

# Neo4j

## A Graph Database Intro

Team members: Jiepeng Zhang, Zhenhua Li & Sha Liu



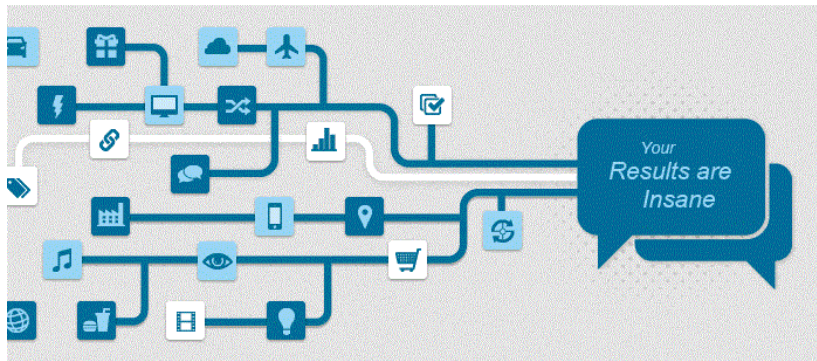
# Content

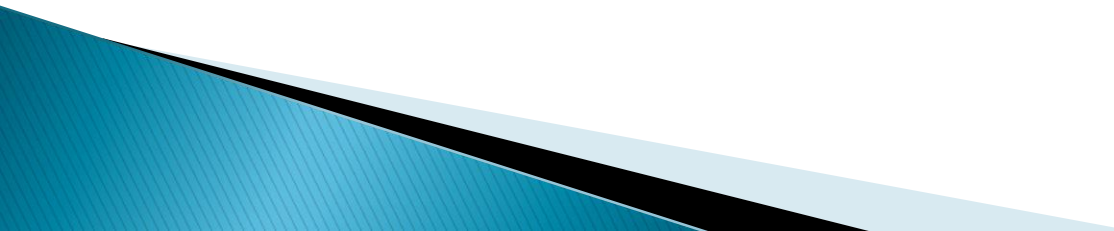
- ▶ 1. Introduction to Neo4j
  - ✓ What is Graph Database?
  - ✓ What is Neo4j?
  - ✓ Why Neo4j?
  - ✓ How to use Neo4j?
- ▶ 2. Proposed Application
- ▶ 3. Application Example

# 1 Introduction to Neo4j

# 1.1 What is Graph Database?

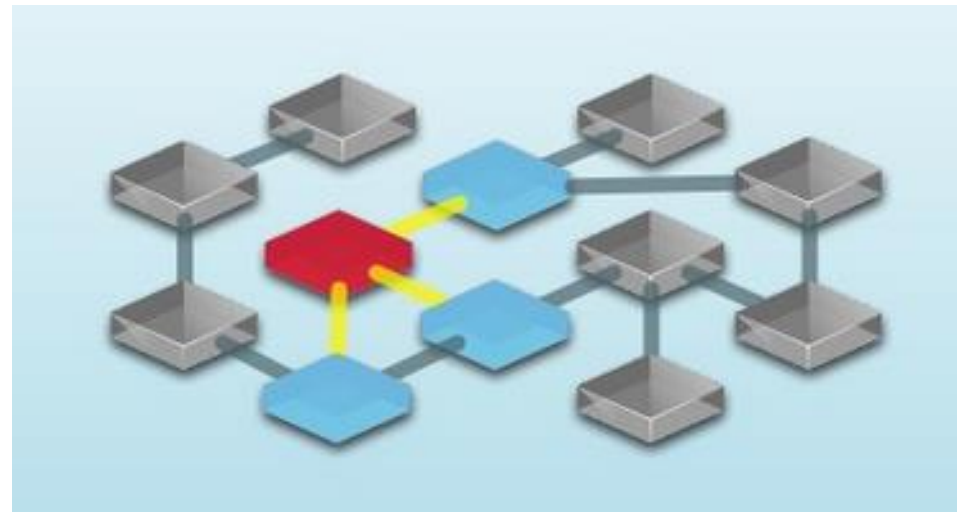
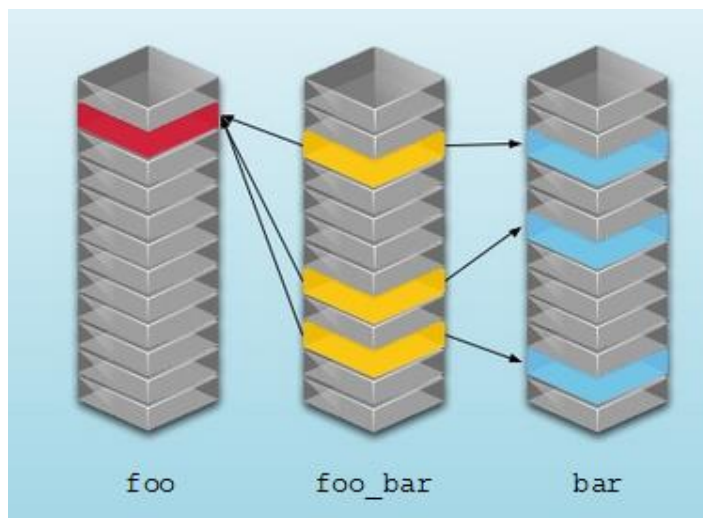
Graphs Everywhere...



- ▶ A graph database?
  - ▶ **No:** not for storing charts & graphs, or vector artwork
  - ▶ **Yes:** for storing data that is structured as a graph
    - ✓ remember linked lists, tree?
    - ✓ graphs are the generalized connected data structure
- 

# 1.1 What is Graph Database?

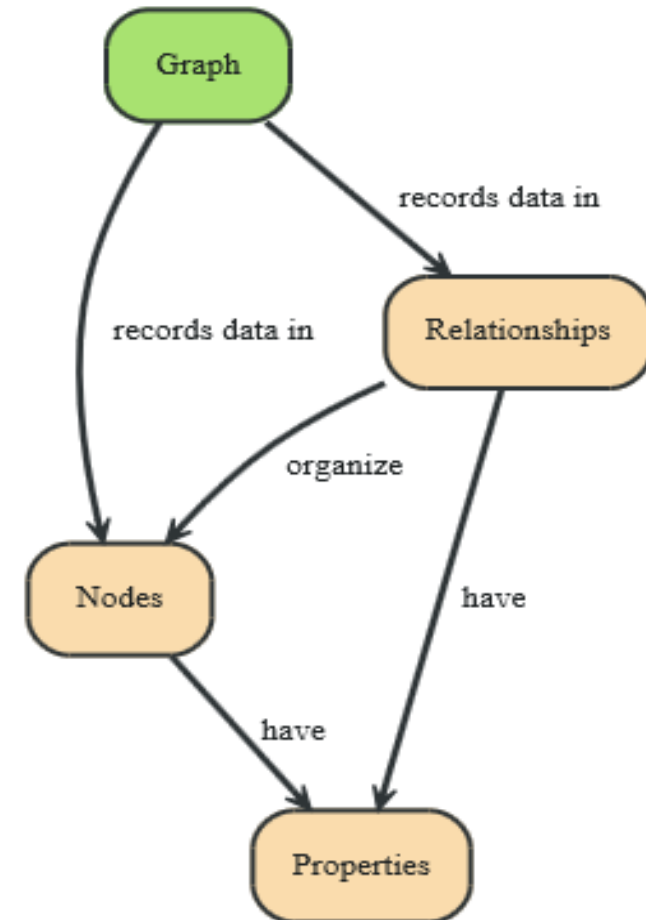
- ▶ You know relational
- ▶ Now consider relationships...



# 1.1 What is Graph Database?

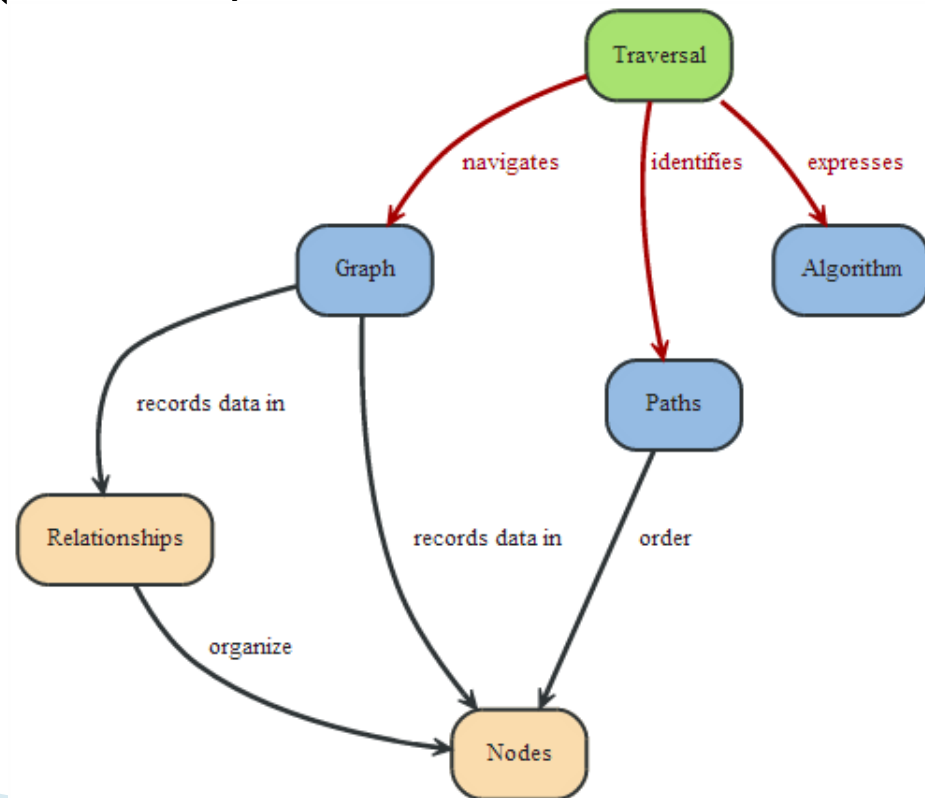
$$G = (V, E)$$

- ▶ A (Graph) - records data in- (Nodes)
- ▶ A (Graph) - records data in - (Relationships)
- ▶ (Nodes) are - organized by - (Relationships)
- ▶ (Nodes & Relationships) - have - (Properties)



# Query a graph with a traversal

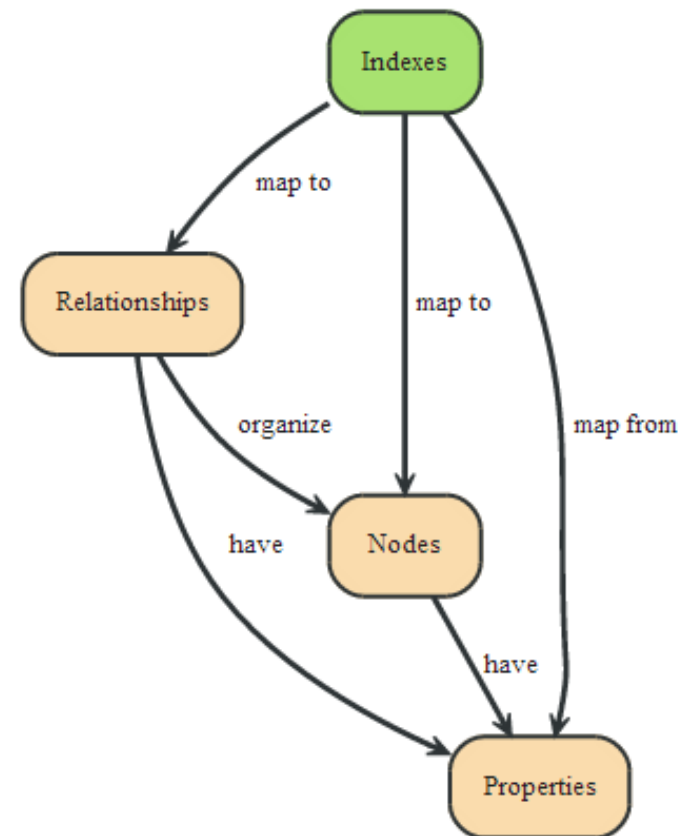
- ▶ A (Traversal) – navigates – a (Graph)
- ▶ A (Traversal) – identifies – (Paths)
- ▶ (Paths) – order – (Nodes)





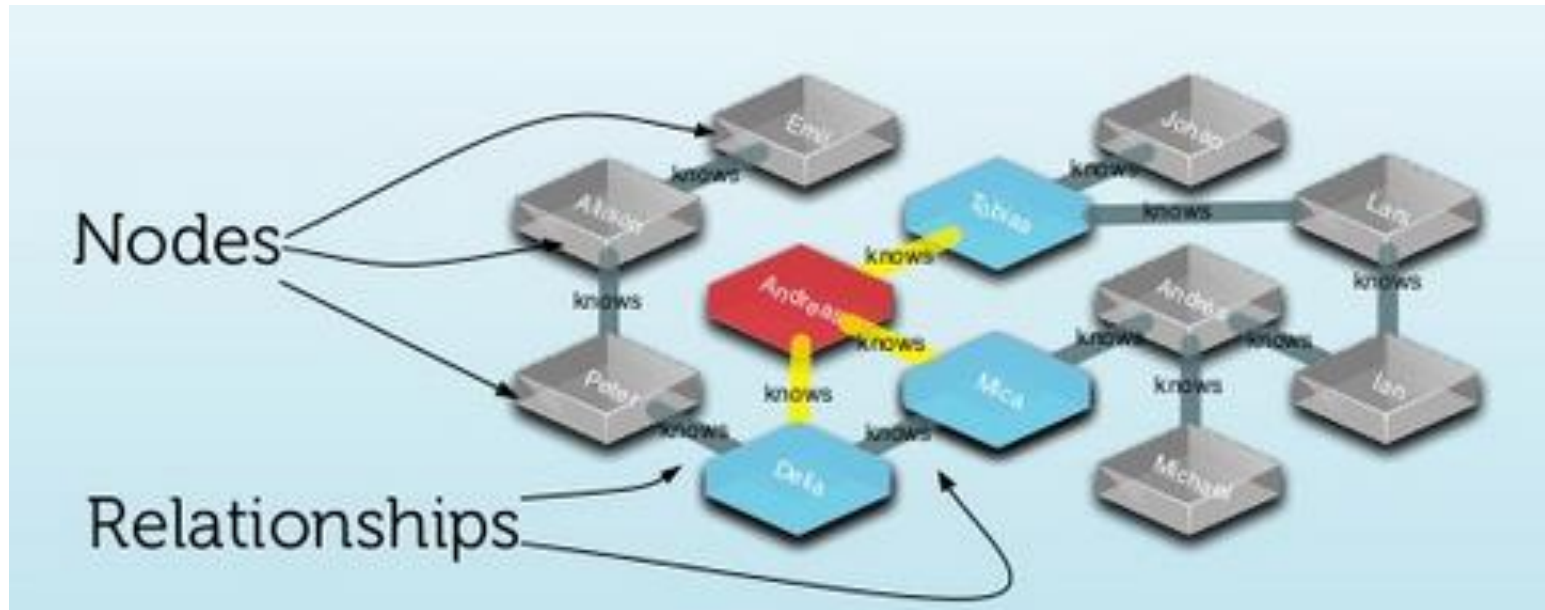
# Indexes look-up Nodes or Relationships

- ▶ An (Index) is a -special- (Traversal)
- ▶ An (Index) - maps from properties -to either - (Nodes or Relationships)



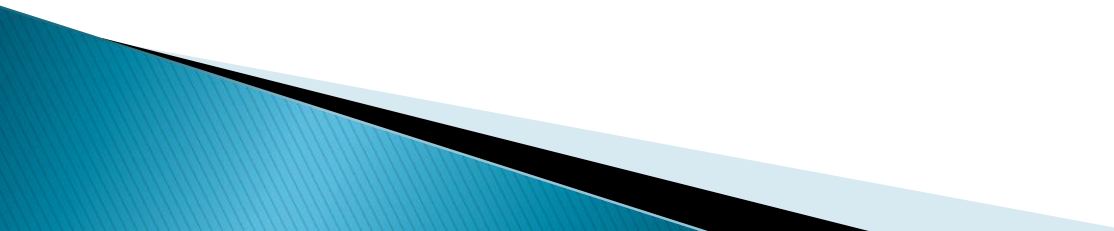
# 1.1 What is Graph Database?

- ▶ We're talking about a Property Graph



Properties (each a key + value)

# A graph database...

- ▶ Optimized for the connections between records
  - ▶ Really, really fast at querying across records
  - ▶ A database: transactional with the usual operations
  - ▶ “ A relational database may tell you the average age of everyone in this session,
  - ▶ but a graph database will tell you who is most likely to buy you a beer.”
- 

# 1.2 What is Neo4j?

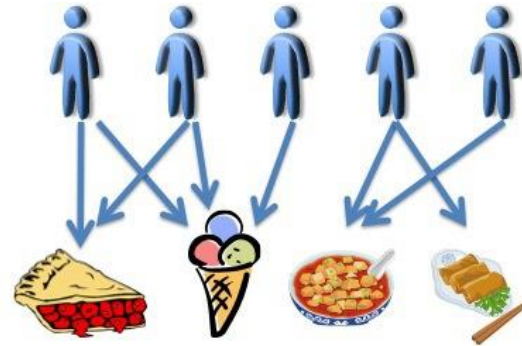
- A robust and high-performance graph database
- Full enterprise deployment or lightweight projects
- ▶ True ACID transactions (Atomicity, Consistency, Isolation, Durability)
- ▶ High availability
- ▶ Scales to billions of nodes and relationships
- ▶ High speed querying through traversals

## 1.2 Why Neo4j?

 What are graphs good for?

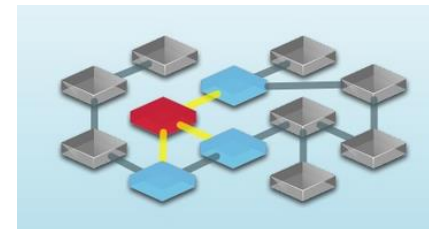
A: highly - connected data

- ✓ Recommendations
- ✓ Business intelligence
- ✓ Social computing
- ✓ Web analytics
- ✓ Geospatial
- ✓ And much more!



# Example: Social Network “path exists”

- ▶ A sample social graph with 1000 persons
- ▶ Average 50 friends per person
- ▶ `pathExists(a, b)` limited to depth 4
- ▶ Caches warmed up to eliminate disk I/O



	# persons	Query time
Relational database	1,000	2000ms
Neo4j	1,000	2ms
Neo4j	1,000,000	2ms

What's a Graph Database?

DEFINED AS

A graph database is a database that uses graph structures with nodes, edges and properties to represent and store information.

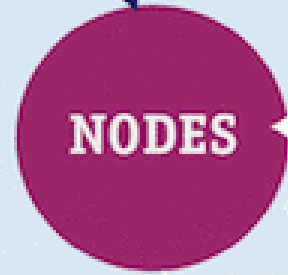


IS A

MANAGES A



RECORDS DATA IN



CONNECT



CONNECT

HAVE

HAVE

NAVIGATES

ORDER

IDENTIFIES



MAPS FROM



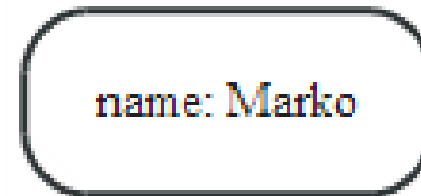
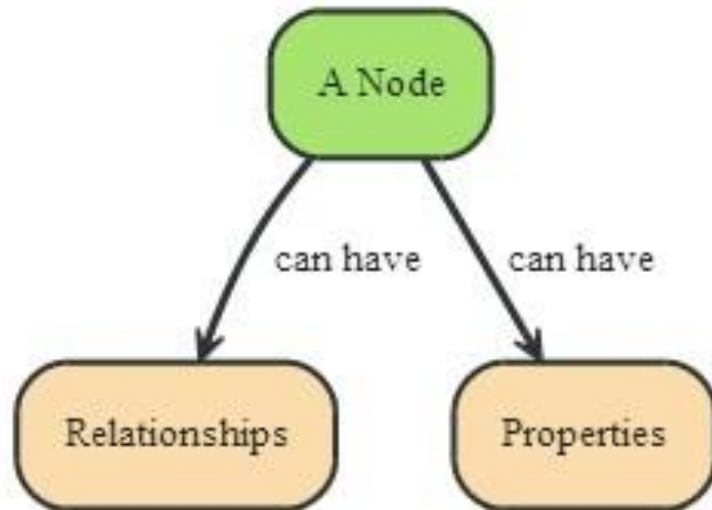
# 1.3 Neo4j Details...

- ▶ Nodes
- ▶ Relationships
- ▶ Properties
- ▶ Paths
- ▶ Traversal
- ▶ Server
- ▶ Language



# Nodes

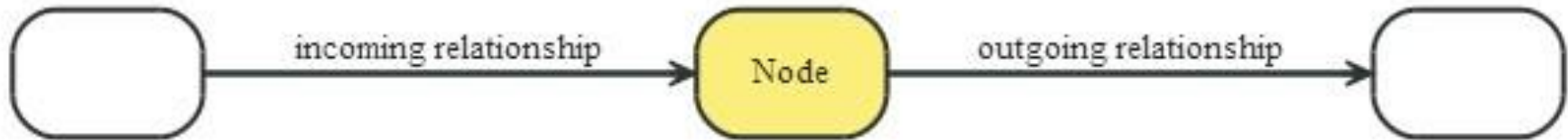
- ▶ The fundamental units that form a graph are nodes and relationships.
- ▶ Nodes are often used to represent entities.



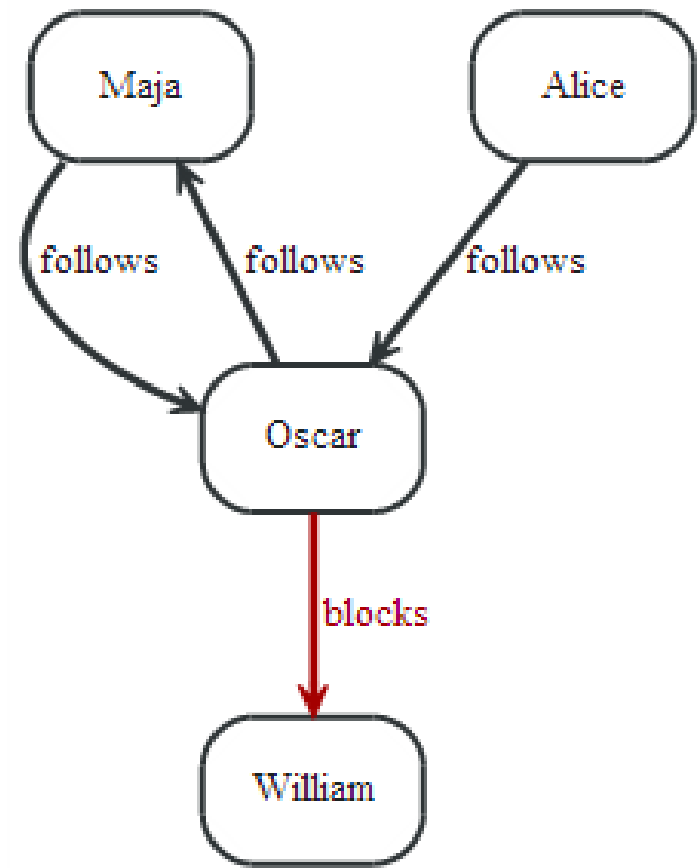
Eg. A single node with one property

# Relationships

- ▶ A relationship connects two nodes, and is always directed, can be viewed as outgoing or incoming relative to a node.

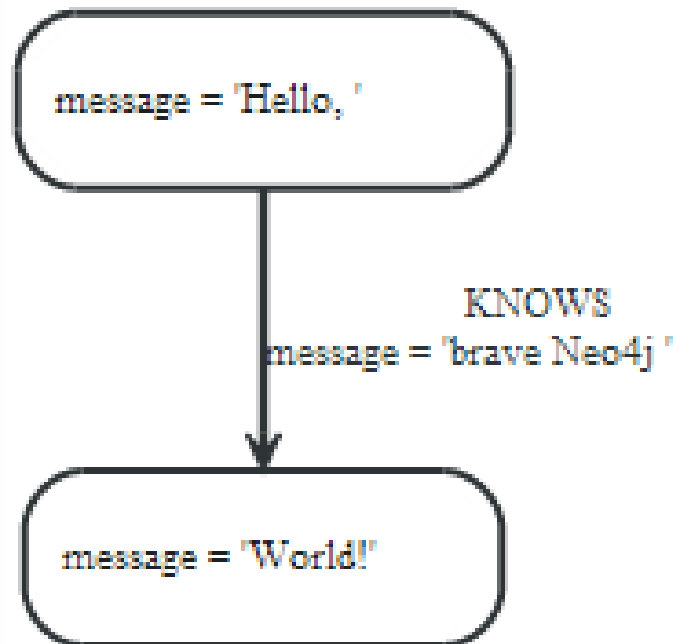


- ▶ Relationships allow for finding related data.



# Properties

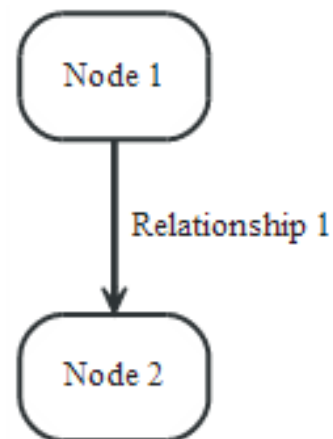
- ▶ Both nodes and relationships can have properties.
- ▶ Properties are key–value pairs where the key is a string. Property values can be either a primitive or an array of one primitive type.



# Paths

- ▶ A path is one or more nodes with connecting relationships, typically retrieved as a query or traversal result.

Eg. A path of length one



# Traversals

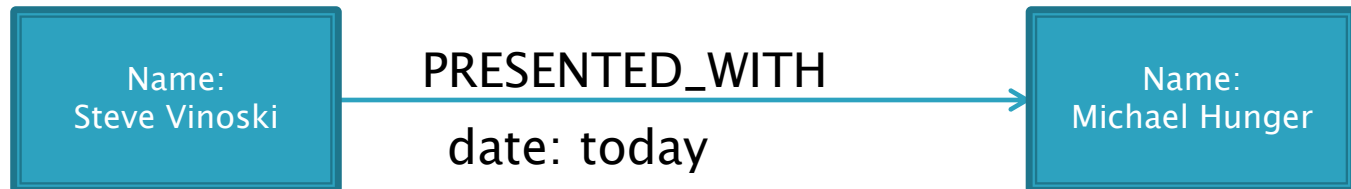
- ▶ Visiting its nodes, following relationships according to some rules.
  - Depth First / Breadth First

# Show me some code, please

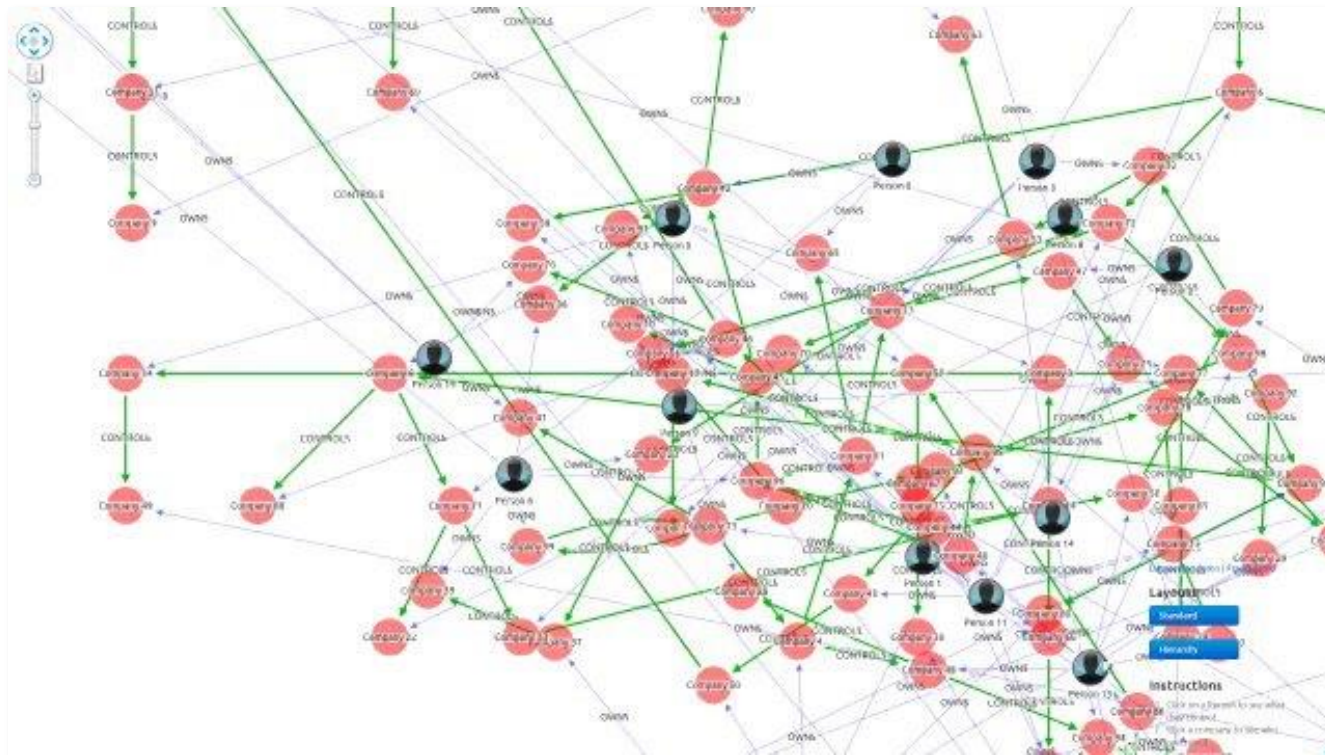
```
GraphDatabaseService graphDb =
    new EmbeddedGraphDatabase("var/neo4j");
Transaction tx = graphDb.beginTx();
try {
    Node steve = graphDb.createNode();
    Node michael = graphDb.createNode();

    steve.setProperty("name", "Steve Vinoski");
    michael.setProperty("name", "Michael Hunger");

    Relationship presentedWith = steve.createRelationshipTo(
        michael, PresentationTypes.PRESENTED_WITH);
    presentedWith.setProperty("date", today);
    tx.success();
} finally {
    tx.finish();
}
```



# 1.4 How you query this “graph” database?



# 1.4 How you query this “graph” database?

- ▶ Gremlin – graph scripting
  - Groovy based Graph Traversal Language
  - Send Gremlin scripts to the Neo4j Server
  - Scripts are executed on the server database
  - Results are returned as Neo4j Node and Relationship representations.



## Try it out:

g- the graph itself

g.v(0) – node 0

g.v(0).in – nodes connected to Node 0

g.v(0).in.name – the names of those Nodes

g.v(l).outE{it.lable == “KNOWS”} – the outgoing “KNOWS”

g.v(l).outE{it.label == “KNOWS”}.inV.name – knows who?

Details:

<https://github.com/tinkerpop/gremlin/>



# 1.4 How you query this “graph” database?

- ▶ Cypher – SQL-like querying
  - WHERE and ORDER BY
- ▶ Pattern-matching
- ▶ Focus on the clarity of expressing *what* to retrieve from a graph, not *how* to do it

## ▶ Clauses

- **START**: Starting points in the graph, obtained via index lookups or by element IDs.
- **MATCH**: The graph pattern to match, bound to the starting points in START.
- **WHERE**: Filtering criteria.
- **RETURN**: What to return.
- **CREATE**: Creates nodes and relationships.
- **DELETE**: Removes nodes, relationships and properties.
- **SET**: Set values to properties.
- **FOREACH**: Performs updating actions once per element in a list.
- **WITH**: Divides a query into multiple, distinct parts.

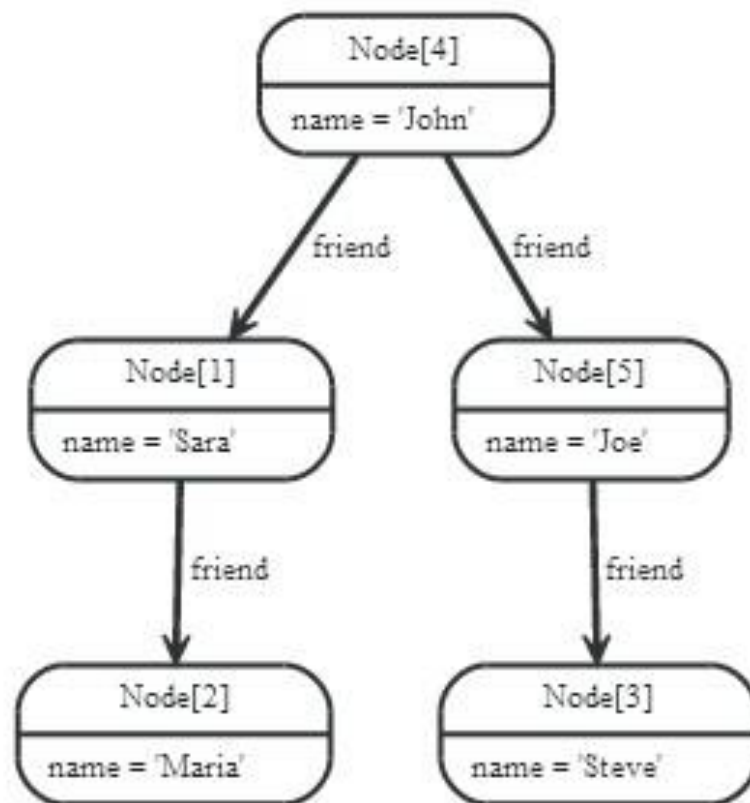
# ▶ Create Graph with Cypher

Demo: <http://console.neo4j.org/>



```
START john=node:node_auto_index(name = 'John')
MATCH john-[:friend]->()-[:friend]->fof
RETURN john, fof
```

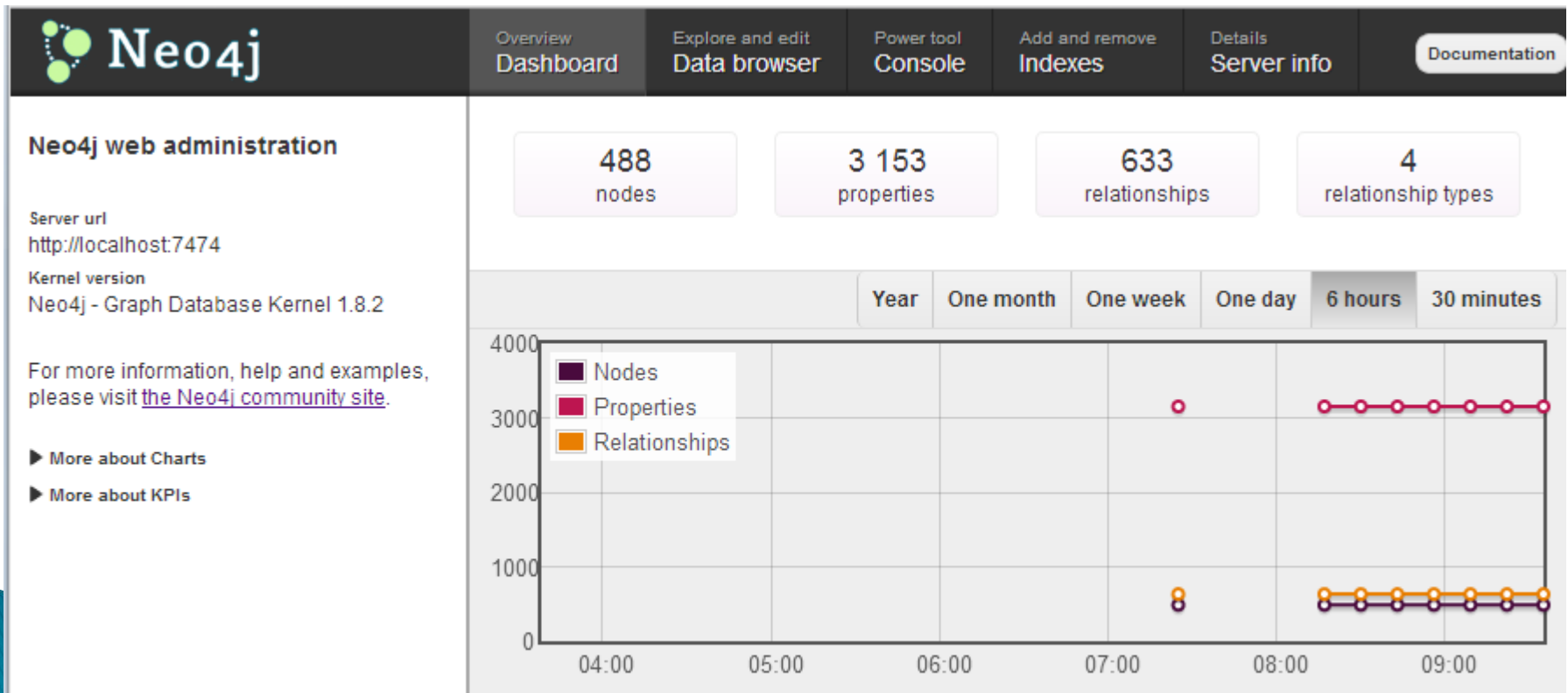
john	fof
Node[4]{name:"John"}	Node[2]{name:"Maria"}
Node[4]{name:"John"}	Node[3]{name:"Steve"}
2 rows	
4 ms	

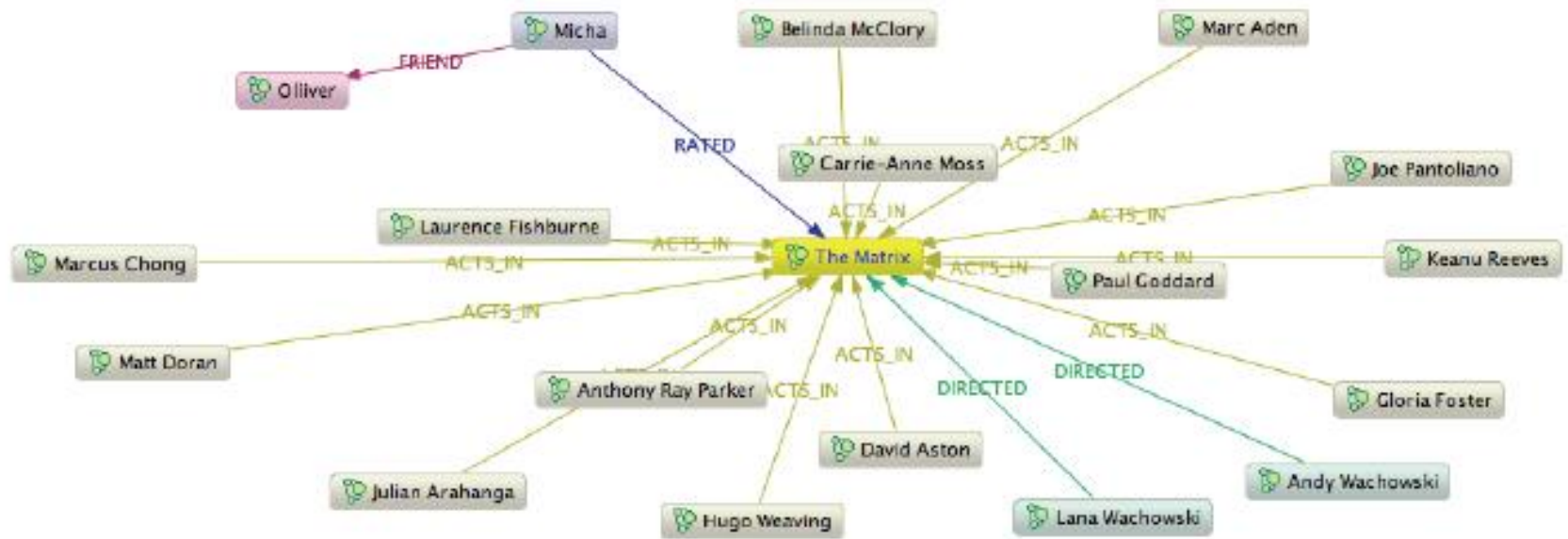


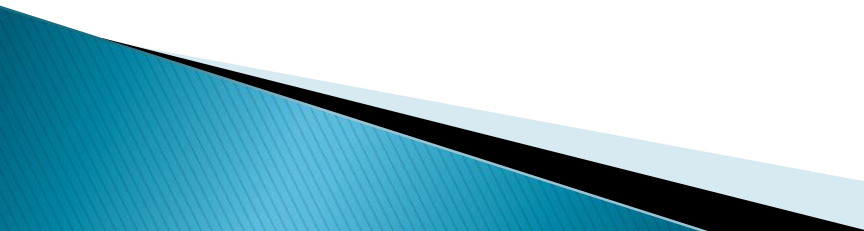
# 1.5 Neo4j Server

## ▶ Server

- <http://localhost:7474/webadmin>





- Embedded in application
  - REST API
    - HTTP protocol and JSON
    - Insert, delete and access data easily from any programming language
  - Ex. Create the nodes
    - Send a HTTP request containing a JSON payload to the server
    - The server will have created a new node in the database
    - Responded with a 201 Created response and a Location header with the URI of the newly created node
- 

# 1.6 Language

Java, JavaScript, Node.js, Python, Rails, PHP, .NET

- Embedded in application
- REST API

name	language / framework	URL
Neo4j.rb	JRuby	<a href="https://github.com/andreasronge/neo4j">https://github.com/andreasronge/neo4j</a>
Neo4django	Python, Django	<a href="https://github.com/scholrly/neo4django">https://github.com/scholrly/neo4django</a>
Neo4js	JavaScript	<a href="https://github.com/neo4j/neo4js">https://github.com/neo4j/neo4js</a>
Gremlin	Java, Groovy	Section 18.18. "Gremlin Plugin", <a href="https://github.com/tinkerpop/gremlin/wiki">https://github.com/tinkerpop/gremlin/wiki</a>
Neo4j-Scala	Scala	<a href="https://github.com/FaKod/neo4j-scala">https://github.com/FaKod/neo4j-scala</a>
Borneo	Clojure	<a href="https://github.com/wagjo/borneo">https://github.com/wagjo/borneo</a>

# More Information...

## ▶ Documentation

- ✓ [docs.neo4j.org-tutorials](http://docs.neo4j.org-tutorials) + reference
- ✓ Neo4j in Action
- ✓ Good Relationships

## ▶ Get Neo4j

- ✓ <http://www.neo4j.org/download>
- ✓ <https://addons.heroku.com/neo4j>





# 2 Proposed Application

- ▶ A Movie Recommendation System

- ▶ Dataset
- ▶ The graph database --- Neo4j
- ▶ The graph traversal language --- Gremlin

MovieLens Data Sets



Shop by Department ▾

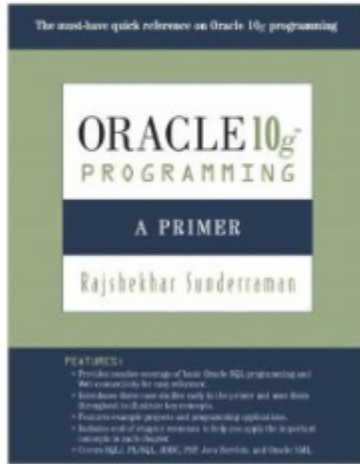
Search Books ▾

Books Advanced Search New Releases Best Sellers The New York Times® Best Sellers Children's Books Textbooks Sell Your Books

Prime

Member: Jiepeng Zhang

Jiepeng Zhang: This item is eligible for Amazon Prime. [Click here to turn on 1-Click](#) and ma



Click to open expanded view

### Oracle 10g Programming: A Primer [Paperback]

Rajshekhar Sunderraman (Author)

★★★★☆ (3 customer reviews)

Buy New

\$47.84 Prime

Only 6 left in stock (more on the way).

Ships from and sold by Amazon.com. Gift-wrap available.

Want it tomorrow, April 20? Order within 7 hrs 21 mins, and choose Saturday Delivery at checkout. [Details](#)

21 new from \$39.63 36 used from \$19.95

EARN \$5 FOR EACH FRIEND YOU REFER TO AMAZON STUDENT [See details](#)

#### Customers Who Bought This Item Also Bought



Fundamentals of Database Systems (6th Edition) \$110.99



Database Systems: The Complete Book (2nd Edition) \$124.72



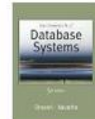
Introduction to Algorithms \$80.00



Computer Networking: A Top-Down Approach ... \$113.47



Programming the World Wide Web (6th Edition) ...



Fundamentals of Database Systems (5th Edition)



New Perspectives on Microsoft Access ... \$104.49



Database Management Systems \$112.99



Database System Concepts \$147.80



WIKIPEDIA  
The Free Encyclopedia

# Recommender system

---

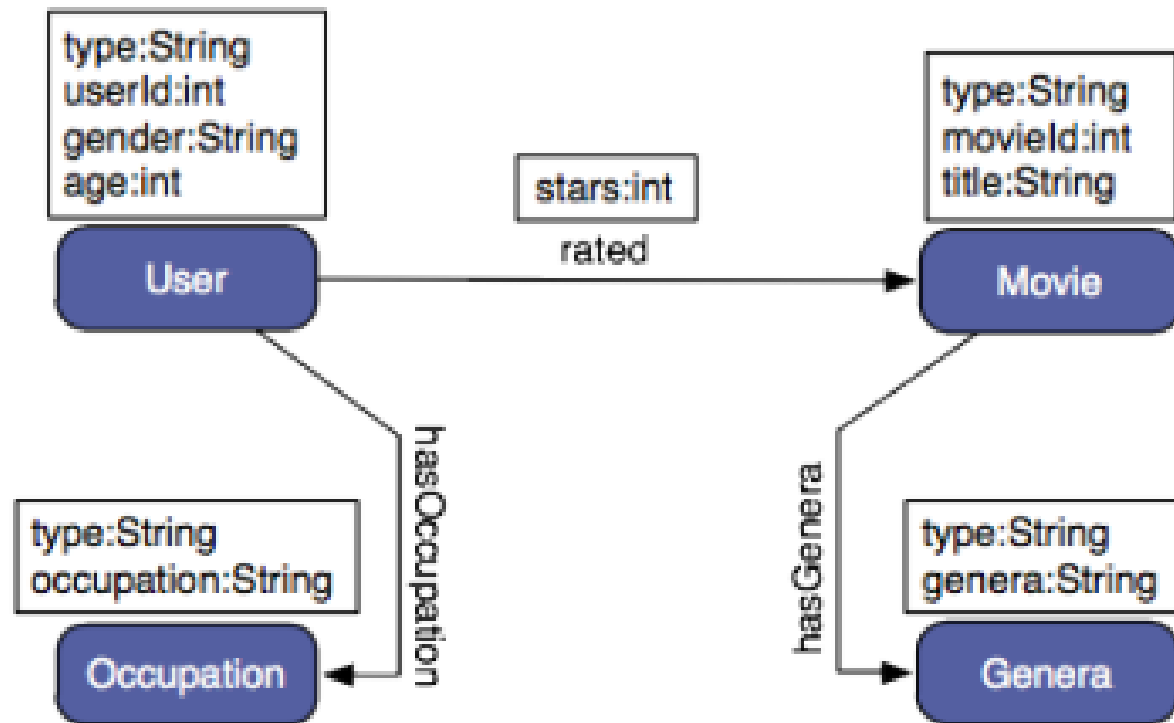
From Wikipedia, the free encyclopedia

**Recommender systems** or **recommendation systems** (sometimes replacing "system" with a synonym such as platform or engine) are a subclass of [information filtering system](#) that seek to predict the 'rating' or 'preference' that user would give to an item (such as [music](#), [books](#), or [movies](#)) or social element (e.g. [people](#) or [groups](#)) they had not yet considered, using a model built from the characteristics of an item (content-based approaches) or the user's social environment (collaborative filtering approaches).<sup>[1][2]</sup>

Recommender systems have become extremely common in recent years. A few examples of such systems:

## 2.1 Generating a MovieRatings Graph

- ▶ Parse the raw MovieLens data according to the graph schema.

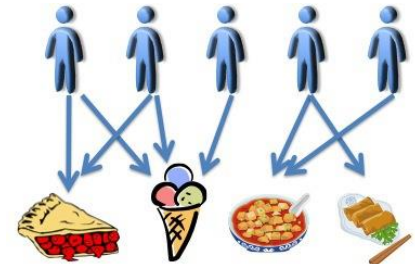


## 2.2 Traversing the MovieLens Graph

- ▶ Recommendation Algorithms

- ▶ Collaborative filtering

- ▶ --- The rating/liking/preference behavior of users is correlated in order to recommend the favorites of one user to another , similar user.



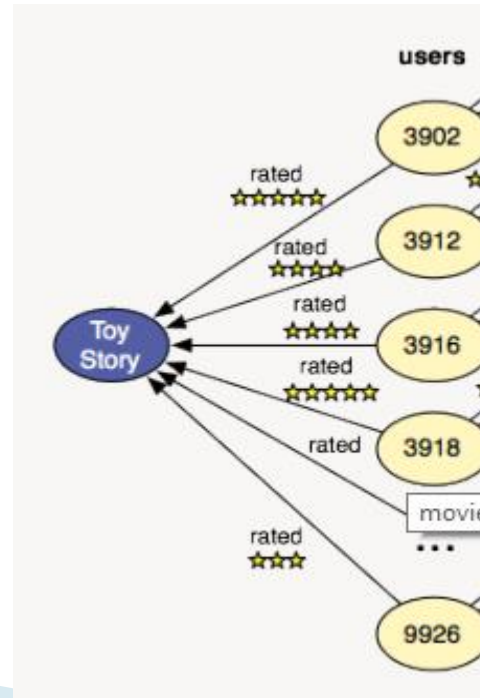
- ▶ Content-based recommendation

- ▶ --- If a particular item is liked, then its features are analyzed in order to find other items with analogous features.

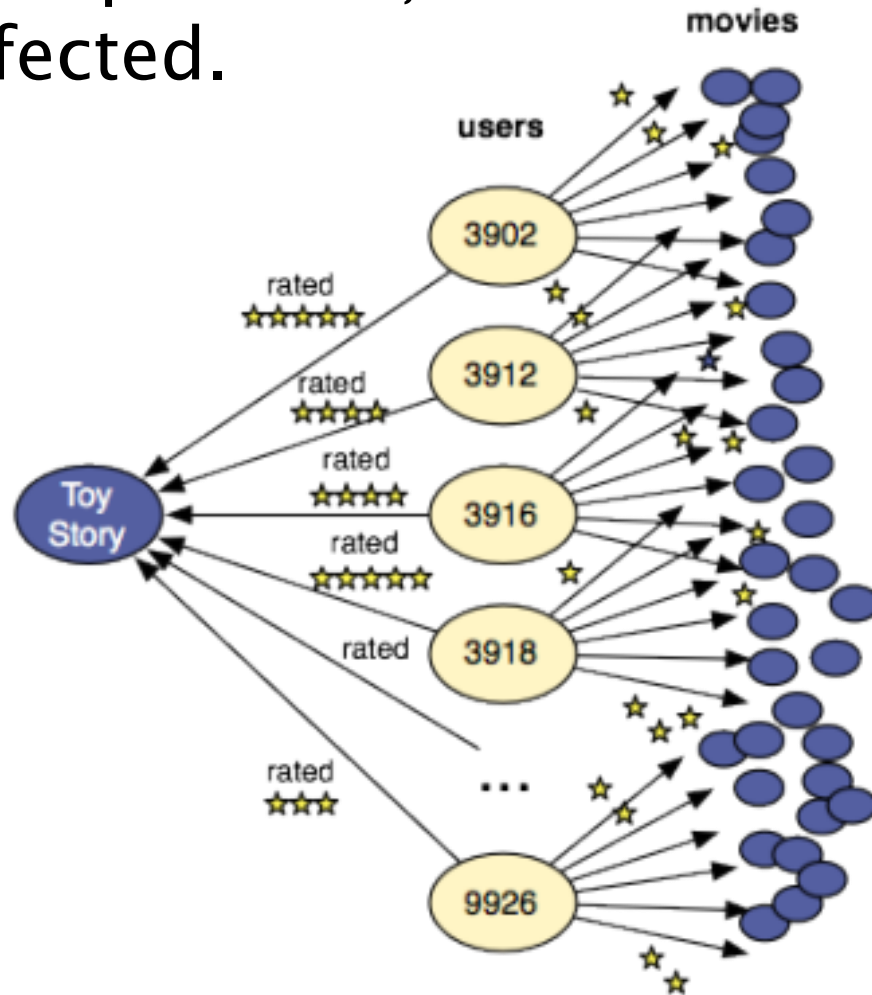
? I like the movies you like, what other movies do you like that I haven't seen?

▶ Let start from Toy Story.

? Which users have Toy Story more than 3 stars?



- ▶ The traversal doesn't yield useful information. However, when it is used within a larger path expression, collaborative filtering is effected.

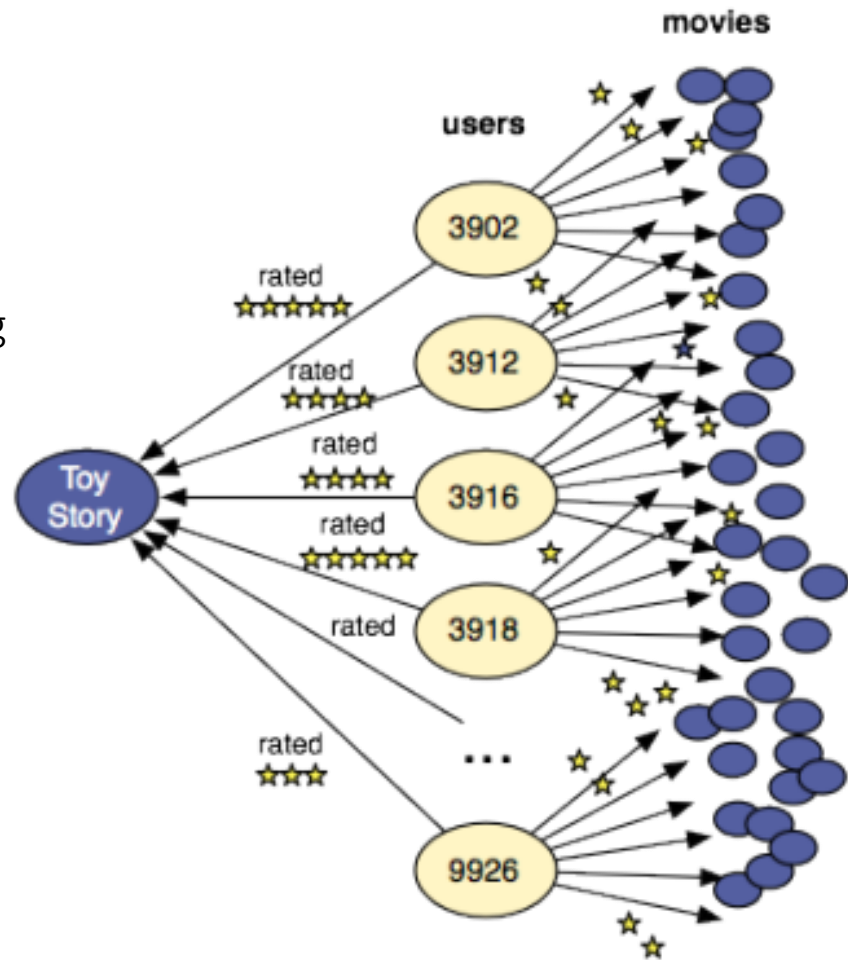






Example: Which users gave Toy Story more than 3 stars and what other movies did they give more than 3 stars to?

- ▶ 1. Start from Toy Story
- ▶ 2. Get the incoming rated edges
- ▶ 3. Filter out those edges whose star property is less than 4.
- ▶ 4. Get the tail user vertices of the remaining edges.
- ▶ 5. Get the rating edges of those user vertices
- ▶ 6. Filter out those edges whose star property is less than 4.
- ▶ 7. Get the head movie vertices of the remaining edges
- ▶ 8. Get the string title property of those movie vertices



```
v.inE('rated').filter{it.getProperty('stars') > 3}.outV.outE('rated').filter{it.getProperty('stars') > 3}.inV.title
```

highly co-rated

- ▶ These atomic-steps together can be bundled into a user defined step. --- “corated”

```
gremlin> v.corated(3).title[0..4]
```

- ▶ Will return only 5 results.
- ▶ Given that there are **268,493** highly rated paths from Toy Story to other movies and only **3,353** of those movies are unique, it is possible to use these duplicates as a ranking mechanism—ultimately, a recommendation.

# Which movies are most highly co-rated with Toy Story?

```
gremlin>m=[:]
```

```
gremlin> v.corated(3).title.groupCount(m) >> -1
```

```
gremlin> m.sort{a,b -> b.value <=> a.value}[0..9]
```



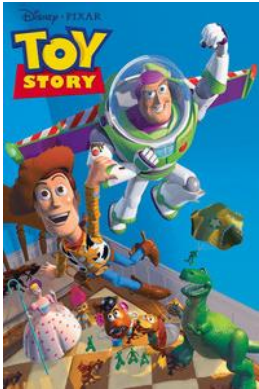
- ▶ Mixing in Content-based Recommendation
- ▶ A traversal that yields Toy Story good movies of the same genre.



Which movies are most highly co-rated with Toy Story that share a genre/all genres with Toy Story?

```
gremlin> m = [:]  
gremlin> x = [] as Set  
gremlin> v.out('hasGenera').aggregate(x).back(2).corated(3).filter{it !=  
v}.filter{it.out('hasGenera')>>[] as Set == x}.title.groupCount(m) >> -1  
==>null  
gremlin> m.sort{a,b -> b.value <=> a.value}[0..9]
```

# Yeah! Got Movies Recommendations!

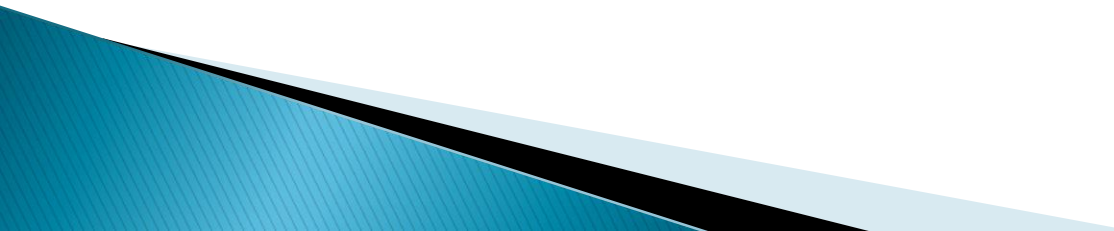


# 3 Application Example

<http://neoflix.herokuapp.com/>

**Thank You!**

**Questions?**



# Conclusion

- ▶ Neo4j is a robust transactional property graph database. Due to its graph data model, Neo4j is highly agile and blazing fast. For connected data operations, Neo4j runs a thousand times faster than relational databases.
  - ▶ More than 20 of the Global 2000, hundreds of startups and thousands of community members use Neo4j in a wide variety of use cases such as social applications, recommendation engines, fraud detection, resource authorization, network & data center management and much more.
- 