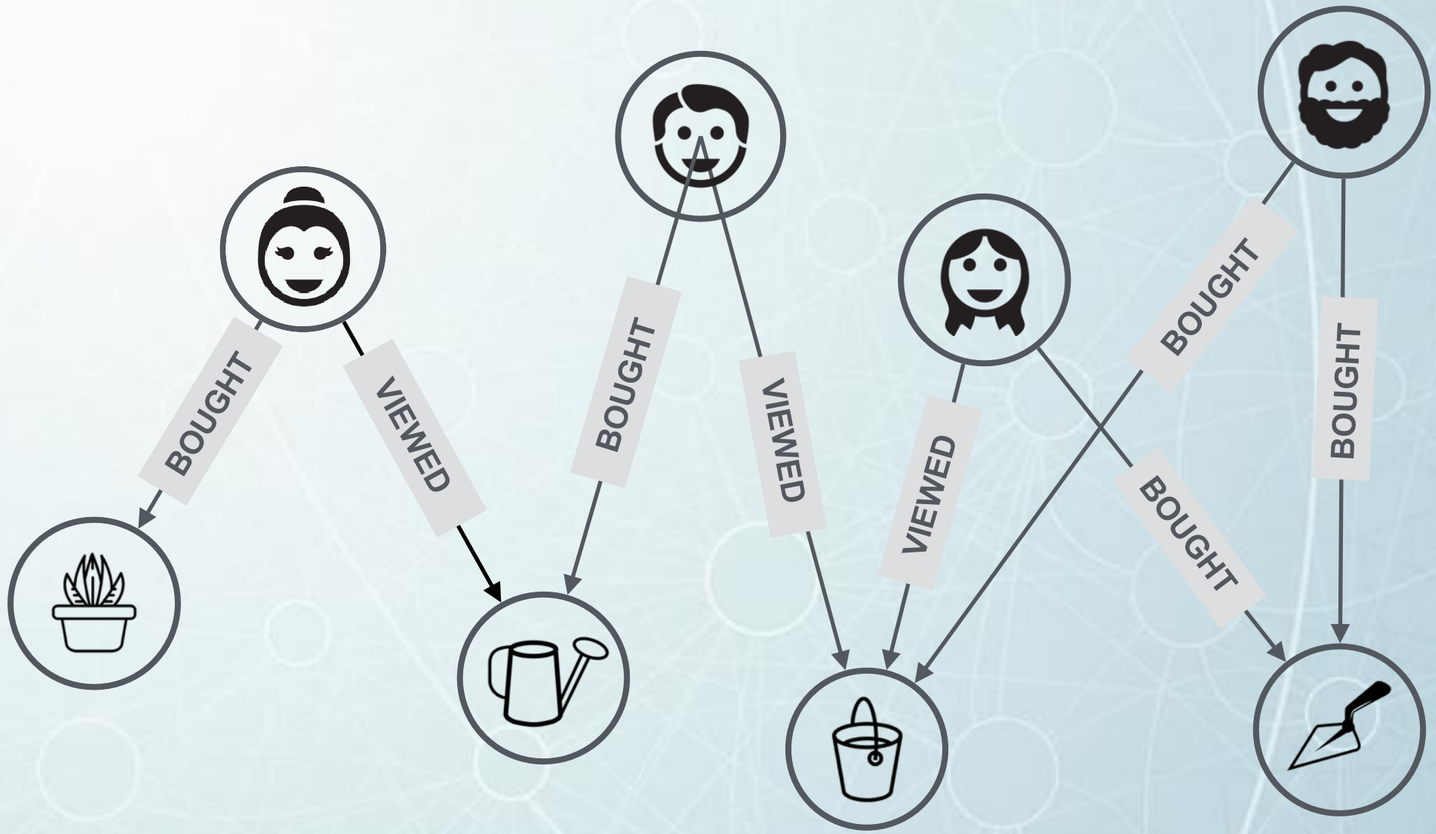
Identity & Access Management

NEO4j USE CASES

- Real Time Recommendations
- Master Data Management
- Fraud Detection
- Graph Based Search
- Network & IT-Operations
- Identity & Access Management

neo4j

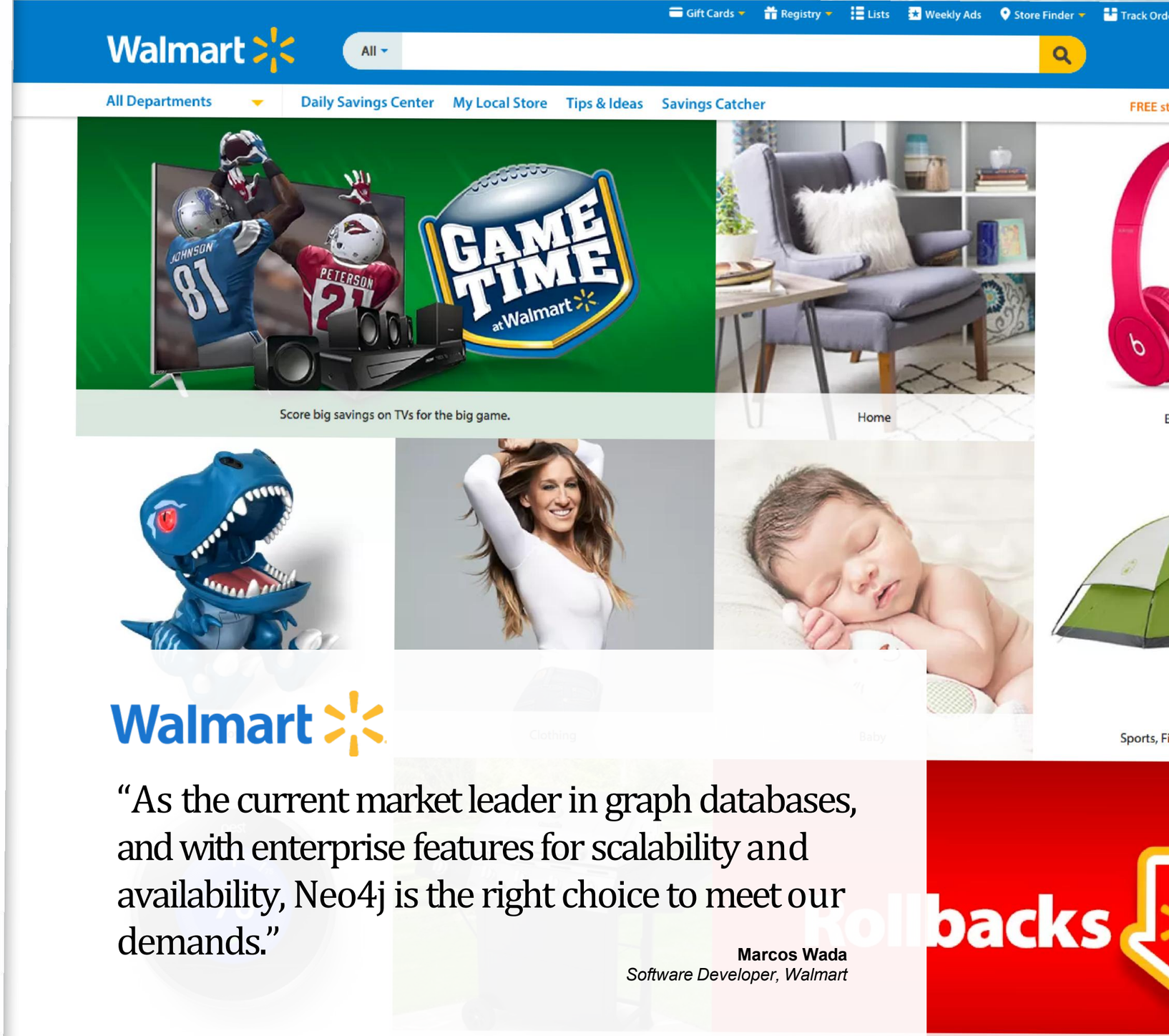
GRAPH THINKING:
Real Time Recommendations



NEO4j USE CASES

- Real Time Recommendations
- Master Data Management
- Fraud Detection
- Graph Based Search
- Network & IT-Operations
- Identity & Access Management

neo4j



NEO4j USE CASES

Real Time Recommendations

Master Data Management

Fraud Detection

Graph Based Search

Network & IT-Operations

Identity & Access Management

neo4j

GRAPH THINKING:
Master Data Management

NEO4j USE CASES

Real Time Recommendations

Master Data Management

Fraud Detection

Graph Based Search

Network & IT-Operations

Identity & Access Management

CISCO

Neo4j is the heart of Cisco HMP: used for governance and single source of truth and a one-stop shop for all of Cisco's hierarchies.

neo4j

NEO4j USE CASES

Real Time Recommendations

Master Data Management

Fraud Detection

Graph Based Search

Network & IT-Operations

Identity & Access Management

GRAPH THINKING:
Fraud Detection

neo4j

NEO4j USE CASES

Real Time Recommendations

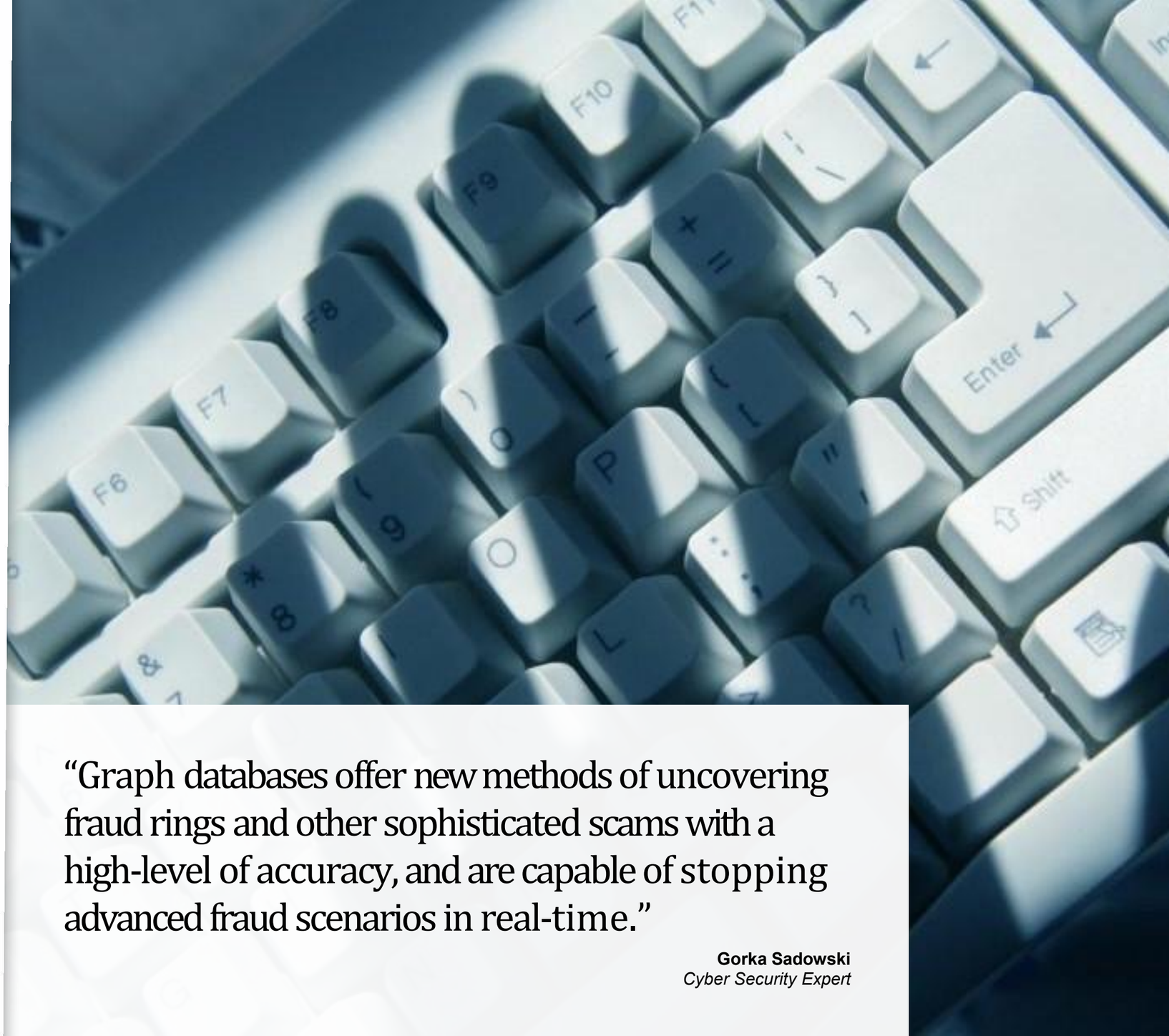
Master Data Management

Fraud Detection

Graph Based Search

Network & IT-Operations

Identity & Access Management



"Graph databases offer new methods of uncovering fraud rings and other sophisticated scams with a high-level of accuracy, and are capable of stopping advanced fraud scenarios in real-time."

Gorka Sadowski
Cyber Security Expert

neo4j

NEO4j USE CASES

Real Time Recommendations

Master Data Management

Fraud Detection

Graph Based Search

Network & IT-Operations

Identity & Access Management

GRAPH THINKING:
Graph Based Search

neo4j

NEO4j USE CASES

Real Time Recommendations

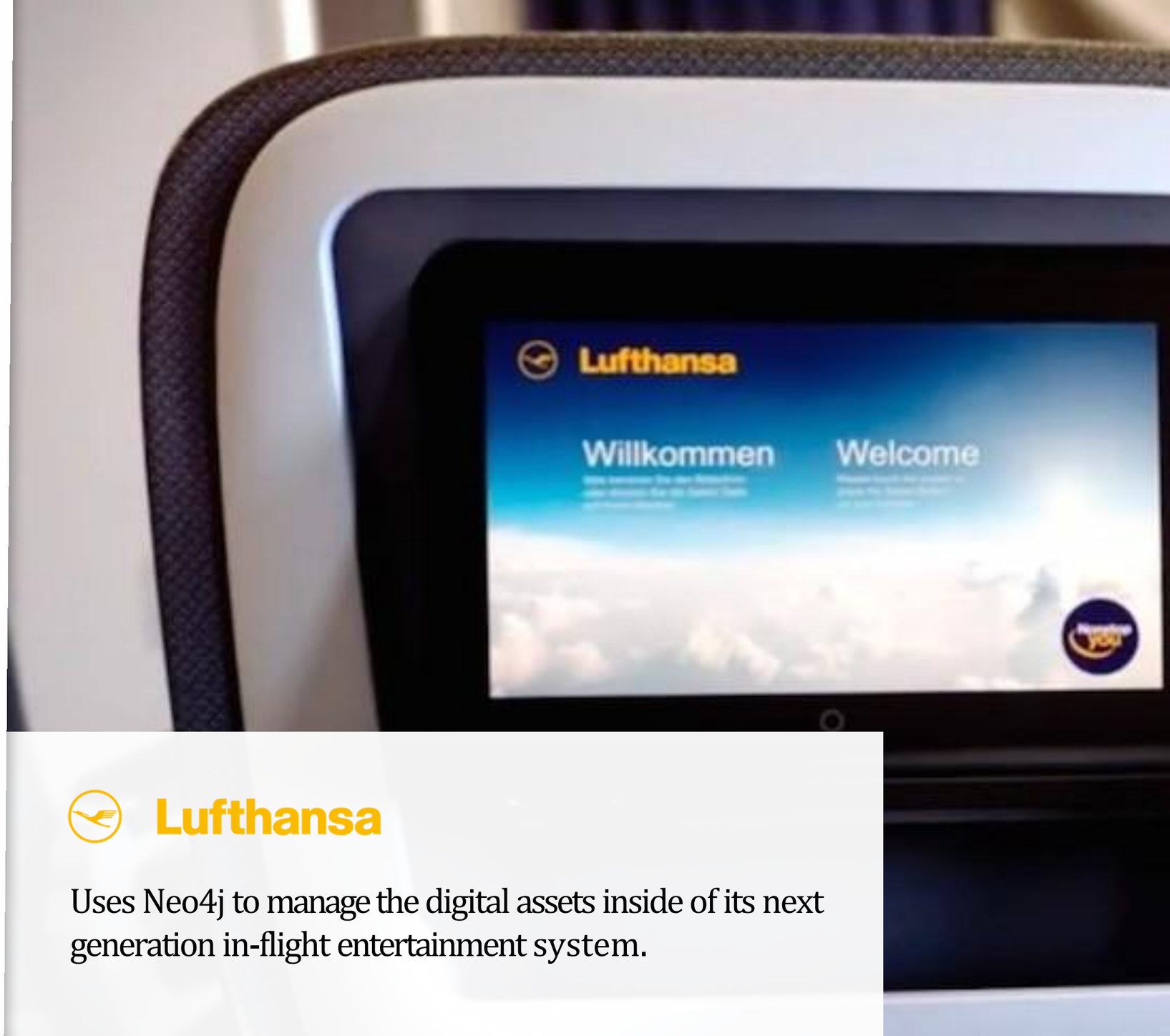
Master Data Management

Fraud Detection

Graph Based Search

Network & IT-Operations

Identity & Access Management



Lufthansa

Uses Neo4j to manage the digital assets inside of its next generation in-flight entertainment system.

NEO4j USE CASES

Real Time Recommendations

Master Data Management

Fraud Detection

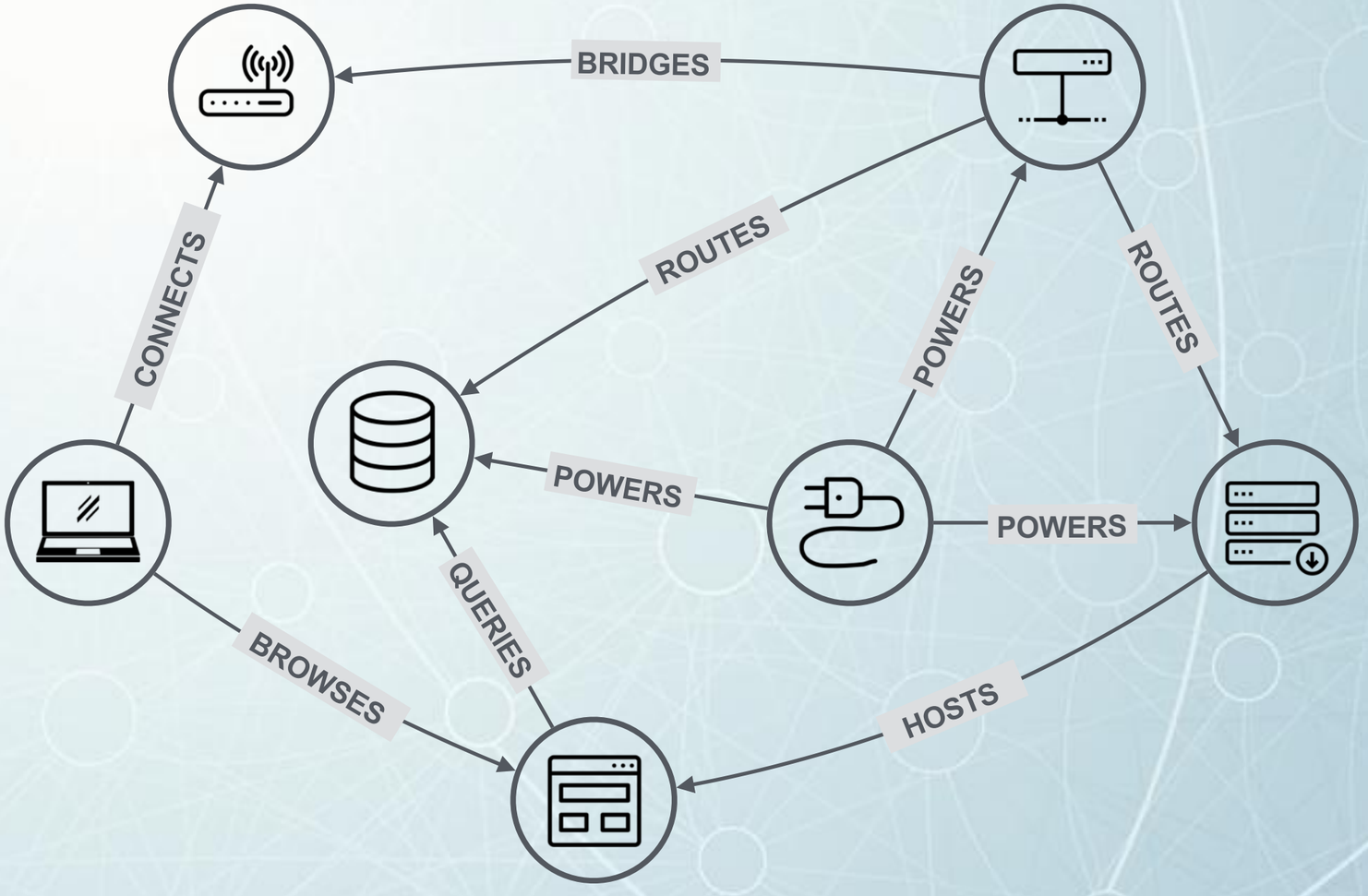
Graph Based Search

Network & IT-Operations

Identity & Access Management

neo4j

GRAPH THINKING:
Network & IT-Operations



NEO4j USE CASES

Real Time Recommendations

Master Data Management

Fraud Detection

Graph Based Search

Network & IT-Operations

Identity & Access Management

Uses Neo4j for network topology analysis for big telco service providers

neo4j

NEO4j USE CASES

Real Time Recommendations

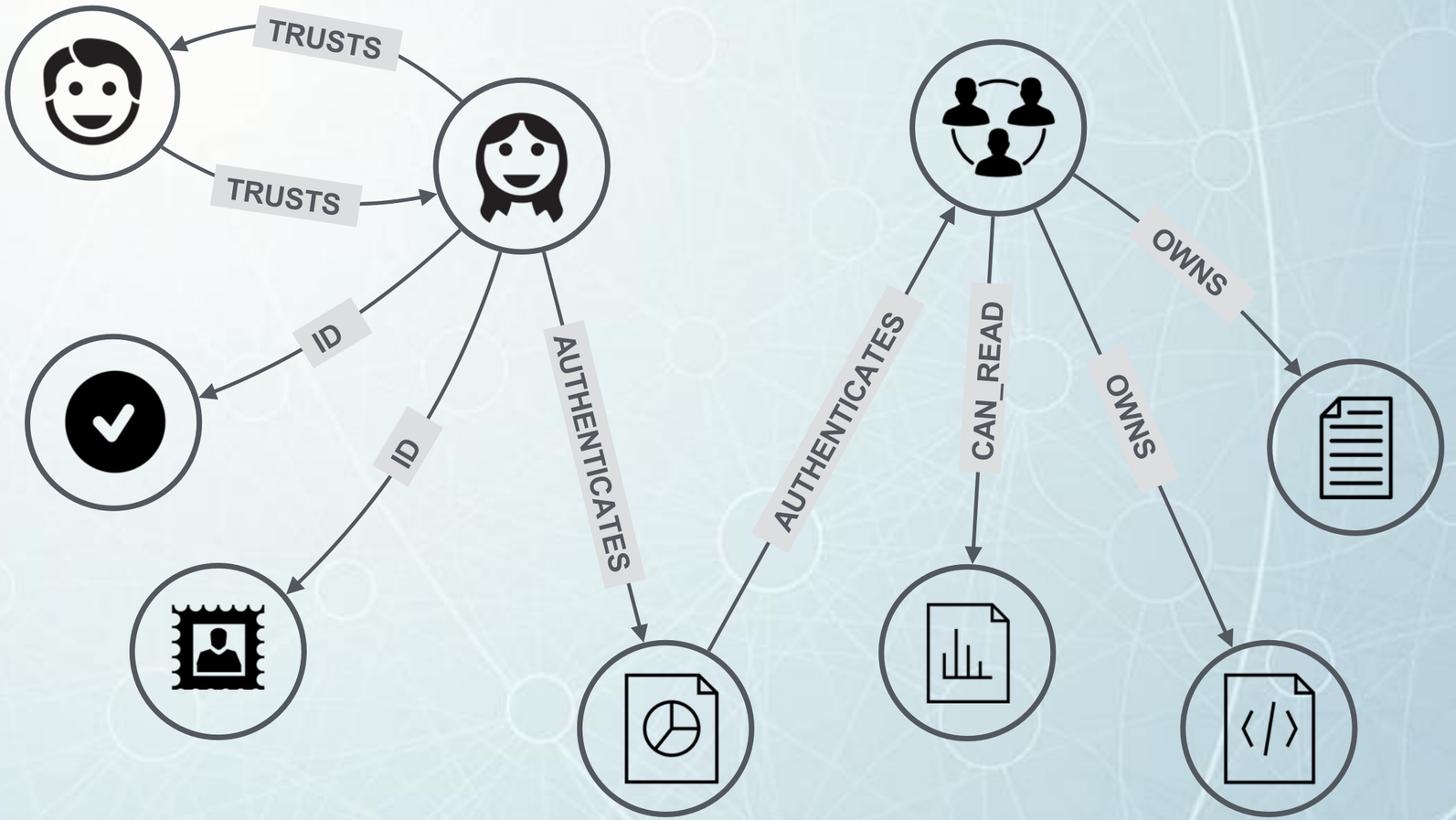
Master Data Management

Fraud Detection

Graph Based Search

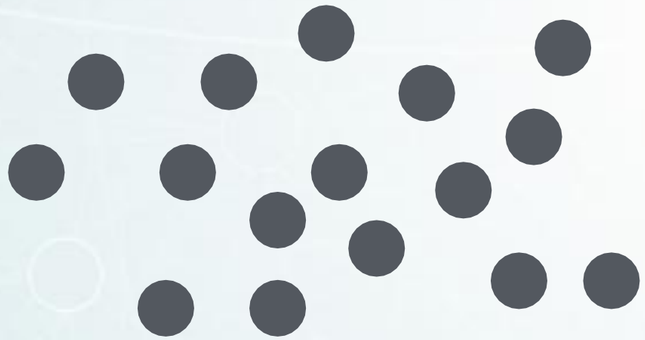
Network & IT-Operations

GRAPH THINKING:
Identity And Access Management



neo4j

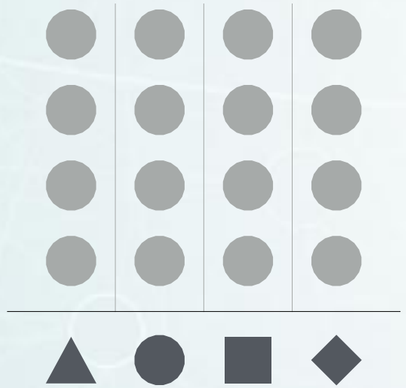
A way of representing data



DATA

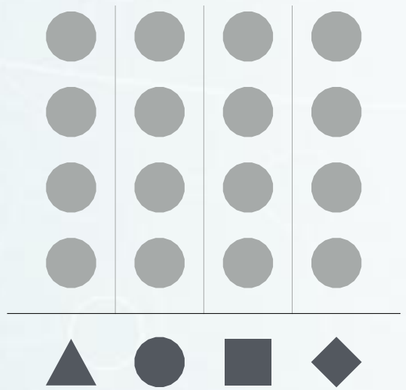
DATA

A way of representing data



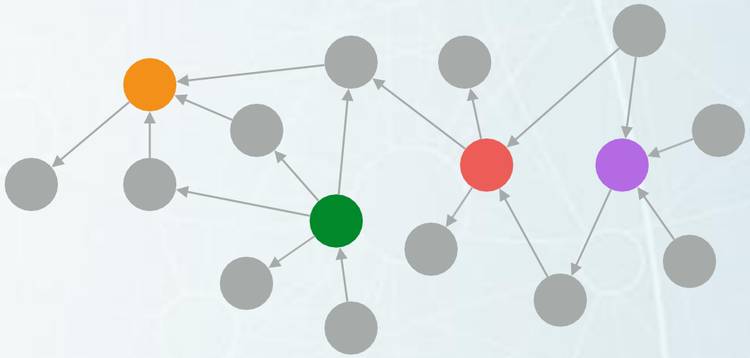
Relational Database

A way of representing data



Relational Database

Graph Database



Good for:

- Well-understood data structures that don’t change too frequently
- Known problems involving discrete parts of the data, or minimal connectivity

Good for:

- Dynamic systems: where the data topology is difficult to predict
- Dynamic requirements: the evolve with the business
- Problems where the relationships in data contribute meaning & value

THE PROPERTY GRAPH DATA MODEL

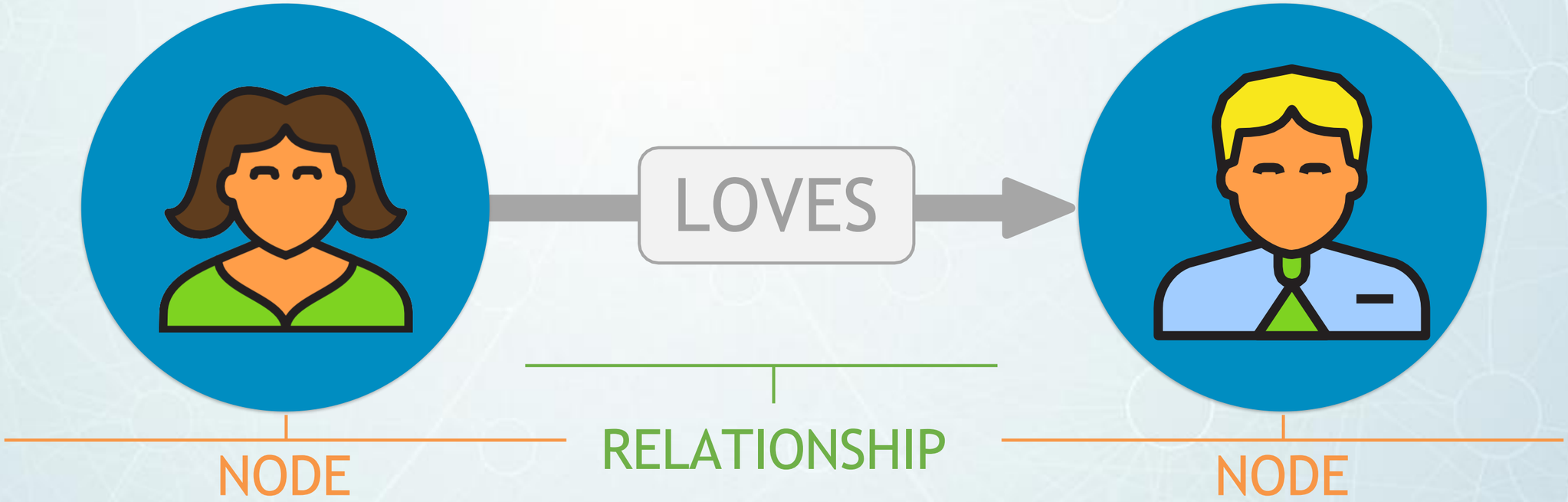
Ann Loves Dan

Ann

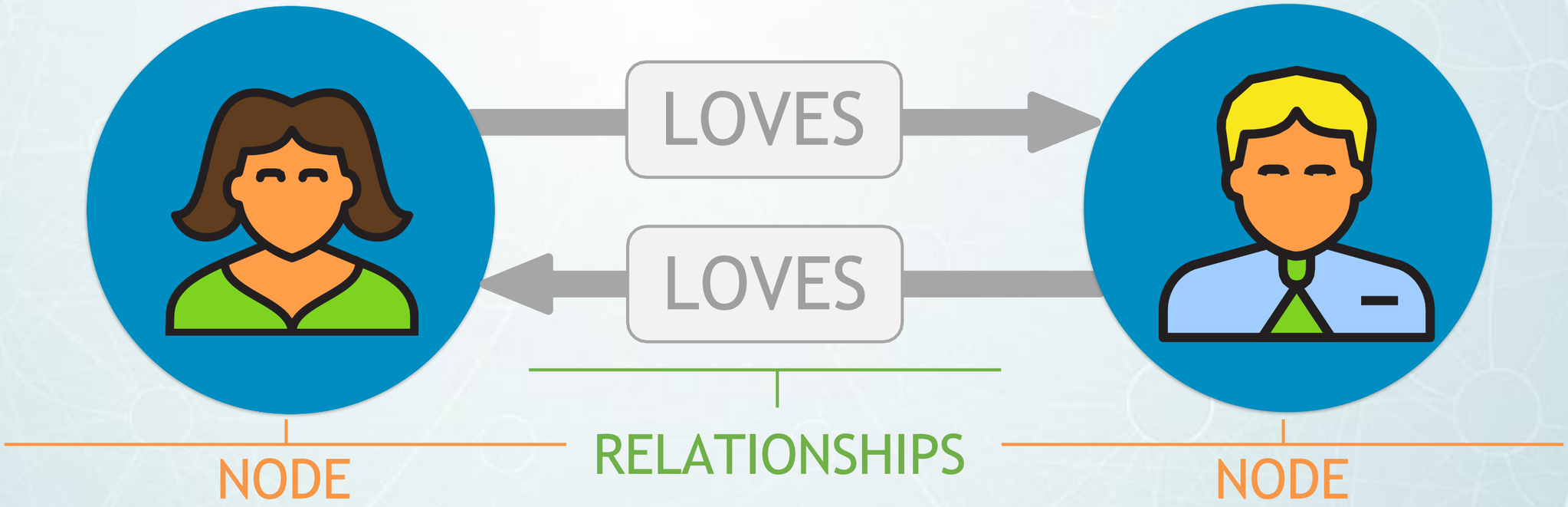
Loves

Dan

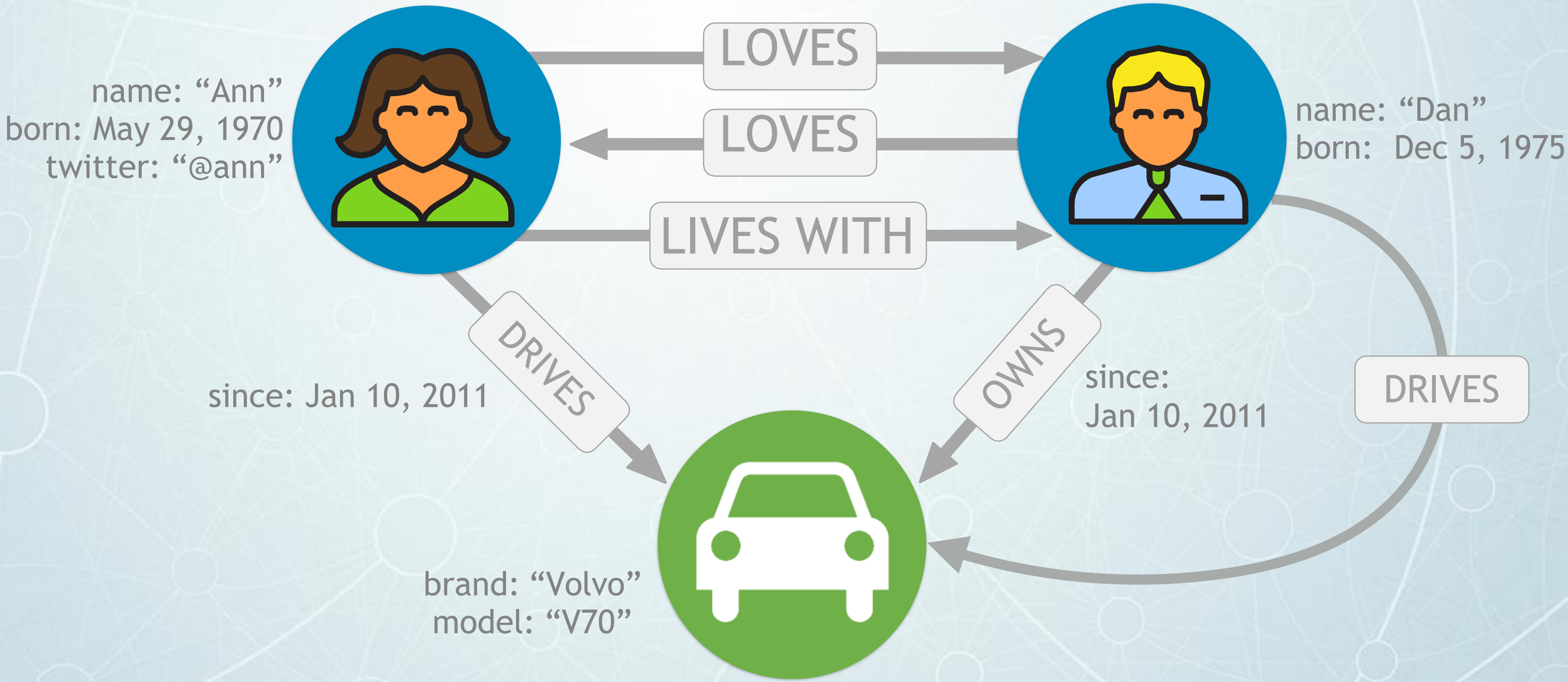
Ann Loves Dan



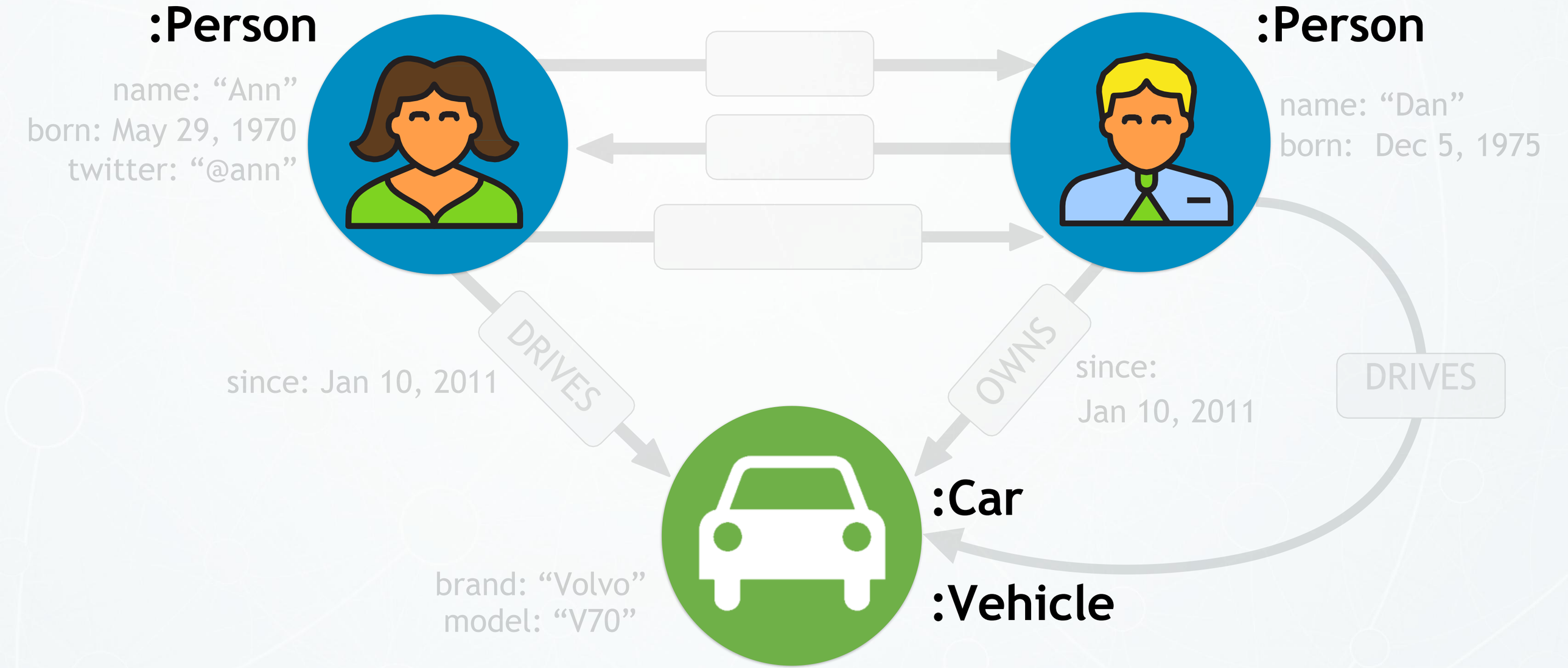
Relationships are Directional



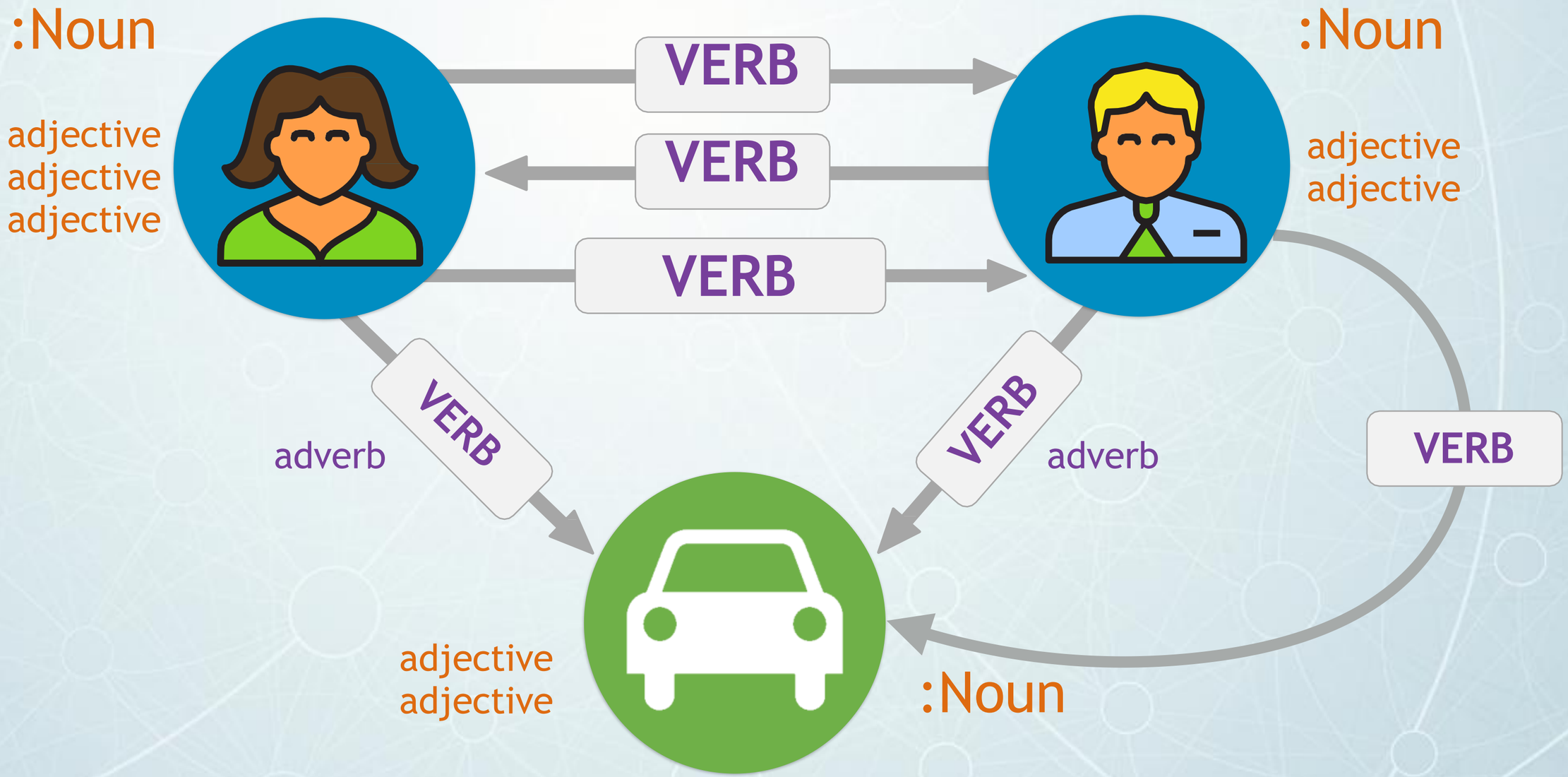
Detailed Property Graph



Labeled Property Graph



Mapping to Languages



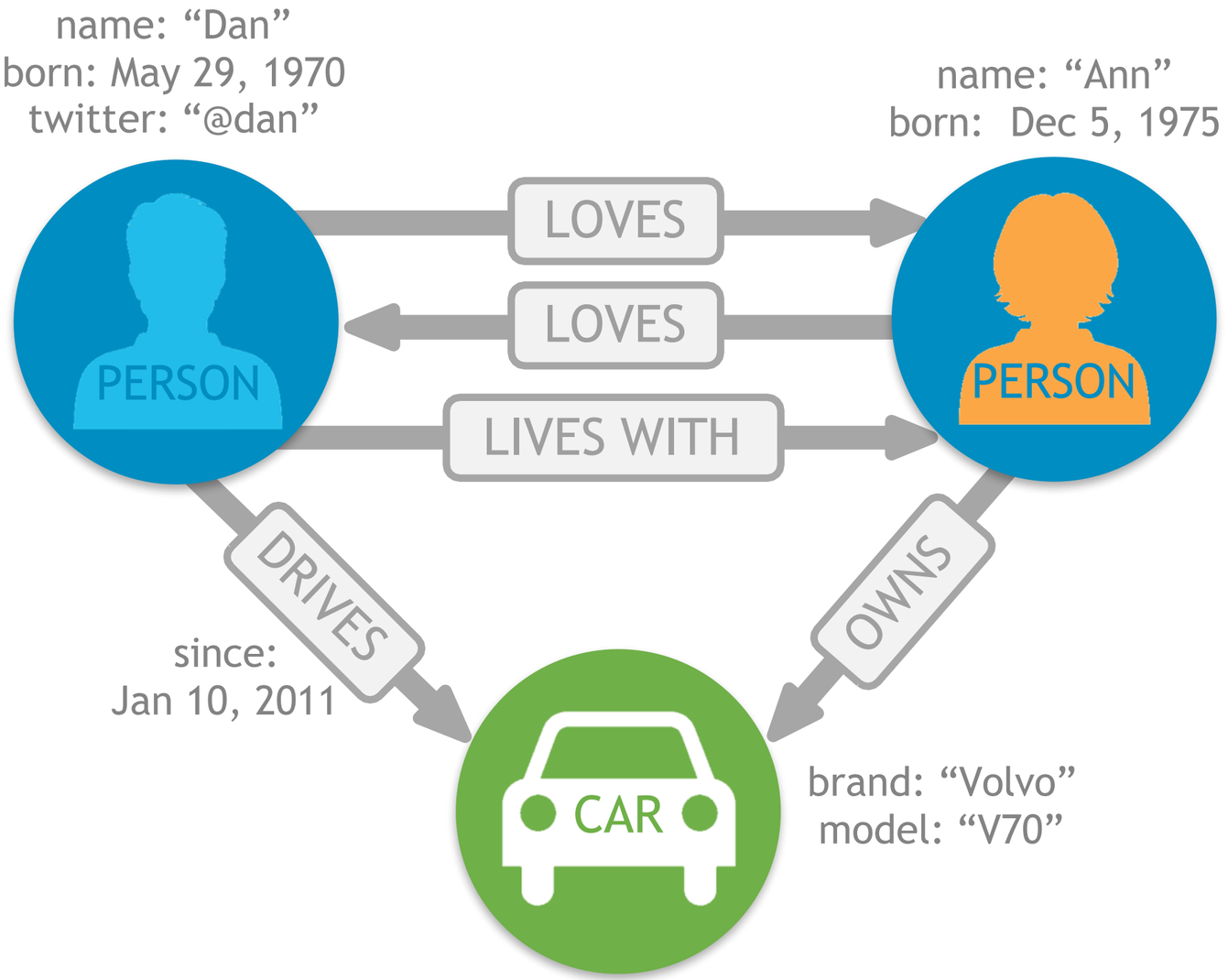
Property Graph Model Components

Nodes

- The objects in the graph
- Can have name-value *properties*
- Can be *labeled*

Relationships

- Relate nodes by type and direction
- Can have name-value *properties*



WHY GRAPHS?

Intuitivness

Speed

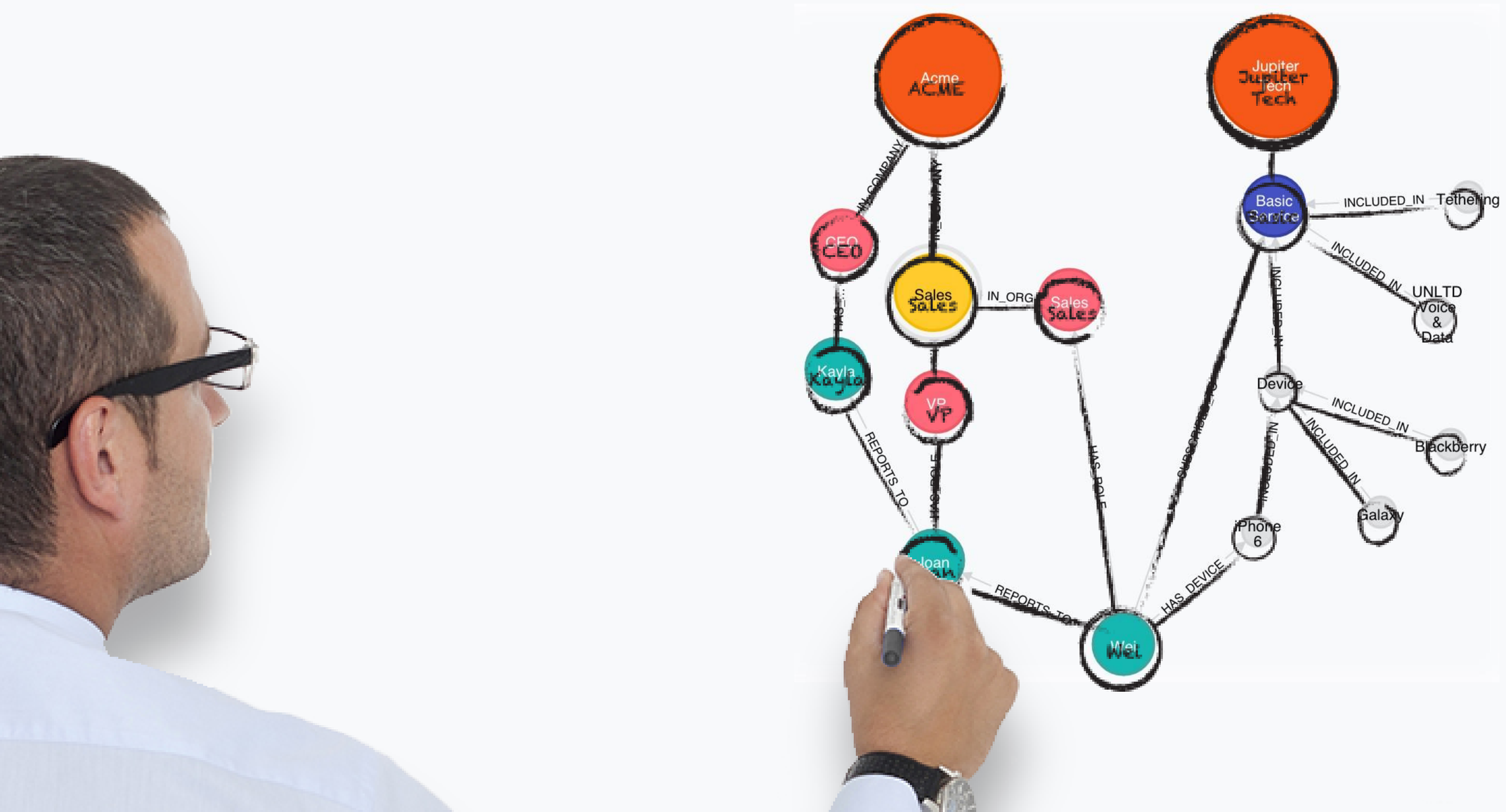
Agility

Intuitiveness

Speed

Agility

Intuitiveness



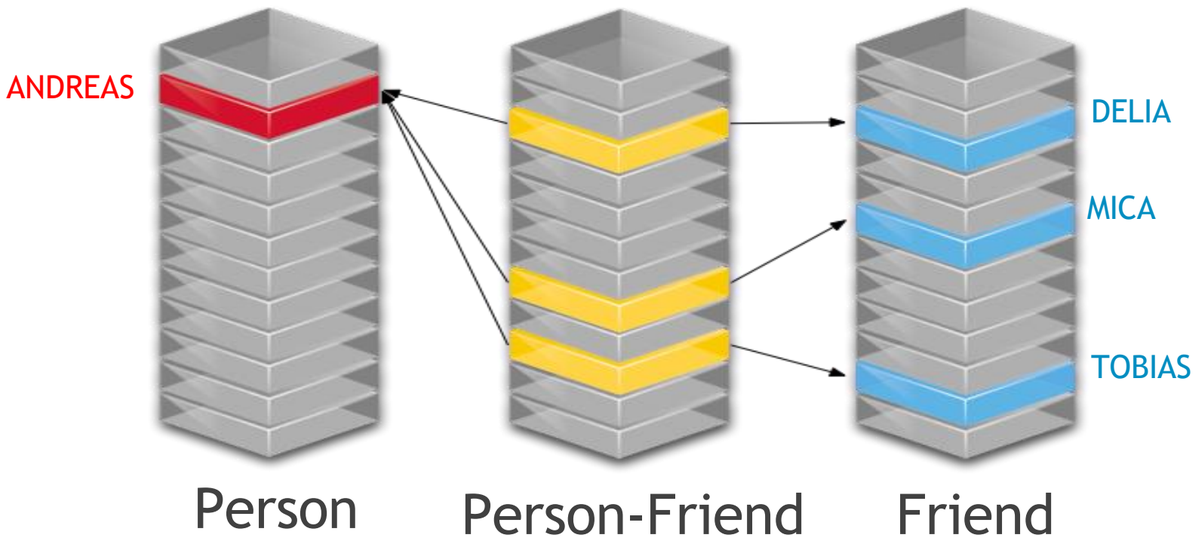
Intuitivness

Speed

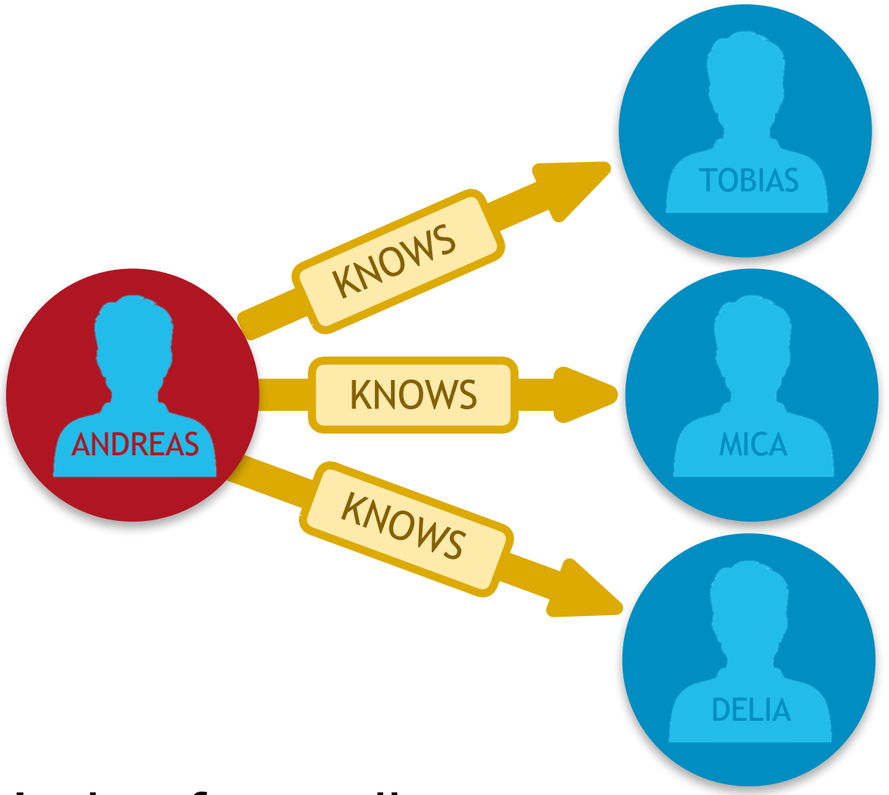
Agility

Relational Versus Graph Models

Relational Model



Graph Model



Index free adjacency

Speed

ebay™

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

- Volker Pacher, Senior Developer

"Minutes to milliseconds" performance

Queries up to 1000x faster than RDBMS or other NoSQL

Intuitivness

Speed

Agility

Agility =

A Naturally Adaptive Model

+

A Query Language Designed for Connectedness

CYPHER

SQL for graphs

(Very) Brief Cypher Tutorial

Creating the Data

NODE

NODE

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

LABEL

PROPERTY

LABEL

PROPERTY

Representing Bi-Directionality

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

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

Cypher

Typical Complex SQL Join

```
 (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")
)
```

The Same Query using Cypher

```
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
```

Project Impact

Less time writing queries

- More time understanding the answers
- Leaving time to ask the next question

Less time debugging queries:

- More time writing the next piece of code
- Improved quality of overall code base

Code that's easier to read:

- Faster ramp-up for new project members
- Improved maintainability & troubleshooting

openCypher

About FAQ News

What is openCypher?

openCypher is an open source project to bring a new public implementation of the industry's most widely adopted graph query language: **Cypher.**

Focus on Your Domain

Graphs naturally describe your domain, and Cypher lets you focus on that domain instead of getting lost in the mechanics of data access. Both expressive and efficient, Cypher is intuitive and immediately familiar, without the steep learning curve of learning a new language.

Human Readable

Cypher is a human-readable query language that makes complex things possible. A combination of English prose and intuitive iconography, Cypher is accessible to developers and operations professionals alike.

Complete Open Source Access

The openCypher project means you can use Cypher as your query language for graph processing capabilities within any product or application. Everyone from Oracle and Apache Spark to Tableau and Structr can (and do) use Cypher's capabilities – now you can too.

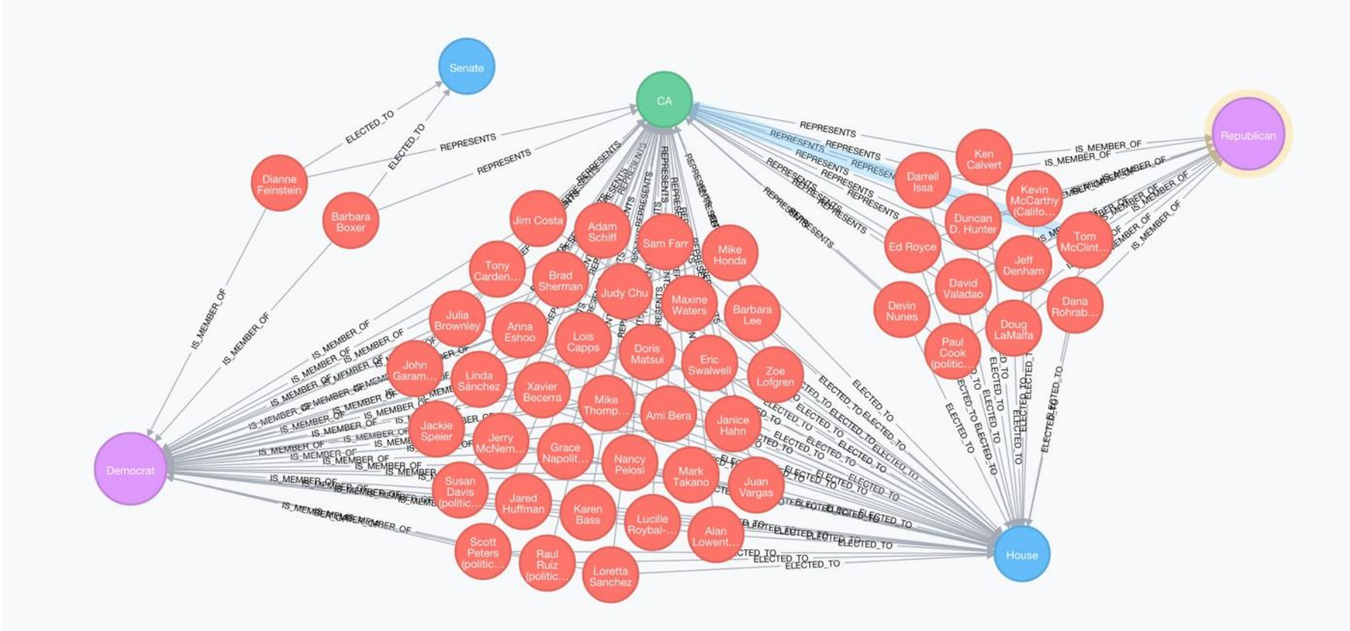
```
MATCH (cypher:QueryLanguage)-[:QUERIES]->(graphs)
MATCH (cypher)<-[:USES]-(u:User) WHERE u.name IN ['Oracle', 'Apache Spark', 'Tableau', 'Structr']
MATCH (openCypher)-[:MAKES_AVAILABLE]->(cypher)
RETURN cypher.attributes
-----------
['awesome',…]
```

http://www.opencypher.org/

Neo4j

Graph Database

- Property graph data model
  - Nodes and relationships
- Native graph processing
- (open)Cypher query language



neo4j.com

Neo4j - Key Product Features

Native Graph Storage
Ensures data consistency and performance

Native Graph Processing
Millions of hops per second, in real time

“Whiteboard Friendly” Data Modeling
Model data as it naturally occurs

High Data Integrity
Fully ACID transactions

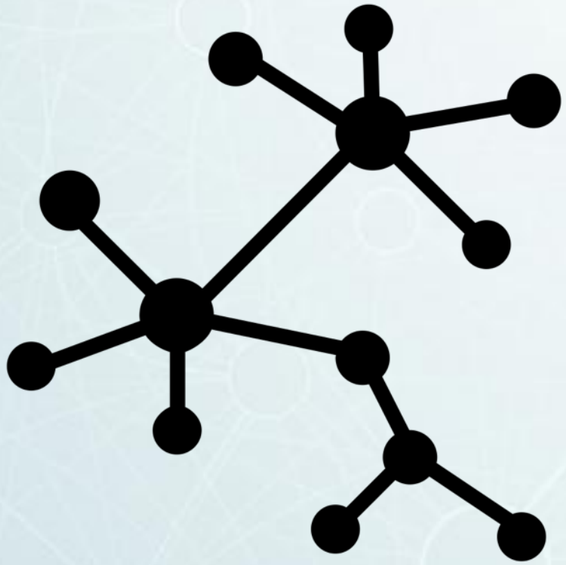
Powerful, Expressive Query Language
Requires 10x to 100x less code than SQL

Scalability and High Availability
Vertical and horizontal scaling optimized for graphs

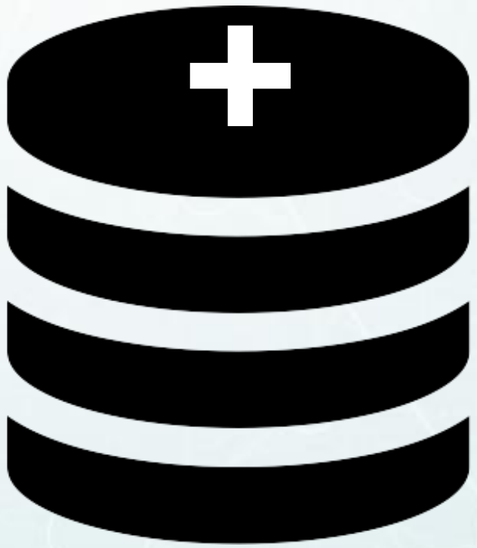
Built-in ETL
Seamless import from other databases

Integration
Drivers and APIs for popular languages

How do you use Neo4j?



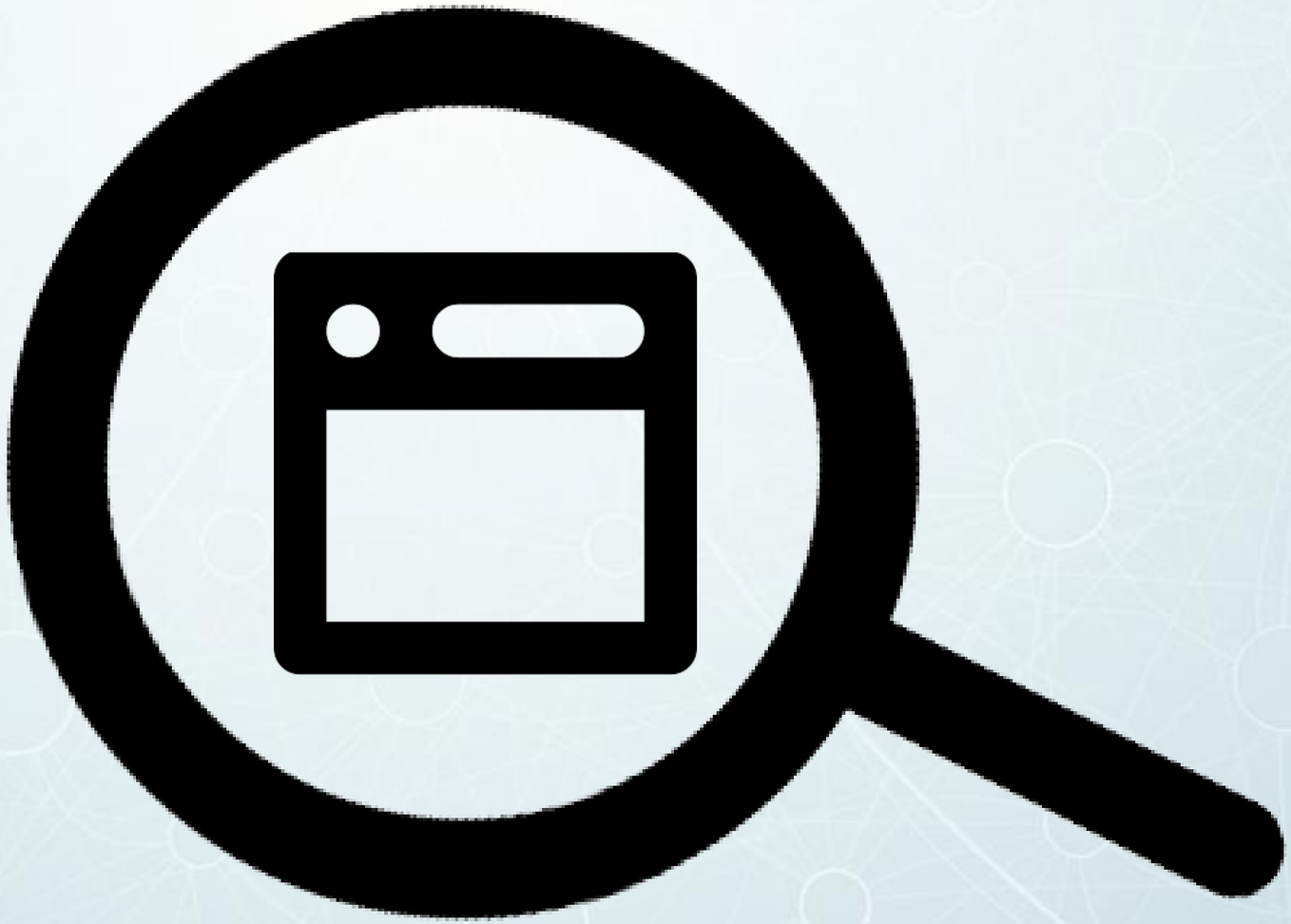
CREATE MODEL



LOAD DATA

QUERY DATA

How do you use Neo4j?



localhost:7474/browser/

Neo4j 2.3.0

Node labels

* Category Customer Employee Order Product Supplier

Relationship types

* PART_OF PRODUCT PURCHASED REPORTS_TO SOLD SUPPLIES

Property keys

_GA_TIMER_MODULE_NR address categoryID categoryName city code companyName contactName contactTitle country created_at customerID customerName description discontinued distance employeeID favorites fax firstName followers following homePage id id_str lastName location method name nodeRank orderID phone picture postalCode productID productName profile_image_url

$

```
MATCH (e:EmpIoyee)‹-[:REP0RTS_TOj-(sub:Employee) OPTIONAL MATCH (e)-[j-(o:...
```

Graph Rows Code

*(32) Employee(2) Order(30)

*(31) REPORTS_TO(1) SOLD(30)

Displaying 32 nodes. 31 relationships (completed with 31 *aad*itional relationships). AUTO-COMPLETE ON

```
MATCH (e:Employee)<- :REP0RTS_TOj-(sub:Employee) OPTIONAL MATCH (e)- j-(o:...
```

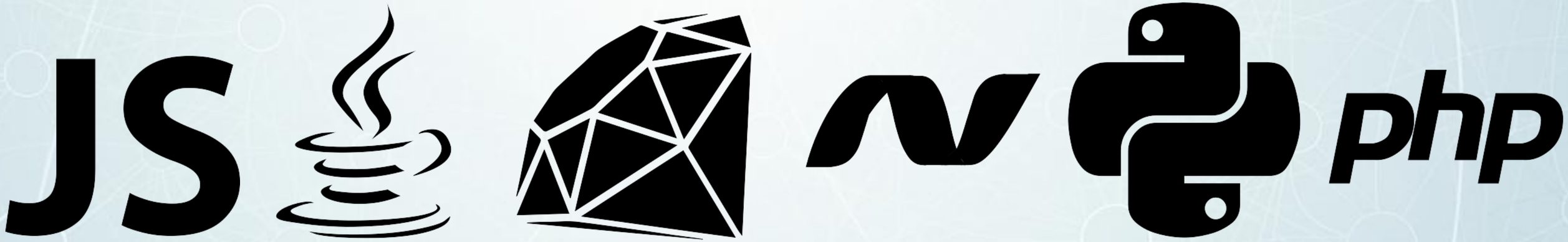
Graph

*(22) Employee(2) Order(20)

*(21) REPORTS_TO(1) SOLD(20)

How do you use Neo4j?

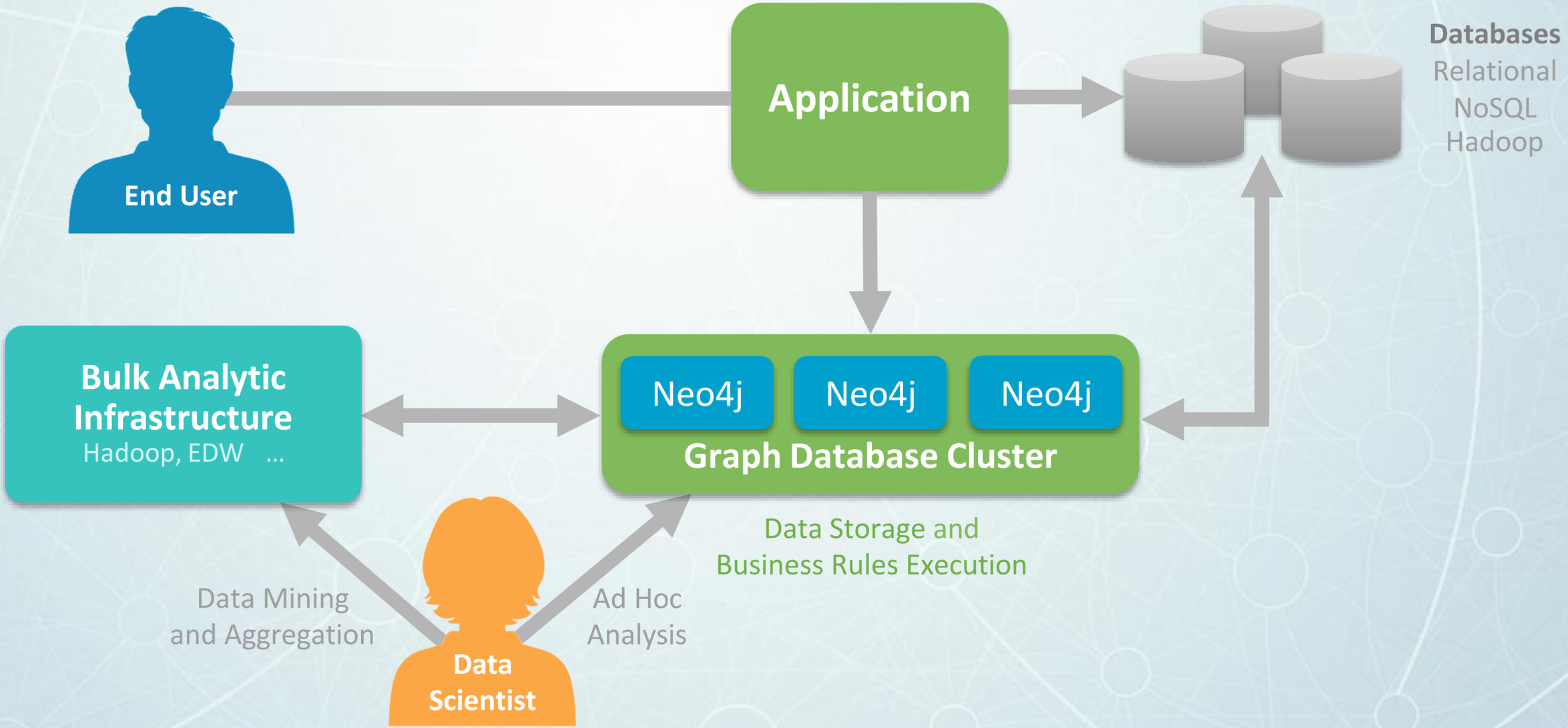
Language Drivers



Language Drivers

Native Server-Side Extensions

Architectural Options



Day in the Life of a RDBMS Developer

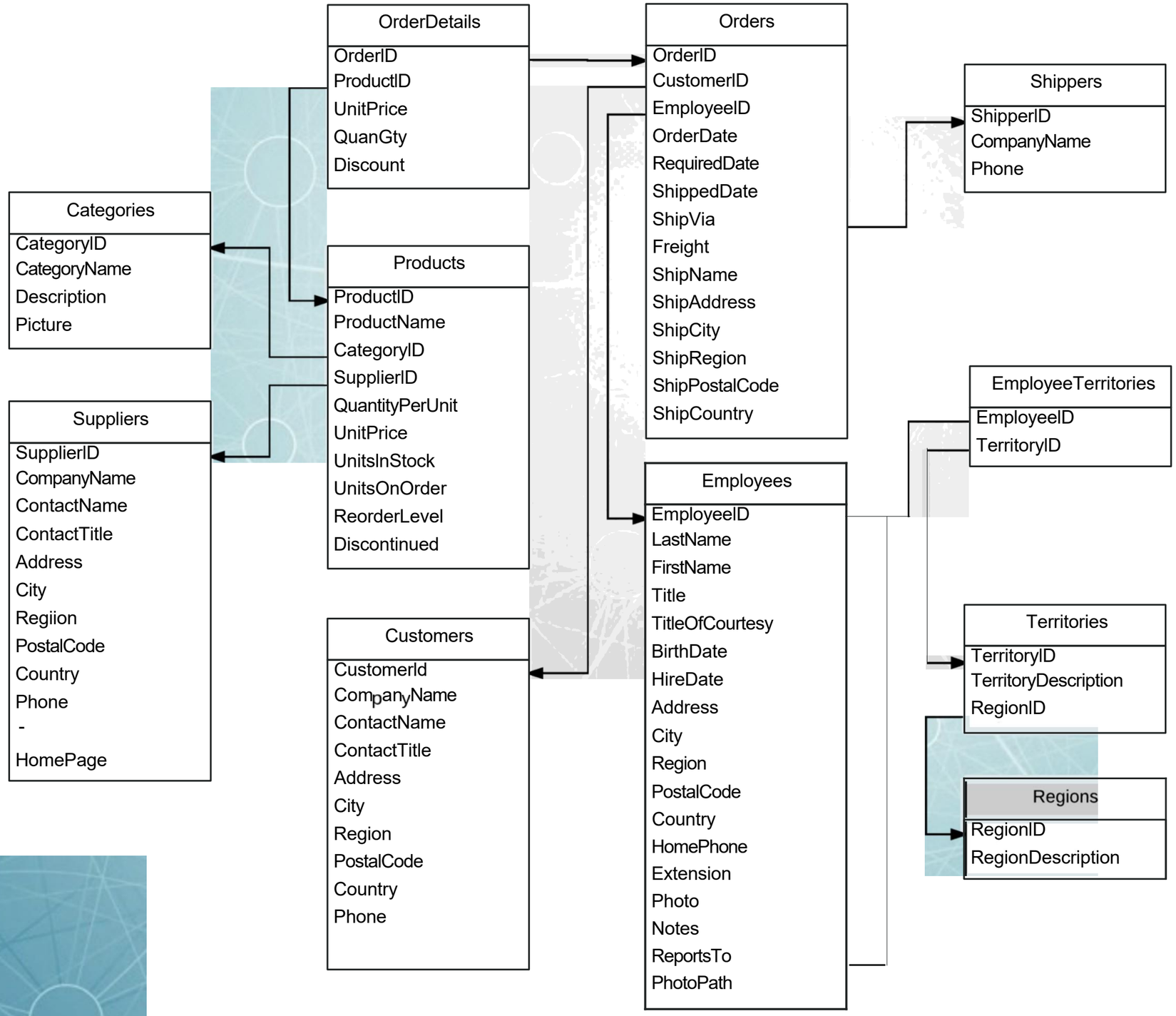
SQL Pains

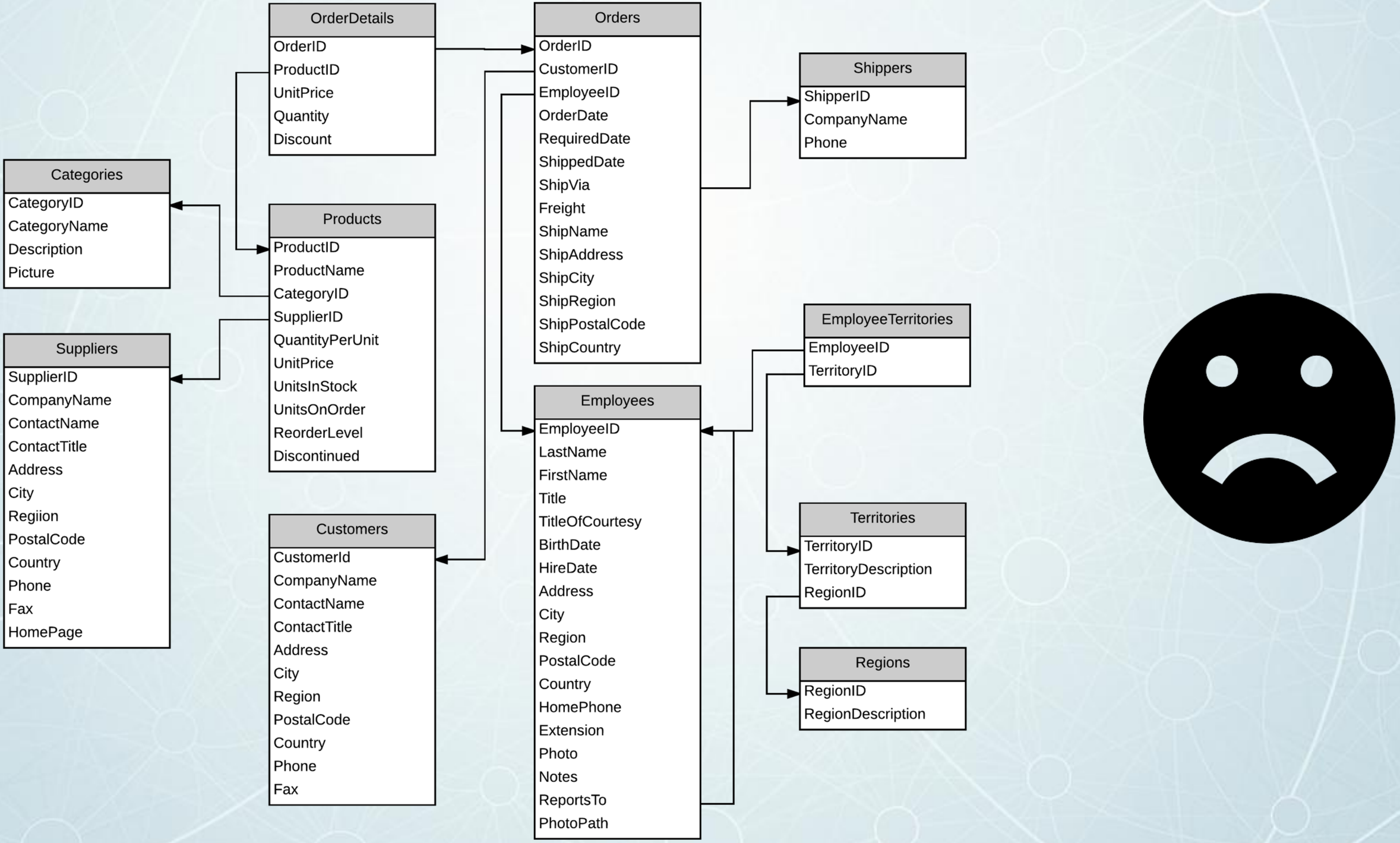
- Complex to model and store relationships
- Performance degrades with increases in data
- Queries get long and complex
- Maintenance is painful

Graph Gains

- Easy to model and store relationships
- Performance of relationship traversal remains constant with growth in data size
- Queries are shortened and more readable
- Adding additional properties and relationships can be done on the fly - no migrations

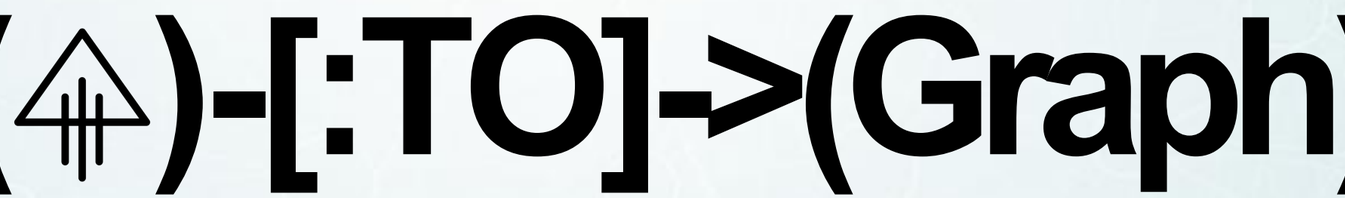
FROM RDBMS TO GRAPHS

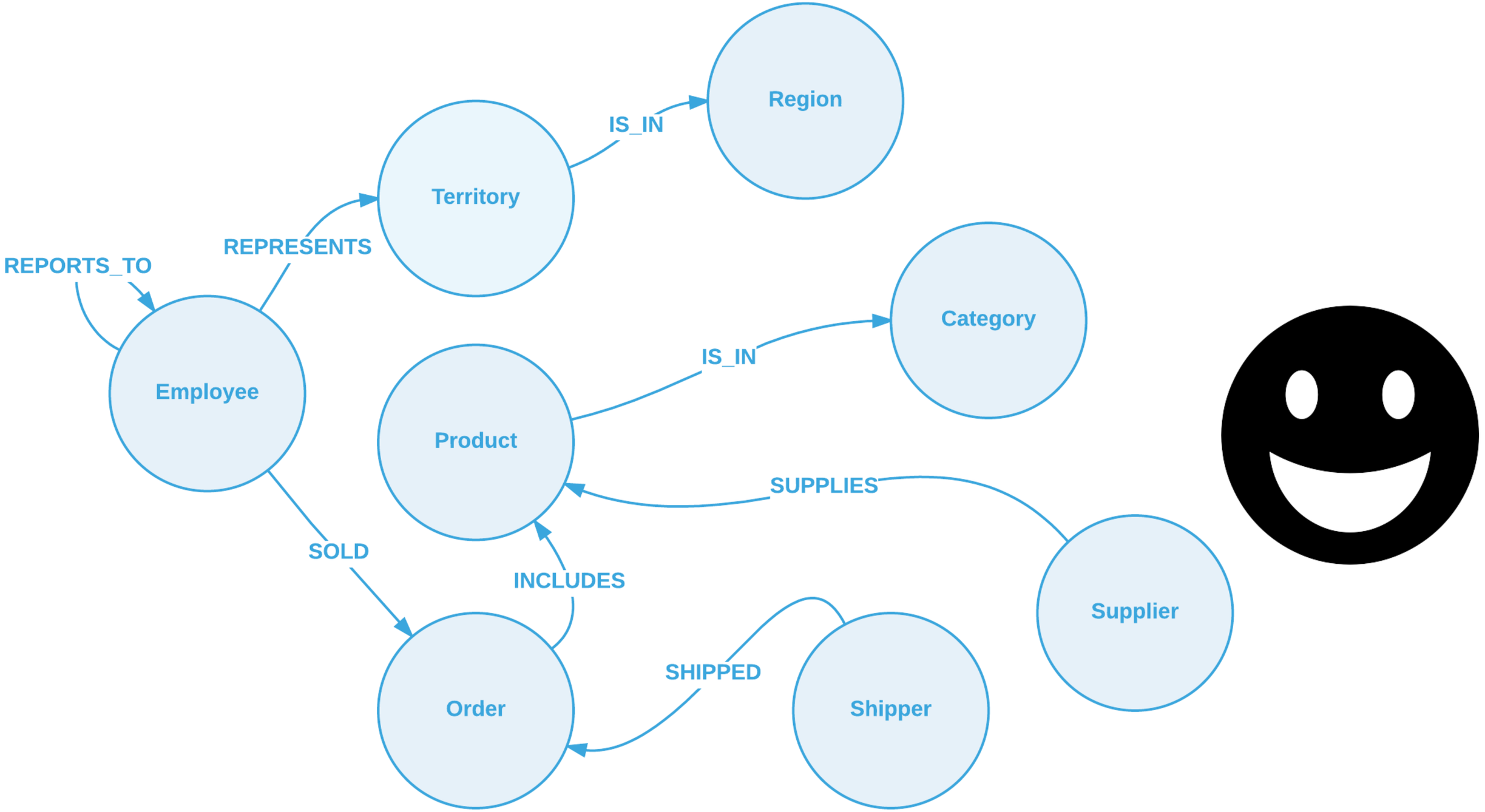

OrderDetails
OrderID
ProductID
UnitPrice
QuanGty
Discount
Orders
OrderID
CustomerID
EmployeeID
OrderDate
RequiredDate
ShippedDate
ShipVia
Freight
ShipName
ShipAddress
ShipCity
ShipRegion
ShipPostalCode
ShipCountry
Shippers
ShipperID
CompanyName
Phone
Categories
CategoryID
CategoryName
Description
Picture
Products
ProductID
ProductName
CategoryID
SupplierID
QuantityPerUnit
UnitPrice
UnitsInStock
UnitsOnOrder
ReorderLevel
Discontinued
Suppliers
SupplierID
CompanyName
ContactName
ContactTitle
Address
City
Regiion
PostalCode
Country
Phone
-
HomePage
EmployeeTerritories
EmployeeID
TerritoryID
Employees
EmployeeID
LastName
FirstName
Title
TitleOfCourtesy
BirthDate
HireDate
Address
City
Region
PostalCode
Country
HomePhone
Extension
Photo
Notes
ReportsTo
PhotoPath
Customers
CustomerId
CompanyName
ContactName
ContactTitle
Address
City
Region
PostalCode
Country
Phone
Territories
TerritoryID
TerritoryDescription
RegionID
Regions
RegionID
RegionDescription


OrderDetails
OrderID
ProductID
UnitPrice
Quantity
Discount
Orders
OrderID
CustomerID
EmployeeID
OrderDate
RequiredDate
ShippedDate
ShipVia
Freight
ShipName
ShipAddress
ShipCity
ShipRegion
ShipPostalCode
ShipCountry
Shippers
ShipperID
CompanyName
Phone
Categories
CategoryID
CategoryName
Description
Picture
Products
ProductID
ProductName
CategoryID
SupplierID
QuantityPerUnit
UnitPrice
UnitsInStock
UnitsOnOrder
ReorderLevel
Discontinued
Suppliers
SupplierID
CompanyName
ContactName
ContactTitle
Address
City
Regiion
PostalCode
Country
Phone
Fax
HomePage
EmployeeTerritories
EmployeeID
TerritoryID
Employees
EmployeeID
LastName
FirstName
Title
TitleOfCourtesy
BirthDate
HireDate
Address
City
Region
PostalCode
Country
HomePhone
Extension
Photo
Notes
ReportsTo
PhotoPath
Customers
CustomerId
CompanyName
ContactName
ContactTitle
Address
City
Region
PostalCode
Country
Phone
Fax
Territories
TerritoryID
TerritoryDescription
RegionID
Regions
RegionID
RegionDescription

Northwind

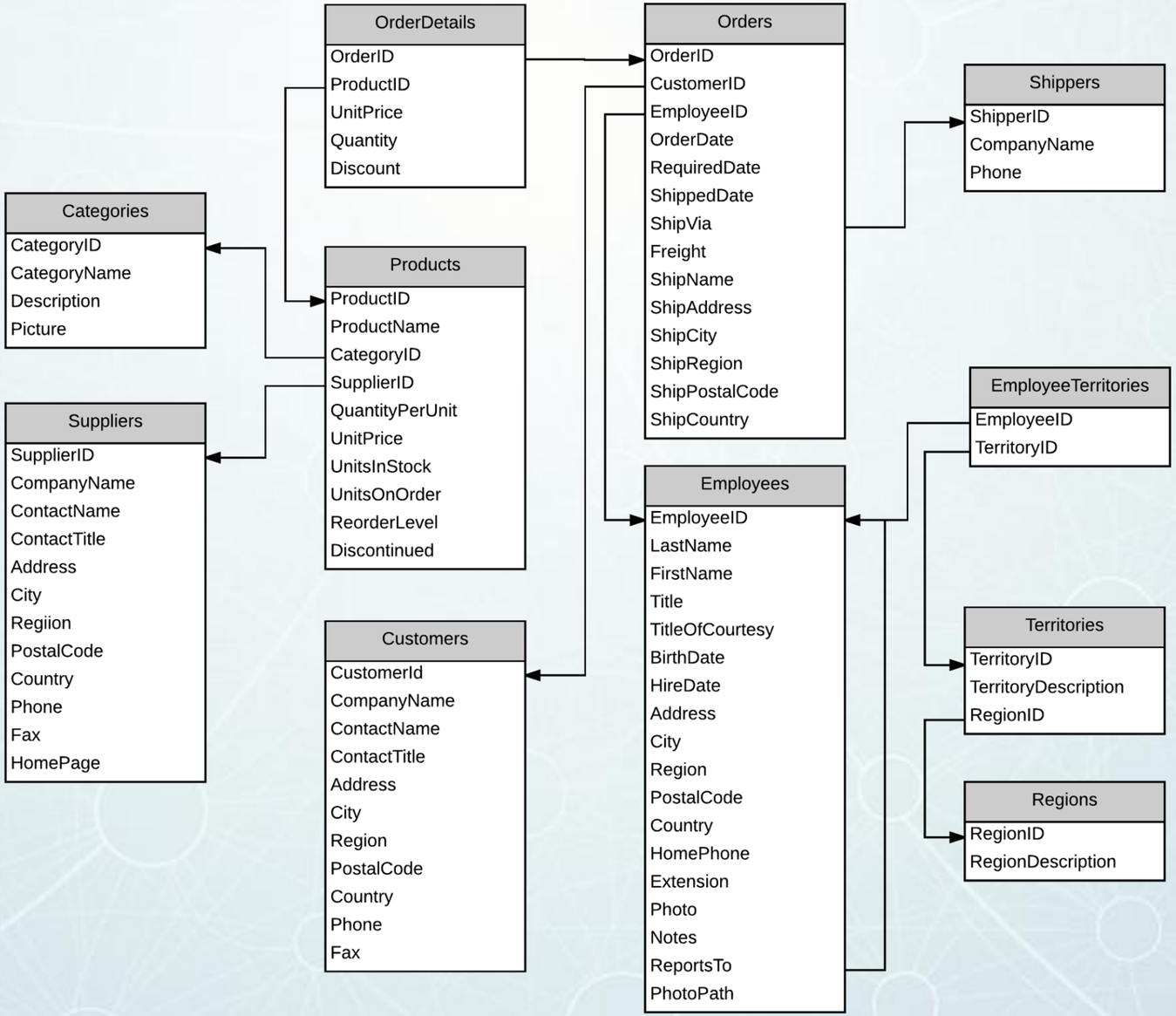
Northwind - the canonical RDBMS Example


(🌲)-[:TO]->(Graph)

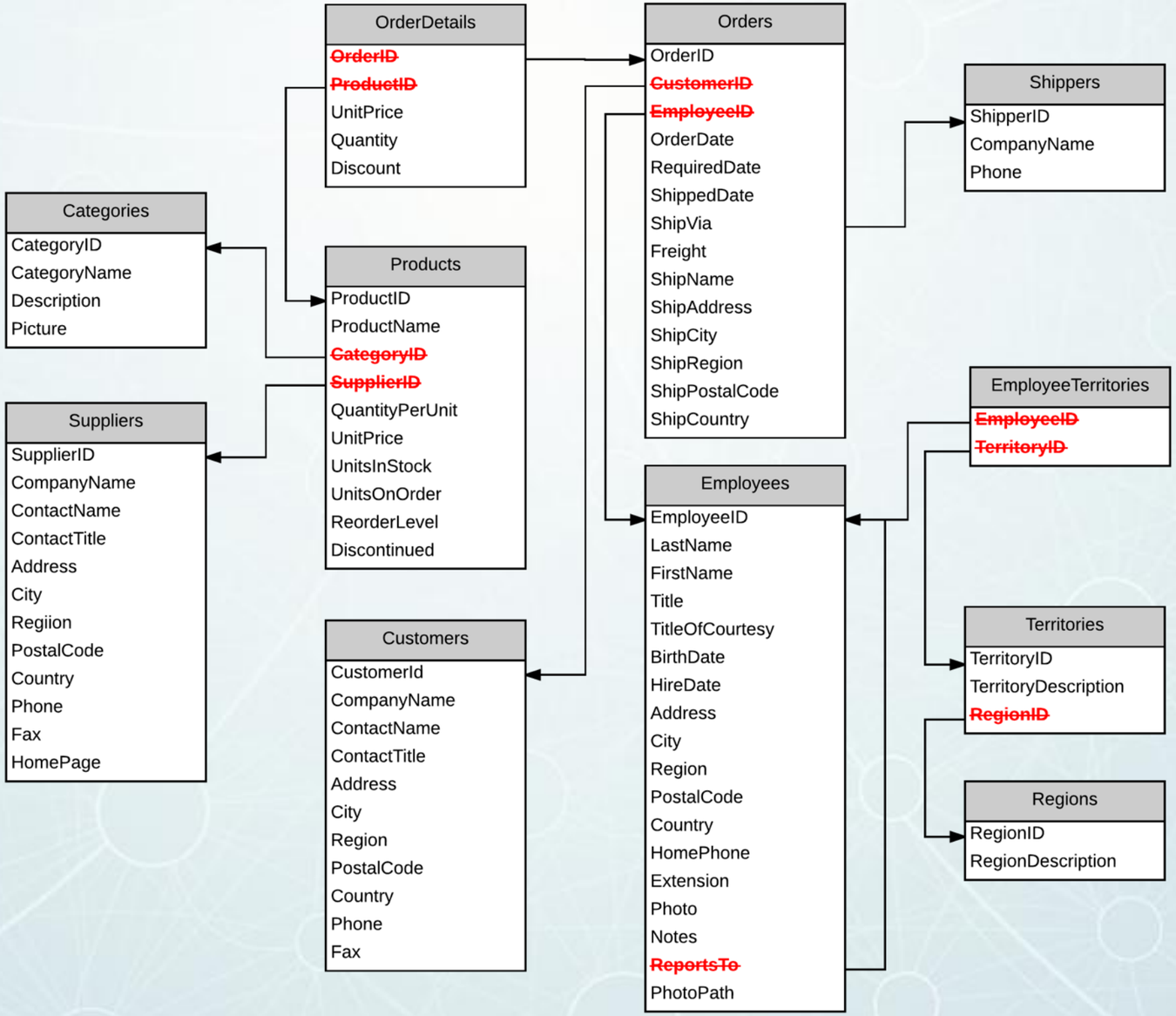

Region
IS_IN
Territory
REPRESENTS
REPORTS_TO
Employee
Category
IS_IN
Product
SUPPLIES
SOLD
INCLUDES
Supplier
SHIPPED
Order
Shipper


(🌲)-[:IS_BETTER_AS]->(Graph)

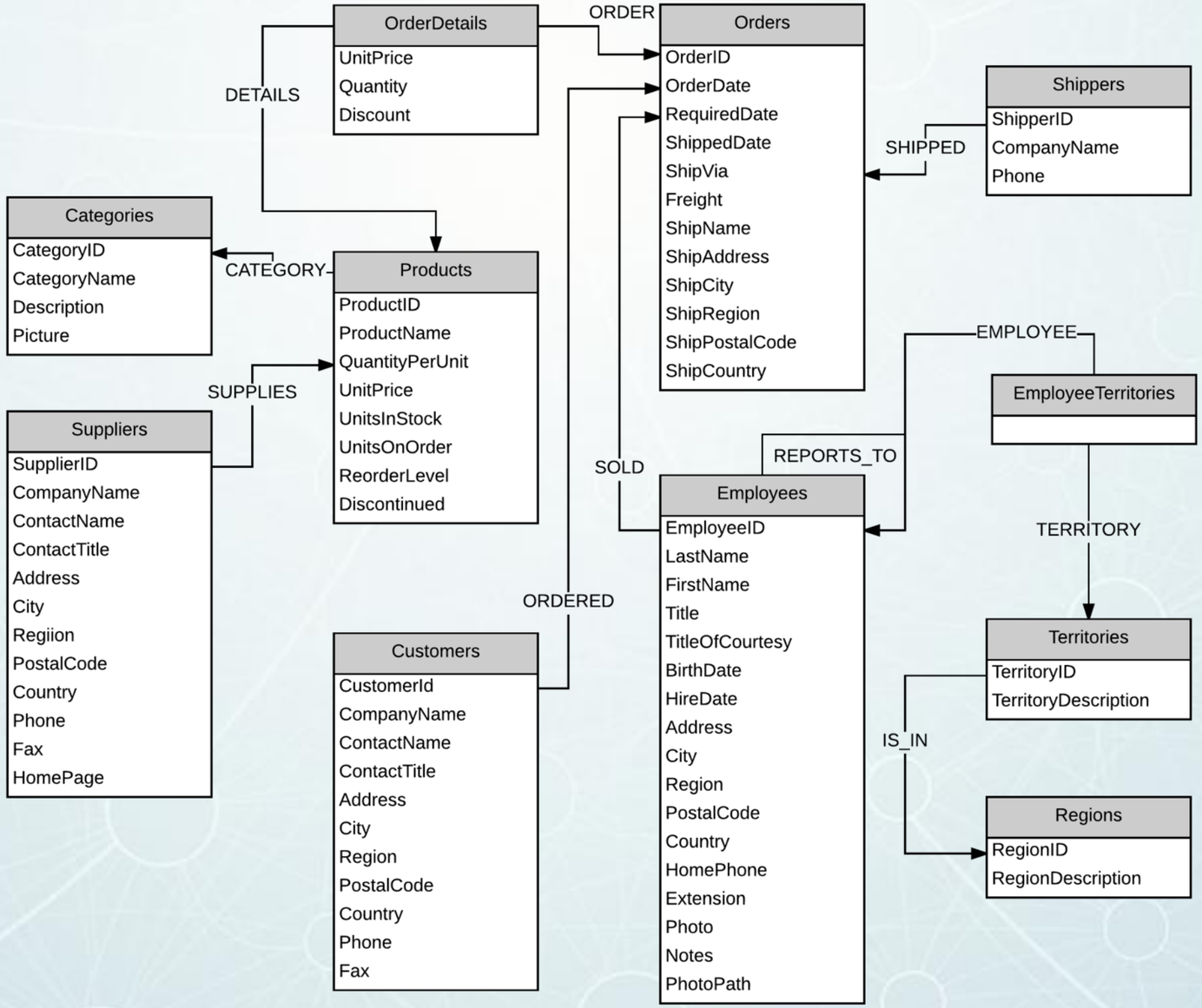
Starting with the ER Diagram



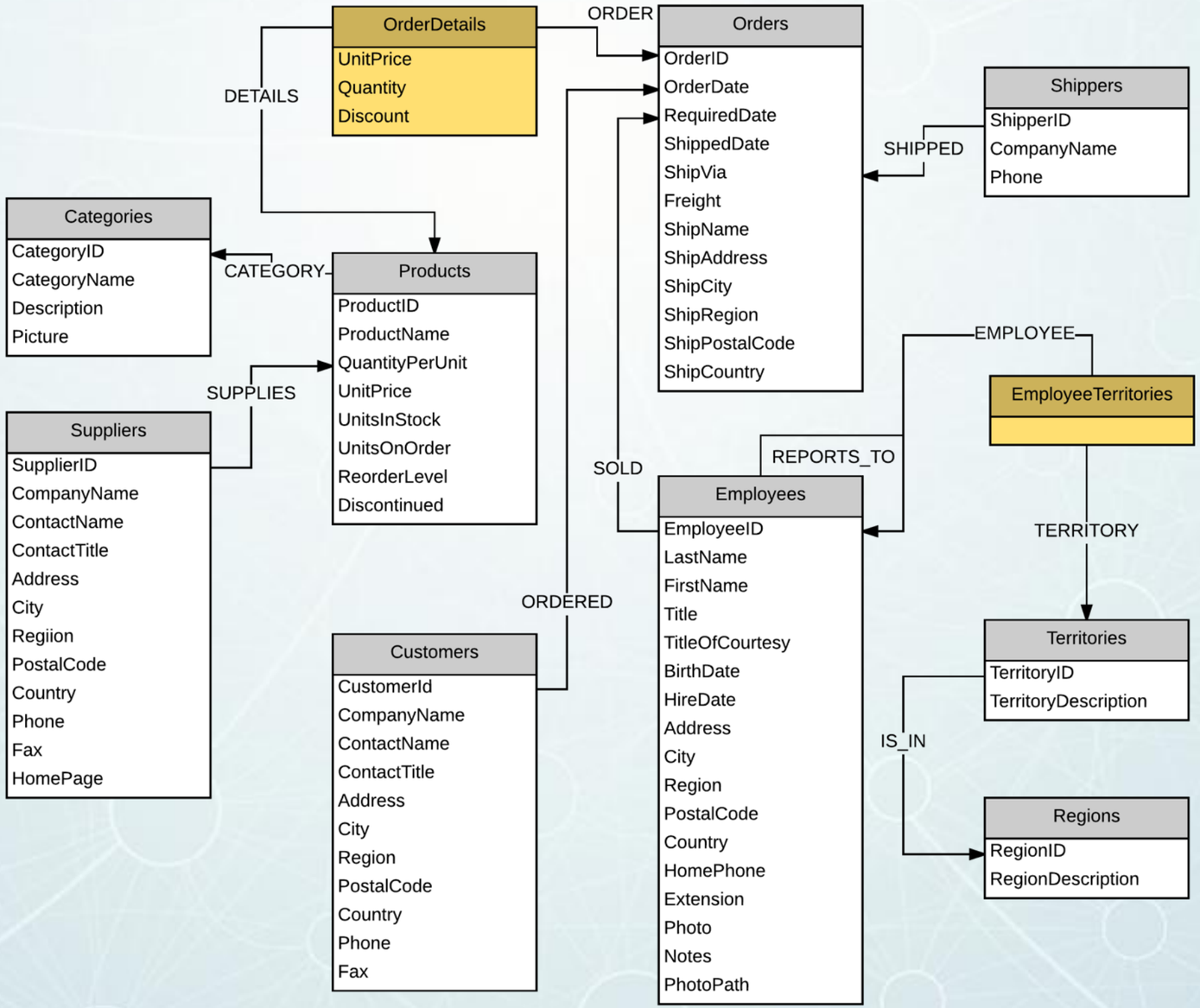
Locate the Foreign Keys



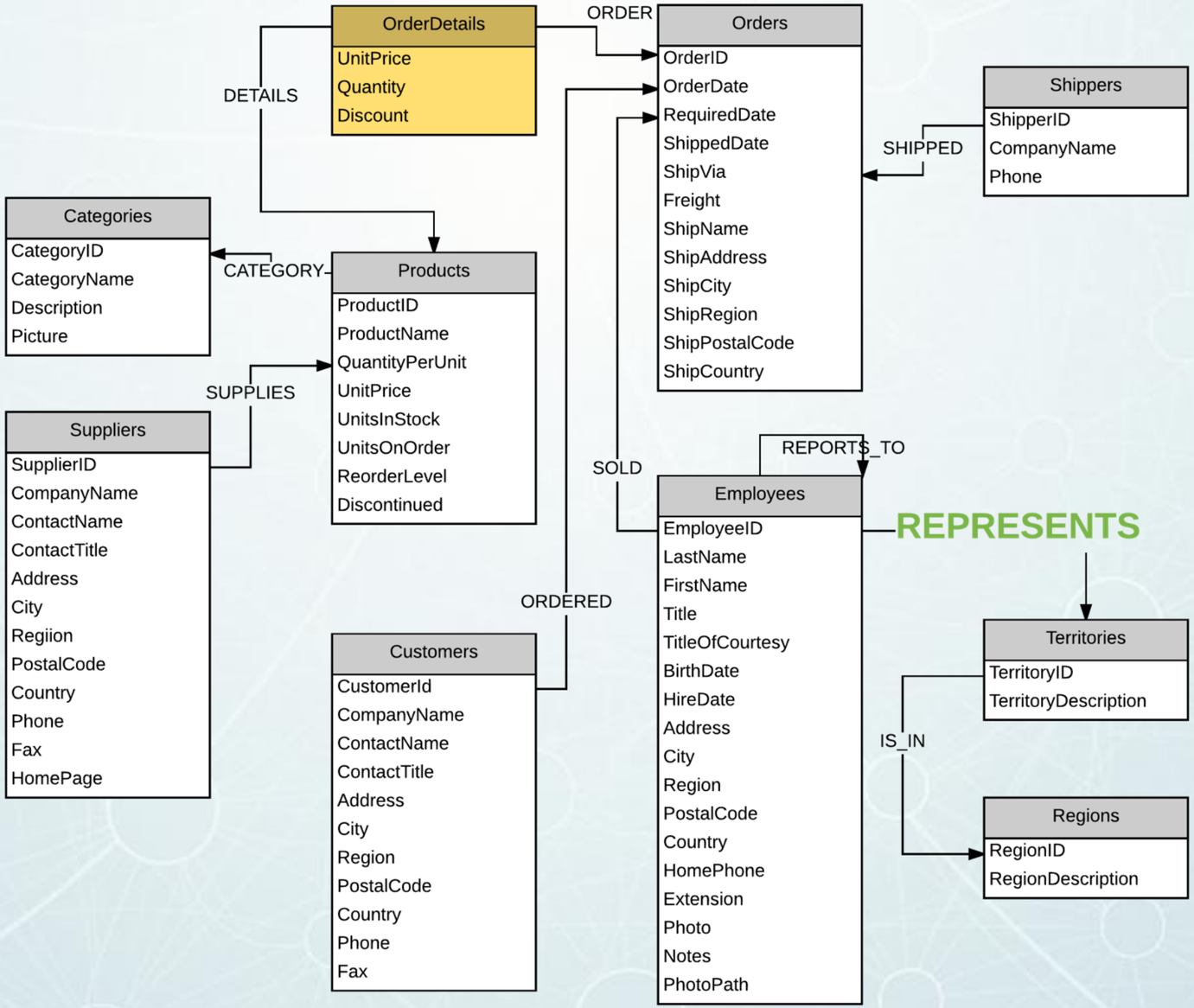
Drop the Foreign Keys



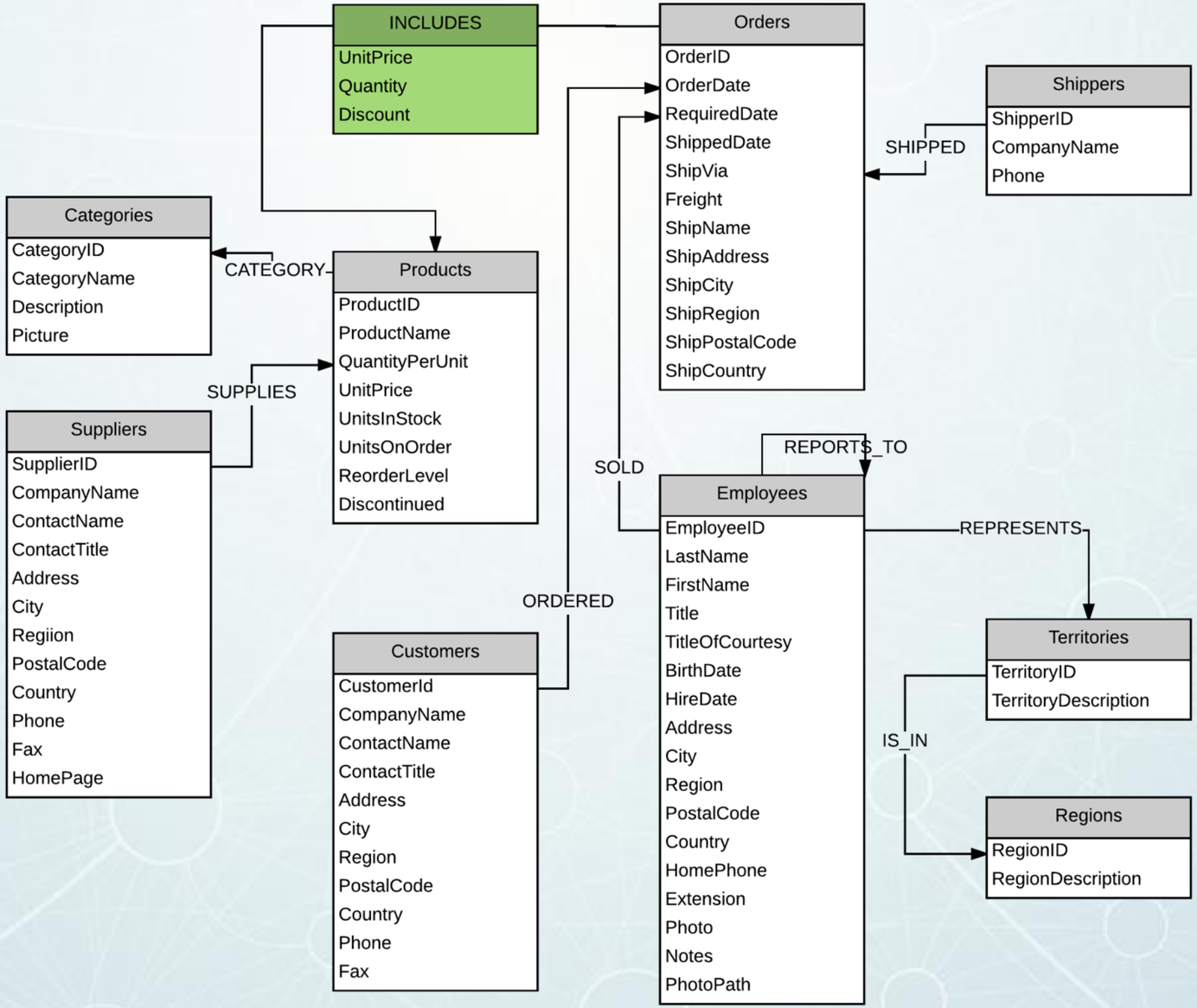
Find the JOIN Tables



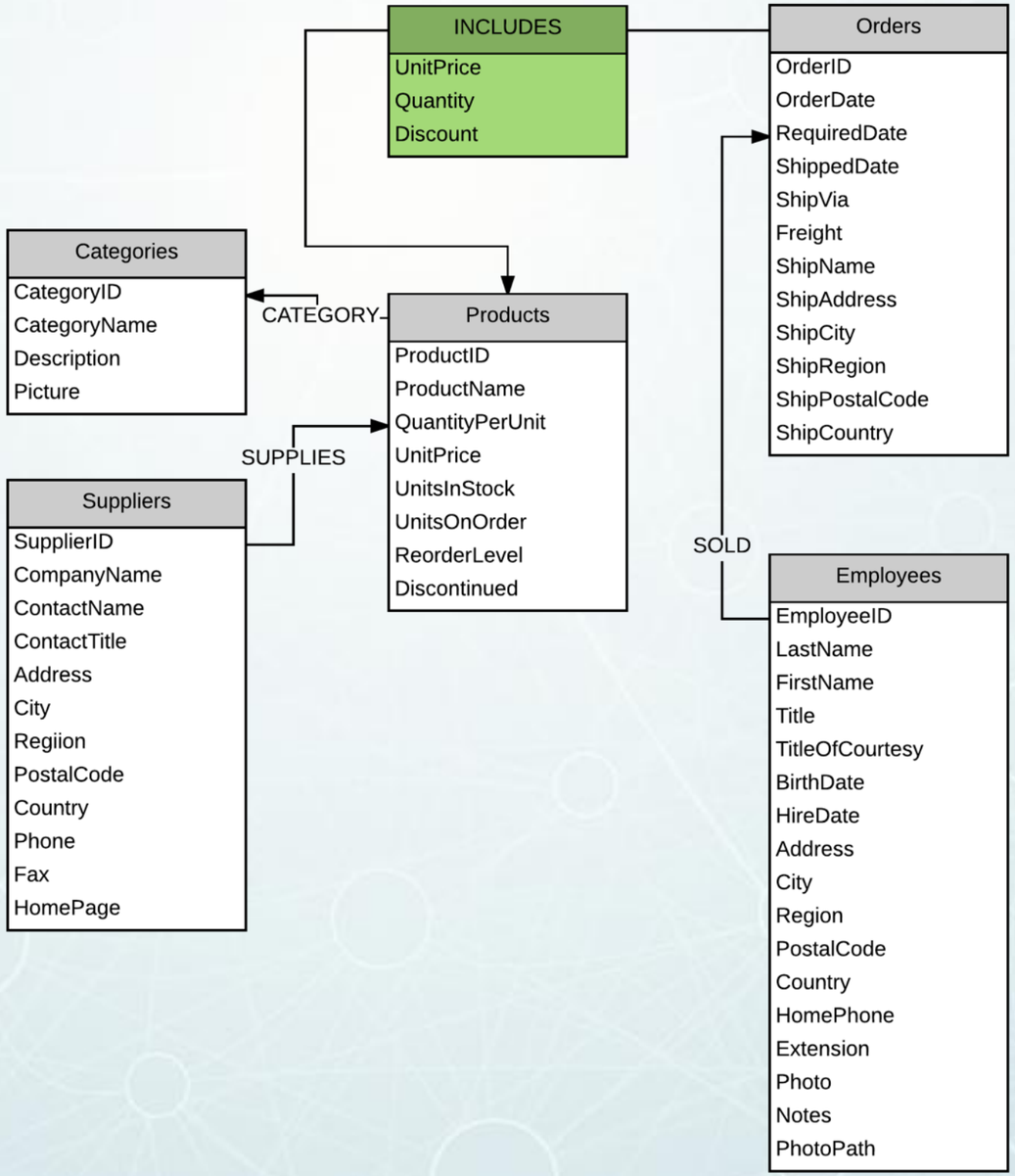
(Simple) JOIN Tables Become Relationships



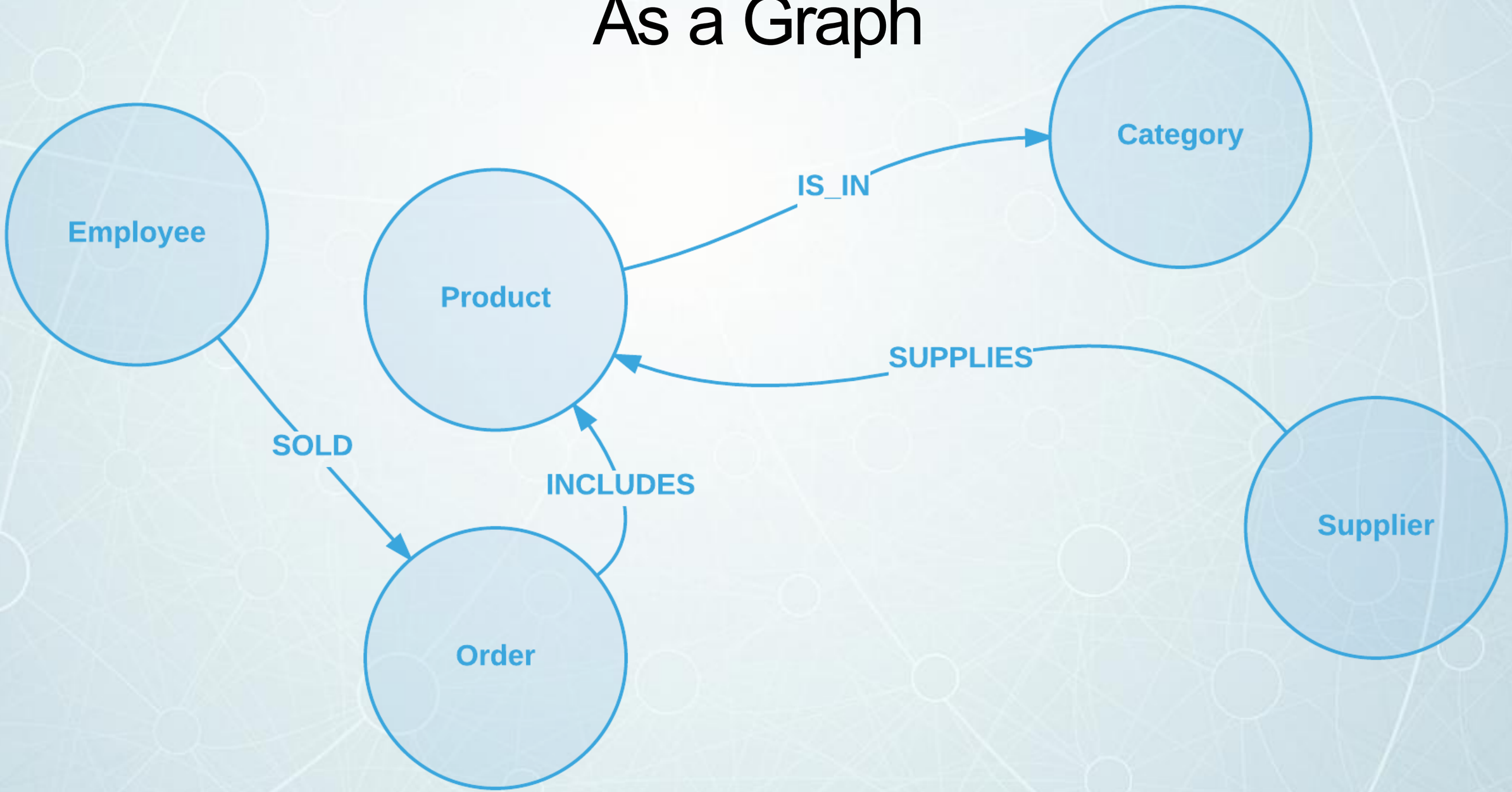
Attributed JOIN Tables -> Relationships with Properties



Querying a Subset Today



As a Graph


Employee
Product
Category
IS_IN
SUPPLIES
SOLD
INCLUDES
Order
Supplier

QUERYING THE GRAPH

using openCypher

Property Graph Model

NODE

NODE

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

LABEL

PROPERTY

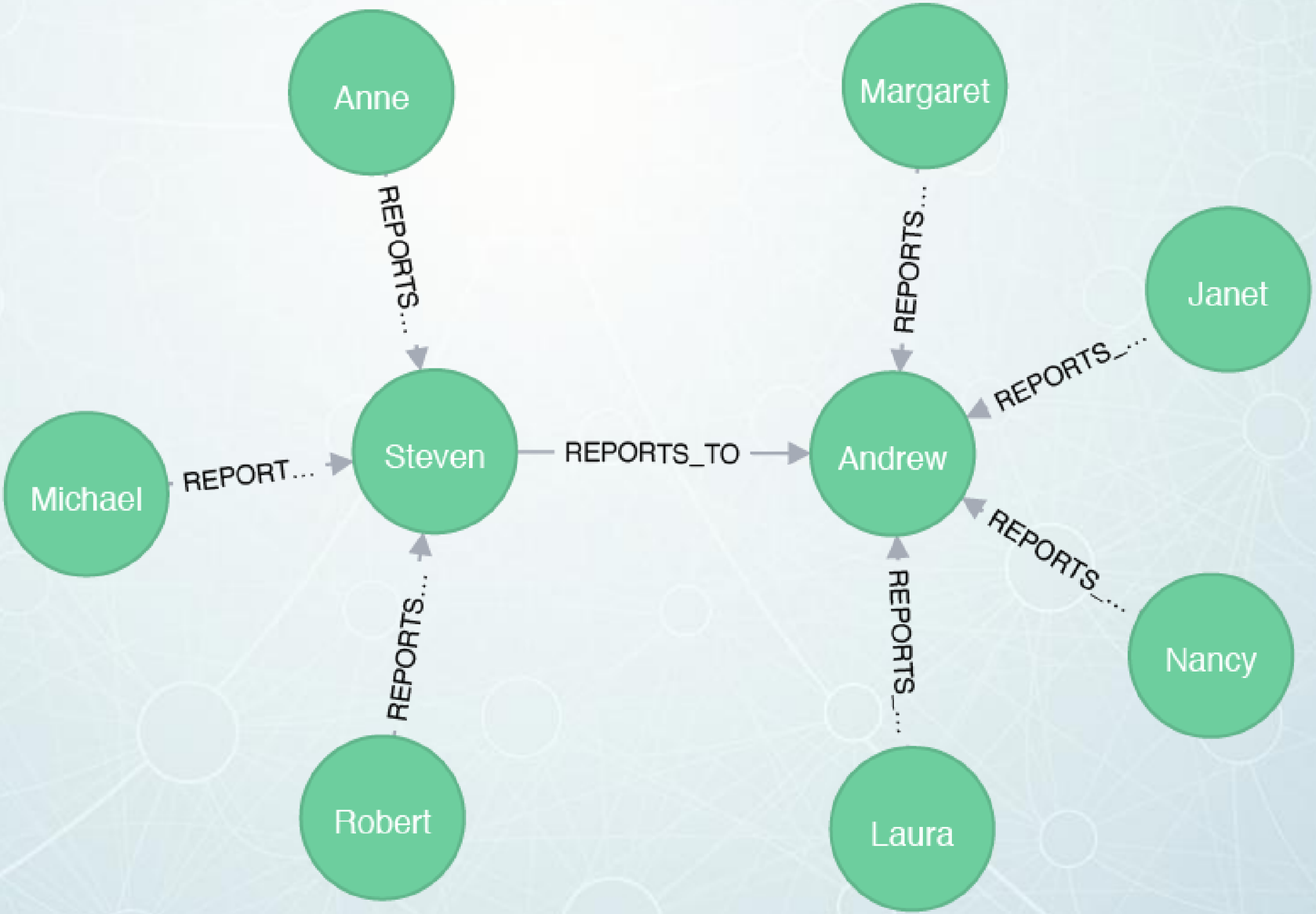
LABEL

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,
   e.firstName AS managerName,
   sub.employeeID AS employeeID,
   sub.firstName AS employeeName;
```

Who do people report to?

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

Rows

Code

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 does Robert report to?

What is Robert's reporting chain?

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

What is Robert's reporting chain?

Who's the Big Boss?

```
MATCH
   (e:Employee)
WHERE
   NOT (e)-[:REPORTS_TO]->()
RETURN
   e.firstName as bigBoss
```

Who’s the Big Boss?

Product Cross-Selling

```
MATCH
  (choc:Product {productName: 'Chocolade'})
   <-[:INCLUDES]-(:Order)<-[:SOLD]-(employee),
  (employee)-[:SOLD]->(o2)-[:INCLUDES]->(other:Product)
RETURN
  employee.firstName,
  other.productName,
  COUNT(DISTINCT o2) as count
ORDER BY
  count DESC
LIMIT 5;
```

Product Cross-Selling

```
$ MATCH (choc:Product {productName: 'Chocolade'}) <-[:INCLUDES]-(:Order)<...
```

Rows

Code

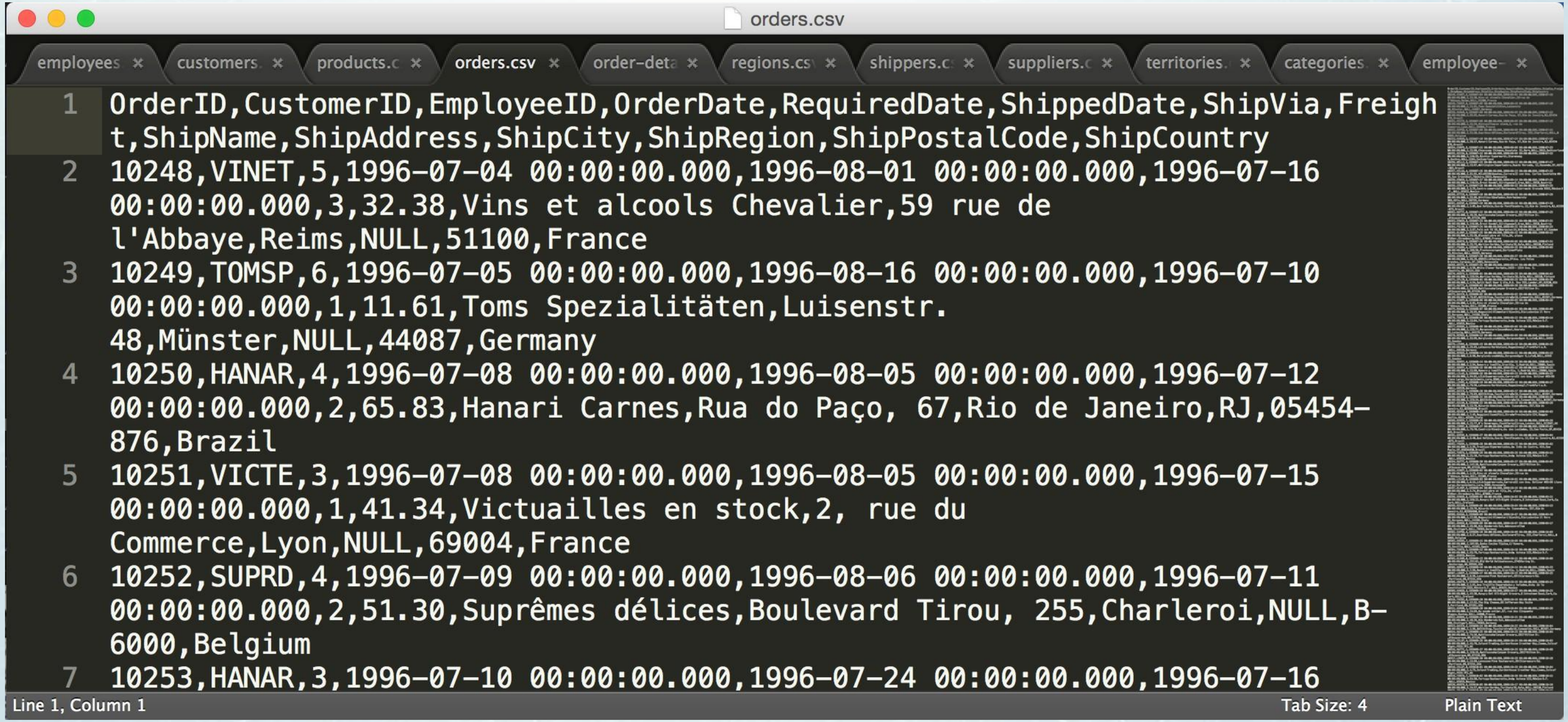
employee.firstName	other.productName	count
Margaret	Gnocchi di nonna Alice	14
Janet	Gumbär Gummibärchen	12
Nancy	Flotemysost	12
Margaret	Pâté chinois	12
Nancy	Camembert Pierrot	11

Returned 5 rows in 319 ms.

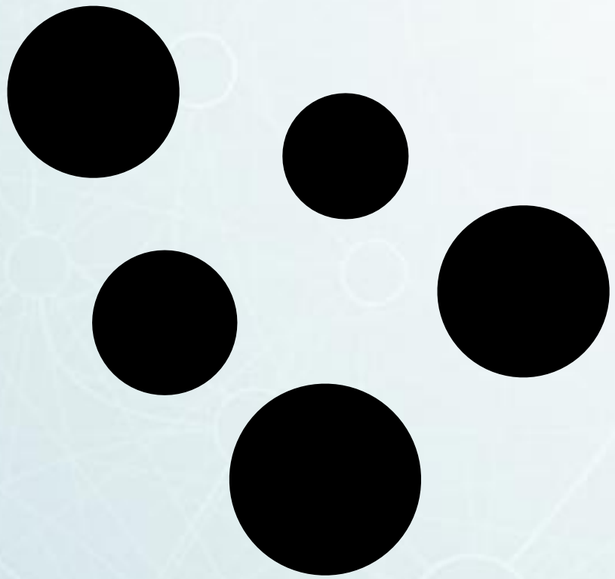
LOADING OUR DATA

CSV

CSV files for Northwind

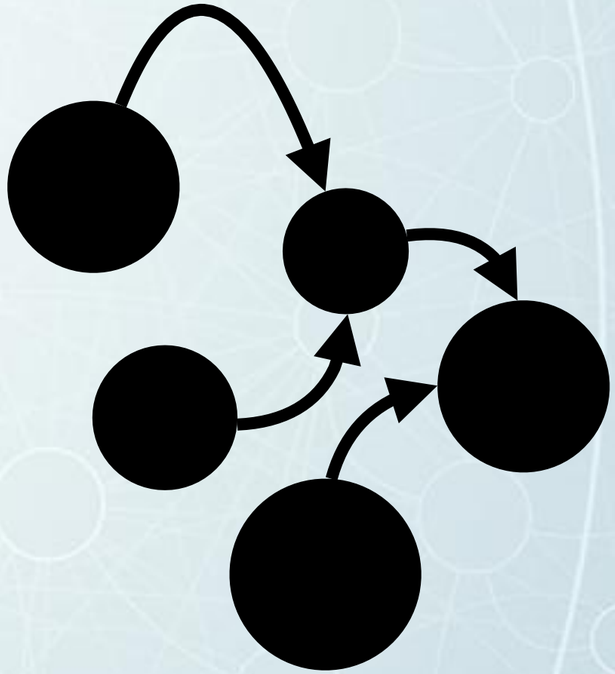


3 Steps to Creating the Graph



IMPORT NODES

CREATE INDEXES



IMPORT RELATIONSHIPS

Importing Nodes

```
// Create customers
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "https://
raw.githubusercontent.com/neo4j-contrib/developer-resources/
gh-pages/data/northwind/customers.csv" AS row
CREATE (:Customer {companyName: row.CompanyName, customerID:
row.CustomerID, fax: row.Fax, phone: row.Phone});

// Create products
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "https://
raw.githubusercontent.com/neo4j-contrib/developer-resources/
gh-pages/data/northwind/products.csv" AS row
CREATE (:Product {productName: row.ProductName, productID:
row.ProductID, unitPrice: toFloat(row.UnitPrice)});
```

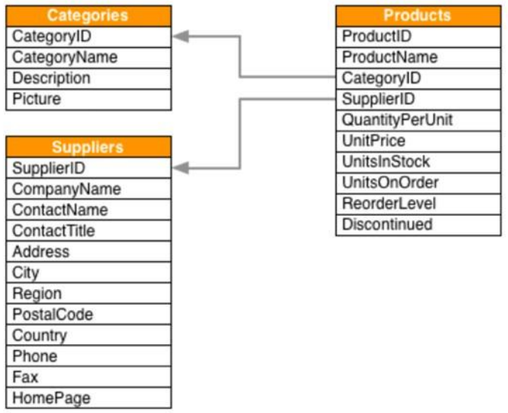
:play northward graph

```
:play northwind graph
```

Product Catalog

Northwind sells food products in a few categories, provided by suppliers. Let's start by loading the product catalog tables.

The load statements to the right require public internet access. `LOAD CSV` will retrieve a CSV file from a valid URL, applying a Cypher statement to each row using a named map (here we're using the name `row`).



:help cypher LOAD CSV

Load records

```
LOAD CSV WITH HEADERS FROM "http://data.neo4j.com/northwind/products.csv" AS row
CREATE (n:Product)
SET n = row,
  n.unitPrice = toFloat(row.unitPrice),
  n.unitsInStock = toInt(row.unitsInStock), n.unitsOnOrder = toInt(row.unitsOnOrder),
  n.reorderLevel = toInt(row.reorderLevel), n.discontinued = (row.discontinued <> "0")
```

```
LOAD CSV WITH HEADERS FROM "http://data.neo4j.com/northwind/categories.csv" AS row
CREATE (n:Category)
SET n = row
```

```
LOAD CSV WITH HEADERS FROM "http://data.neo4j.com/northwind/suppliers.csv" AS row
CREATE (n:Supplier)
SET n = row
```

Create indexes

```
CREATE INDEX ON :Product(productID)
```

```
CREATE INDEX ON :Category(categoryID)
```

```
CREATE INDEX ON :Supplier(supplierID)
```

High Performance LOADing

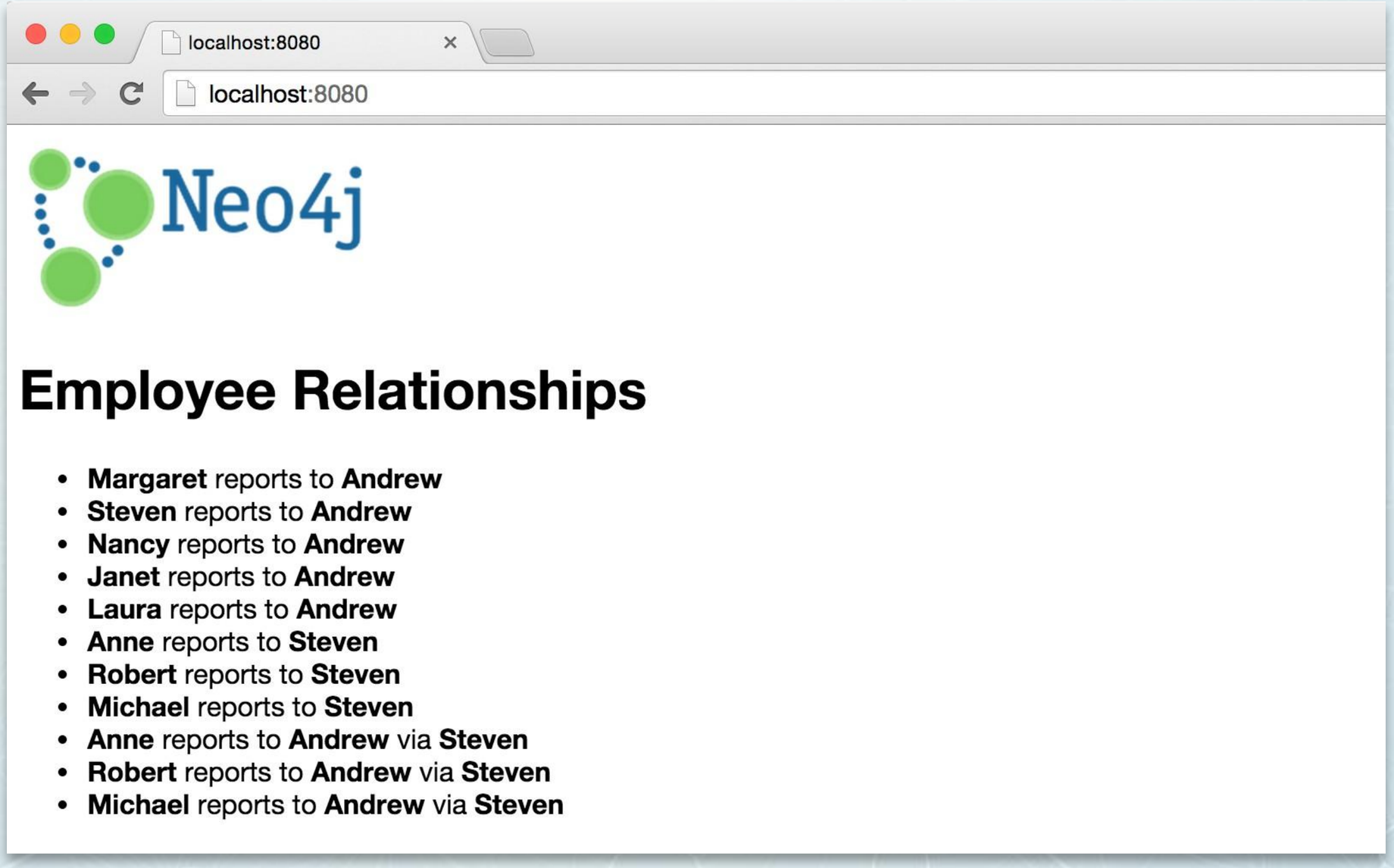
neo4j-import

4.58 million things
and their relationships…

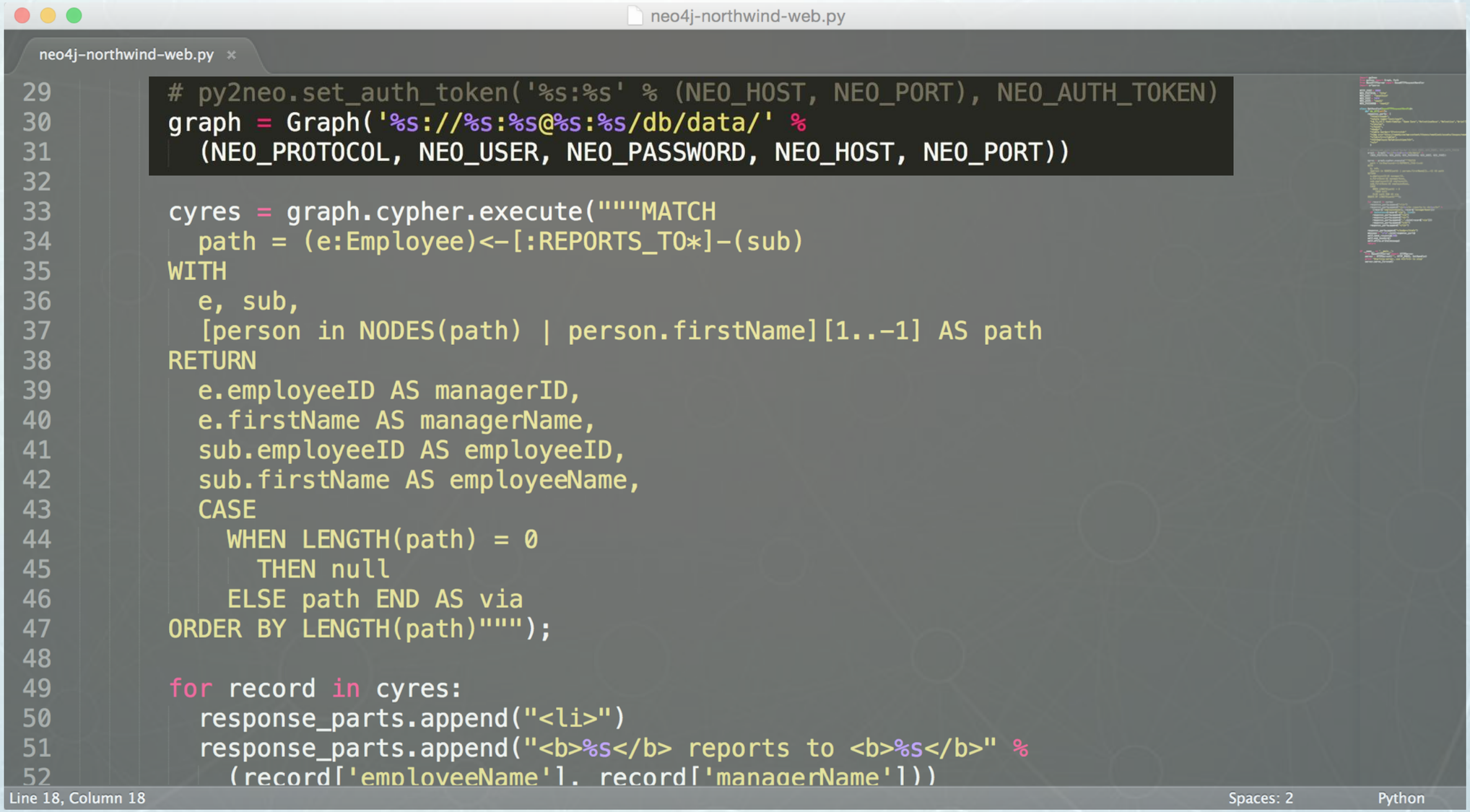
Loads in 100 seconds!

POWERING AN APP

Simple App


localhost:8080
localhost:8080
Neo4j
Employee Relationships
Margaret reports to Andrew
Steven reports to Andrew
Nancy reports to Andrew
Janet reports to Andrew
Laura reports to Andrew
Anne reports to Steven
Robert reports to Steven
Michael reports to Steven
Anne reports to Andrew via Steven
Robert reports to Andrew via Steven
Michael reports to Andrew via Steven

Simple Python Code



Simple Python Code

neo4j-northwind-web.py

```
29      # py2neo.set_auth_token('%s:%s' % (NEO_HOST, NEO_PORT), NEO_AUTH_TOKEN)
30      graph = Graph('%s://%s:%s@%s:%s/db/data/' %
31        (NEO_PROTOCOL, NEO_USER, NEO_PASSWORD, NEO_HOST, NEO_PORT))
32
33      cyres = graph.cypher.execute("""MATCH
34        path = (e:Employee)<-[:REPORTS_TO*]-(sub)
35      WITH
36        e, sub,
37        [person in NODES(path) | person.firstName][1..-1] AS path
38      RETURN
39        e.employeeID AS managerID,
40        e.firstName AS managerName,
41        sub.employeeID AS employeeID,
42        sub.firstName AS employeeName,
43        CASE
44          WHEN LENGTH(path) = 0
45            THEN null
46          ELSE path END AS via
47      ORDER BY LENGTH(path)""");
48
49      for record in cyres:
50        response_parts.append("<li>")
51        response_parts.append("<b>%s</b> reports to <b>%s</b>" %
52          (record['employeeName'], record['managerName']))
```

Line 18, Column 18 Spaces: 2 Python

Simple Python Code

neo4j-northwind-web.py

```
29        # py2neo.set_auth_token('%s:%s' % (NEO_HOST, NEO_PORT), NEO_AUTH_TOKEN)
30        graph = Graph('%s://%s:%s@%s:%s/db/data/' %
31          (NEO_PROTOCOL, NEO_USER, NEO_PASSWORD, NEO_HOST, NEO_PORT))
32
33        cyres = graph.cypher.execute("""MATCH
34          path = (e:Employee)<-[:REPORTS_TO*]-(sub)
35        WITH
36          e, sub,
37          [person in NODES(path) | person.firstName][1..-1] AS path
38        RETURN
39          e.employeeID AS managerID,
40          e.firstName AS managerName,
41          sub.employeeID AS employeeID,
42          sub.firstName AS employeeName,
43          CASE
44            WHEN LENGTH(path) = 0
45              THEN null
46            ELSE path END AS via
47        ORDER BY LENGTH(path)""");
48
49        for record in cyres:
50          response_parts.append("<li>")
51          response_parts.append("<b>%s</b> reports to <b>%s</b>" %
52            (record['employeeName'], record['managerName']))
```

Line 18, Column 18 Spaces: 2 Python

Simple Python Code

neo4j-northwind-web.py

```
29        # py2neo.set_auth_token('%s:%s' % (NEO_HOST, NEO_PORT), NEO_AUTH_TOKEN)
30        graph = Graph('%s://%s:%s@%s:%s/db/data/' %
31          (NEO_PROTOCOL, NEO_USER, NEO_PASSWORD, NEO_HOST, NEO_PORT))
32
33        cyres = graph.cypher.execute("""MATCH
34          path = (e:Employee)<-[:REPORTS_TO*]-(sub)
35        WITH
36          e, sub,
37          [person in NODES(path) | person.firstName][1..-1] AS path
38        RETURN
39          e.employeeID AS managerID,
40          e.firstName AS managerName,
41          sub.employeeID AS employeeID,
42          sub.firstName AS employeeName,
43          CASE
44            WHEN LENGTH(path) = 0
45              THEN null
46            ELSE path END AS via
47        ORDER BY LENGTH(path)""");
48
49        for record in cyres:
50          response_parts.append("<li>")
51          response_parts.append("<b>%s</b> reports to <b>%s</b>" %
52            (record['employeeName'], record['managerName']))
```

Line 18, Column 18 Spaces: 2 Python

Resources

Install Neo4j and Take it for a Spin

Experience powerful scalability, blazing speed and unparalleled flexibility – download and try Neo4j today.

For Business

The Neo4j Enterprise Edition offers incredible power and flexibility, with enterprise-grade availability, management and scale-up & scale-out capabilities.

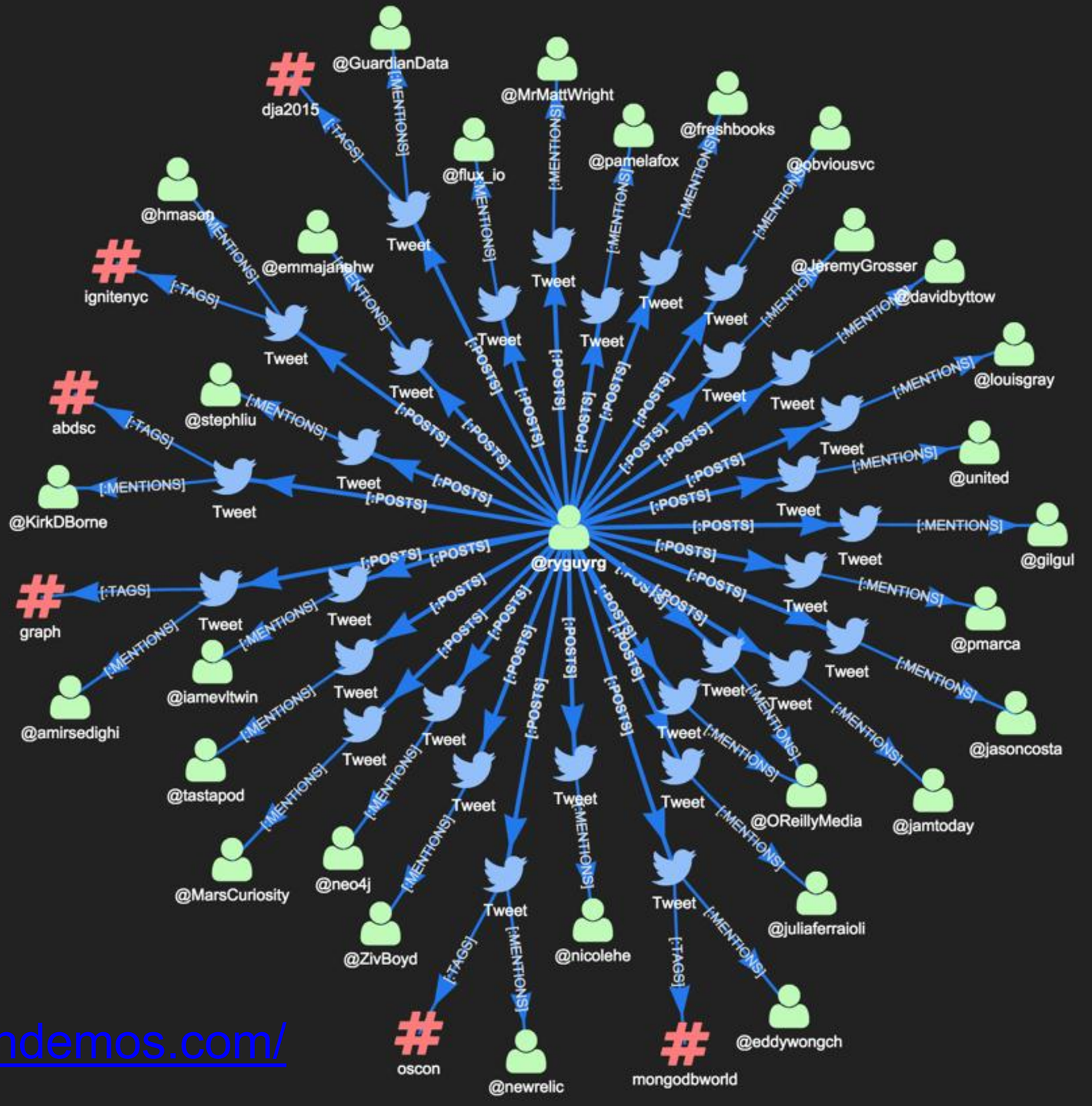
Download Free Enterprise Trial

For Individuals

Ideal for learning, and smaller do-it-yourself projects that do not require high levels of scaling. Excludes professional services and support.

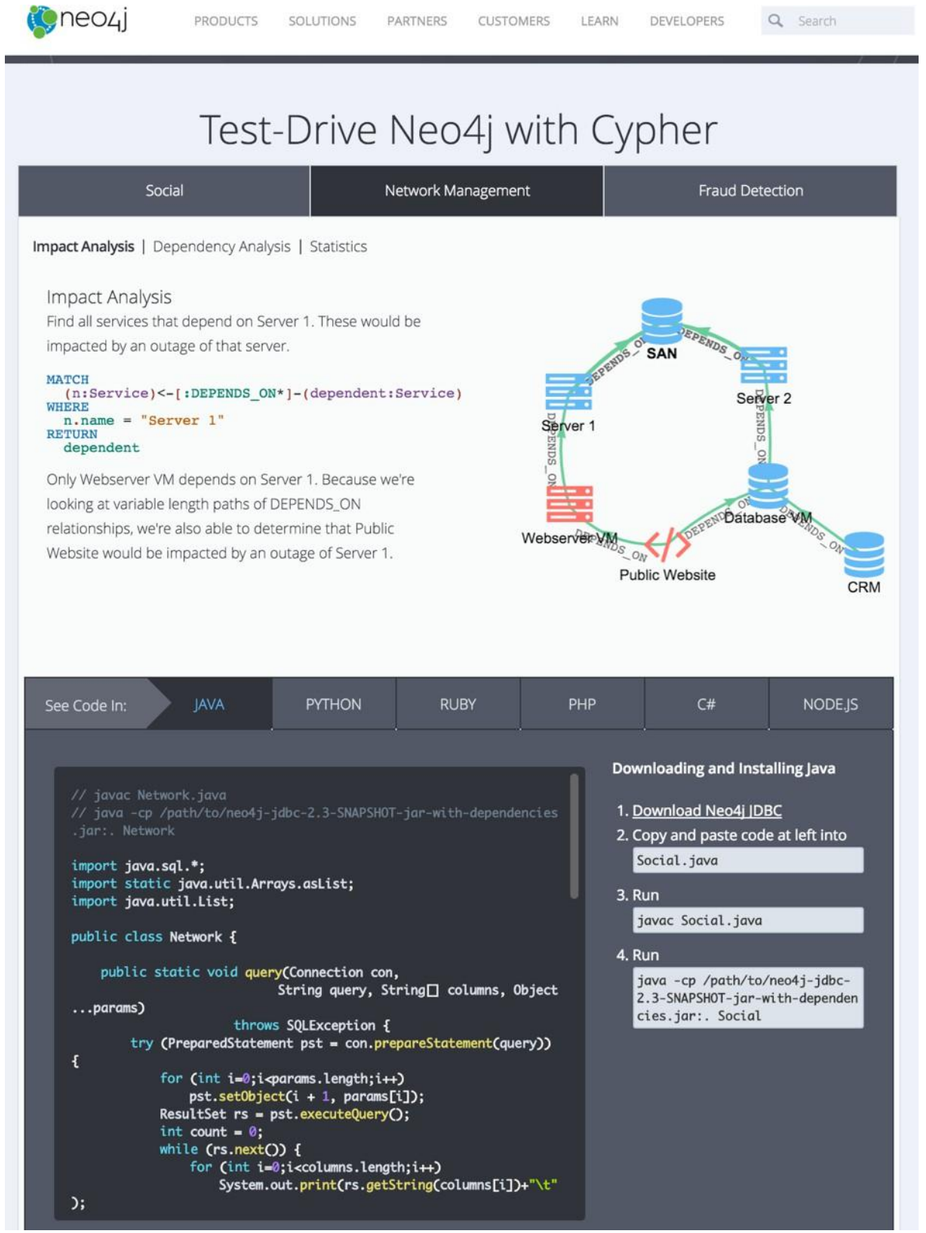
Download Community Edition

neo4j.com/download



http://network.graphdemos.com/

neo4j.com/developer



There Are Lots of Ways to Easily Learn Neo4j

neo4j

community

Graph Academy
Learn. Graph. Deploy.

stackoverflow

slack

On-site Training

BOOKS

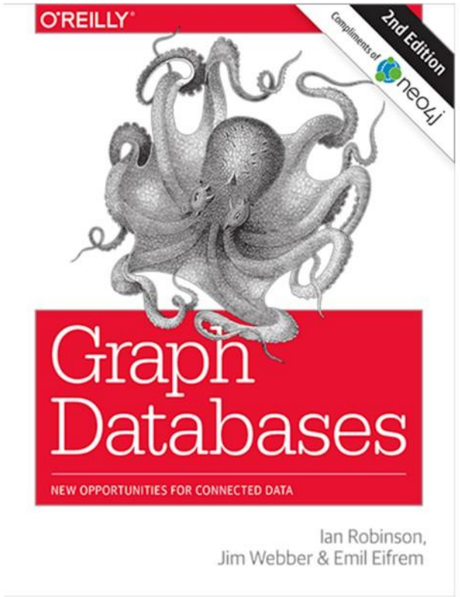
Documentation

GraphGist

Built-in Guides

Google Groups

Online Training



***UPDATE!* 2nd Edition**

O'Reilly's *Graph Databases*

The Definitive Book on Graph Databases.

Official Released Version of O'Reilly's Graph Databases.

Graph Databases

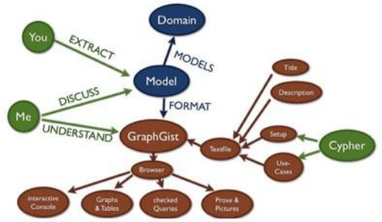
The Definitive Book on Graph Databases and Introduction to Neo4j

graphdatabases.com

GraphGists

What is a GraphGist?

With Neo4j GraphGists you can describe and model your domain in a simple text file (AsciiDoc) and render it as a rich, interactive, database-backed page in any browser. It is perfect to document a specific domain, use-case, question or graph problem.



Examples

GraphGists work like any AsciiDoc document, but they allow you to insert special comments to define how data from Neo4j can be displayed and interacted with.

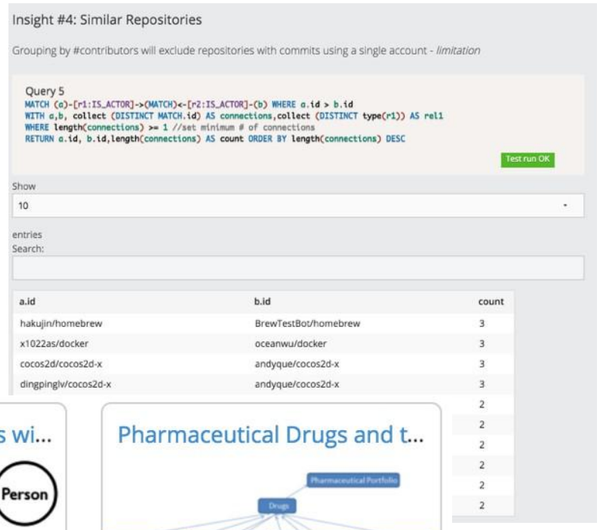
First define a block of Cypher code to setup the database:

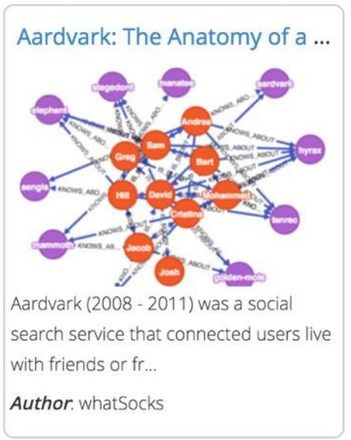
```
//setup
//hide
[source,cypher]
----
CREATE (:Database {name:'Neo4j'})-[:SUPPORTS]->(:Language {name:'Cypher'})
----
```

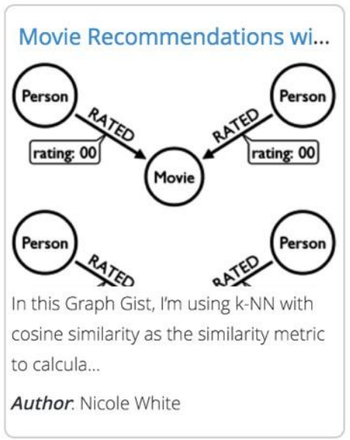
You can then make queries to the data and output them as a table and/or a graph:

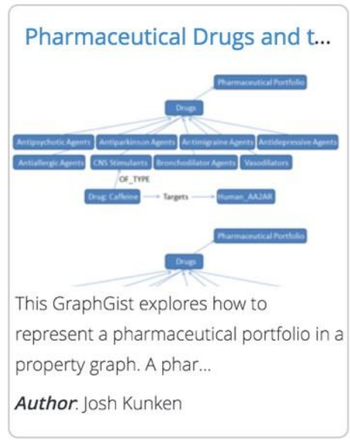
```
[source,cypher]
----
MATCH (db:Database)-[:SUPPORTS]->(language:Language)
RETURN db.name, collect(language.name)
----

//table
```

```
[source,cypher]
----
MATCH (:Database)-[rel:SUPPORTS]->(:Language)
RETURN rel
----

//graph
```









http://neo4j.com/graphgists/

THANK YOU!

will@neo4j.com

@lyonwj