

Neo4j: graph-oriented database

Software Languages Team
University of Koblenz-Landau
Ralf Lämmel and Andrei Varanovich

Motivation

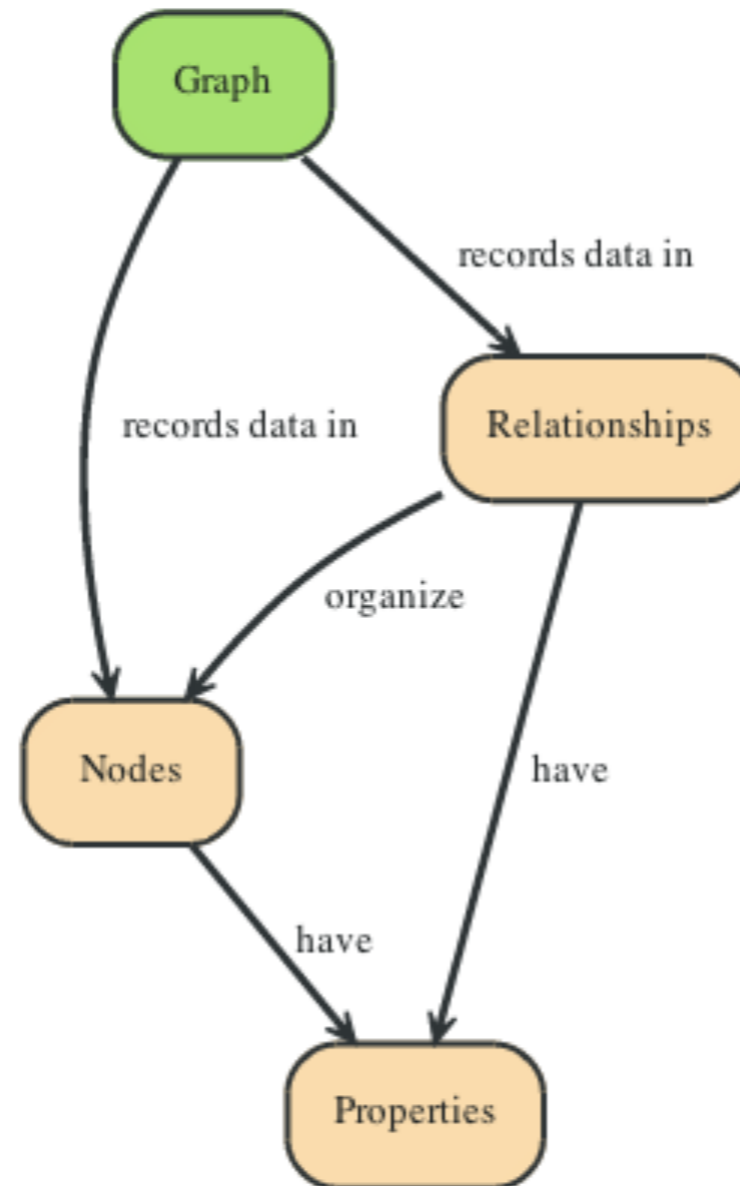
highly-connected data

Use cases

Connected Data (e.g., social graph)

- Routing, Dispatch, and Location-Based Services (delivery services, logistics)
- Recommendation Engines (with product A you want to buy product B)

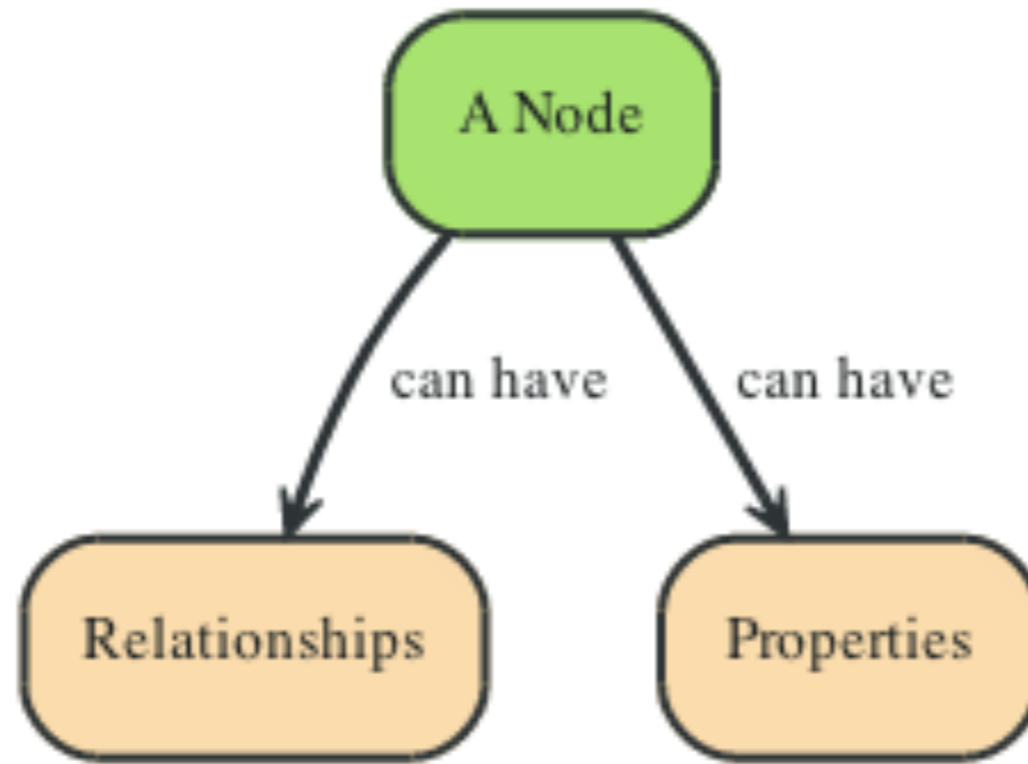
Graph



<http://docs.neo4j.org/chunked/milestone/images/graphdb-GVE.svg>

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

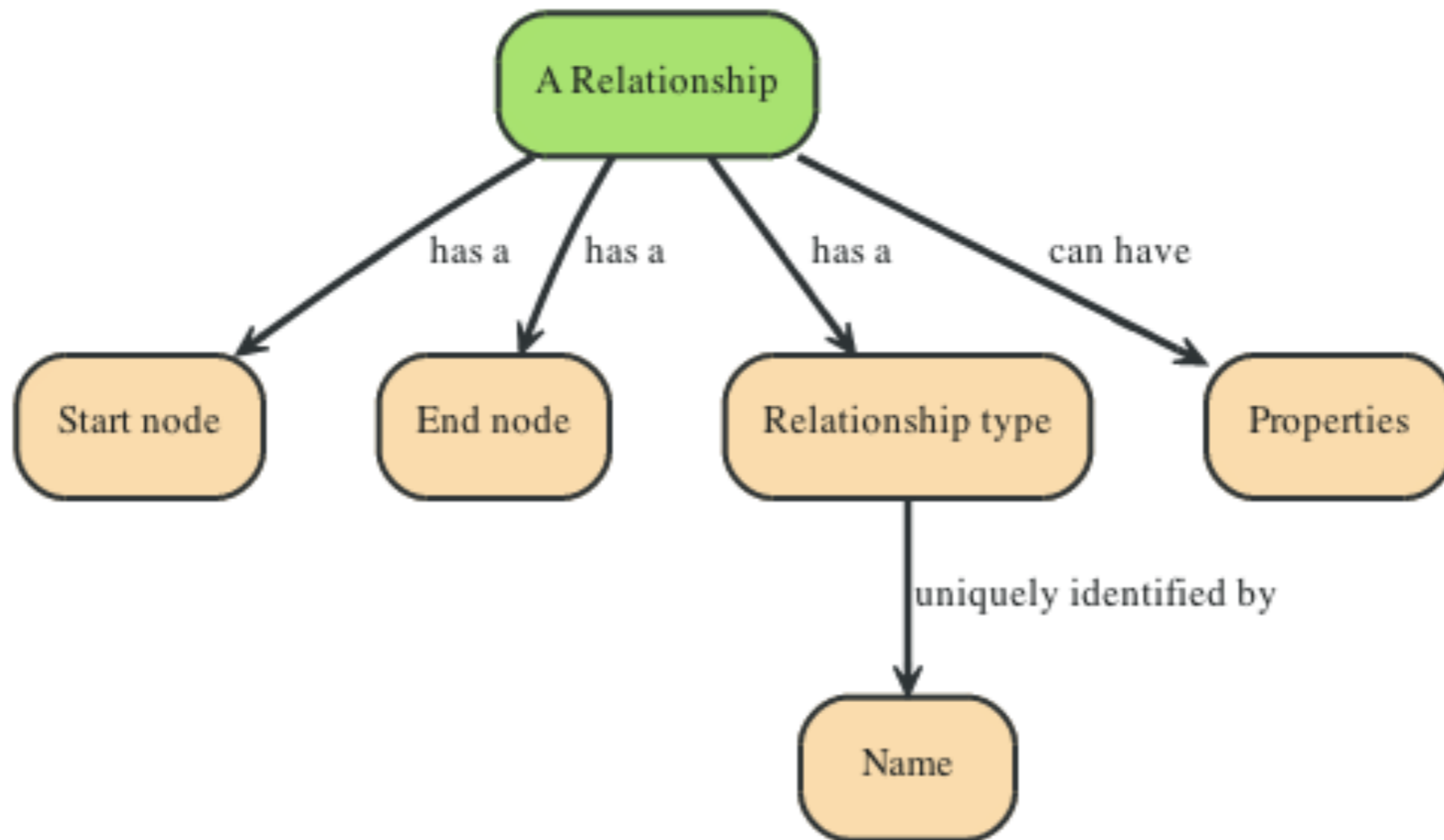
Nodes



<http://docs.neo4j.org/chunked/stable/images/graphdb-nodes-overview.svg>

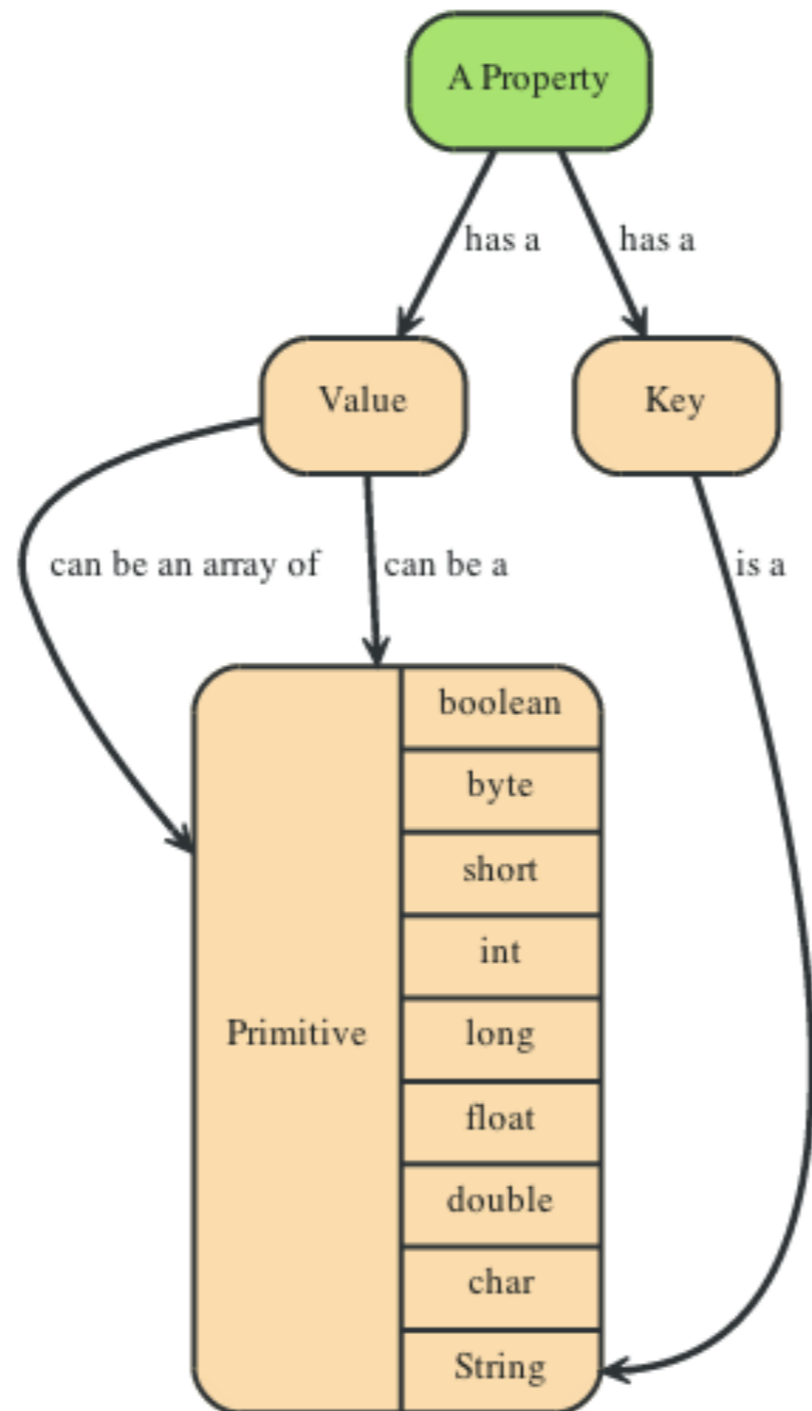
“Nodes —are organized by→ Relationships —which also have→ Properties”

Relationships



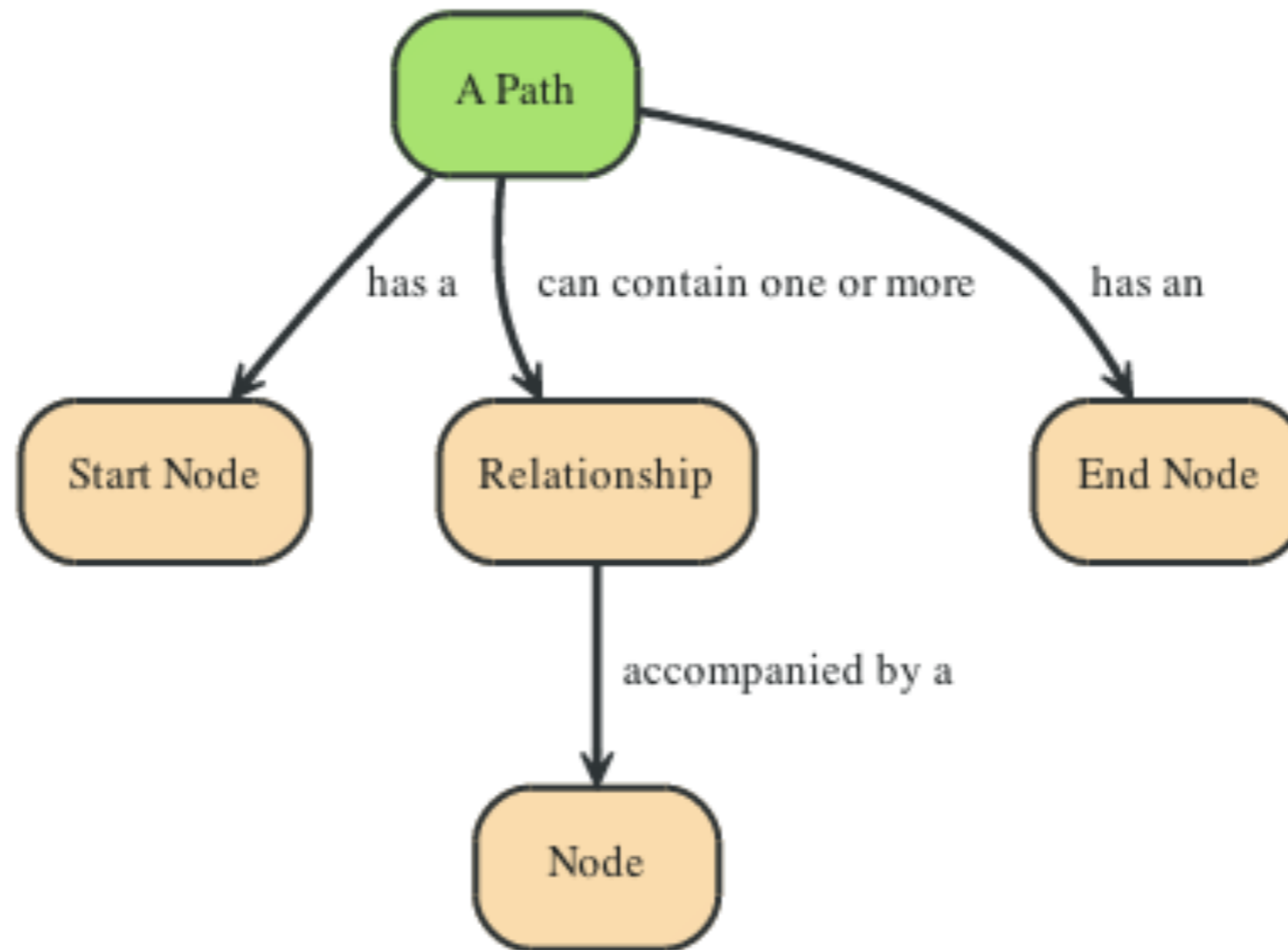
<http://docs.neo4j.org/chunked/stable/images/graphdb-rels-overview.svg>

Properties



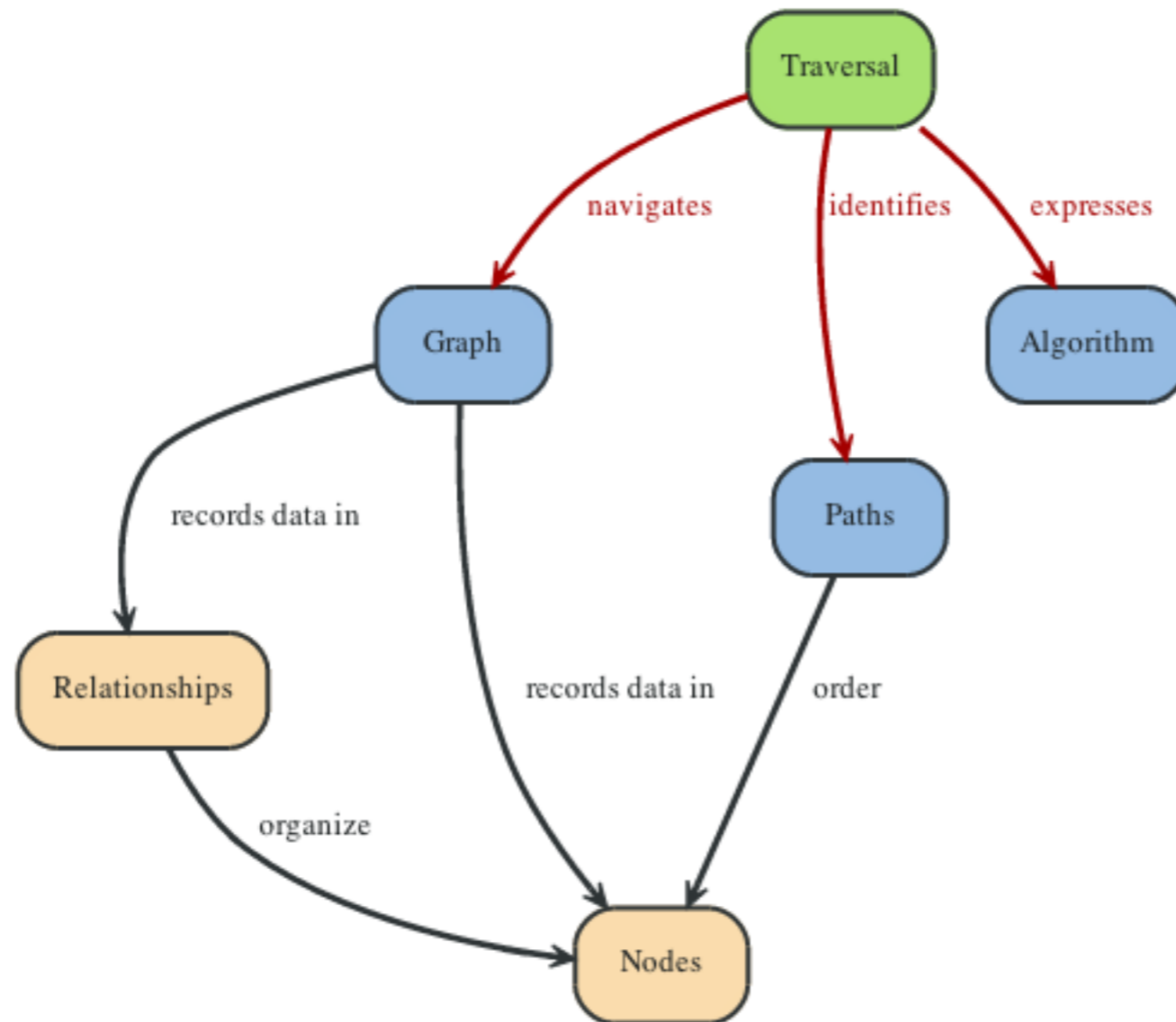
<http://docs.neo4j.org/chunked/stable/images/graphdb-properties.svg>

Paths



<http://docs.neo4j.org/chunked/stable/images/graphdb-path.svg>

Traversal



<http://docs.neo4j.org/chunked/milestone/images/graphdb-traversal.svg>

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

Working with neo4j

- REST
- Web admin tool
- neo4j shell
- Gremlin -- a domain-specific language for graph traversals
- Java, Ruby, etc.

Basic operation: create

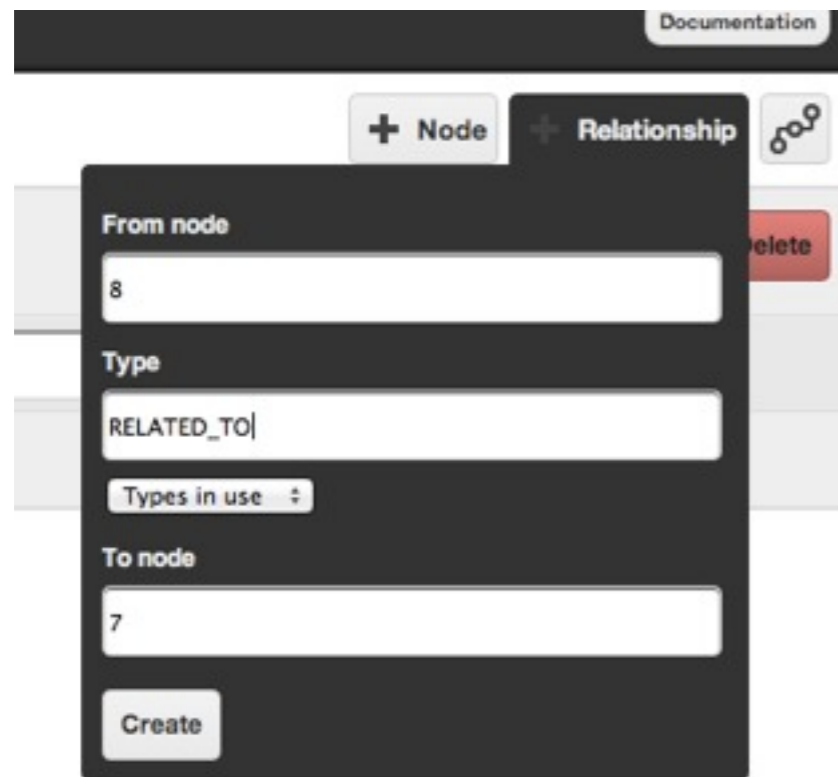


Assign property (shell):

```
cd 8  
set -t String name "Andrei"
```

Uses REST (POST request)

Adding relationships



Documentation

+ Node + Relationship

From node

8

Type

RELATED_TO

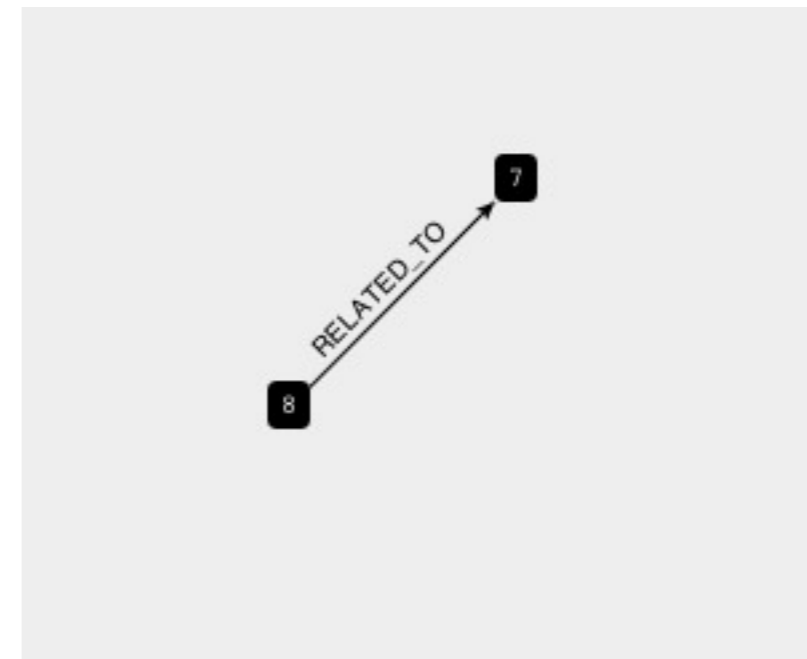
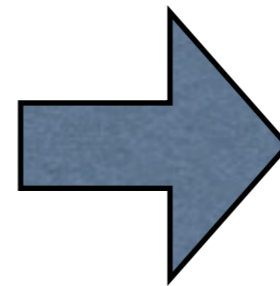
Types in use

To node

7

Create

delete



Basic operation: read

Using gremlin:

```
gremlin> g.V
```

```
==> v[2]
```

```
==> v[3]
```

```
==> v[4]
```

```
==> v[5]
```

```
==> v[6]
```

```
==> v[7]
```

```
==> v[8]
```

All vertices

```
gremlin> g.E
```

```
==> e[2][8-RELATED_TO->7]
```

All edges

```
gremlin> g.v(8).map()
```

```
==> name=Andrei
```

Type of the relationship

Basic operation: update

```
node = g.v(8)  
node.name = "Ralf"  
node.save
```

Find the node
Change the property
Save changes

```
g.v(8).map()  
==> name=Ralf
```

Basic operation: delete

HTTP Request

DELETE <http://debeka.uni-koblenz.de:7474/db/data/node/21>

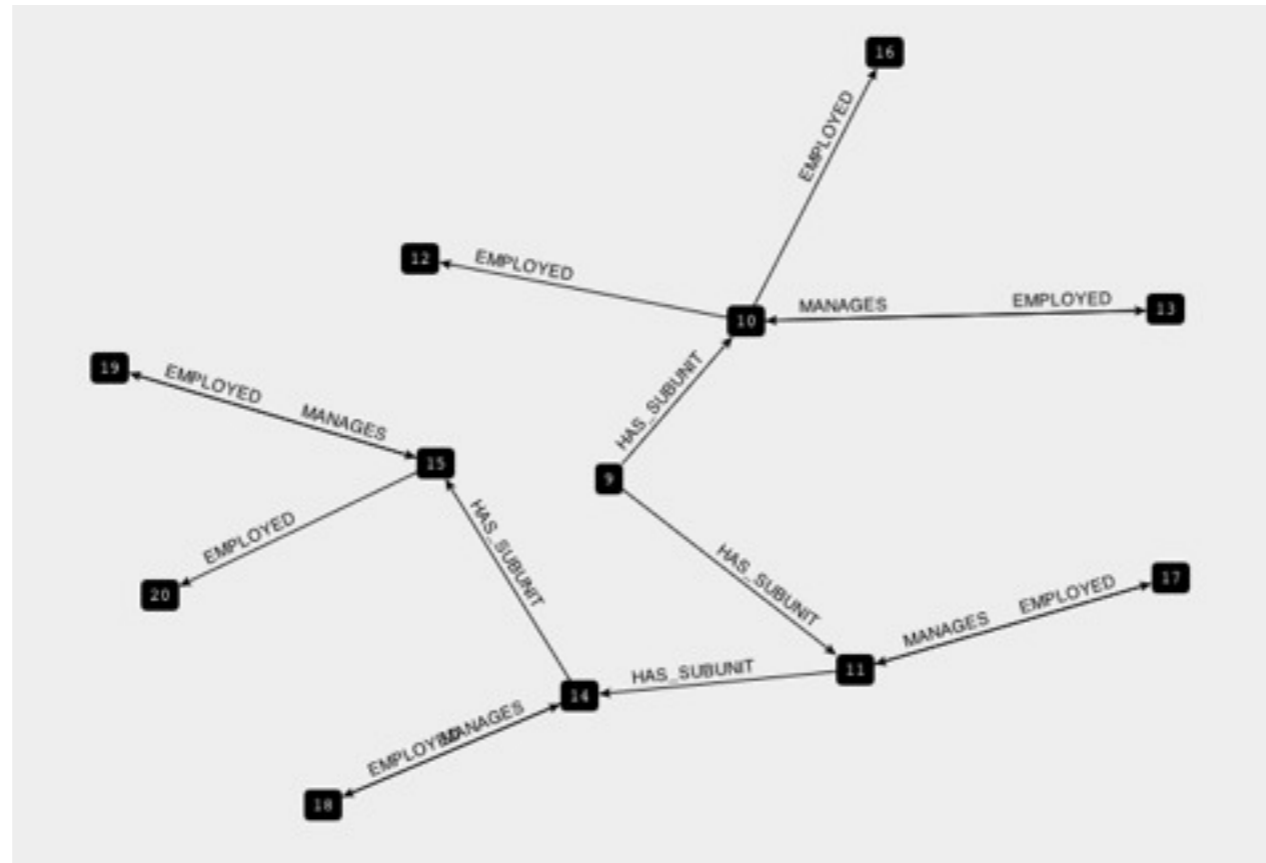


node id

DEMO

IOI implementation:gremlin-neo4j

Company graph



```
g.loadGraphML(https://raw.githubusercontent.com/DerDackel/gremlin-neo4j101companies/master/graphml/meganalysis.graphml)
```

Total salaries

```
g.V.outE('EMPLOYED').  
collect{it.salary}.  
flatten().  
sum()
```



all vertices with
'Employed' relationship

get 'salary' value

to array

sum of the array elements

Cut salaries



outbound relation

```
g.V.outE('EMPLOYED').  
sideEffect{it.salary = (double)it.salary / 2.0}
```

Transactions

```
Transaction transaction = database.beginTx();  
try {  
    Node node = database.createNode();  
    node.setProperty("name", "NoSQL Distilled");  
    node.setProperty("published", "2012");  
    transaction.success();  
} finally {  
    transaction.finish();  
}
```

Supported on the API level

Consistency

- Distributing the nodes on different servers -- not supported.
- Within a single server, data is always consistent (Neo4J is fully ACID-compliant)
- Ensure consistency through transactions.
- In a cluster: a write to the master is eventually synchronized to the slaves, while slaves are always available for read. Writes to slaves are allowed and are immediately synchronized to the master; other slaves will not be synchronized immediately.

Availability

- Replicated slaves: *horizontally scaling read-mostly architecture* that enables the system to handle more read load than a single Neo4j database instance can handle.

Summary

You learned about ...

- principles of graph-oriented databases,
- key properties of neo4j,
- how to use neo4j with web admin console, REST, and Gremlin.

Resources

- neo4j documentation:

<http://docs.neo4j.org/>

- Gremlin wiki:

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