Using AsciiArt to Analyse your SourceCode with Neo4j and OSS Tools

Who the hell is this guy?

- Michael Hunger
- Developer Advocate Neo Technology
- Love People and Graphs
- @mesirii | michael@neotechnology.com

What will he talk about?

What will he talk about?

- What is this Neo4j Graphdatabase thing?

What will he talk about?

- What is this Neo4j Graphdatabase thing?
- Ascii-Art Rocks

What will he talk about?

- What is this Neo4j Graphdatabase thing?
- Ascii-Art Rocks
- Graphs in Your Code - The Idea

What will he talk about?

- What is this Neo4j Graphdatabase thing?
- Ascii-Art Rocks
- Graphs in Your Code - The Idea
- Having Fun with your Code and jQAssistant

What will he talk about?

- What is this Neo4j Graphdatabase thing?
- Ascii-Art Rocks
- Graphs in Your Code - The Idea
- Having Fun with your Code and jQAssistant
- Gimme More

What is a Graph Database ?

What is a Graph Database ?

- labeled Nodes

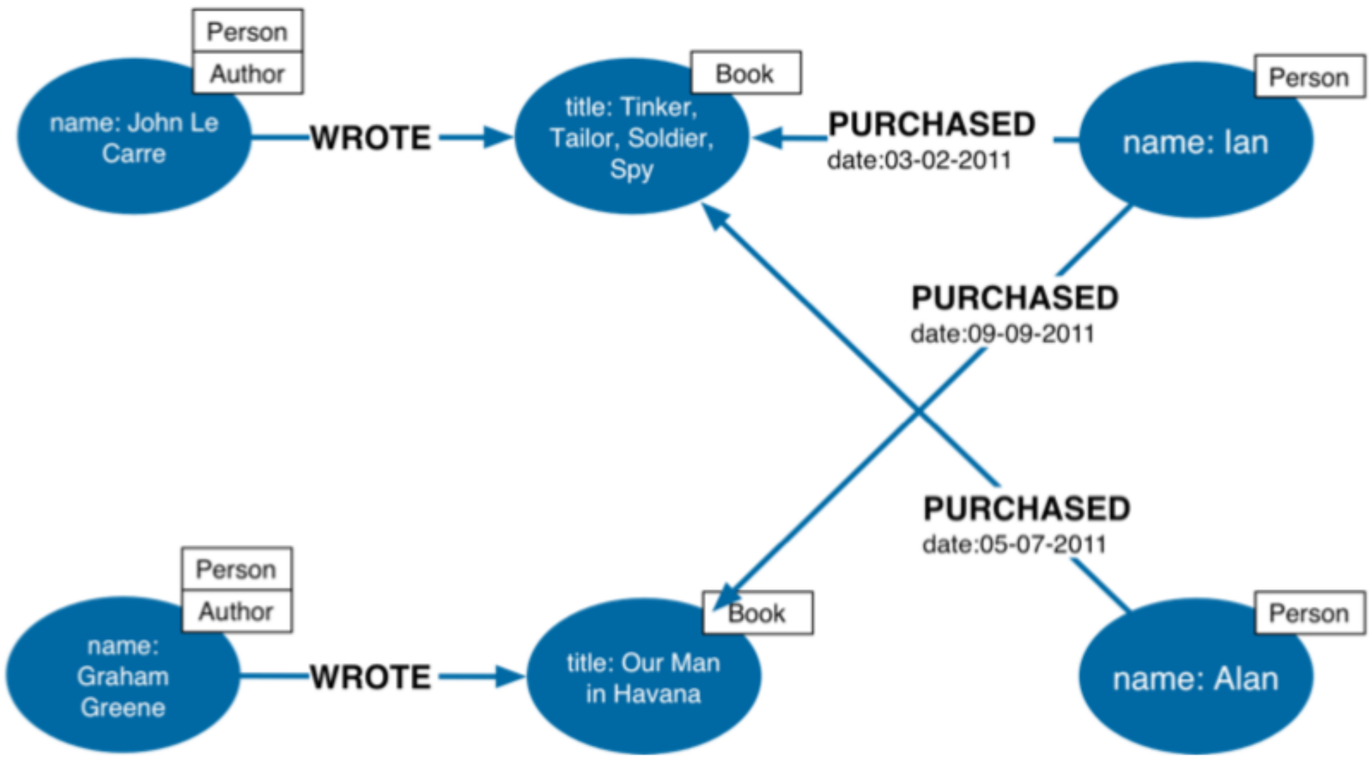
What is a Graph Database ?

- labeled Nodes
- directed, typed Relationships

What is a Graph Database ?

- labeled Nodes
- directed, typed Relationships
- arbitrary Properties on each

Property Graph Model



What makes it special ?

What makes it special ?

- close to the object model

What makes it special ?

- close to the object model
- prematerialize relationships

What makes it special ?

- close to the object model
- prematerialize relationships
- traversals in linear time

What makes it special ?

- close to the object model
- prematerialize relationships
- traversals in linear time
- sparse, heterogenous data + schema free

What makes it special ?

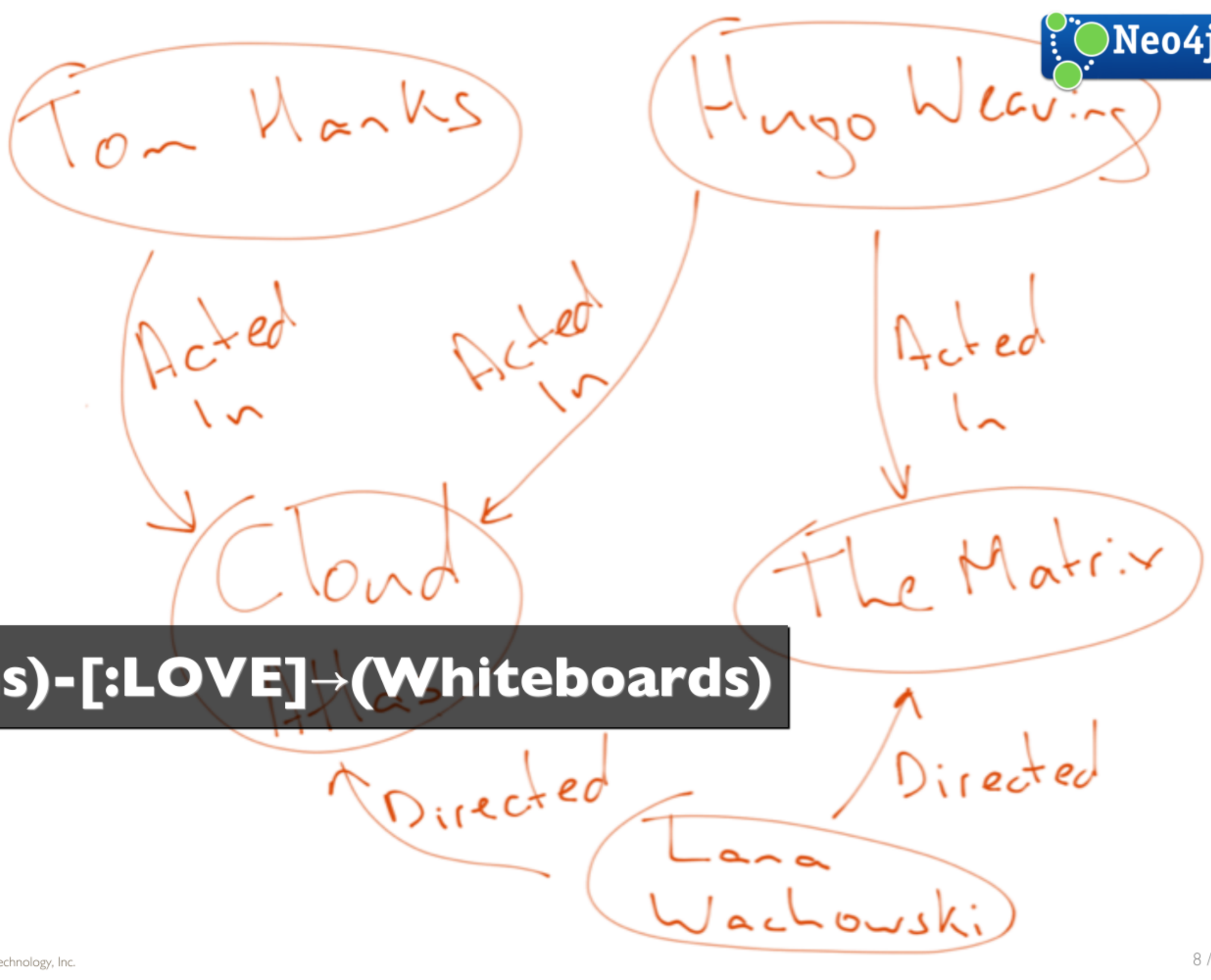
- close to the object model
- prematerialize relationships
- traversals in linear time
- sparse, heterogenous data + schema free
- local queries - explore the neighbourhood

What makes it special ?

- close to the object model
- prematerialize relationships
- traversals in linear time
- sparse, heterogenous data + schema free
- local queries - explore the neighbourhood
- whiteboard-friendly

Where can/should I use it ?

- Impact Analysis (Network, Software)
- Routing / Logistics
- Recommendation, Dating, Job-Search
- Sciene (Metadata, Drug Research)
- Masterdata, Hierarchy-Mgmt
- Fraud-Detection, Market-Analysis
- Social, …. and many more



(Graphs)-[:LOVE]→(Whiteboards)

Tom Hanks

Hugo Weaving

ACTED_IN

ACTED_IN

ACTED_IN

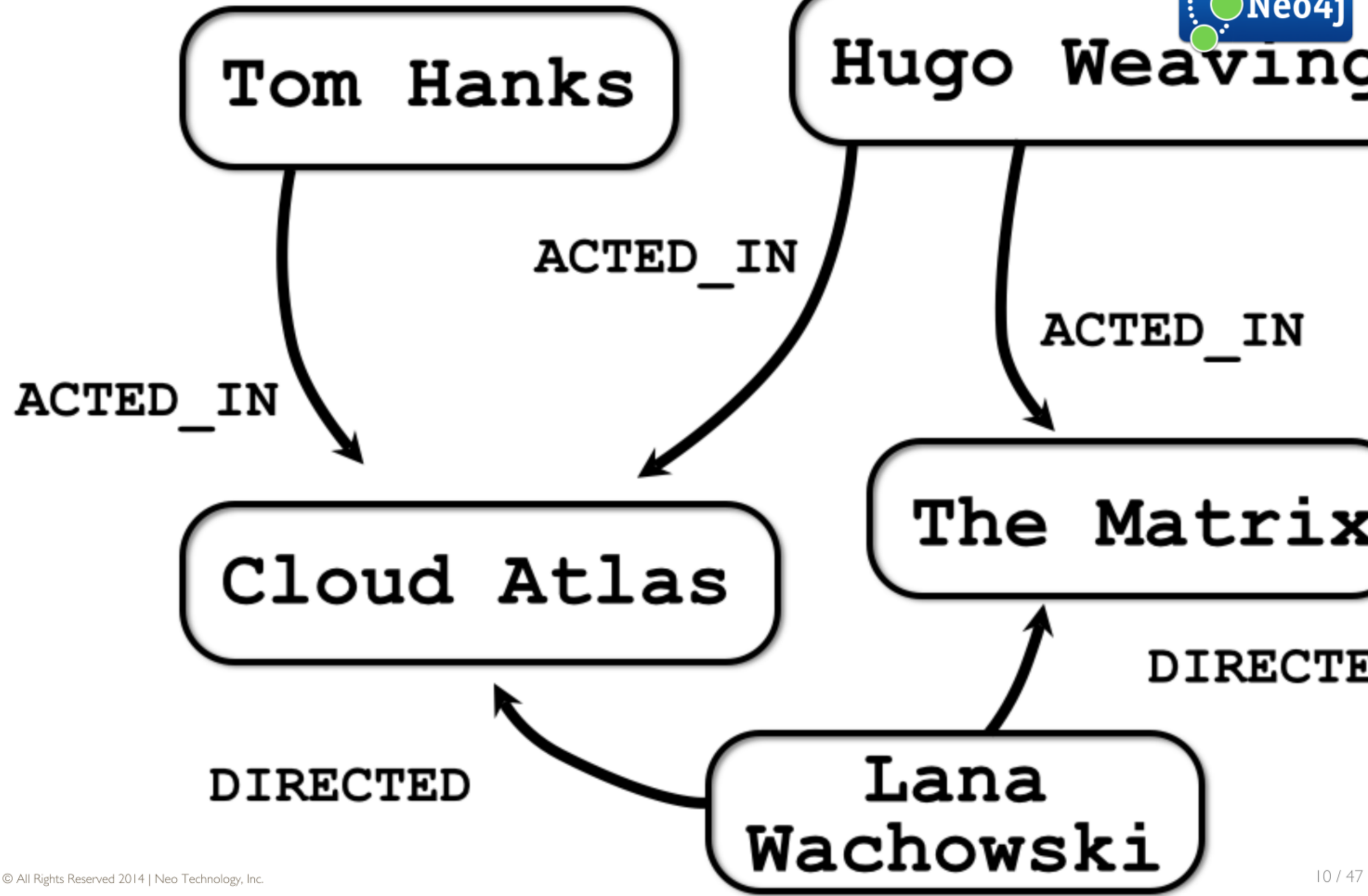
Cloud Atlas

The Matrix

DIRECTED

DIRECTE

Lana Wachowski



Person
name: Tom Hanks
nationality: USA
won: Oscar, Emmy

Person
name: Hugo Weaving
nationality: Austra
won: MTV Movie Awar

Neo4j

ACTED_IN
role: Bill Smoke

ACTED_IN
role: Zachry

ACTED_I
role: Agent S

Movie
title: Cloud Atlas
genre: drama, sci-fi

Movie
title: The Matri
genre: sci-fi

DIRECTED

DIRECTED

Person
name: Lana Wachowski
nationality: USA
won: Razzie, Hugo

Ascii-Art Rocks

Ascii-Art Rocks

- Turn text into pictures

Ascii-Art Rocks

- Turn text into pictures
- Turn picture into text

Ascii-Art Rocks

- Turn text into pictures
- Turn picture into text
- The Power of Symbols

Ascii-Art Rocks

- Turn text into pictures
- Turn picture into text
- The Power of Symbols
- Graph Patterns Made easy

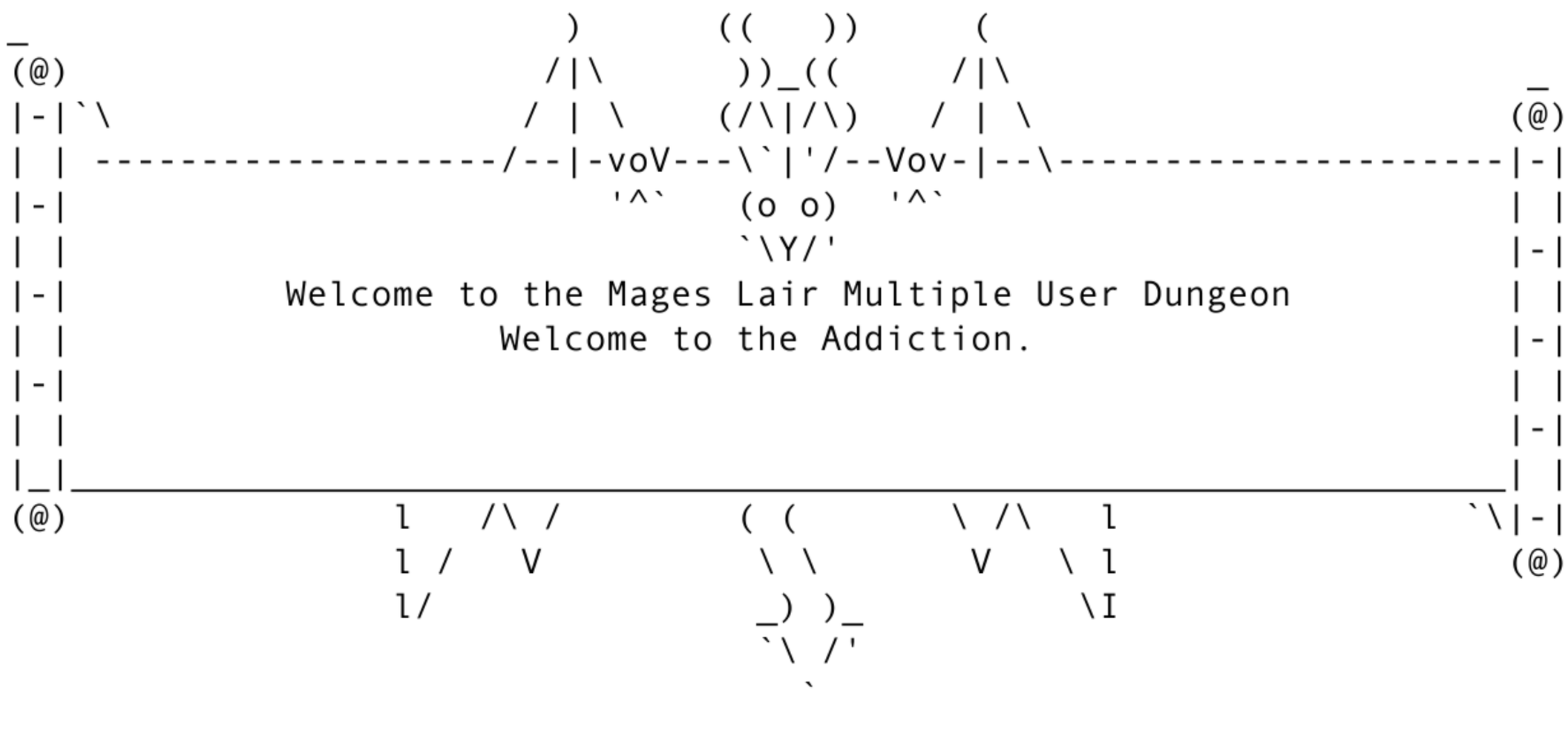
Ascii-Art Rocks

- Turn text into pictures
- Turn picture into text
- The Power of Symbols
- Graph Patterns Made easy
- Hacker and Mudder Friendly

Ascii-Art Rocks

- Turn text into pictures
- Turn picture into text
- The Power of Symbols
- Graph Patterns Made easy
- Hacker and Mudder Friendly
- Diffs, VCS

Ascii-Art Rocks



Cypher

(Cypher)-[:USES]→(Ascii-Art)

(Cypher)-[:USES]→(Ascii-Art)

- Declarative Graph Query Language

(Cypher)-[:USES]→(Ascii-Art)

- Declarative Graph Query Language
- Graph Pattern Matching

(Cypher)-[:USES]→(Ascii-Art)

- Declarative Graph Query Language
- Graph Pattern Matching
- Humane, Readable

(Cypher)-[:USES]→(Ascii-Art)

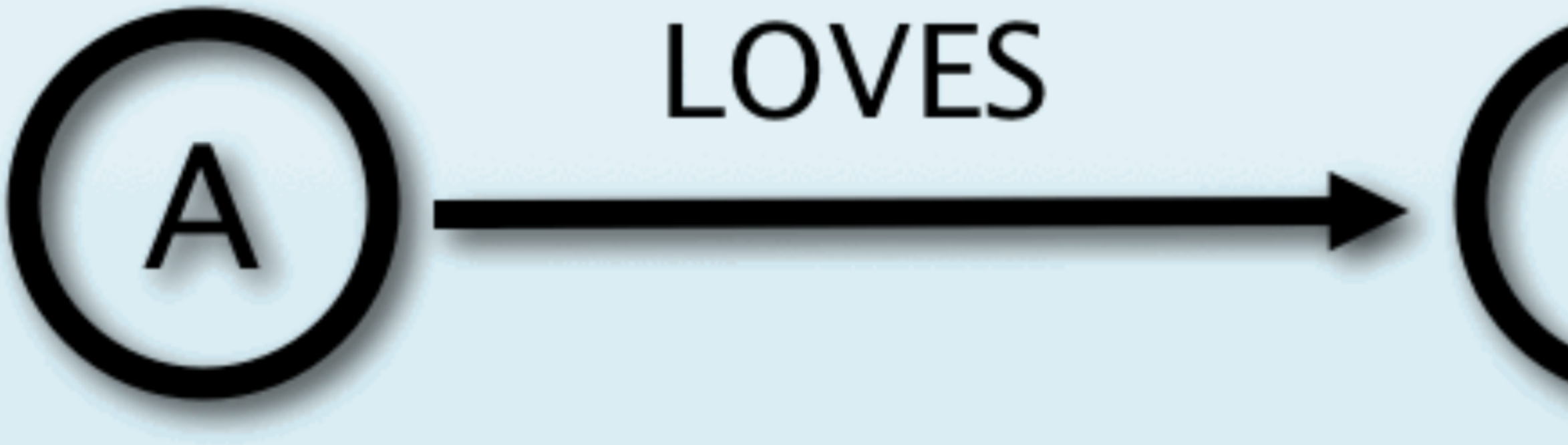
- Declarative Graph Query Language
- Graph Pattern Matching
- Humane, Readable
- Expressive

(Cypher)-[:USES]→(Ascii-Art)

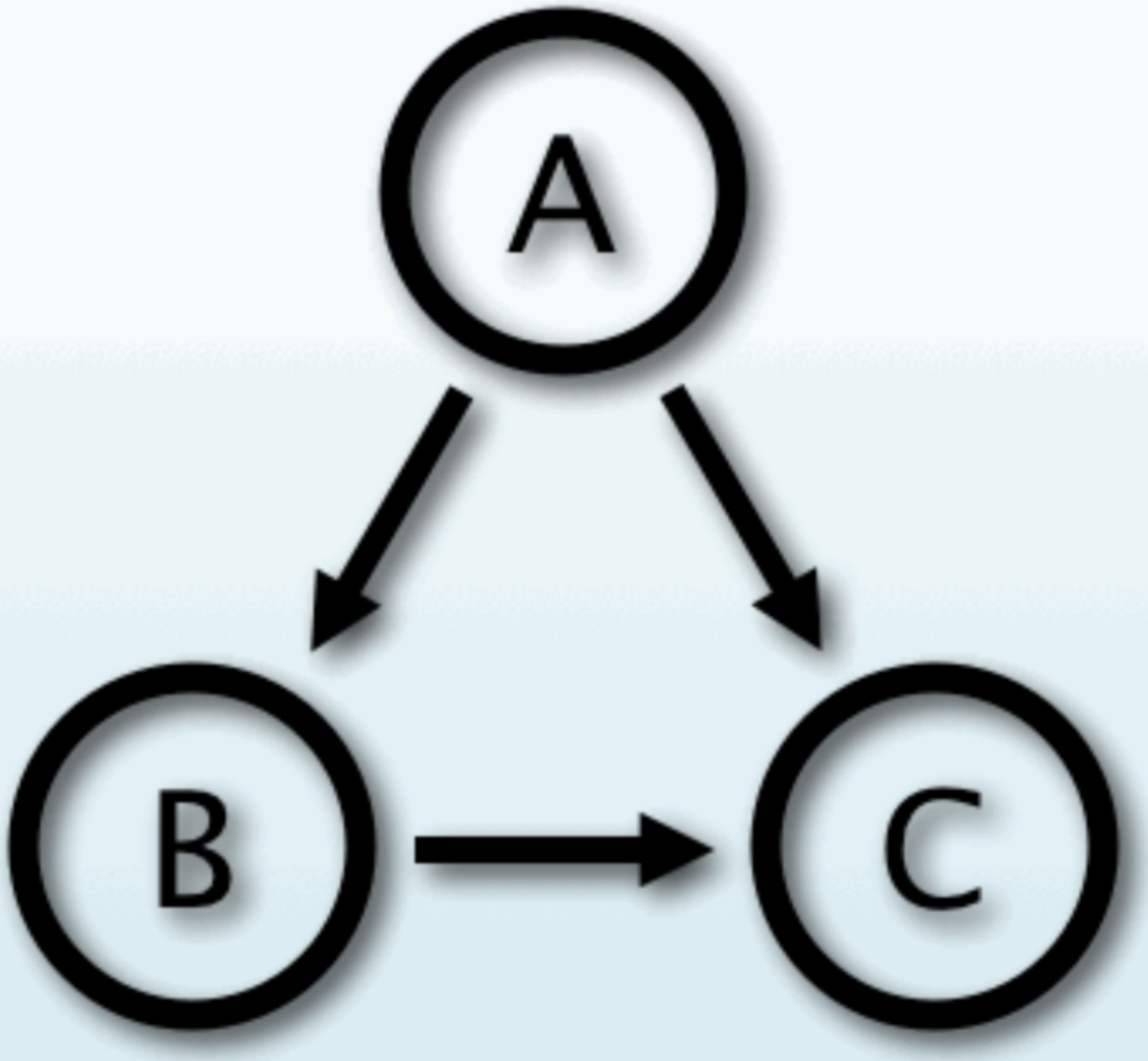
- Declarative Graph Query Language
- Graph Pattern Matching
- Humane, Readable
- Expressive
- Read and Write Graphs

(Cypher)-[:USES]→(Ascii-Art)

- Declarative Graph Query Language
- Graph Pattern Matching
- Humane, Readable
- Expressive
- Read and Write Graphs
- Tabular Results



A -[:LOVES]->



A --> B --> C, A -->

A --> B --> C <--

Cypher Query - Example Geekout

Cypher Query - Example Geekout

Setup

```
CREATE (:Year {year:2014})<-[:IN_YEAR]-(geekout:Conference {name:"Geekout"})
        -[:LOCATION]->(:City {name:"Tallinn"})
CREATE (track:Track {name:"Room 1"})-[:TRACK_OF]->(geekout)

MERGE (speaker:Attendee:Speaker {name:"Hadi Hariri"}) MERGE (geekout)<-[:ATTENDS]-(speaker)

CREATE (speaker)-[:PRESENTS]->(session:Session {title:"Mouseless IDE"})<-[:IN_TRACK]-(track)

FOREACH (name in ["Java","IDE","Development"] |
   MERGE (topic:Topic {name:name})
   CREATE (session)-[:HAS_TOPIC]->(topic)))
```

Cypher Query - Example Geekout

Setup

```
CREATE (:Year {year:2014})<-[:IN_YEAR]-(geekout:Conference {name:"Geekout"})
        -[:LOCATION]->(:City {name:"Tallinn"})
CREATE (track:Track {name:"Room 1"})-[:TRACK_OF]->(geekout)

MERGE (speaker:Attendee:Speaker {name:"Hadi Hariri"}) MERGE (geekout)<-[:ATTENDS]-(speaker)

CREATE (speaker)-[:PRESENTS]->(session:Session {title:"Mouseless IDE"})<-[:IN_TRACK]-(track)

FOREACH (name in ["Java","IDE","Development"] |
   MERGE (topic:Topic {name:name})
   CREATE (session)-[:HAS_TOPIC]->(topic)))
```

Query

```
// Which Speakers entertain you with Java ?
MATCH (t:Topic {name:"Java"})<-[:HAS_TOPIC]-(session)<-[:PRESENTS]-(speaker),
      (session)<-[:IN_TRACK]-(track)

RETURN speaker.name as speaker, {time:session.time, session:session.title} as session,
       track.name as track
ORDER BY session.time
```

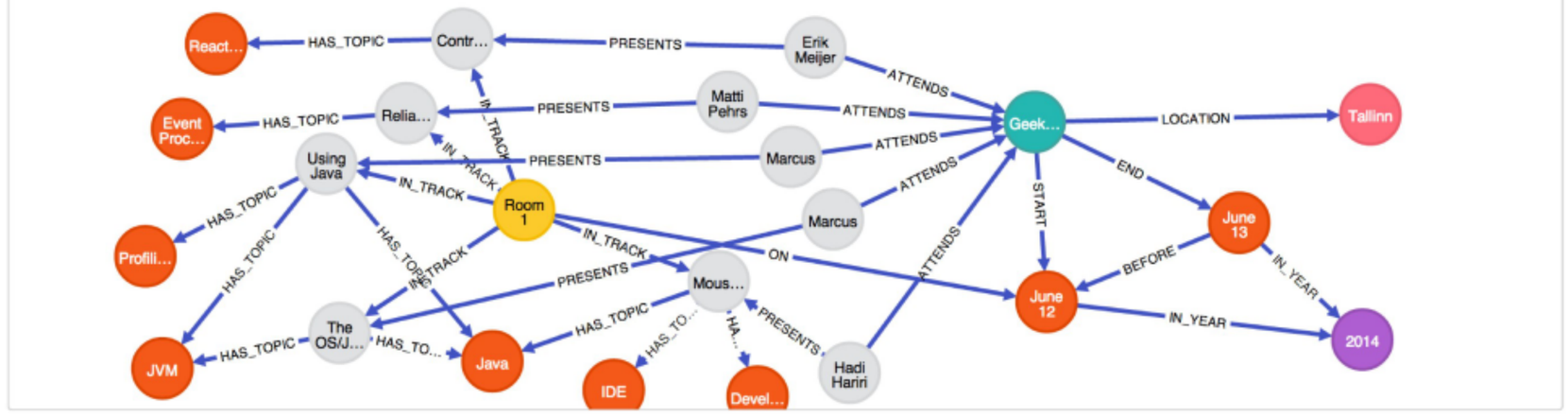
Geekout 2014 GraphGist

Let me graph that for you: GeekOut 2014

What better way to show what a graph database is good at, is to show how it helps you to connect people, sessions, topics, time and more.

Query 1 — this query has been used to initialize the console

Test run OK



Which Speakers entertain you with *Java* ?

Query 2

```
MATCH (t:Topic {name:"Java"})<-[:HAS_TOPIC]-(session)<-[:PRESENTS]-(speaker), (session)<-[:IN_TRACK]-(track)
RETURN speaker.name as speaker, {time:session.time, session:session.title} as session
ORDER BY session.time
```

Test run OK

speaker	session
Hadi Hariri	{time:"11.15", session:"Mouseless Driven Development"}
Marcus Hirt	{time:"14:30", session:"Using Java Flight Recorder"}
Marcus Lagergren	{time:"15:50", session:"The OS/JVM dialectic – ending the struggle through unity"}

Interactive GraphGist Document WorldCup

Software Analytics - Approach

Software Analytics - Approach

1. look at one interesting aspect

Software Analytics - Approach

1. look at one interesting aspect
2. which insights would be cool?

Software Analytics - Approach

1. look at one interesting aspect
2. which insights would be cool?
3. model it as a graph

Software Analytics - Approach

1. look at one interesting aspect
2. which insights would be cool?
3. model it as a graph
4. get data

Software Analytics - Approach

1. look at one interesting aspect
2. which insights would be cool?
3. model it as a graph
4. get data
5. import into graph model

Software Analytics - Approach

1. look at one interesting aspect
2. which insights would be cool?
3. model it as a graph
4. get data
5. import into graph model
6. enrich graph model with concepts / structure

Software Analytics - Approach

1. look at one interesting aspect
2. which insights would be cool?
3. model it as a graph
4. get data
5. import into graph model
6. enrich graph model with concepts / structure
7. query for insights

(Code)-[:IS_A]→(Graph)

(Code)-[:IS_A]→(Graph)

- AST, ByteCode, Source-Code

(Code)-[:IS_A]→(Graph)

- AST, ByteCode, Source-Code
- Inheritance, Composition, Dependencies

(Code)-[:IS_A]→(Graph)

- AST, ByteCode, Source-Code
- Inheritance, Composition, Dependencies
- Transitive Module and Library dependencies

(Code)-[:IS_A]→(Graph)

- AST, ByteCode, Source-Code
- Inheritance, Composition, Dependencies
- Transitive Module and Library dependencies
- Dependency injection config

(Code)-[:IS_A]→(Graph)

- AST, ByteCode, Source-Code
- Inheritance, Composition, Dependencies
- Transitive Module and Library dependencies
- Dependency injection config
- Data model (db) <→ object model

(Code)-[:IS_A]→(Graph)

- AST, ByteCode, Source-Code
- Inheritance, Composition, Dependencies
- Transitive Module and Library dependencies
- Dependency injection config
- Data model (db) <→ object model
- Runtime characteristics: call graph, heap

(Code)-[:IS_A]→(Graph)

- AST, ByteCode, Source-Code
- Inheritance, Composition, Dependencies
- Transitive Module and Library dependencies
- Dependency injection config
- Data model (db) <→ object model
- Runtime characteristics: call graph, heap
- Version control, repositories, issues

jQAssistant

jQAssistant

- Open Source Software Analytics Tool

jQAssistant

- Open Source Software Analytics Tool
- Plugins for Java, JEE, JPA, Maven, Gradle, SonarJ …

jQAssistant

- Open Source Software Analytics Tool
- Plugins for Java, JEE, JPA, Maven, Gradle, SonarJ …
- All Cypher based

jQAssistant

- Open Source Software Analytics Tool
- Plugins for Java, JEE, JPA, Maven, Gradle, SonarJ …
- All Cypher based
- Technical and Domain Concept Definitions

jQAssistant

- Open Source Software Analytics Tool
- Plugins for Java, JEE, JPA, Maven, Gradle, SonarJ …
- All Cypher based
- Technical and Domain Concept Definitions
- Compute Software Metrics

jQAssistant

- Open Source Software Analytics Tool
- Plugins for Java, JEE, JPA, Maven, Gradle, SonarJ …
- All Cypher based
- Technical and Domain Concept Definitions
- Compute Software Metrics
- Declare and Validate Architectural Rules

jQAssistant

- Open Source Software Analytics Tool
- Plugins for Java, JEE, JPA, Maven, Gradle, SonarJ …
- All Cypher based
- Technical and Domain Concept Definitions
- Compute Software Metrics
- Declare and Validate Architectural Rules
- Integrated in Build Process

jQAssistant

Actively Looking for Contributions

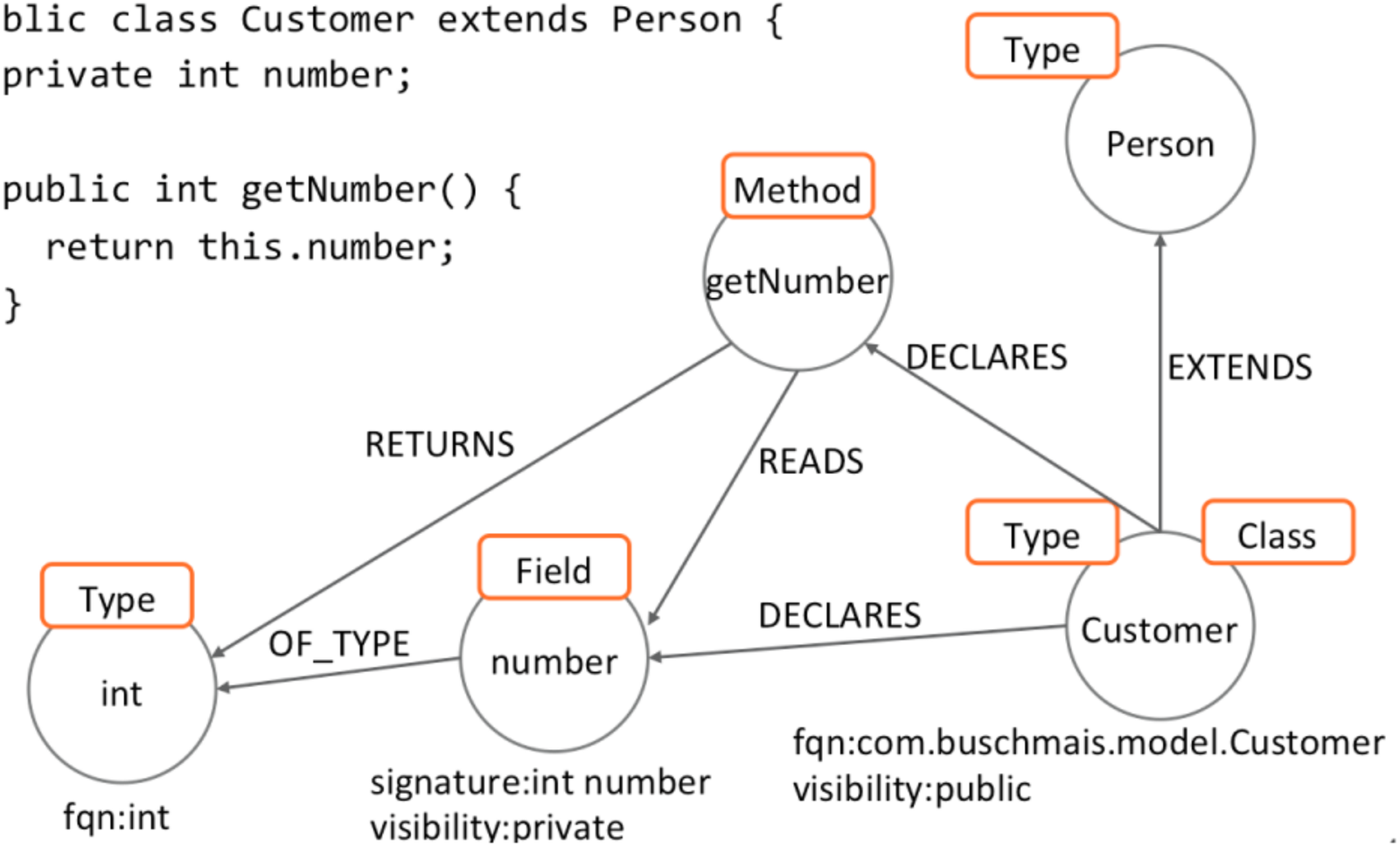
http://github.com/buschmais/jqassistant

Modeling Source Code as a Graph

Modeling Software Structures As A Graph

buschmais
Beratung . Technologie . Innovation

```
public class Customer extends Person {
  private int number;

  public int getNumber() {
    return this.number;
  }
}
```



Java software graph model : Nodes

- Artifact
- Package
- Type, Class, Interface, Annotation, Enum
- Method, Constructor, Parameter
- Field
- Value, Class, Annotation, Enum, Primitive, Array

Java software graph model: Relationships

- CONTAINS, DECLARES
- EXTENDS, IMPLEMENTS
- RETURNS, THROWS, INVOKES, HAS, IS
- ANNOTATED_BY, OF_TYPE

Approach

Approach

1. Scan your project with
   Plugins for Code (Java-ASM), Config, Metadata

Approach

1. Scan your project with
   Plugins for Code (Java-ASM), Config, Metadata
2. Import into Neo4j

Approach

1. Scan your project with
   Plugins for Code (Java-ASM), Config, Metadata
2. Import into Neo4j
3. Enrich with declared technical and domain concepts

Approach

1. Scan your project with
   Plugins for Code (Java-ASM), Config, Metadata
2. Import into Neo4j
3. Enrich with declared technical and domain concepts
4. On top of those concepts

Approach

1. Scan your project with
   Plugins for Code (Java-ASM), Config, Metadata
2. Import into Neo4j
3. Enrich with declared technical and domain concepts
4. On top of those concepts
5. Software-Metrics queries

Approach

1. Scan your project with
   Plugins for Code (Java-ASM), Config, Metadata
2. Import into Neo4j
3. Enrich with declared technical and domain concepts
4. On top of those concepts
5. Software-Metrics queries
6. Architectural-Rules queries

Query the Data

Pattern matching is the core principle of Cypher!

```
MATCH
  (c1:Class)-[:EXTENDS]->(c2:Type)
RETURN
  c1.fqn, c2.fqn
```

Demo

Analyzing A Maven Repository

Rickard Öberg

This little nifty tool will allow you to import your local Maven repository information into a Neo4j graph, in particular dependencies between artifacts.

You can then take this graph and put it into a Neo4j server, and perform Cypher queries on it.

Or whatever else awesome you want to do.

```
mvn compile exec:java -Dexec.mainClass=com.github.rickardoberg.neomvn.Main \
  -Dexec.arguments="$HOME/.m2/repository"
```

https://github.com/rickardoberg/neomvn#example-queries

NeoMVN: Example Queries

NeoMVN: Example Queries

Find all transitive dependencies of all artifacts with "org.neo4j" groups

```
MATCH (group:Group {groupId:'org.neo4j'}),
      (group)-[:HAS_ARTIFACT]->(artifact)-[:HAS_VERSION]->(version)<-[:HAS_DEPENDENCY]-(dependent)

WHERE left(dependent.groupId,9)<>group.groupId
RETURN DISTINCT dependent.artifactId, dependent.groupId
```

NeoMVN: Example Queries

Find all transitive dependencies of all artifacts with "org.neo4j" groups

```
MATCH (group:Group {groupId:'org.neo4j'}),
      (group)-[:HAS_ARTIFACT]->(artifact)-[:HAS_VERSION]->(version)<-[:HAS_DEPENDENCY]-(dependent)

WHERE left(dependent.groupId,9)<>group.groupId
RETURN DISTINCT dependent.artifactId, dependent.groupId
```

Which version of JUnit is the most popular

```
MATCH (group:Group {groupId:'junit'})
MATCH (group)-[:HAS_ARTIFACT]->(artifact)-[:HAS_VERSION]->(version)<-[:HAS_DEPENDENCY]-(dependent)
RETURN version.version, count(dependent) as depCount
ORDER BY depCount DESC
```

Query a JVM Heapdump

Query a JVM Heapdump

1. Get a heap-dump with jmap

   jmap -dump:format=b,file=dump.hprof <pid>

Query a JVM Heapdump

1. Get a heap-dump with jmap
   jmap -dump:format=b,file=dump.hprof <pid>
2. Use jhat dump.hprof find the /oql endpoint

Query a JVM Heapdump

1. Get a heap-dump with jmap
   jmap -dump:format=b,file=dump.hprof <pid>
2. Use jhat dump.hprof find the /oql endpoint
3. run the *OQL* script to generate *Cypher* code

Query a JVM Heapdump

1. Get a heap-dump with jmap
   jmap -dump:format=b,file=dump.hprof <pid>
2. Use jhat dump.hprof find the /oql endpoint
3. run the *OQL* script to generate *Cypher* code
4. Import into Neo4j

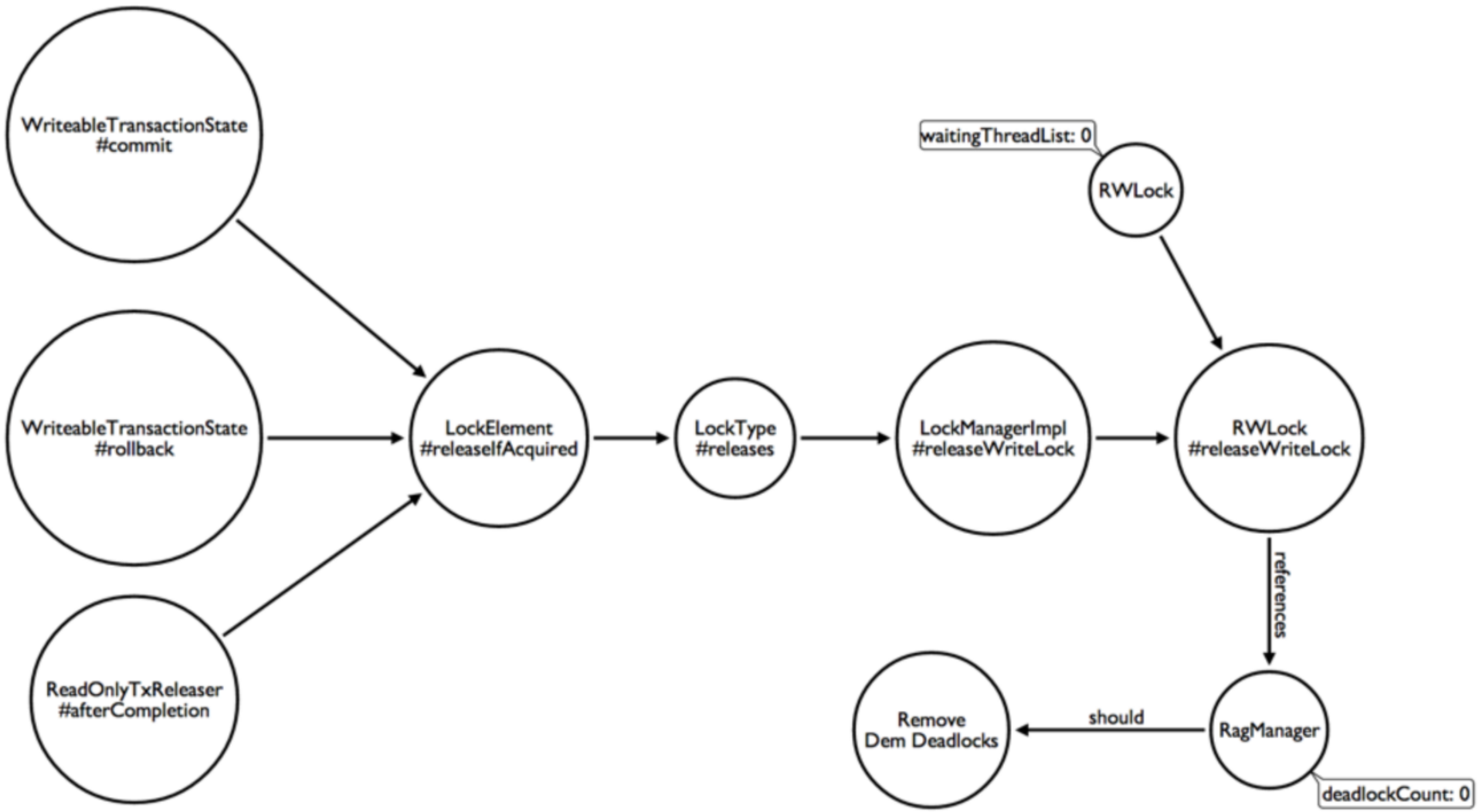
Query a JVM Heapdump

1. Get a heap-dump with jmap
   jmap -dump:format=b,file=dump.hprof <pid>
2. Use jhat dump.hprof find the /oql endpoint
3. run the *OQL* script to generate *Cypher* code
4. Import into Neo4j
5. Run arbitrary queries

Query a JVM Heapdump

1. Get a heap-dump with jmap
   jmap -dump:format=b,file=dump.hprof <pid>
2. Use jhat dump.hprof find the /oql endpoint
3. run the *OQL* script to generate *Cypher* code
4. Import into Neo4j
5. Run arbitrary queries
6. Visualize

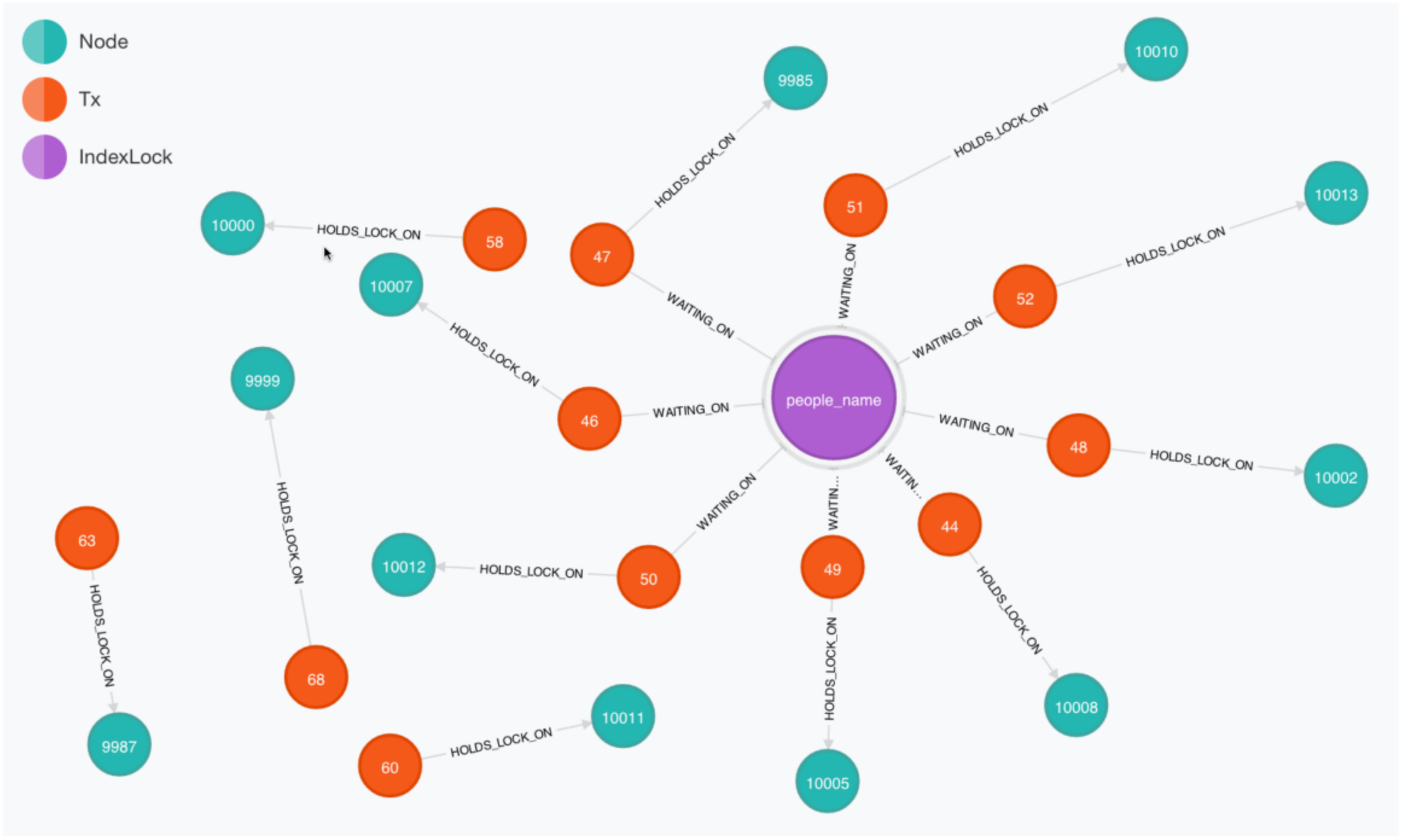
Graph Domain Model



OQL to Generate Cypher

```
select (function () {
  var entries = filter(p.waitingTxMap.data.table, function(it) { return it; });
  var resources = "";

  var idSeq = 0;

  for ( var i = 0; i < entries.length; i++ ) {
    var resource = entries[i].value.resource;
    var index = resource.index.toString();
    var key = resource.key.toString();
    var indexNameIndexKey = index + "_" + key;
    var indexName = "n" + idSeq++;

    var resourceName = "";

    var tx = entries[i].key;
    var txName = "n" + idSeq++;

    var cypher =  "MERGE (" + txName + ":Tx {txId:" + tx.eventIdentifier + "})<br />";
    cypher     += "MERGE (" + indexName + ":IndexLock {indexLockId:'" + indexNameIndexKey + "'})<br
    cypher     += "CREATE (" + txName + ")-[:WAITING_ON]->(" + indexName + ")<br />";

    resources += cypher + "<br />";
  }

...

  return resources;
})()
from org.neo4j.kernel.impl.transaction.RagManager p
```

Visualization of a Deadlock Scenario

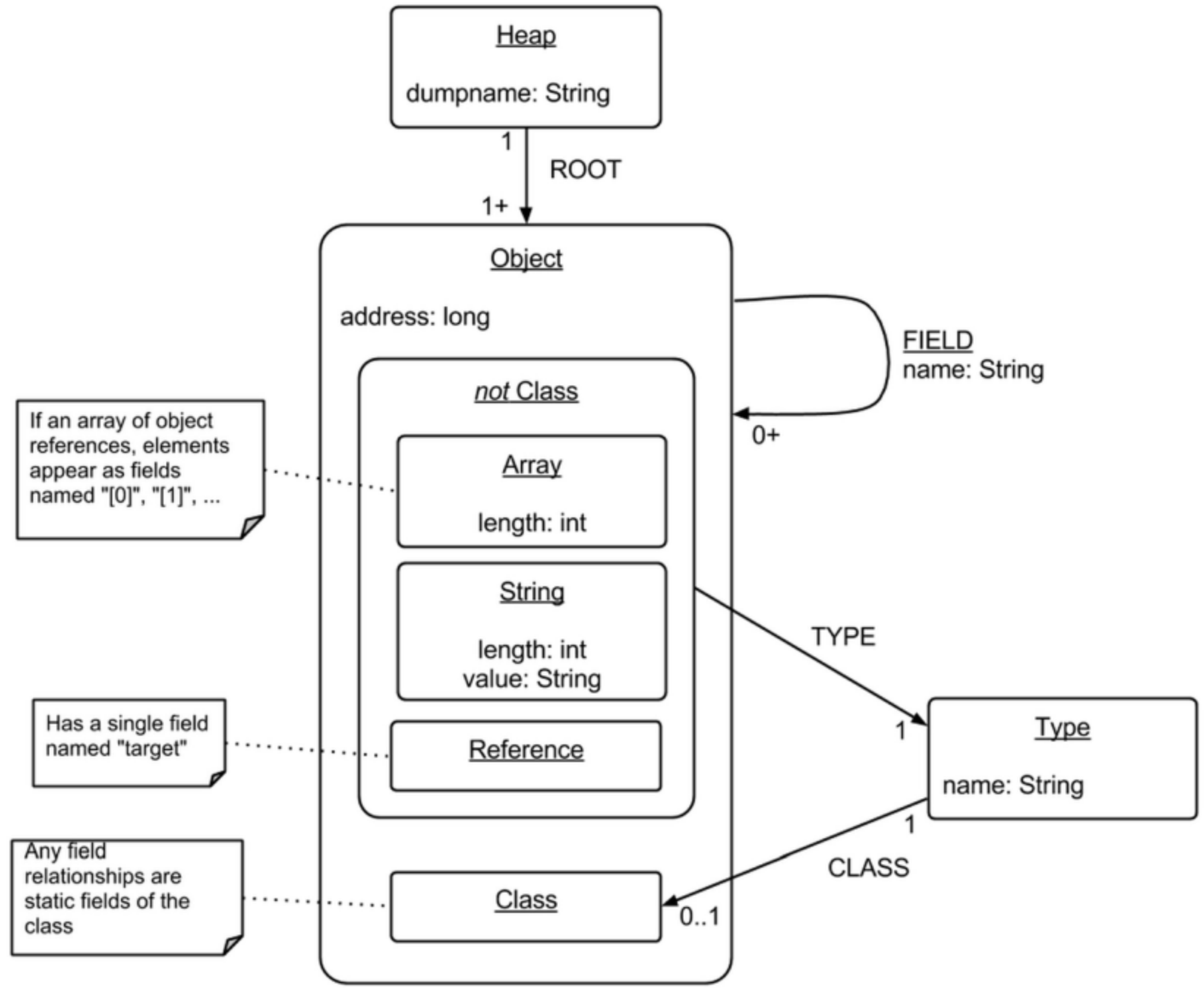


JVM heap analysis using Neo4j

Nat Pryce, James Richardson (Software Engineers, Sky)

- Use Neo4J for ad-hoc analysis of heap use in a proprietary embedded JVM that's deployed in one of the most widely used consumer products in the UK (Sky Box).
- Used Cypher queries that uncovered surprising aspects of their code, platform and the Java compiler.
- **And finding a memory leaking JSON parser with a Cypher query.**

Heap Model



From Nat's Graph Schema Modeling Approach

Software Analytics & Evolution

Analytics of (legacy) Software Systems

for modernization, insight gathering, testing, architecture retrieval

- Domain Knowledge Extraction
- Software Architecture Analysis
- Query Software Systems
- Knowledge Driven Testing
- Measure Software Systems
- Domain Specific Programming

Software Competence Center Hagenberg, Austria image::http://scch.at/files/assets/images/scch-logo.gif[right]

Graph Querying

Graph Querying

- Parsing of C, FORTRAN - source-code using Antlr

Graph Querying

- Parsing of C, FORTRAN - source-code using Antlr
- Building of a graph model

Graph Querying

- Parsing of C, FORTRAN - source-code using Antlr
- Building of a graph model
- Allow for querying, software analytics, metrics, visualization

Graph Querying

- Parsing of C, FORTRAN - source-code using Antlr
- Building of a graph model
- Allow for querying, software analytics, metrics, visualization

Quick Demo

Demo

Putty Source Code - functions reading structures

> one last visu for @GraphConnect #ShowMeYourGraph putty source code functions reading structures, complexity + effort pic.twitter.com/etN495kVkB
>
> — Wolfgang Beer (@wolfgangscch) October 1, 2014

Using graphs for source code analysis

Raoul-Gabriel Urma (PhD in Computer Science, The University of Cambridge)

Did you know that the source code of software that you engineer every day can also be represented as graphs?

In this talk, we demonstrate how you can perform program analysis using Neo4j.

We describe a prototype that stores a graph representation of the source code of Java programs in Neo4j. We can then query the graph using Cypher and we'll show various examples of possible queries such as "find me all recursive methods" and "find me all subtypes of a given type".

This talk is based on Raoul-Gabriel's research paper 'Source-Code Queries with Graph Databases—with Application to Programming Language Usage and Evolution'

Source Control, Issues, Social Coding

- What can you learn from commits
- about the code
- Class Toxicity, Frequency of Change (Feathers)
- about the people
- checkin times, collaboration, commit-size, commit-frequency
- Issues
- bug-rich classes (separation of concerns?)

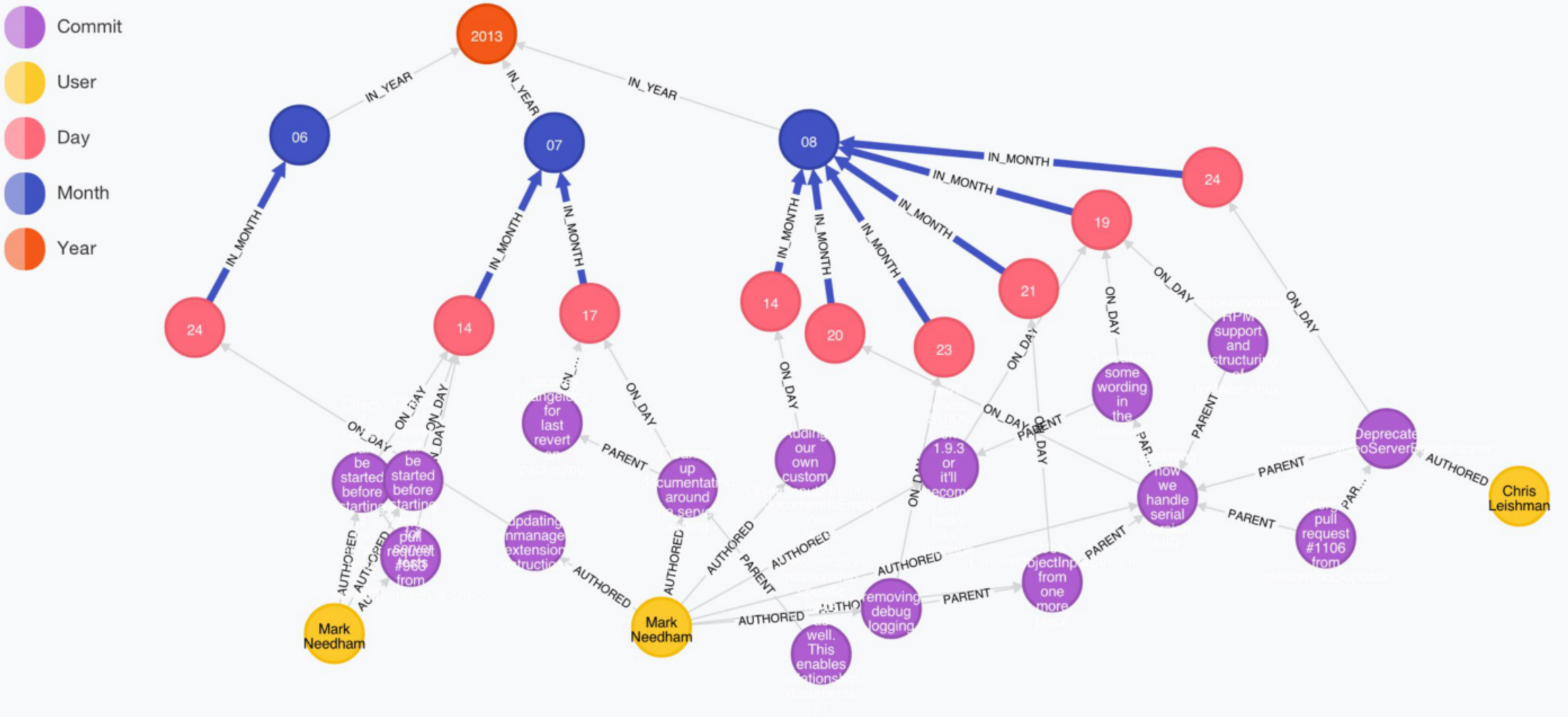
Example: Import Git Commit Logs into Neo4j

- git log --format emits CSV
- Graph Model
- LOAD CSV with Cypher
- Create / Update complex Graph Structure

Blog Post

Visualization

```
CYPHER MATCH path=(u:User {name:"Mark Needham"})-[:AUTHORED]->(:Commit)-[*..3]->(:Year) RETURN path
```



Finally: Some Eye Candy

Isaac & Nash (Software Engineers at Leap Motion)

- Leap Motion Software
- Inheritance Hierarchy, Call Graph
- Render to .dot file
- Use dotparse.js to read it in
- WebGL enabled Three.js rendering
- LeapMotion SDK 2.x beta

Let's have a look: Demo Source

Questions ? Thank You!

Status of the presenation

Cypher queries execution

Done with errors.

```
statement:
CREATE (:Year {year:2014})<-[:IN_YEAR]-(geekout:Conference {name:"Geekout"})
        -[:LOCATION]->(:City {name:"Tallinn"})
CREATE (track:Track {name:"Room 1"})-[:TRACK_OF]->(geekout)

MERGE (speaker:Attendee:Speaker {name:"Hadi Hariri"}) MERGE (geekout)<-
[:ATTENDS]-(speaker)

CREATE (speaker)-[:PRESENTS]->(session:Session {title:"Mouseless IDE"})<-
[:IN_TRACK]-(track)

FOREACH (name in ["Java","IDE","Development"] |
   MERGE (topic:Topic {name:name})
   CREATE (session)-[:HAS_TOPIC]->(topic)))
Message:
Invalid input ')': expected whitespace, LOAD CSV, START, MATCH, MERGE, CREATE,
SET, DELETE, REMOVE, FOREACH, WITH, RETURN, UNION, ';' or end of input (line 11,
column 43)
"   CREATE (session)-[:HAS_TOPIC]->(topic)))"
```