NOSQL Databases and Neo4j

Database and DBMS

- Database - Organized collection of data
- The term database is correctly applied to the data and their supporting data structures.

- DBMS - Database Management System: a software package with computer programs that controls the creation, maintenance and use of a database.

Types of Happening Databases

- Relational database – nothing new but still in use and it seems it will always be a happening one.
- Cloud databases – everything is cloudy.
- Data warehouse – Huge! Huge! Huge! archives.
- Embedded databases – you can’t see them :P
- Document oriented database – the In thing.
- Hypermedia database – WWW.
- Graph database – facebook, twitter, social network.

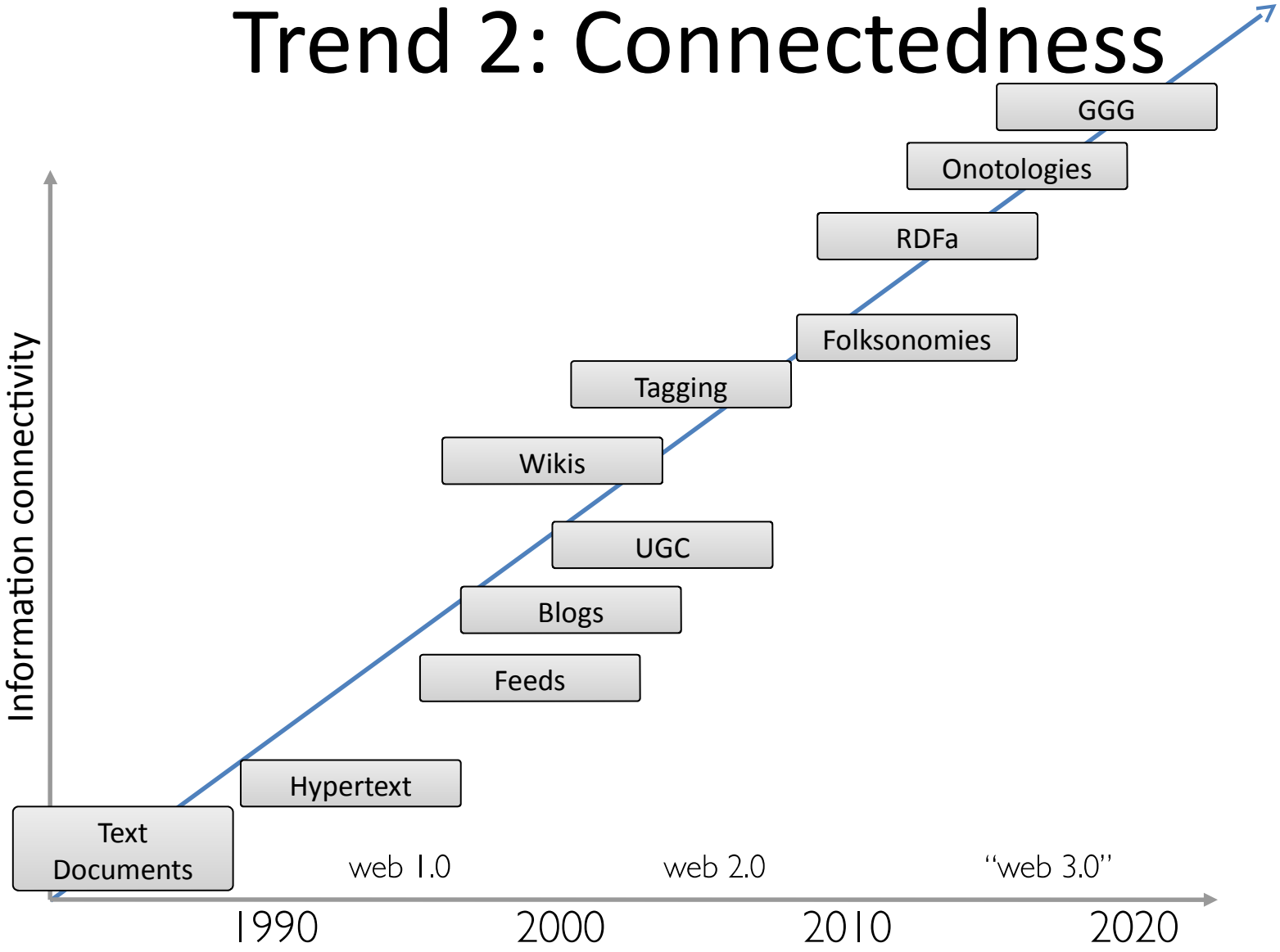
NOSQL is simply…

Not **O**nly **S**QL

Why NOSQL now?

Driving trends

Trend 2: Connectedness



Trend 3: Semi-structured information

- Individualisation of content
  - 1970's salary lists, all elements exactly one job
  - 2000's salary lists, we need many job columns!
- Store more data about each entity
- Trend accelerated by the decentralization of content generation
  - Age of participation ("web 2.0")

Why Graph Databases?

- Schema Less and Efficient storage of Semi Structured Information
- No O/R mismatch – very natural to map a graph to an Object Oriented language like Ruby.
- Express Queries as Traversals. Fast deep traversal instead of slow SQL queries that span many table joins.
- Very natural to express graph related problem with traversals (recommendation engine, find shortest parth etc.)

Social Network “path exists” Performance

- Experiment:
  - ~1k persons
  - Average 50 friends per person
  - `pathExists(a,b)` limited to depth 4

	# persons	query time
Relational database	1000	2000ms

Social Network “path exists” Performance

- Experiment:
  - ~1k persons
  - Average 50 friends per person
  - `pathExists(a,b)` limited to depth 4

	# persons	query time
Relational database	1000	2000ms
Neo4j	1000	2ms

Social Network “path exists” Performance

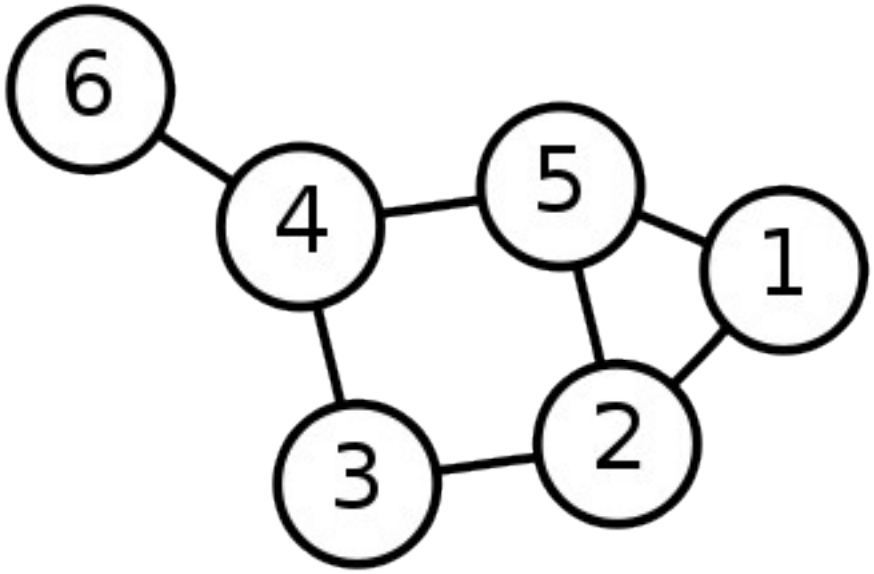
- Experiment:
  - ~1k persons
  - Average 50 friends per person
  - `pathExists(a,b)` limited to depth 4

	# persons	query time
Relational database	1000	2000ms
Neo4j	1000	2ms
Neo4j	1000000	2ms

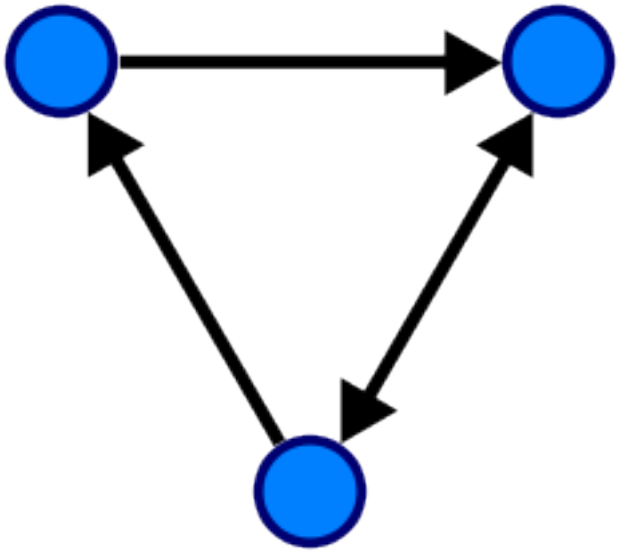
What are graphs good for?

- Recommendations
- Business intelligence
- Social computing
- Geospatial
- Systems management
- Web of things
- Genealogy
- Time series data
- Product catalogue
- Web analytics
- Scientific computing (especially bioinformatics)

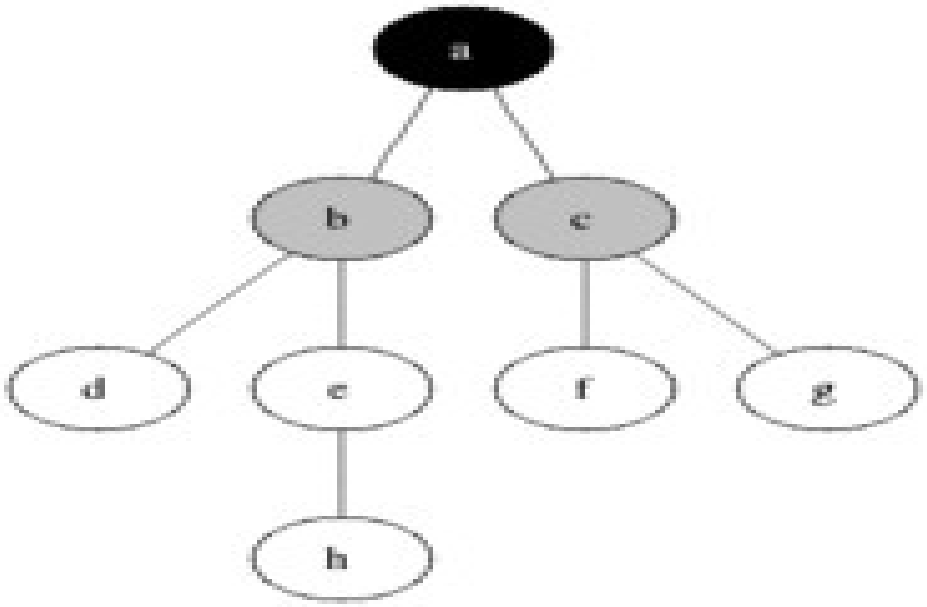
Graphs



Directed Graphs



Breadth First Search



Depth First Search

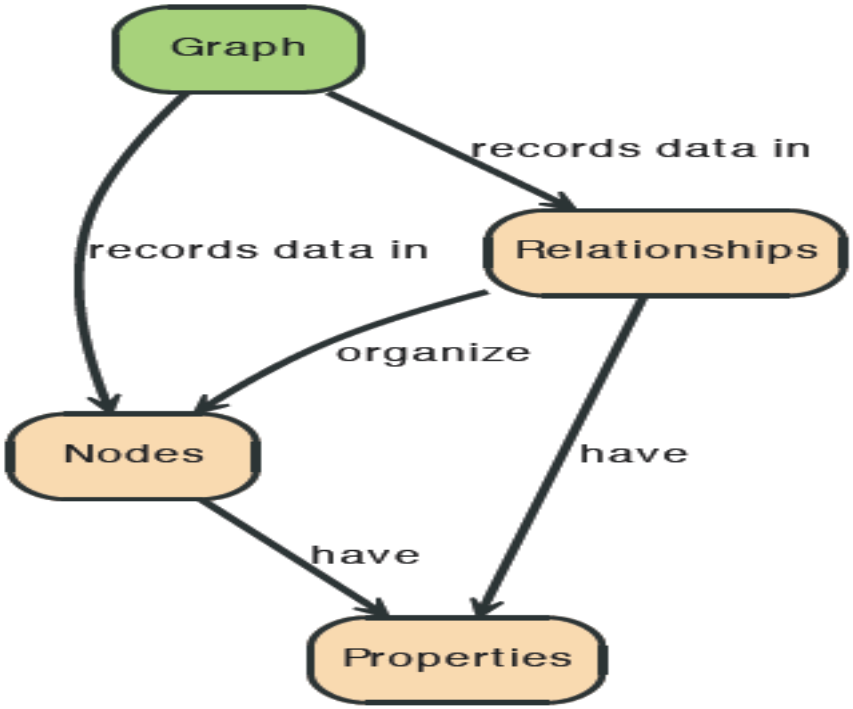
??????????????????

Graph Databases

- A graph database stores data in a graph, the most generic of data structures, capable of elegantly representing any kind of data in a highly accessible way.

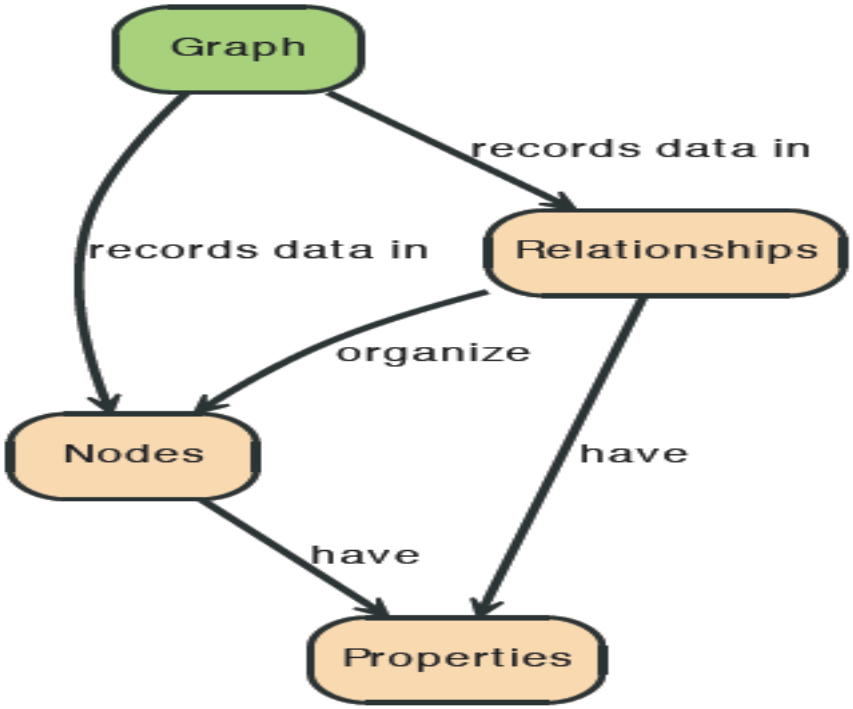
Graphs

- “A Graph —records data in→ Nodes —which have→ Properties”



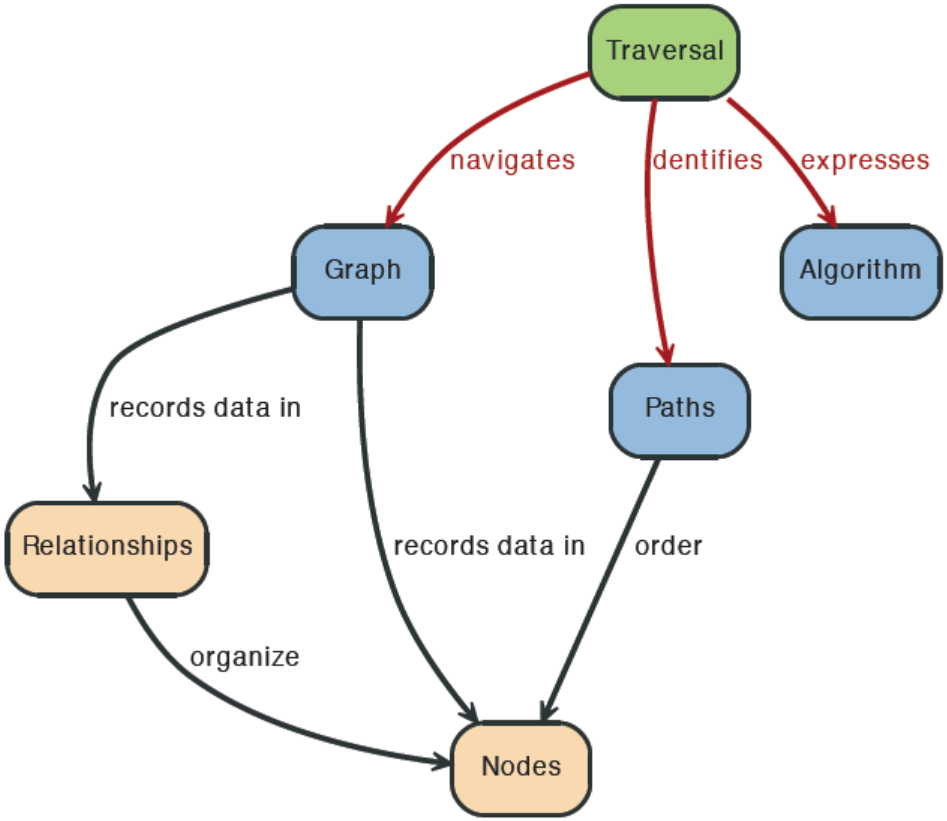
Graphs

- “Nodes —are organized by→ Relationships — which also have→ Properties”



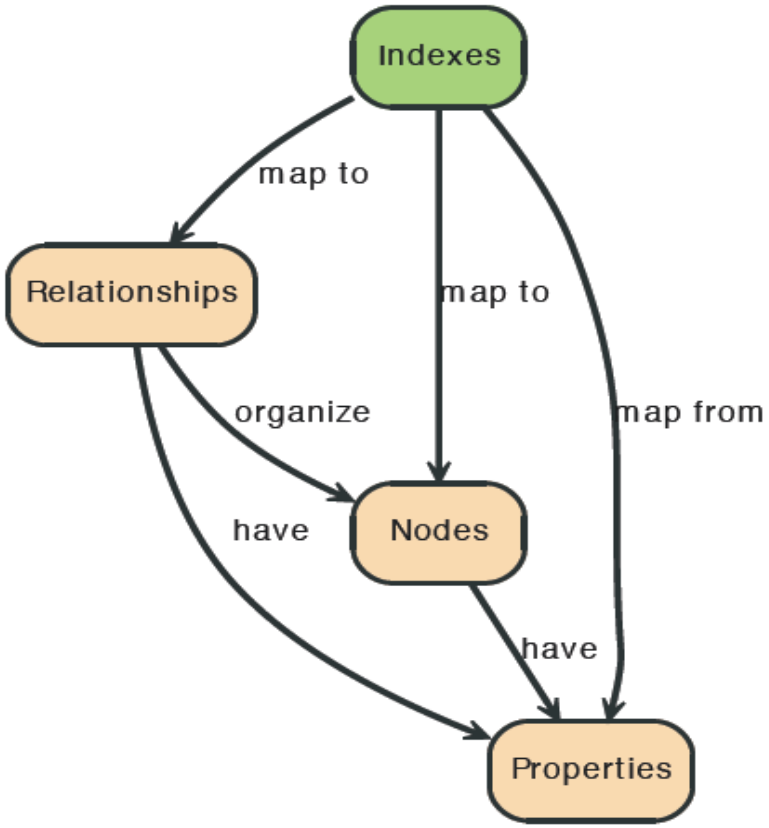
Query a graph with Traversal

- “A Traversal —navigates→ a Graph; it —identifies→ Paths —which order→ Nodes”



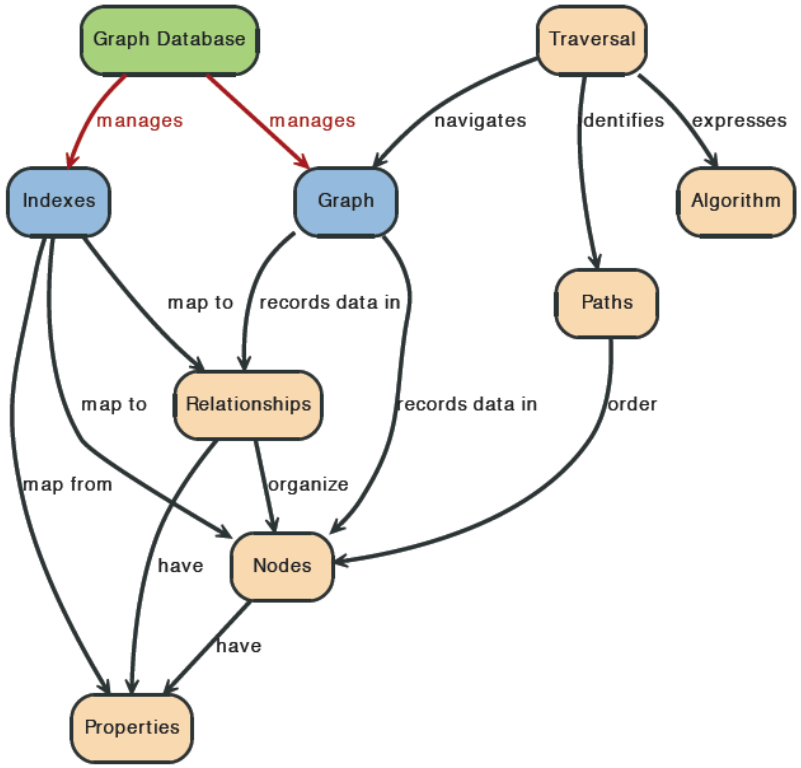
Indexes

- “An Index —maps from→ Properties —to either→ Nodes or Relationships”



Neo4j is a Graph Database

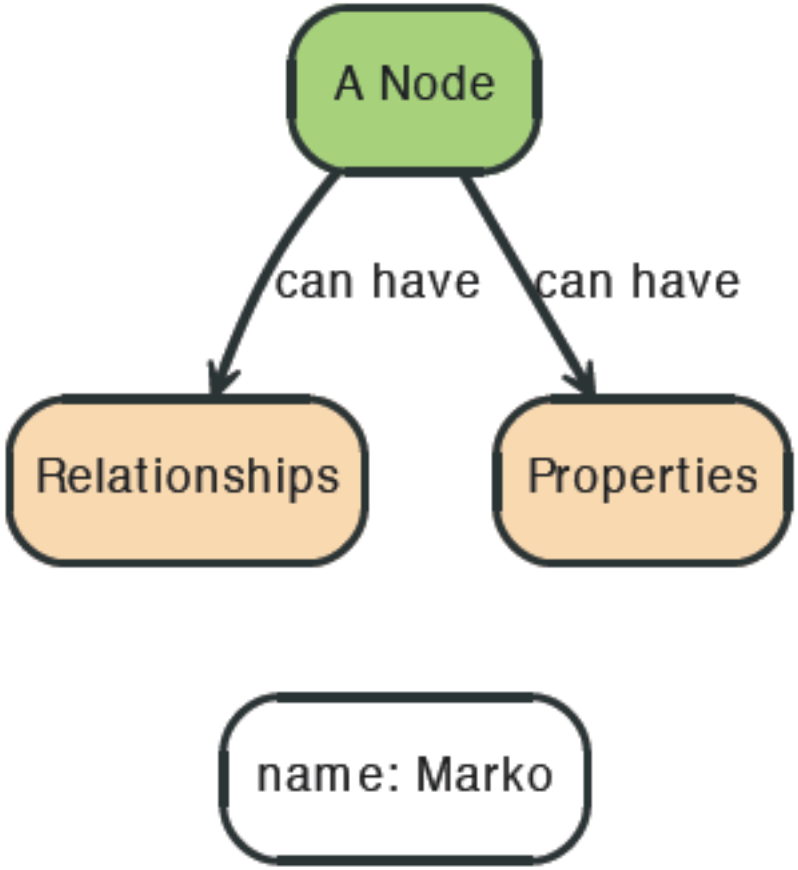
- "A Graph Database —manages a→ Graph and —also manages related→ Indexes"



Neo4j – Hey! This is why I am a Graph Database.

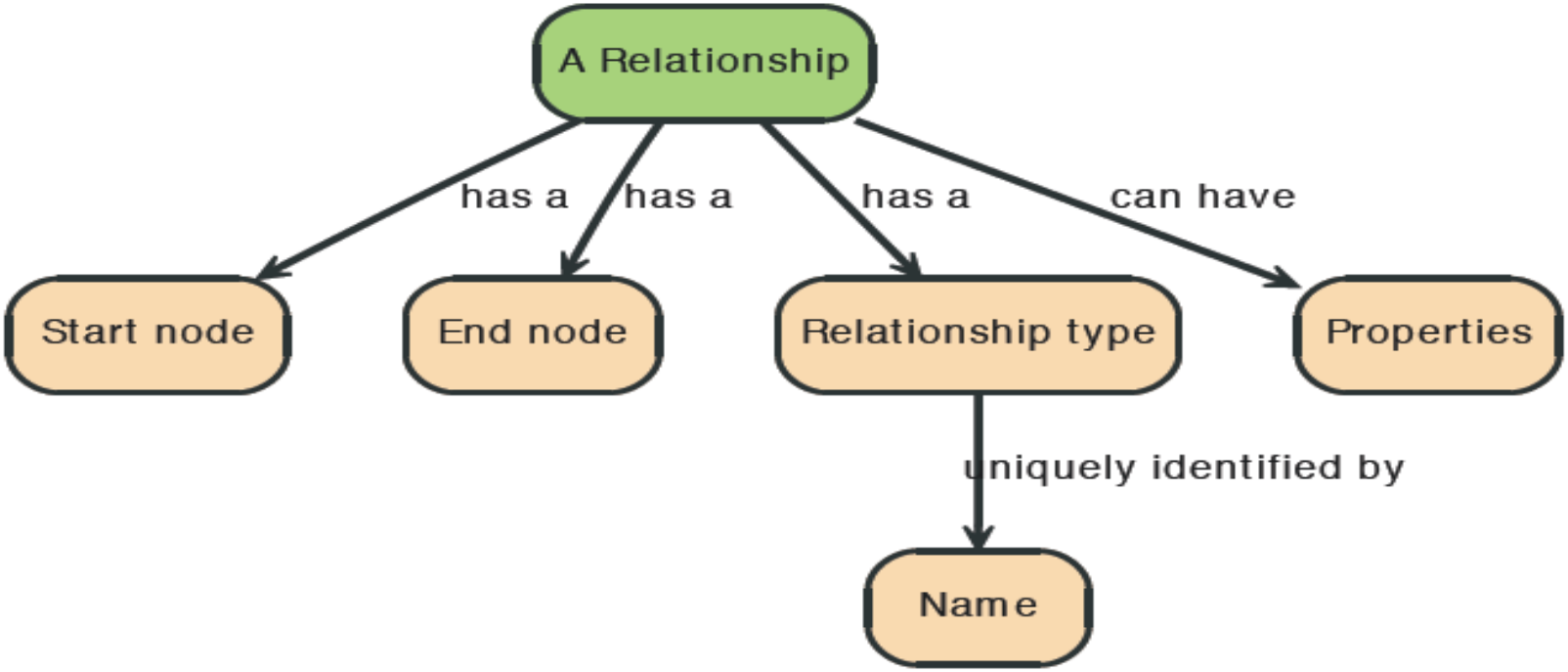
- The fundamental units that form a graph are nodes and relationships.
- In Neo4j, both nodes and relationships can contain properties.
- Nodes are often used to represent entities, but depending on the domain relationships may be used for that purpose as well.

Node in Neo4j

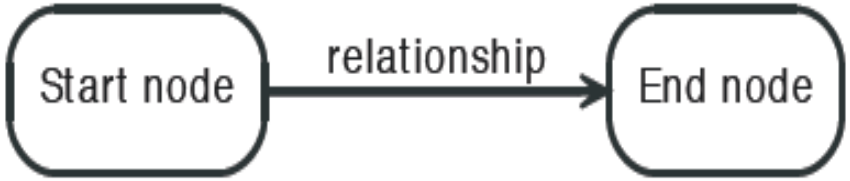


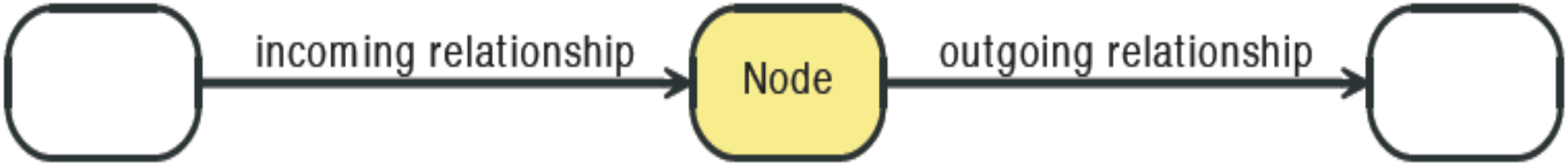
Relationships in Neo4j

- Relationships between nodes are a key part of Neo4j.

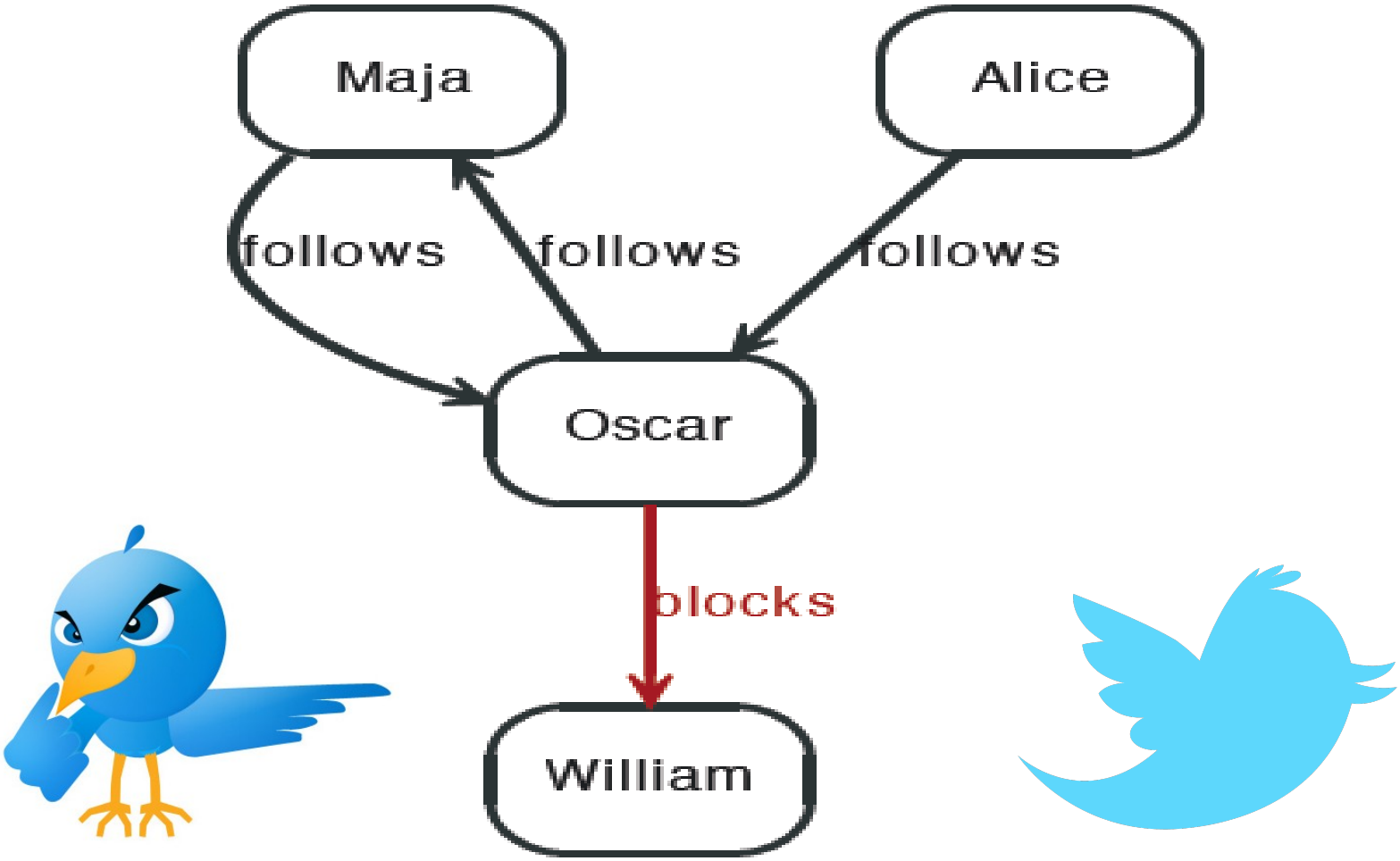


Relationships in Neo4j





Twitter and relationships

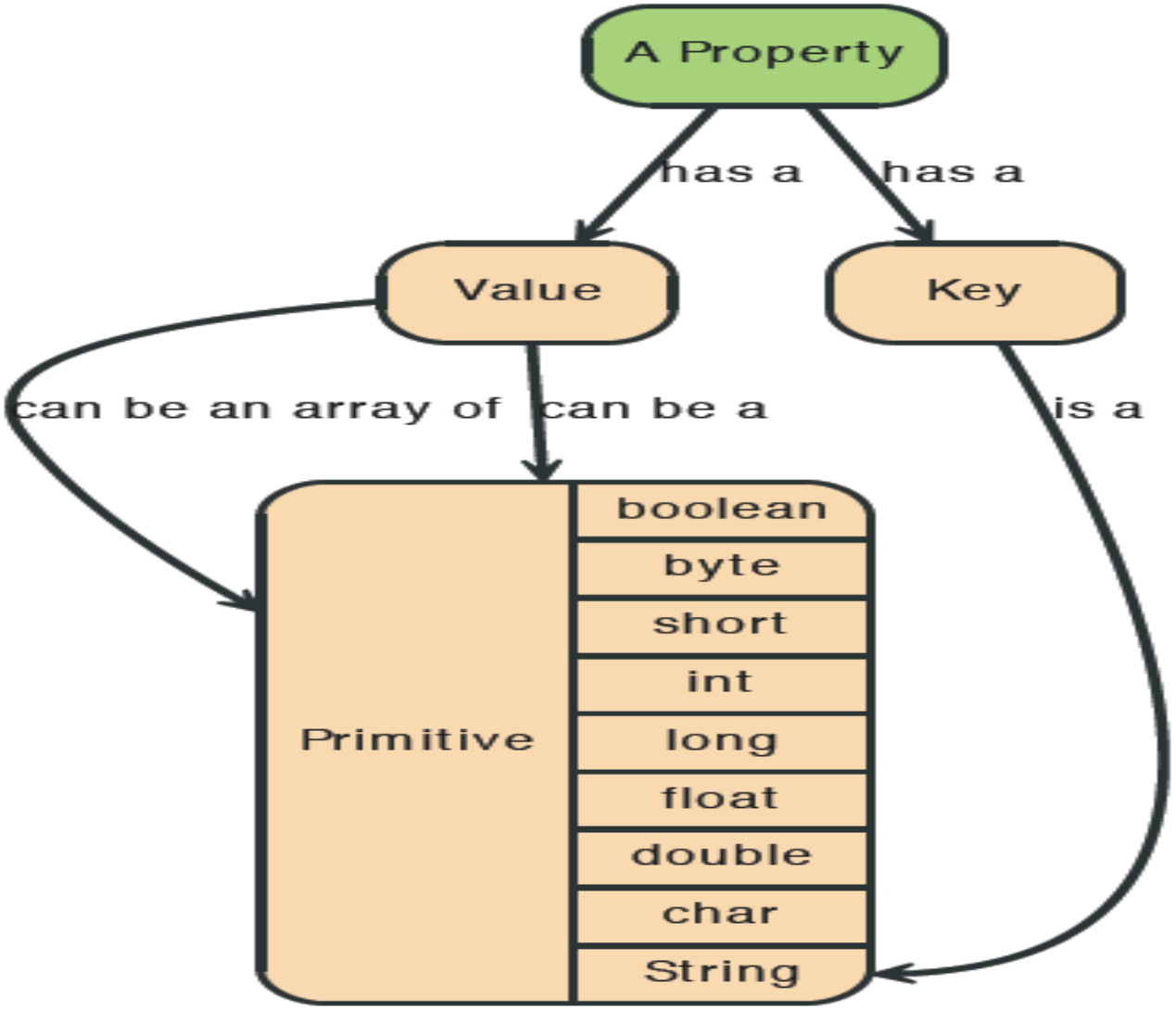


Properties

- 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.

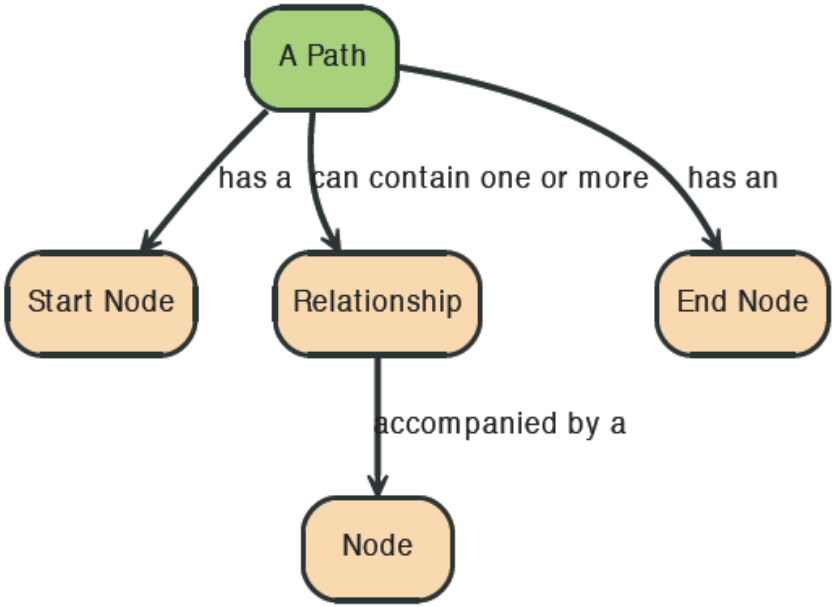
For example String, int and int[] values are valid for properties.

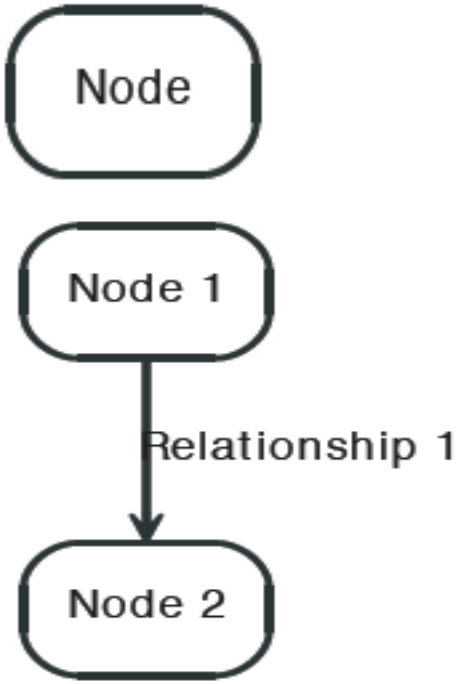
Properties


A Property
has a
has a
Value
Key
can be an array of
can be a
is a
Primitive
boolean
byte
short
int
long
float
double
char
String

Paths in Neo4j

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





Traversals in Neo4j

- Traversing a graph means visiting its nodes, following relationships according to some rules.
- In most cases only a subgraph is visited, as you already know where in the graph the interesting nodes and relationships are found.
- Traversal API
- Depth first and Breadth first

Starting and Stopping

```
graphDb = new EmbeddedGraphDatabase( DB_PATH );
registerShutdownHook( graphDb );
```

```
graphDb.shutdown();
```

```
private static void registerShutdownHook( final GraphDatabaseService graphDb )
{
    // Registers a shutdown hook for the Neo4j instance so that it
    // shuts down nicely when the VM exits (even if you "Ctrl-C" the
    // running example before it's completed)
    Runtime.getRuntime().addShutdownHook( new Thread()
    {
        @Override
        public void run()
        {
            graphDb.shutdown();
        }
    } );
}
```

Preparing the database

```
private static enum RelTypes implements RelationshipType
{
    KNOWS
}
```

```
GraphDatabaseService graphDb;
Node firstNode;
Node secondNode;
Relationship relationship;
```

```
graphDb = new EmbeddedGraphDatabase( DB_PATH );
registerShutdownHook( graphDb );
```

Wrap mutating operations in a transaction.

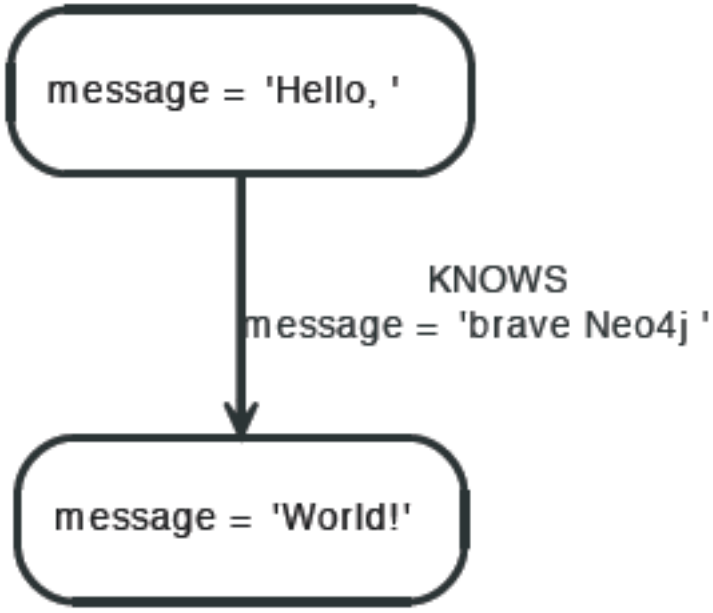
```
Transaction tx = graphDb.beginTx();
```

```
try
{
    // Mutating operations go here
    tx.success();
}
finally
{
    tx.finish();
}
```

Creating a small graph

```
firstNode = graphDb.createNode();
firstNode.setProperty( "message", "Hello, " );
secondNode = graphDb.createNode();
secondNode.setProperty( "message", "World!" );

relationship = firstNode.createRelationshipTo( secondNode, RelTypes.KNOWS );
relationship.setProperty( "message", "brave Neo4j " );
```



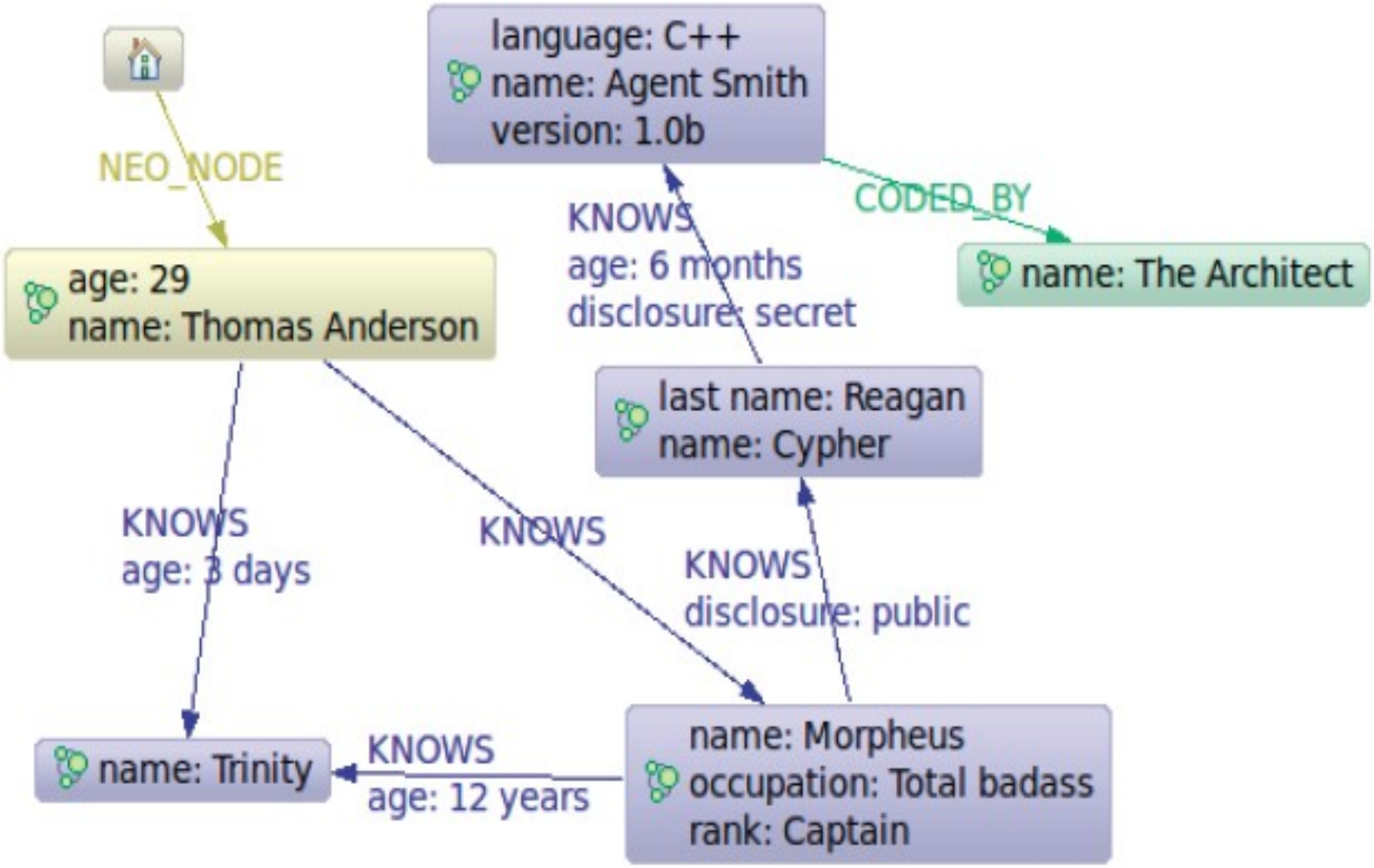
Print the data

```
System.out.print( firstNode.getProperty( "message" ) );
System.out.print( relationship.getProperty( "message" ) );
System.out.print( secondNode.getProperty( "message" ) );
```

Remove the data

```
firstNode.getSingleRelationship( RelTypes.KNOWS, Direction.OUTGOING ).delete();
firstNode.delete();
secondNode.delete();
```

The Matrix Graph Database



Traversing the Graph

```
private static Traverser getFriends( final Node person )
{
    return person.traverse( Order.BREADTH_FIRST,
            StopEvaluator.END_OF_GRAPH,
            ReturnableEvaluator.ALL_BUT_START_NODE, RelTypes.KNOWS,
            Direction.OUTGOING );
}
```

Resources & References

- Neo4j website : http://neo4j.org/
- Neo4j learning resources: http://neo4j.org/resources/
- Videos about Neo4j: http://video.neo4j.org/
- Neo4j tutorial: http://docs.neo4j.org/chunked/snapshot/tutorials.html
- Neo4j Java API documentation: http://api.neo4j.org/current/