

No SQL?



Neo4j

the benefits of
graph databases

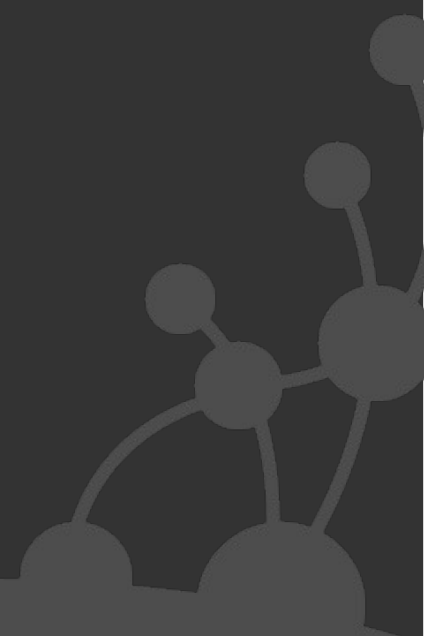
Emil Eifrem
CEO, Neo Technology

#neo4j
@emileifrem
emil@neotechnology.com



NoSQL

overview



First off: the name

- NoSQL is NOT “Never SQL”
- NoSQL is NOT “No To SQL”

NOSQL

is simply

Not **Only** SQL!

Four (emerging) NOSQL categories

◎ Key-value stores

- Based on Amazon's Dynamo paper
- Data model: (global) collection of K-V pairs
- Example: Dynamite, Voldemort, Tokyo*

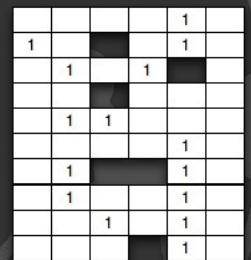
Key-Value



◎ BigTable clones

- Based on Google's BigTable paper
- Data model: big table, column families
- Example: HBase, Hypertable, Cassandra

BigTable

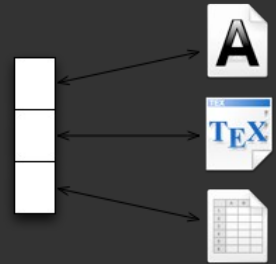


Four (emerging) NOSQL categories

◎ Document databases

- Inspired by Lotus Notes
- Data model: collections of K-V collections
- Example: CouchDB, MongoDB

Document



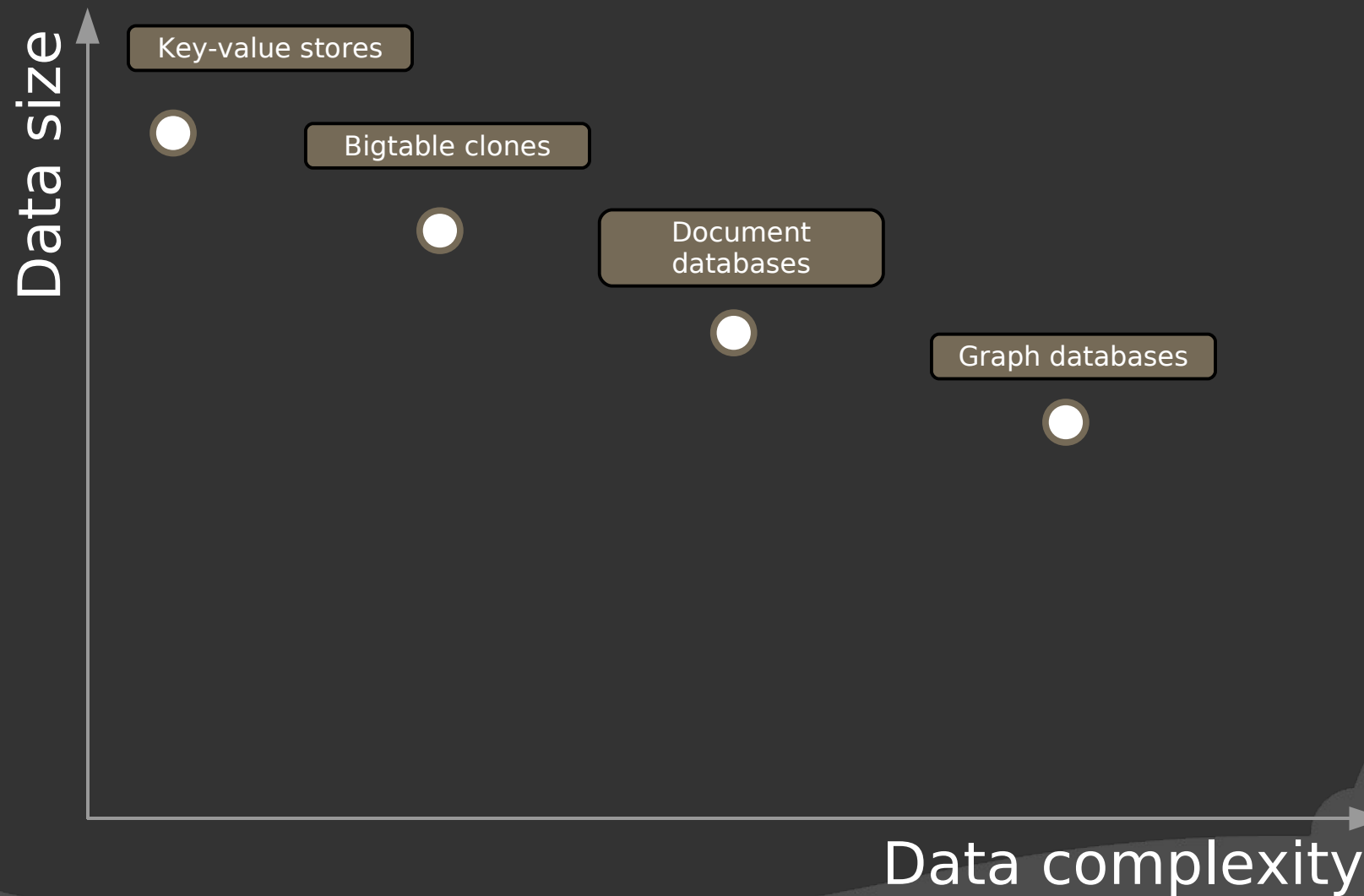
◎ Graph databases

- Inspired by Euler & graph theory
- Data model: nodes, rels, K-V on both
- Example: AllegroGraph, Sones, Neo4j

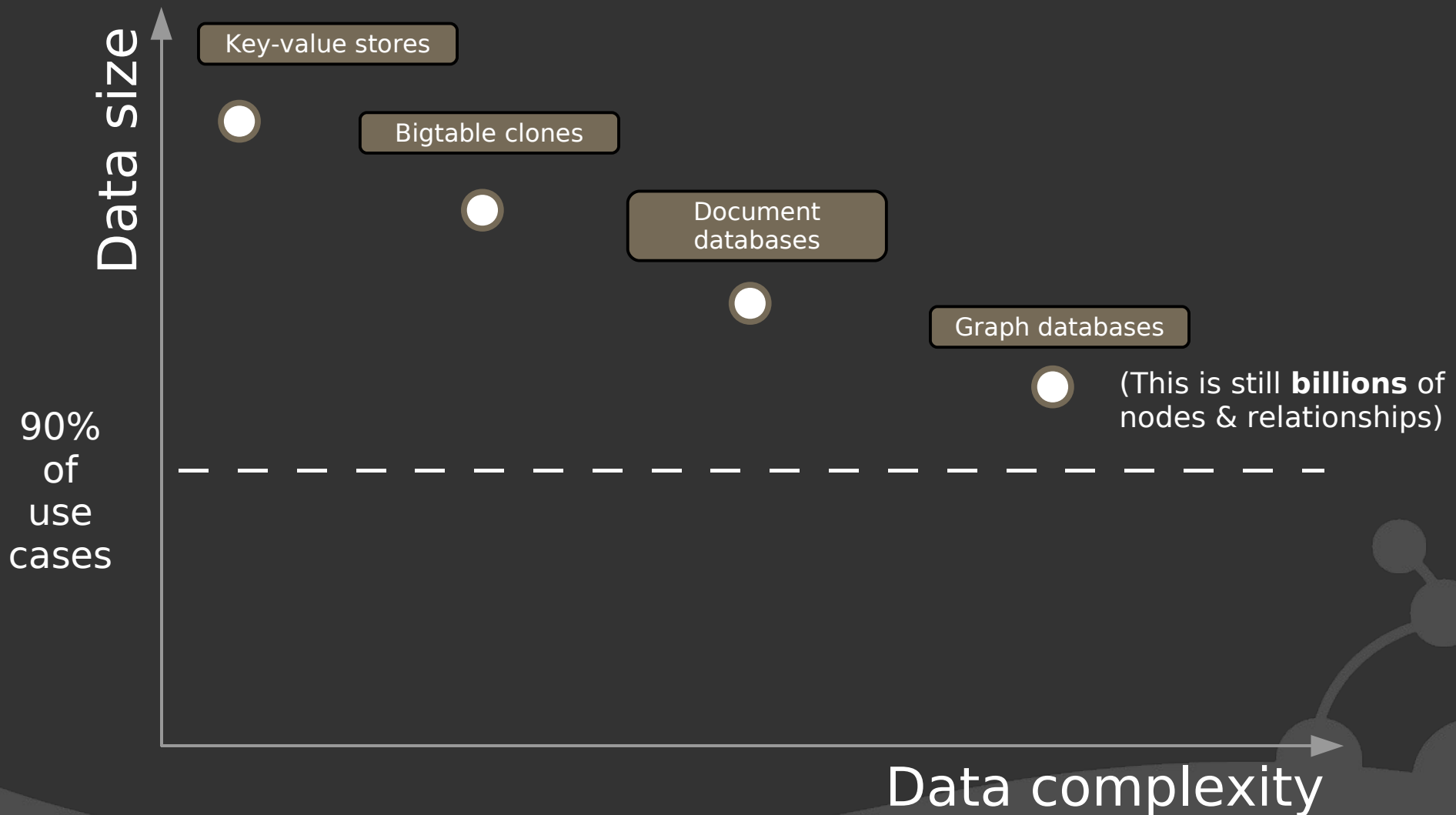
Graph DB



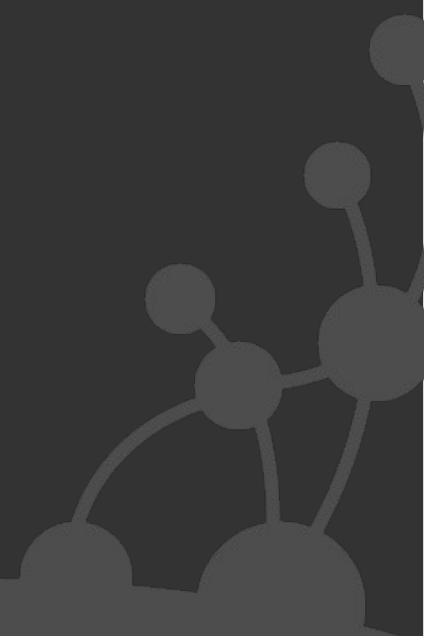
NOSQL data models



NOSQL data models



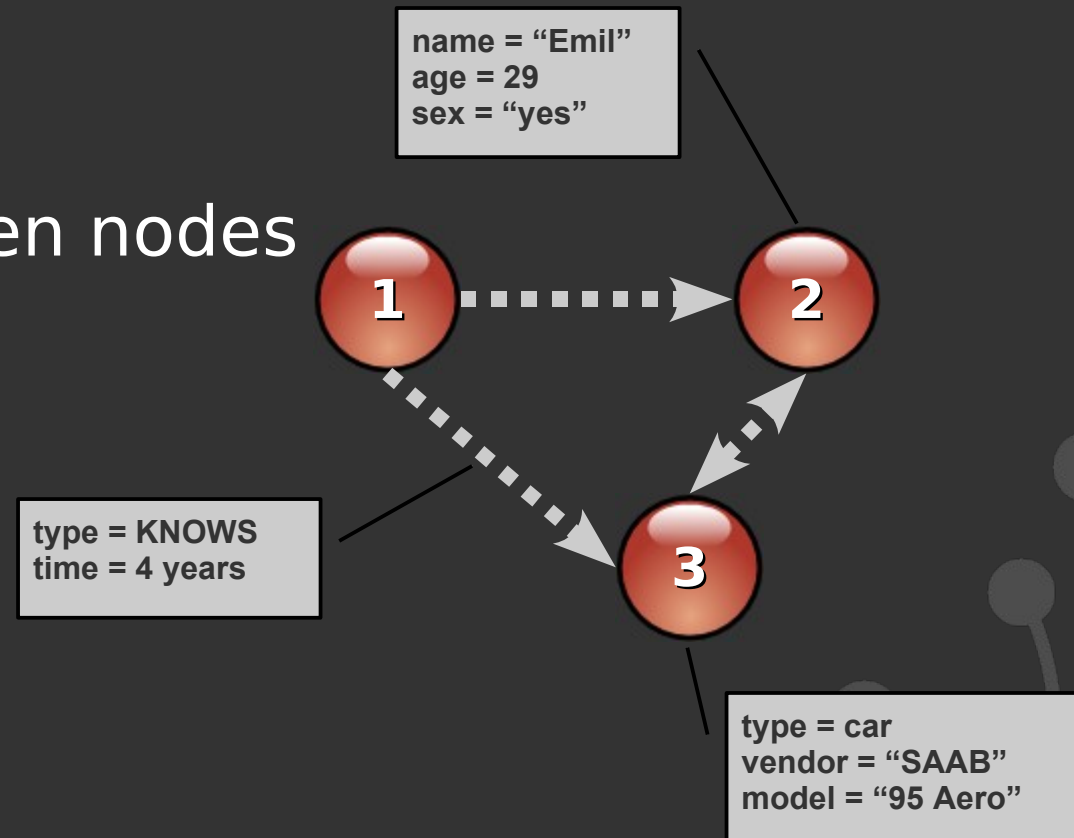
Graph DBs & Neo4j intro



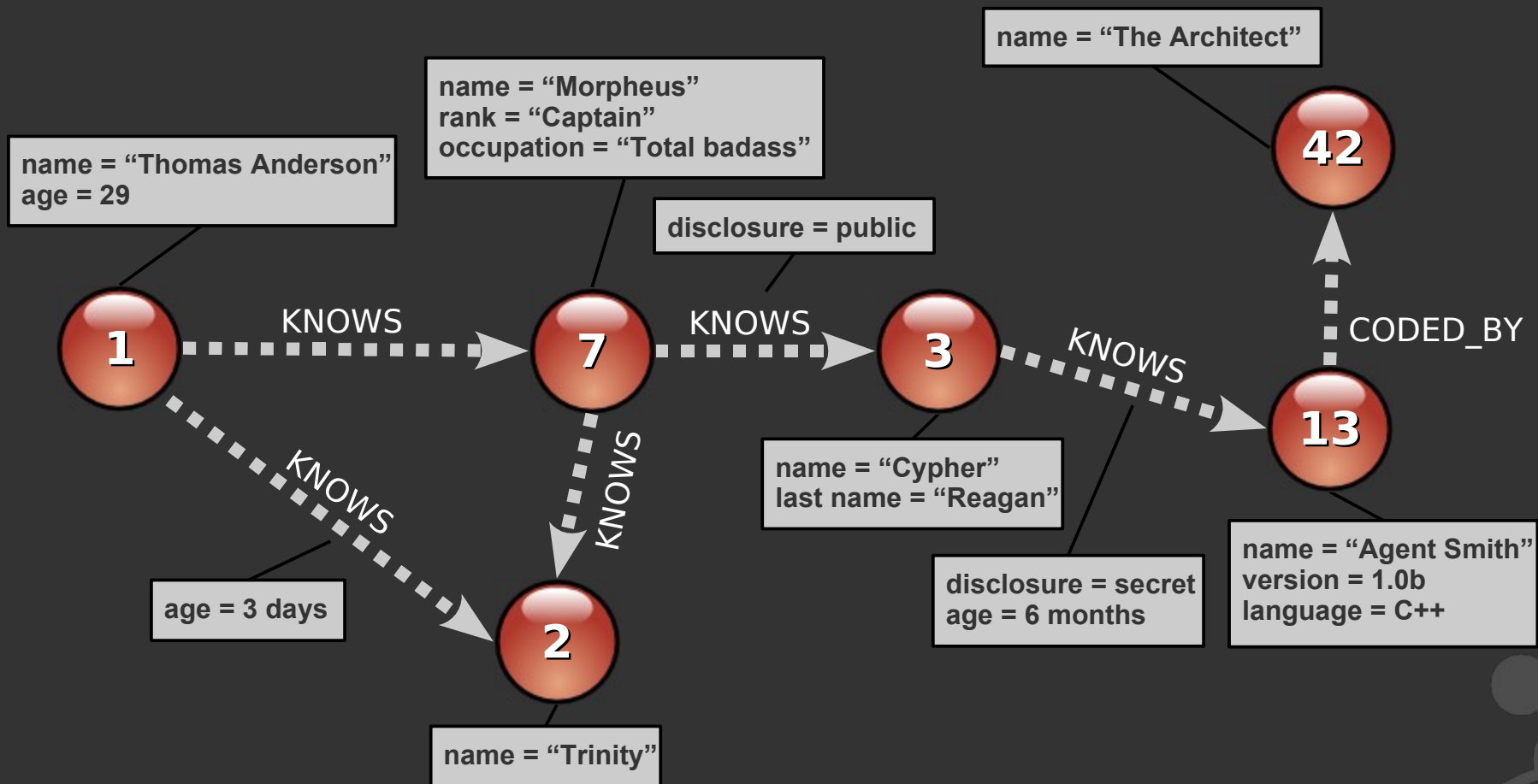
The Graph DB model: representation

Core abstractions:

- Nodes
- Relationships between nodes
- Properties on both



Example: The Matrix



Code (1): Building a node space

```
GraphDatabaseService graphDb = ... // Get factory

// Create Thomas 'Neo' Anderson
Node mrAnderson = graphDb.createNode();
mrAnderson.setProperty( "name", "Thomas Anderson" );
mrAnderson.setProperty( "age", 29 );

// Create Morpheus
Node morpheus = graphDb.createNode();
morpheus.setProperty( "name", "Morpheus" );
morpheus.setProperty( "rank", "Captain" );
morpheus.setProperty( "occupation", "Total bad ass" );

// Create a relationship representing that they know each other
mrAnderson.createRelationshipTo( morpheus, RelTypes.KNOWS );
// ...create Trinity, Cypher, Agent Smith, Architect similarly
```

Code (1): Building a node space

```
GraphDatabaseService graphDb = ... // Get factory
Transaction tx = neo.beginTx();

// Create Thomas 'Neo' Anderson
Node mrAnderson = graphDb.createNode();
mrAnderson.setProperty( "name", "Thomas Anderson" );
mrAnderson.setProperty( "age", 29 );

// Create Morpheus
Node morpheus = graphDb.createNode();
morpheus.setProperty( "name", "Morpheus" );
morpheus.setProperty( "rank", "Captain" );
morpheus.setProperty( "occupation", "Total bad ass" );

// Create a relationship representing that they know each other
mrAnderson.createRelationshipTo( morpheus, RelTypes.KNOWS );
// ...create Trinity, Cypher, Agent Smith, Architect similarly

tx.commit();
```

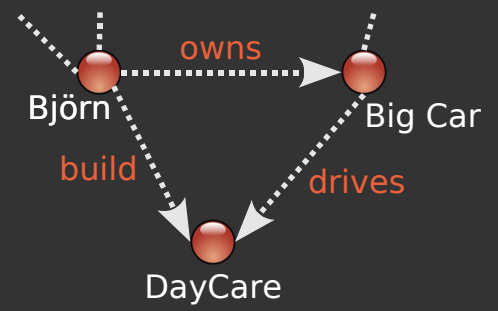
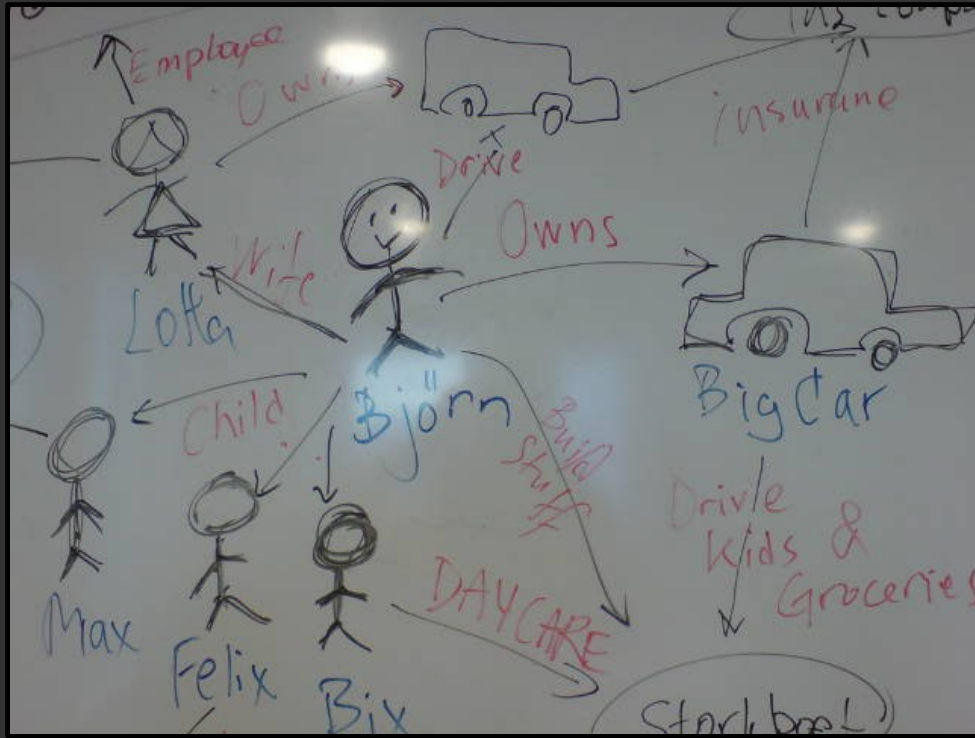
Code (1b): Defining RelationshipTypes

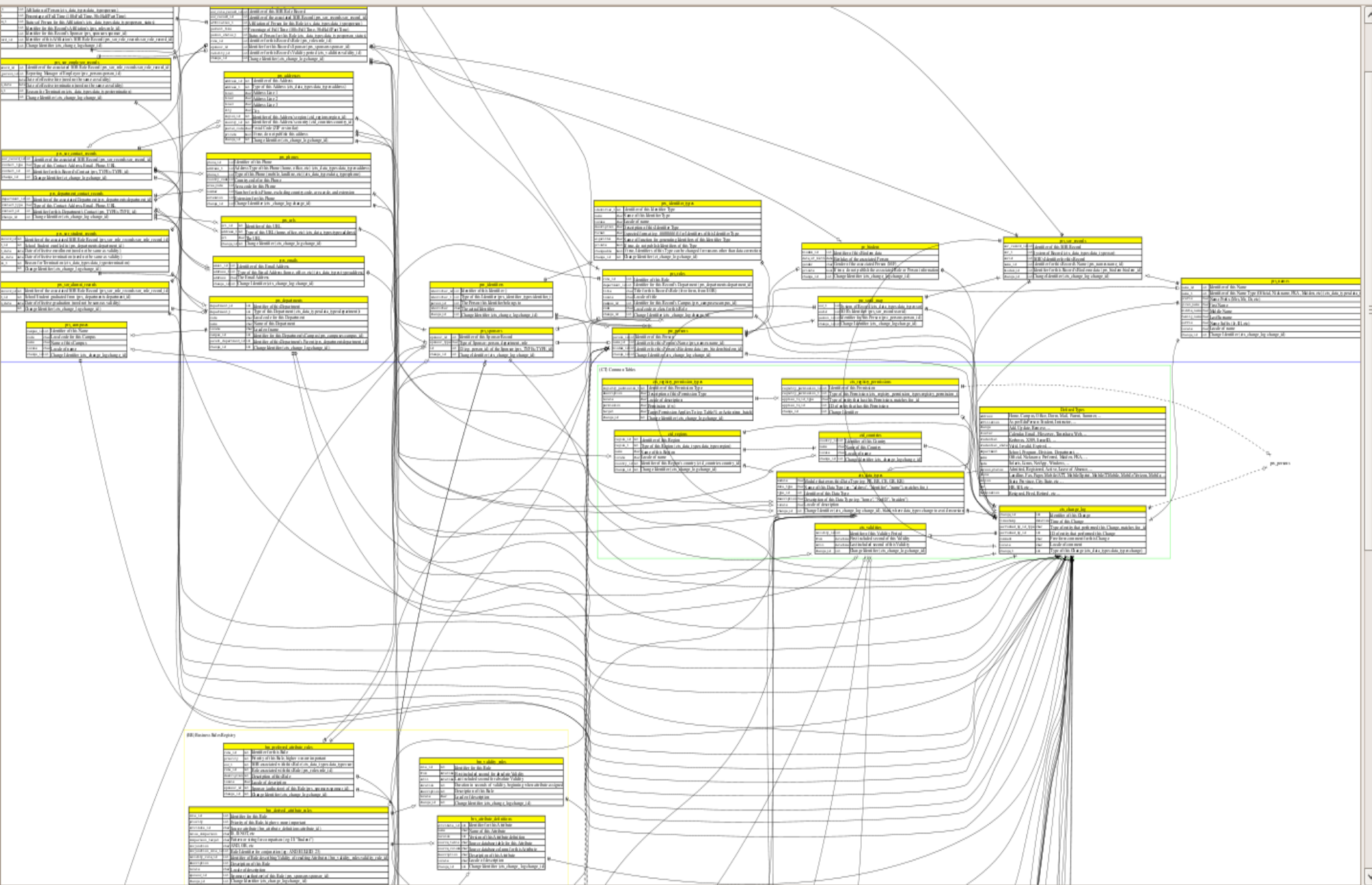
```
// In package org.neo4j.graphdb
public interface RelationshipType
{
    String name();
}

// In package org.yourdomain.yourapp
// Example on how to roll dynamic RelationshipTypes
class MyDynamicRelType implements RelationshipType
{
    private final String name;
    MyDynamicRelType( String name ){ this.name = name; }
    public String name() { return this.name; }
}

// Example on how to kick it, static-RelationshipType-like
enum MyStaticRelTypes implements RelationshipType
{
    KNOWS,
    WORKS_FOR,
}
```

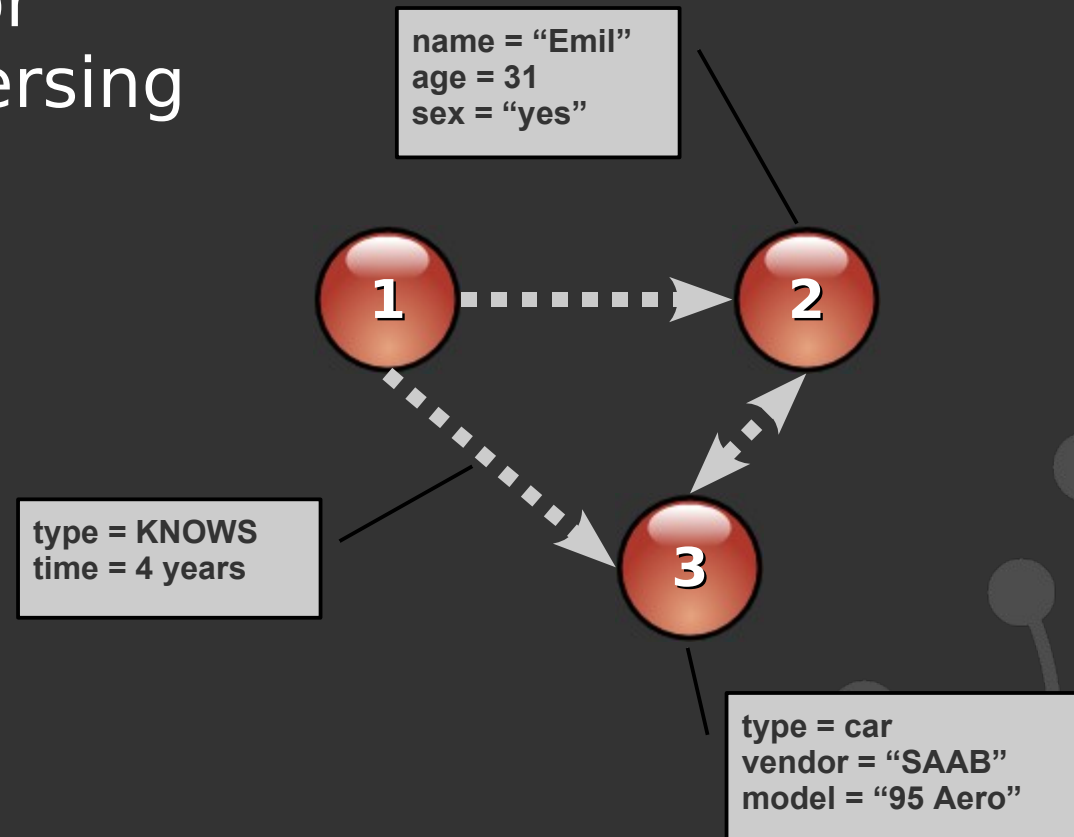

Whiteboard friendly



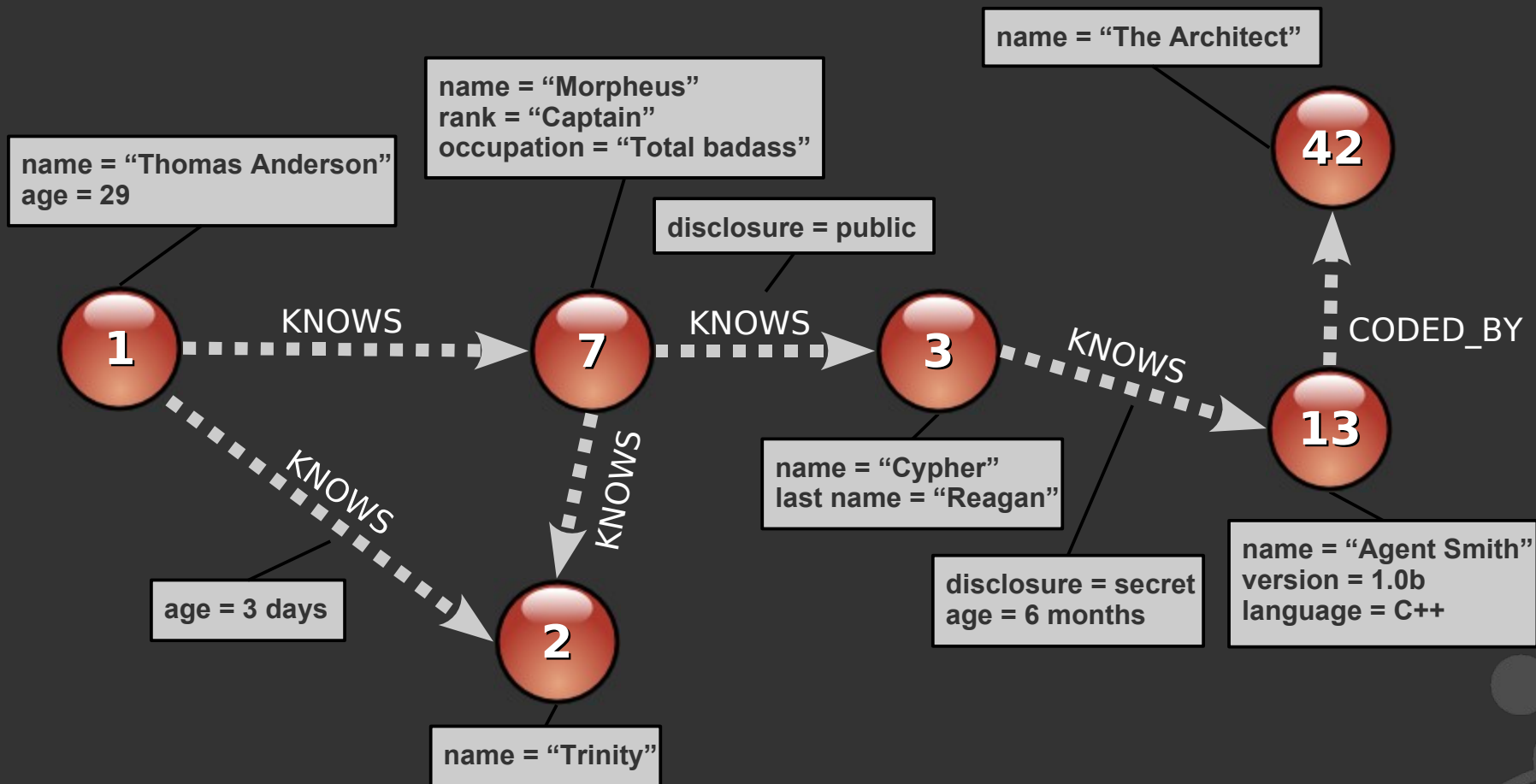


The Graph DB model: traversal

- ◎ Traverser framework for high-performance traversing across the node space



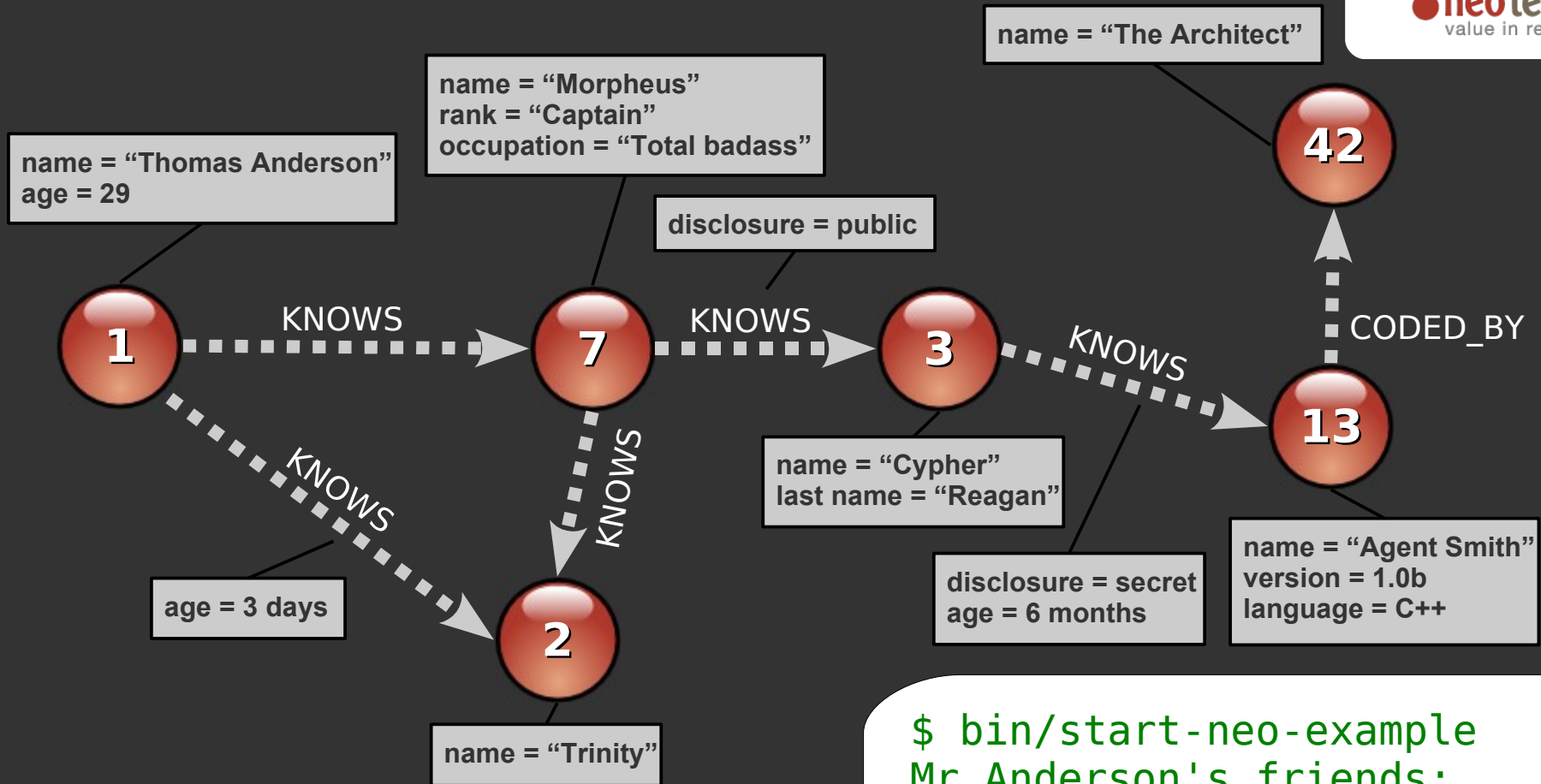
Example: Mr Anderson's friends



Code (2): Traversing a node space

```
// Instantiate a traverser that returns Mr Anderson's friends
Traverser friendsTraverser = mrAnderson.traverse(
    Traverser.Order.BREADTH_FIRST,
    StopEvaluator.END_OF_GRAPH,
    ReturnableEvaluator.ALL_BUT_START_NODE,
    RelTypes.KNOWS,
    Direction.OUTGOING );

// Traverse the node space and print out the result
System.out.println( "Mr Anderson's friends:" );
for ( Node friend : friendsTraverser )
{
    System.out.printf( "At depth %d => %s%n",
        friendsTraverser.currentPosition().getDepth(),
        friend.getProperty( "name" ) );
}
```



```

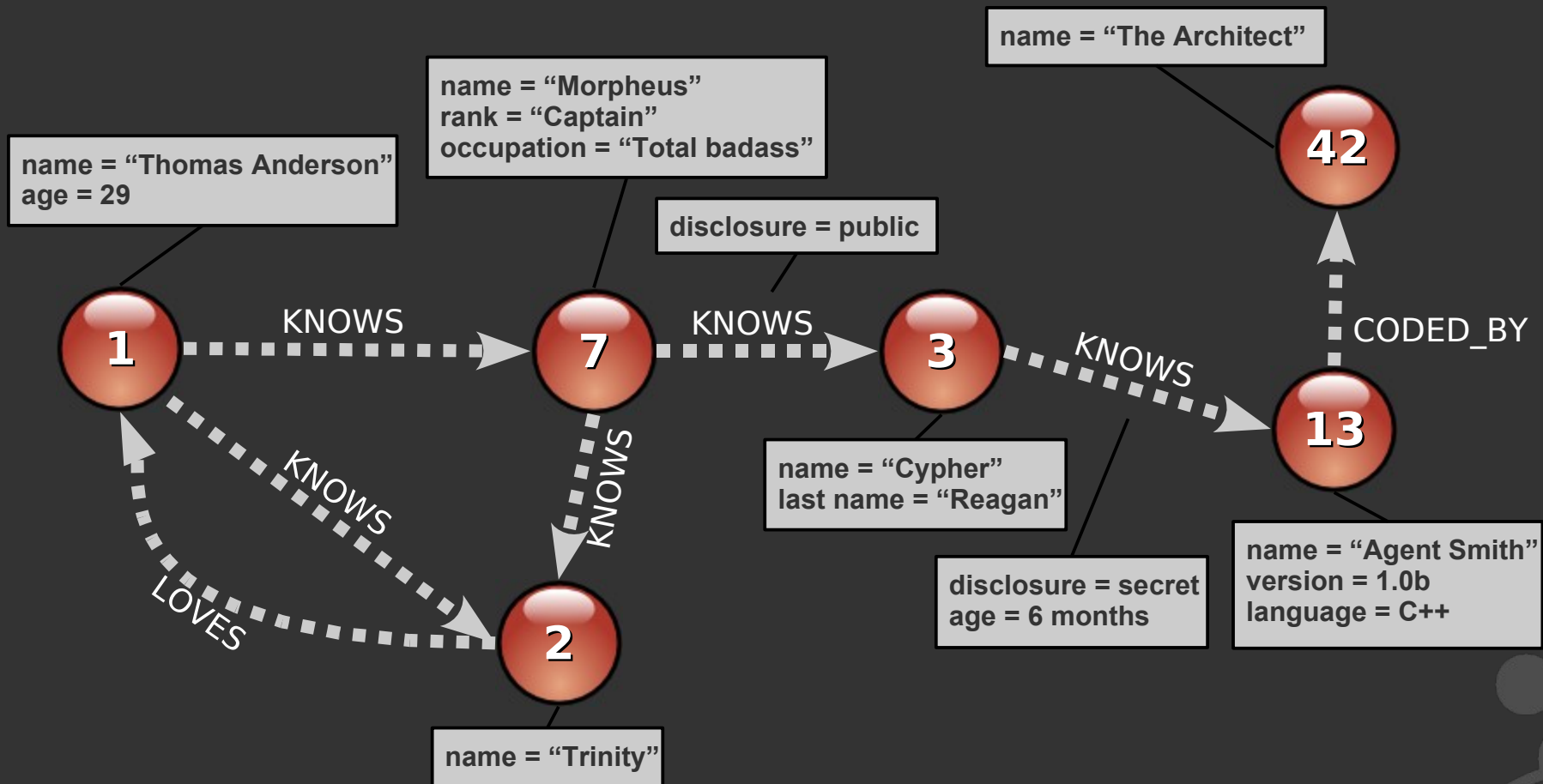
friendsTraverser = mrAnderson.traverse(
  Traverser.Order.BREADTH_FIRST,
  StopEvaluator.END_OF_GRAPH,
  ReturnableEvaluator.ALL_BUT_START_NODE,
  RelTypes.KNOWS,
  Direction.OUTGOING );
  
```

```

$ bin/start-neo-example
Mr Anderson's friends:

At depth 1 => Morpheus
At depth 1 => Trinity
At depth 2 => Cypher
At depth 3 => Agent Smith
$
  
```

Example: Friends in love?

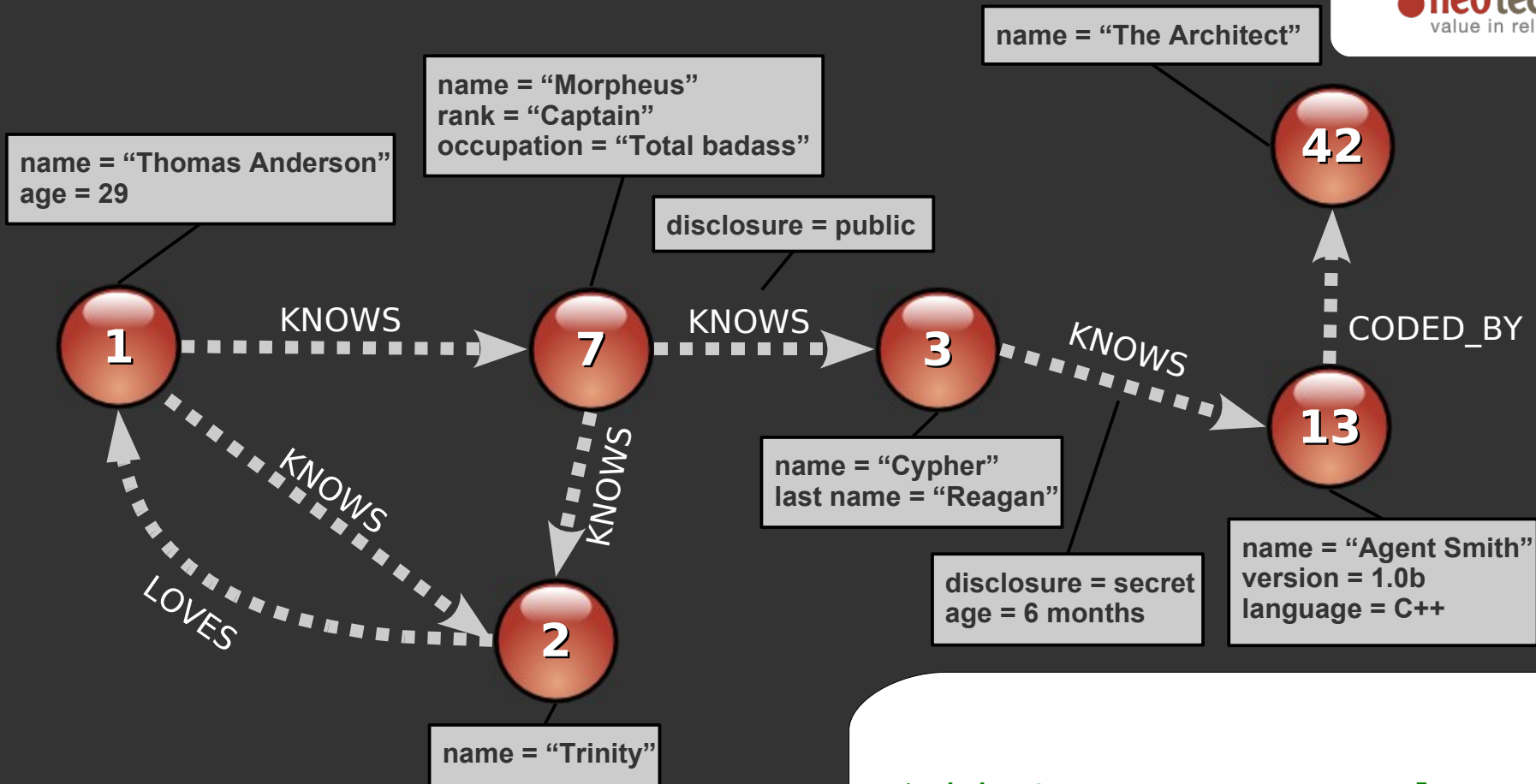


Code (3a): Custom traverser

```
// Create a traverser that returns all "friends in love"
Traverser loveTraverser = mrAnderson.traverse(
    Traverser.Order.BREADTH_FIRST,
    StopEvaluator.END_OF_GRAPH,
    new ReturnableEvaluator()
    {
        public boolean isReturnableNode( TraversalPosition pos )
        {
            return pos.currentNode().hasRelationship(
                RelTypes.LOVES, Direction.OUTGOING );
        }
    },
    RelTypes.KNOWS,
    Direction.OUTGOING );
```


Code (3a): Custom traverser

```
// Traverse the node space and print out the result  
System.out.println( "Who's a lover?" );  
for ( Node person : loveTraverser )  
{  
    System.out.printf( "At depth %d => %s%n",  
        loveTraverser.currentPosition().getDepth(),  
        person.getProperty( "name" ) );  
}
```



```

new ReturnableEvaluator()
{
  public boolean isReturnableNode(
    TraversalPosition pos)
  {
    return pos.currentNode().
      hasRelationship( RelTypes.LOVES,
        Direction.OUTGOING );
  }
},

```

```

$ bin/start-neo-example
Who's a lover?

```

```

At depth 1 => Trinity
$

```

Bonus code: domain model

- ◎ How do you implement your domain model?
- ◎ Use the delegator pattern, i.e. every domain entity wraps a Neo4j primitive:

```
// In package org.yourdomain.yourapp
class PersonImpl implements Person
{
    private final Node underlyingNode;
    PersonImpl( Node node ){ this.underlyingNode = node; }

    public String getName()
    {
        return (String) this.underlyingNode.getProperty( "name" );
    }
    public void setName( String name )
    {
        this.underlyingNode.setProperty( "name", name );
    }
}
```

Domain layer frameworks

◎ Qi4j (www.qi4j.org)

- Framework for doing DDD in pure Java5
- Defines Entities / Associations / Properties
 - Sound familiar? Nodes / Rel's / Properties!
- Neo4j is an “EntityStore” backend



◎ Jo4neo (<http://code.google.com/p/jo4neo>)

- Annotation driven
- Weaves Neo4j-backed persistence into domain objects at runtime

Neo4j system characteristics

◎ Disk-based

- Native graph storage engine with custom binary on-disk format

◎ Transactional

- JTA/JTS, XA, 2PC, Tx recovery, deadlock detection, MVCC, etc

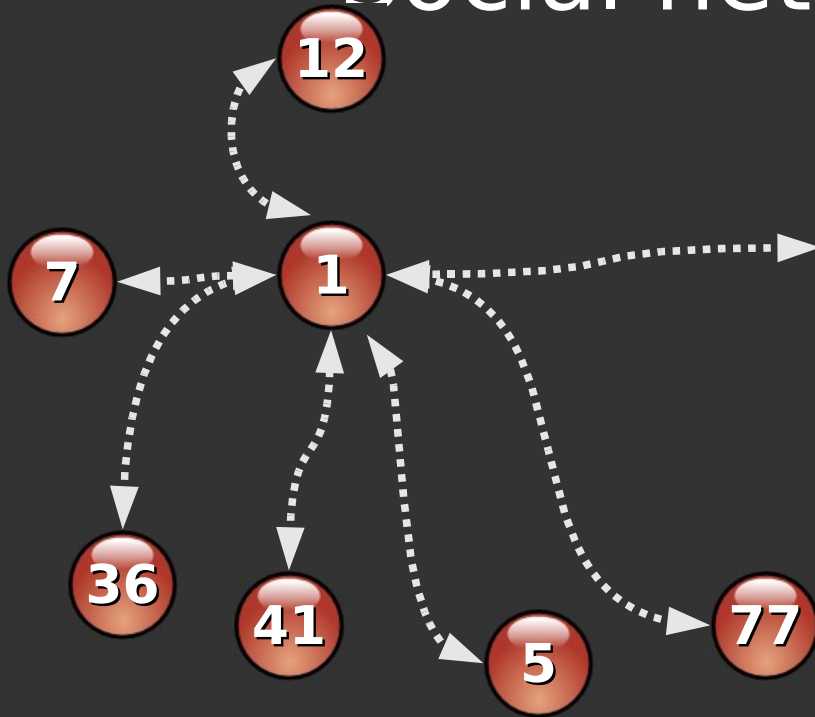
◎ Scales up

- Many **billions** of nodes/rels/props on single JVM

◎ Robust

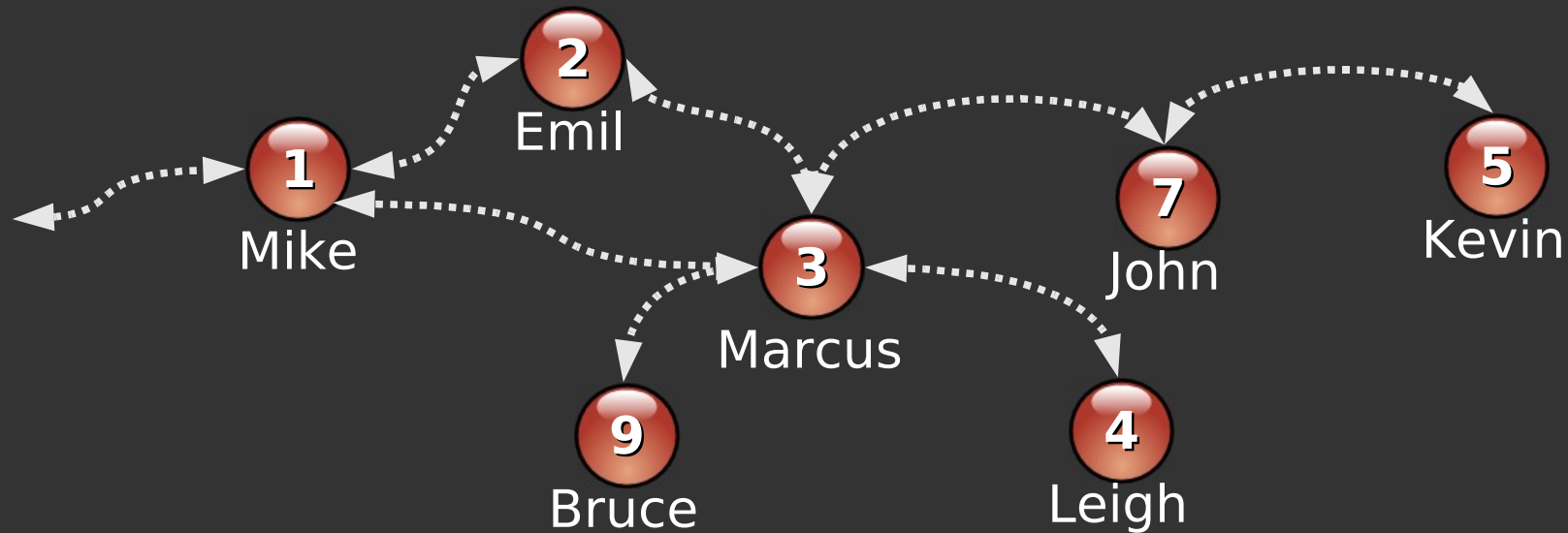
- 6+ years in 24/7 production

Social network *pathExists()*



- ~1k persons
- Avg 50 friends per person
- *pathExists(a, b)* limit depth 4
- Two backends
- Eliminate disk IO so warm up caches

Social network *pathExists()*



Relational database
 Graph database (Neo4j)
 Graph database (Neo4j)

persons query time



Got neo4j to do a do a lookup in 2 seconds, that sql server did in 45 minutes. neo4j rocks!



6:28 AM Jun 30th from web



turboCodr

John Conwell





Home Profile Find People Settings Help Sign out

Getting 40Mb/s write speeds out of Neo4J+Lucene, go Neo go!



9:12 PM Jun 9th from twhirl



peepwl
peepwl.com



Pros & Cons compared to RDBMS

- + No O/R impedance mismatch (*whiteboard friendly*)
- + Can easily evolve schemas
- + Can represent semi-structured info
- + Can represent graphs/networks (*with* performance)
- Lacks in tool and framework support
- Few other implementations => potential lock in
- + ~~No support for ad-hoc queries~~

Query languages

◎ SPARQL – “SQL for linked data”

- EX:

```
"SELECT ?person WHERE {  
  ?person neo4j:KNOWS ?friend .  
  ?friend neo4j:KNOWS ?foe .  
  ?foe neo4j:name "Larry Ellison" .  
}"
```

◎ Gremlin – “perl for graphs”

- EX:

```
"./outE[@label='KNOWS']/inV[@age > 30]/@name"
```

The Neo4j ecosystem

◎ Neo4j is an embedded database

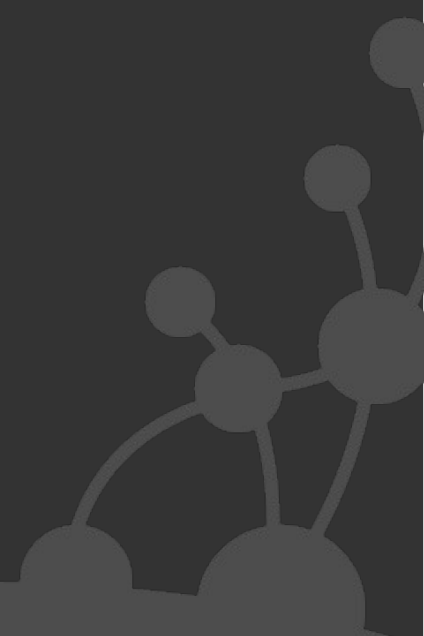
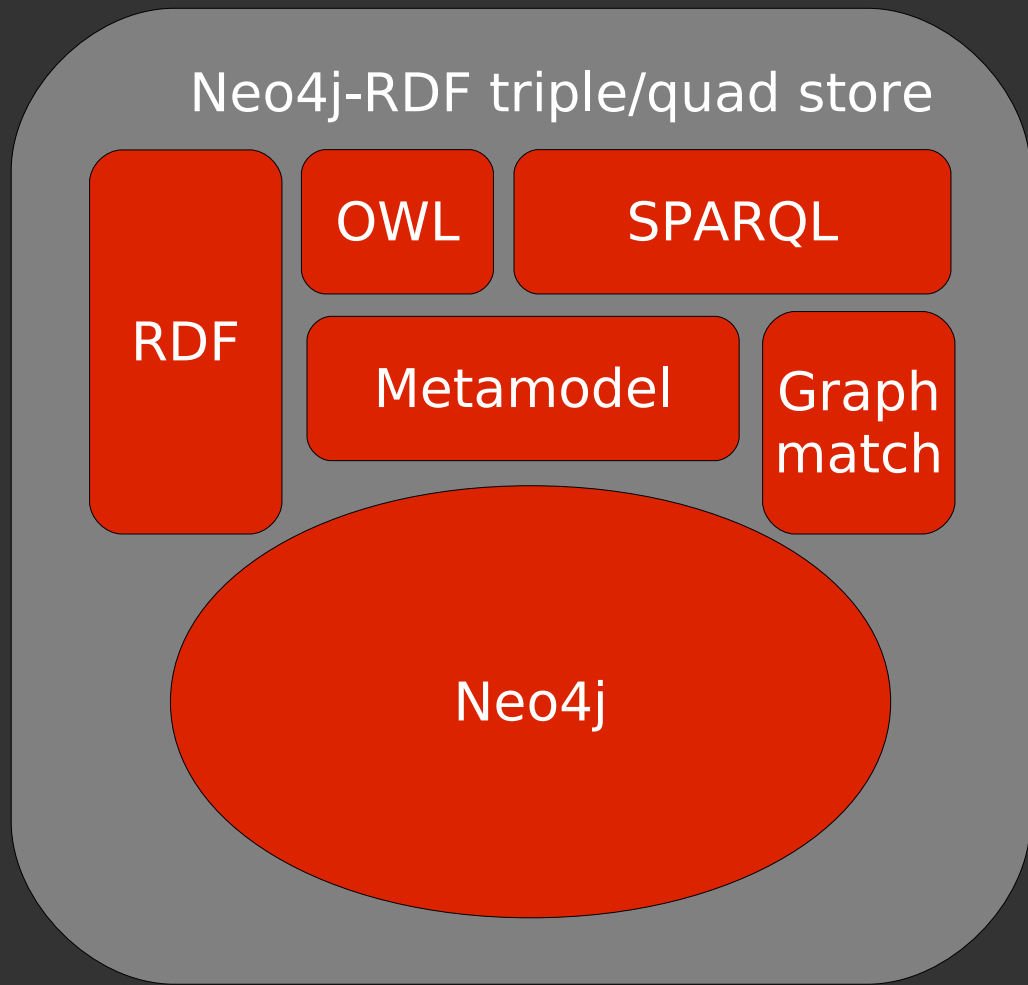
- Tiny teeny lil jar file

◎ Component ecosystem

- index
- meta-model
- graph-matching
- remote-graphdb
- sparql-engine
- ...

◎ See <http://components.neo4j.org>

Example: Neo4j-RDF



Language bindings

- ◎ Neo4j.py – bindings for Jython and CPython
 - <http://components.neo4j.org/neo4j.py>
- ◎ Neo4jrb – bindings for JRuby (incl RESTful API)
 - <http://wiki.neo4j.org/content/Ruby>
- ◎ Neo4jrb-simple
 - <http://github.com/mdeiters/neo4jr-simple>
- ◎ Clojure
 - <http://wiki.neo4j.org/content/Clojure>
- ◎ Scala (incl RESTful API)
 - <http://wiki.neo4j.org/content/Scala>



Integration of Neo4j in Grails



☆☆☆☆☆
(0 Ratings)

AUTHOR(S):

Stefan Armbruster

CURRENT RELEASE:

0.1

GRAILS VERSION:

1.1.1 > *

TAGS

neo4j - persistence +

Fisheye Docs Edit Plugin

Installation

Description

Faq

Screenshots

Edit View Info

The plugin's goal is to provide an alternative approach for storing Grails domain classes: in the Neo4j database.

Neo4j is a relative new and very interesting approach for persistence in a non-SQLish way. Neo4j is a graph database and uses the concept of

Nodes

A node is the basic building block. It normally represents a "something", a entity.

Relationships

1. Create your sample application:

```
grails create-app neo4jtest; cd neo4jtest
```

2. Remove the hibernate plugin:

```
grails uninstall-plugin hibernate
```

3. Add the Neo4j plugin:

```
grails install-plugin neo4j
```

4. create some sample domain classes:

```
grails create-domain-class Author
grails create-domain-class Book
```

5. create a controller for the domain class

```
grails create-controller Author
grails create-controller Book
```

6. modify the domain classes:

```
class Author {
    String name
    Date dob

    static hasMany = [ books: Book ]
}
```

and

```
class Book {
    String title
    static belongsTo = [author:Author]
}
```

7. modify the controller to use dynamic scaffolding:

```
class AuthorController {
    def scaffold = true
}
```

```
class BookController {
    def scaffold = true
}
```

8. start up the application:

```
grails run-app
```

9. use it, love it: go to <http://localhost:8080/neo4jtest>, add some authors and books.

10. to explore the Neo4j node space created with your grails app, check out [Neoclipse](#).

- [Gnome and CTI \(computer-telephone-integration\)](#)

Tag Cloud

cti gnome **grails** linux **neo4j** telephone ubuntu

Blogroll

- [GroovyBlogs](#)

Categories

- [Uncategorized](#)

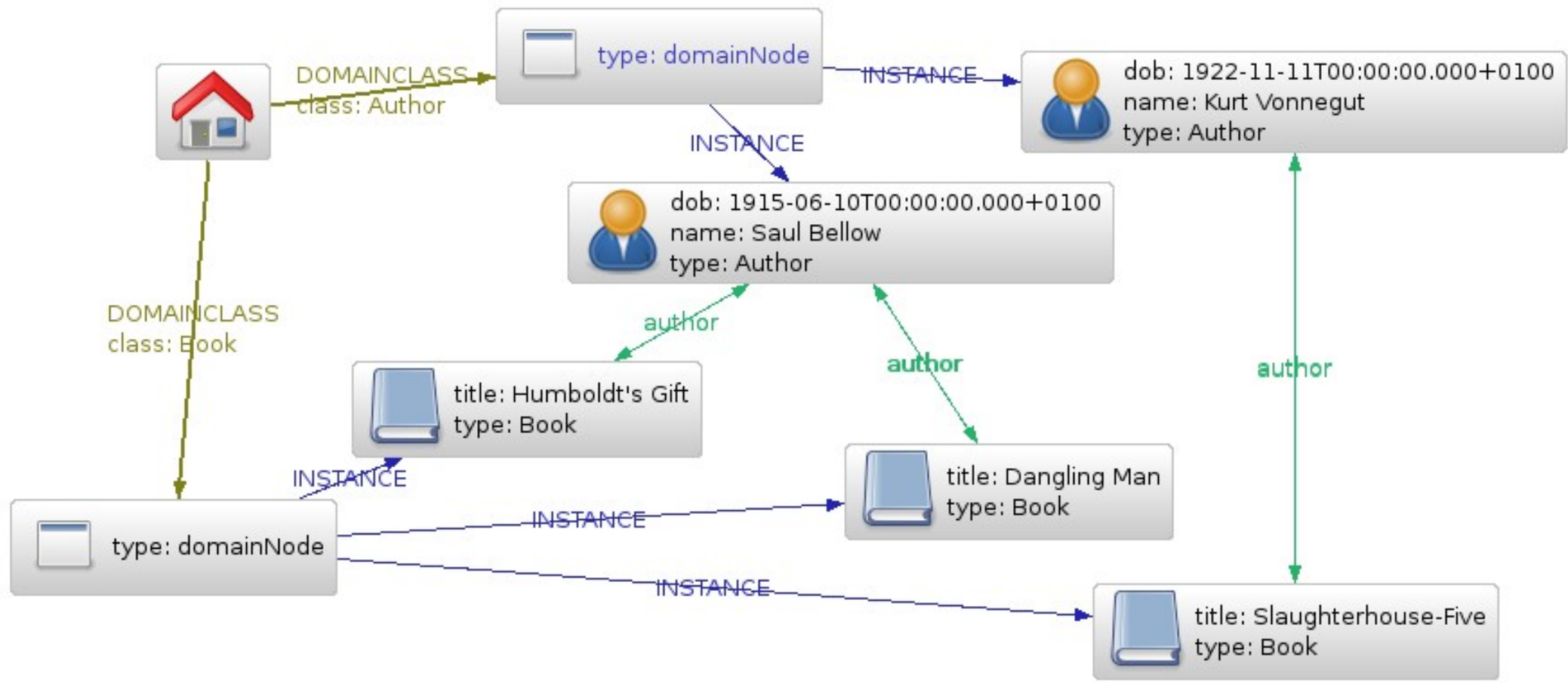
Archives

- [October 2009](#)
- [September 2009](#)

Meta

- [Log in](#)

Grails Neoclipse screendump




Scale out – replication

- ◎ Rolling out Neo4j HA... now :)
- ◎ Master-slave replication, 1st configuration
 - MySQL style... ish
 - Except all instances can write, synchronously between writing slave & master (strong consistency)
 - Updates are asynchronously propagated to the other slaves (eventual consistency)
- ◎ This can handle billions of entities...
- ◎ ... but not 100B

Scale out – partitioning

◎ Sharding possible today

- ... but you have to do manual work
- ... just as with MySQL
- Great option: shard on top of resilient, scalable OSS app server  Newton, see: www.codecauldron.org

◎ Transparent partitioning? Neo4j 2.0

- 100B? Easy to say. Sliiiiightly harder to do.
- Fundamentals: BASE & eventual consistency
- Generic clustering algorithm as base case, but give lots of knobs for developers

How ego are you? (aka other impls?)

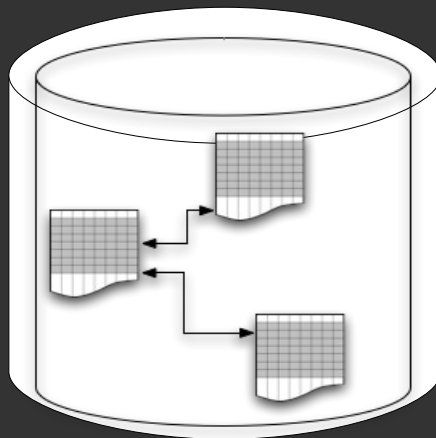
- ◎ Franz' **AllegroGraph** (<http://agraph.franz.com>)
 - Proprietary, Lisp, RDF-oriented but real graphdb
- ◎ Sones **graphDB** (<http://sones.com>)
 - Proprietary, .NET, cloud-only, req invite for test
- ◎ **Kloudshare** (<http://kloudshare.com>)
 - Graph database in the cloud, still stealth mode
- ◎ Google **Pregel** (<http://bit.ly/dP9IP>)
 - We are oh-so-secret
- ◎ Some academic papers from ~10 years ago
 - $G = \{V, E\}$ #FAIL

Conclusion

- ◎ Graphs & Neo4j => teh awesome!
- ◎ Available NOW under AGPLv3 / commercial license
 - AGPLv3: “if you’re open source, we’re open source”
 - If you have proprietary software? Must buy a commercial license
 - But up to 1M primitives it’s free for all uses!
- ◎ Download
 - <http://neo4j.org>
- ◎ Feedback
 - <http://lists.neo4j.org>

Looking ahead: polyglot persistence

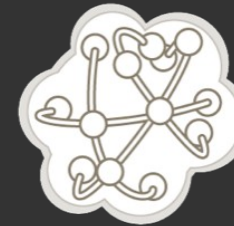
SQL && NoSQL



BigTable



Graph DB



Document

Key-Value



: julianbrowne **twitter**

Home Profile Find People Settings Help Sign out

Going to play with Neo4j this w/e.
Seems to me that even after arguments
about ACID/scale/CAP it's just more
human & agile to be graph-based



5:42 PM Jul 3rd from web



julianbrowne



Questions?



Image credit: lost again! Sorry :(



<http://neotechnology.com>