Agenda

(1) Einleitung

(2) neo4j

  (1) neo4j Embedded

  (2) neo4j Server (REST-Interface)

  (3) Cypher Query Language

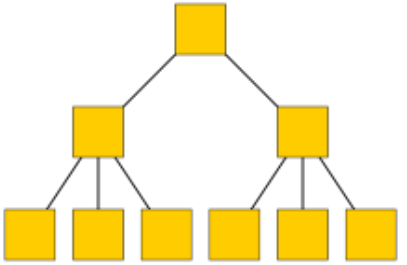
  (4) High Availability Cluster

(3) Beispiel + Demo

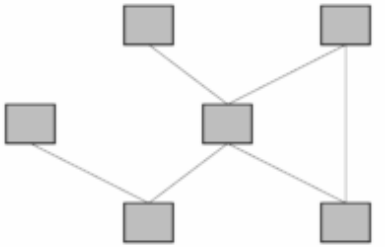
Einleitung

Einleitung: Geschichte

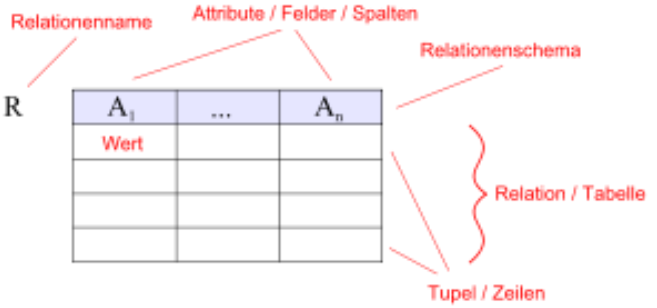
- 1960er: Hierarchische Datenbanken (IBM)



- 1969: Netzwerkdatenbanken (Charles Bachman, CODASYL)



- 1970: Relationale Datenbank (E.F. Codd)



- 2000er: Graphendatenbanken (NoSQL)

Einleitung: Warum Graphendatenbanken?

User-generiert

- Hohe Nutzerzahlen
- Hohes Datenaufkommen

- RDBs skalieren schlecht

User-zentriert

- „Freunde“
- Sozialer Graph

- RDBs sind umständlich

Einleitung: Typische Probleme

- „Gib mir alle Personen, die mit meinen Freunden befreundet sind, mit denen ich noch nicht befreundet bin“
- „Welches sind die TOP10 Bücher, die Leute mögen, die Buch x auch mögen?“
- „Welche Orte sind auf kürzester Strecke nicht weiter als x von meiner aktuellen Position entfernt?“

neo4j

neo4j

	Neo4j the graph database
Hersteller:	Neo Technology
Erscheinungsjahr:	2007 (Version 1.0: 2010)
Aktuelle Version:	1.7.1
Programmiersprache	Java
Website:	http://neo4j.org/
Lizenz	GPL3/AGPL3

neo4j: Anwender

Adobe

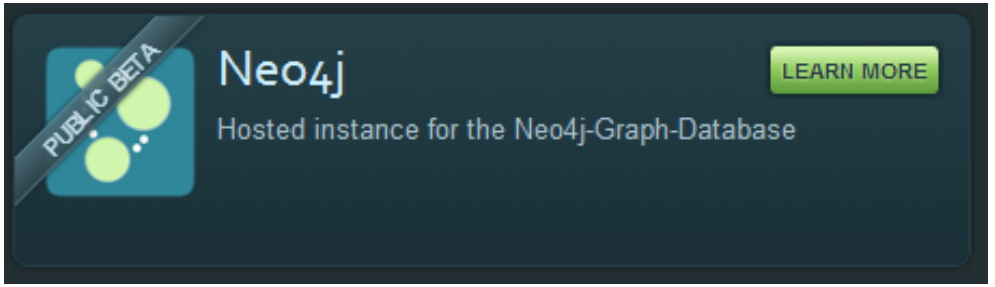
CISCO

Deutsche Telekom

amazon.com

"For anything with multiple relationships, multiple connections, Neo4j absolutely ROCKS!"
--Werner Vogels, CTO Amazon

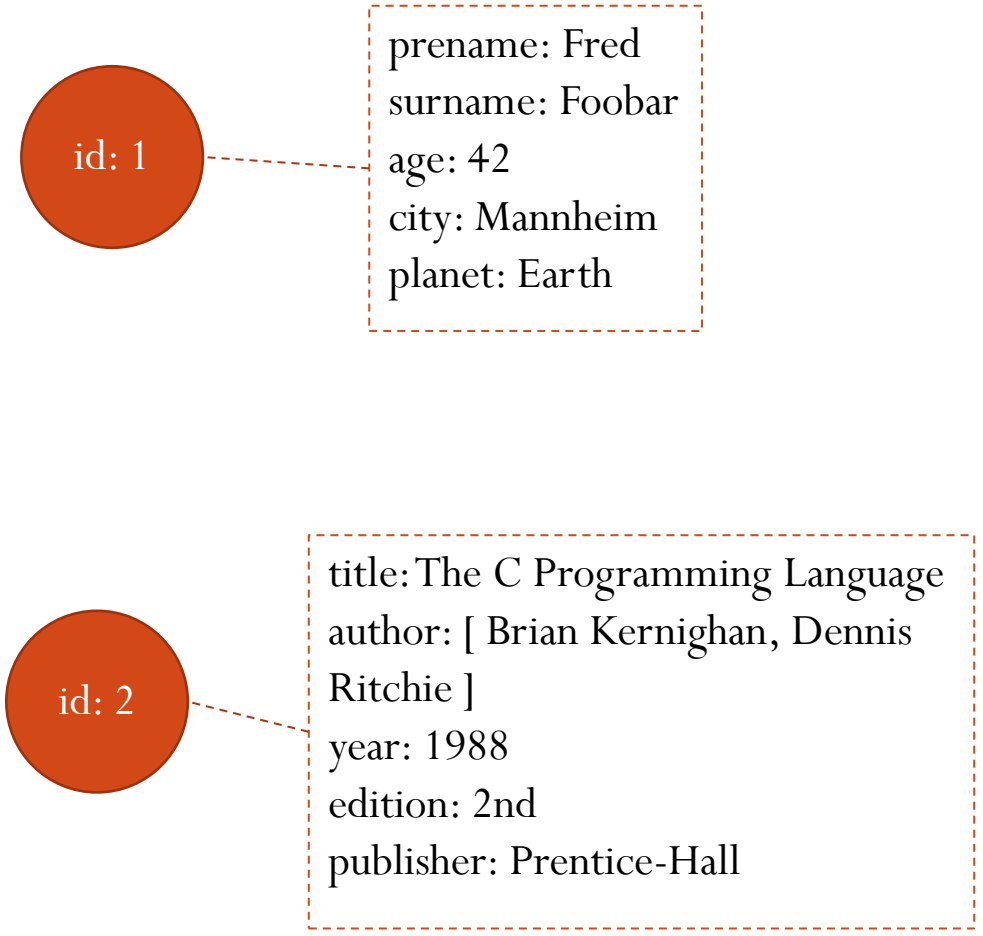
heroku



neo4j: Features

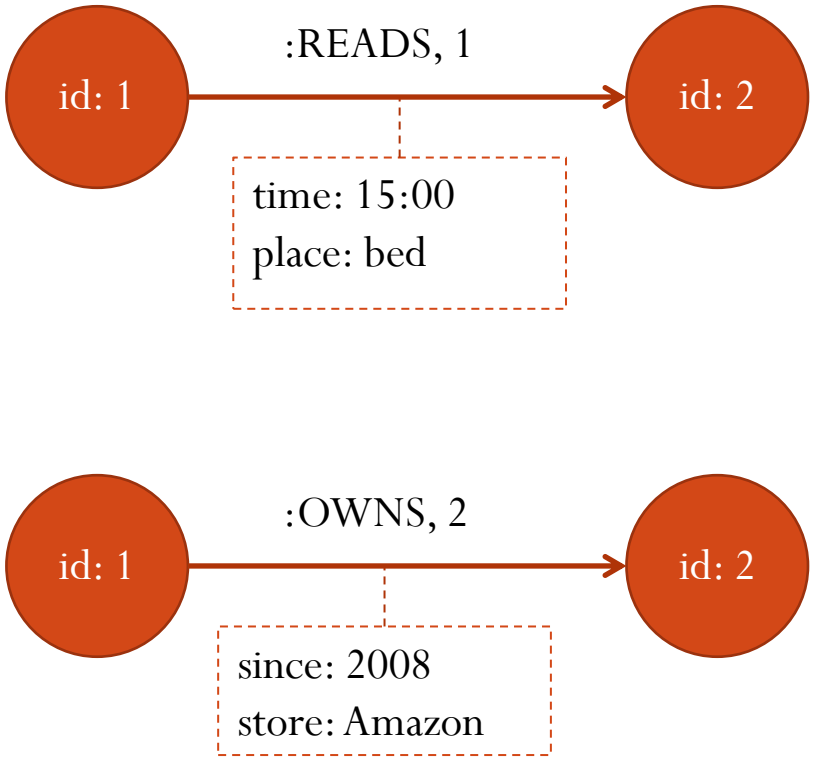
- Transaktionsfähig (ACID)
- Embedded und Server
- REST-Interface
- Cypher Query Language
- High Availability Cluster

neo4j: Knoten (Node)



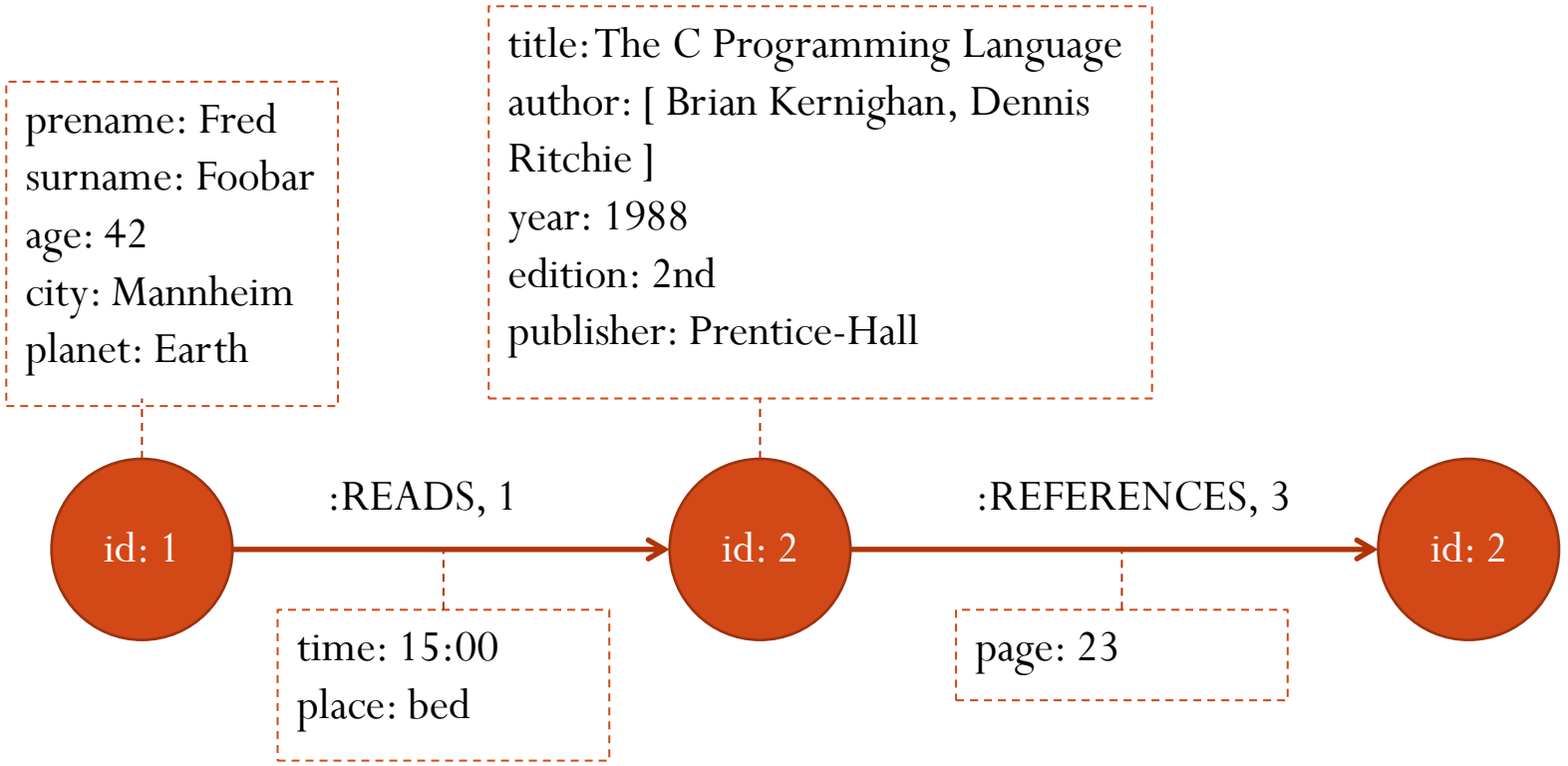
- Fortlaufende ID
- Optional: Eigenschaften
  - boolean
  - byte
  - short
  - int
  - long
  - float
  - double
  - char
  - String
  - array of …

neo4j: Kanten (Relationship)

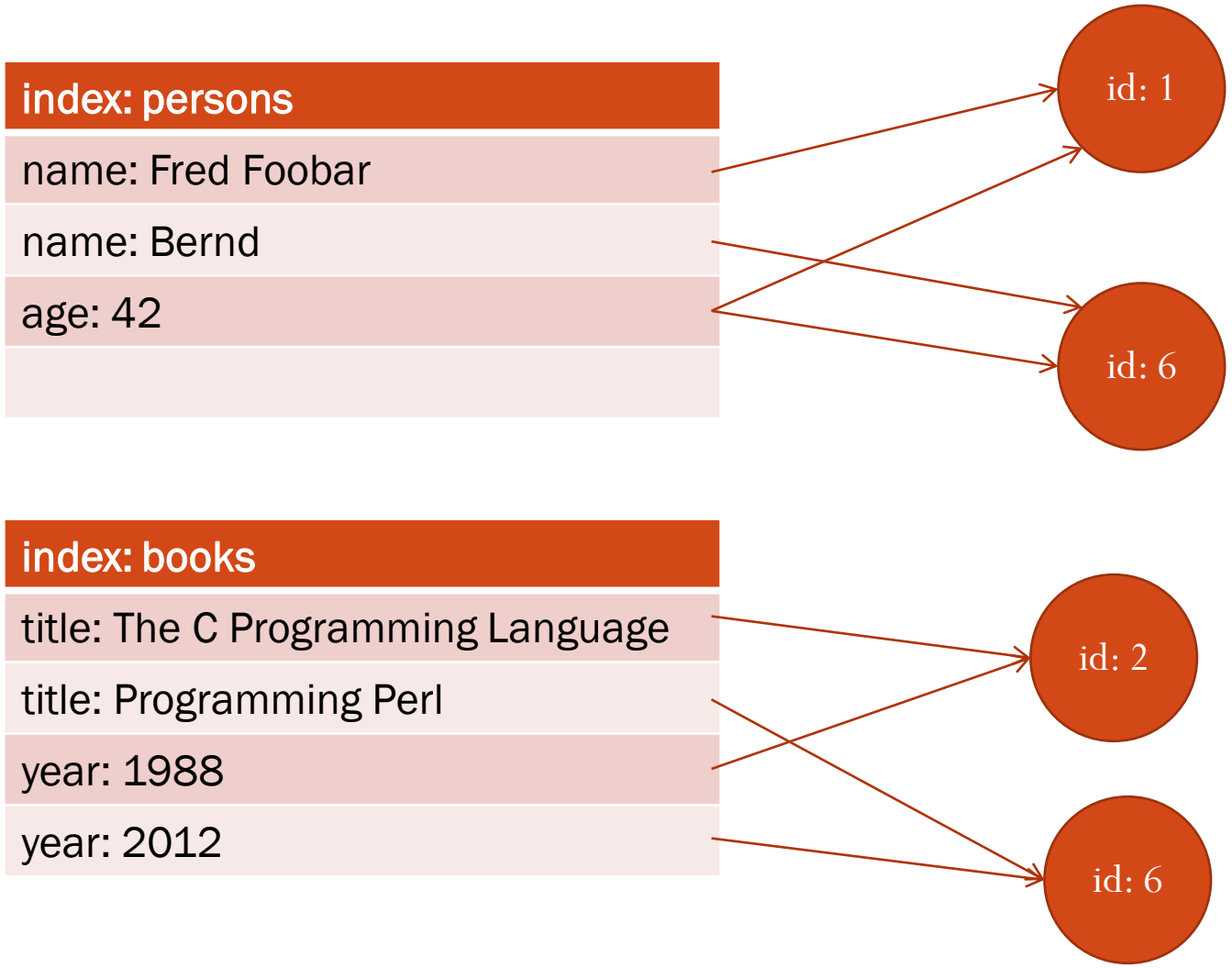


- Fortlaufende ID
- Start-Knoten
- End-Knoten
- Typ
- Richtung
- Optional: Eigenschaften
  - boolean
  - byte
  - short
  - int
  - long
  - float
  - double
  - char
  - String
  - array of …

neo4j: Pfade (Paths)

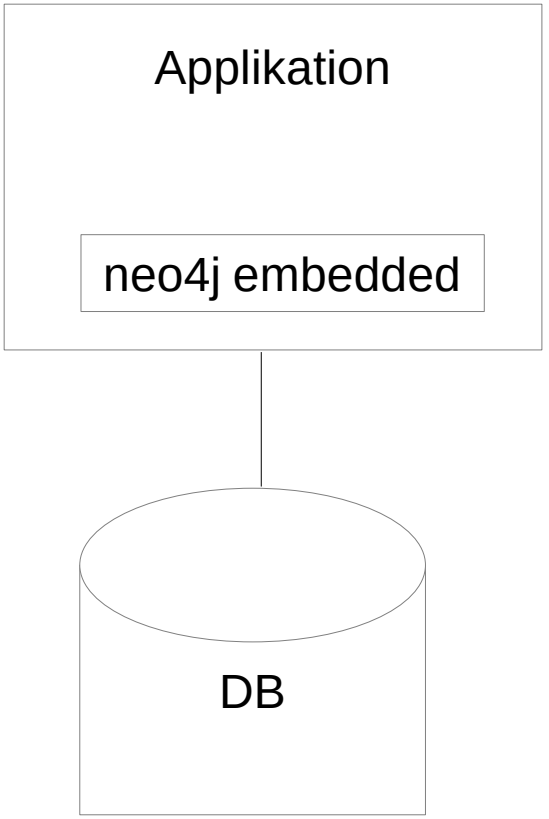


neo4j: Indizes

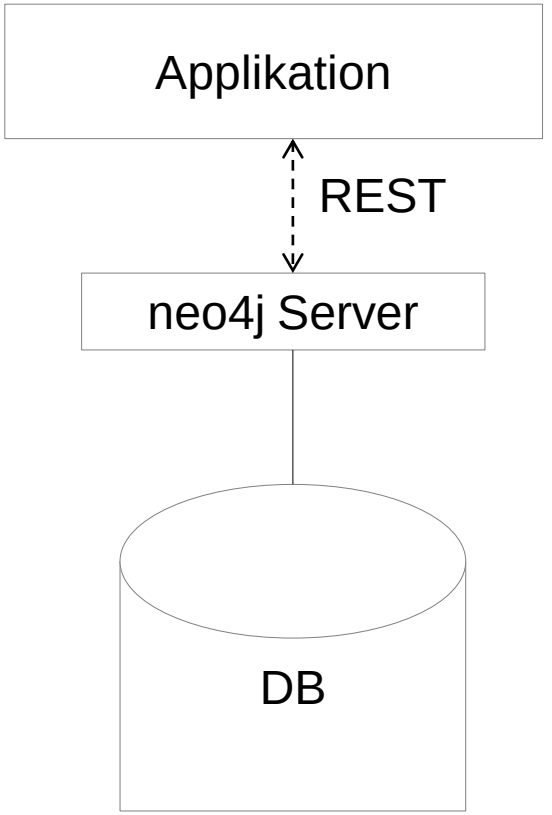


neoj4: Embedded vs. Server

Embedded



Server



neoj4: Embedded

- Neue Graphendatenbank öffnen:

```
GraphDatabaseService graphDb =
   new GraphDatabaseFactory().
   newEmbeddedDatabase("target/dir");
```

neoj4: Embedded

- Neuen Knoten erstellen:

```
Node fred = graphDb.createNode();
fred.setProperty("name", "Fred");
fred.setProperty("age", 42);
```

- Neuen Knoten indizieren:

```
IndexManager index = graphDb.index();
Index<Node> persons = index.forNodes( "persons" );
persons.add(fred, "name", fred.getProperty("name"));
```

neoj4: Embedded

- Kantentyp definieren:

```
public enum RelTypes implements RelationshipType {
   FRIENDS,
   ENEMIES
}
```

- Neue Kante erstellen:

```
Relationship rel =
   fred.createRelationshipTo(bernd, RelTypes.FRIENDS);
rel.setProperty("since", "2008");
```

neoj4: Embedded

Traversal

Durchlaufen des Graphen nach bestimmten Kriterien

- *Starting node:* Wo soll ich die Suche starten?
- *Expander:* Welchen Kanten soll ich folgen?
- *Order*: Tiefensuche oder Breitensuche?
- *Uniqueness*: Knoten mehrfach besuchen?
- *Evaluator*: Wann soll die Suche abbrechen?

neoj4: Embedded

Beispiel

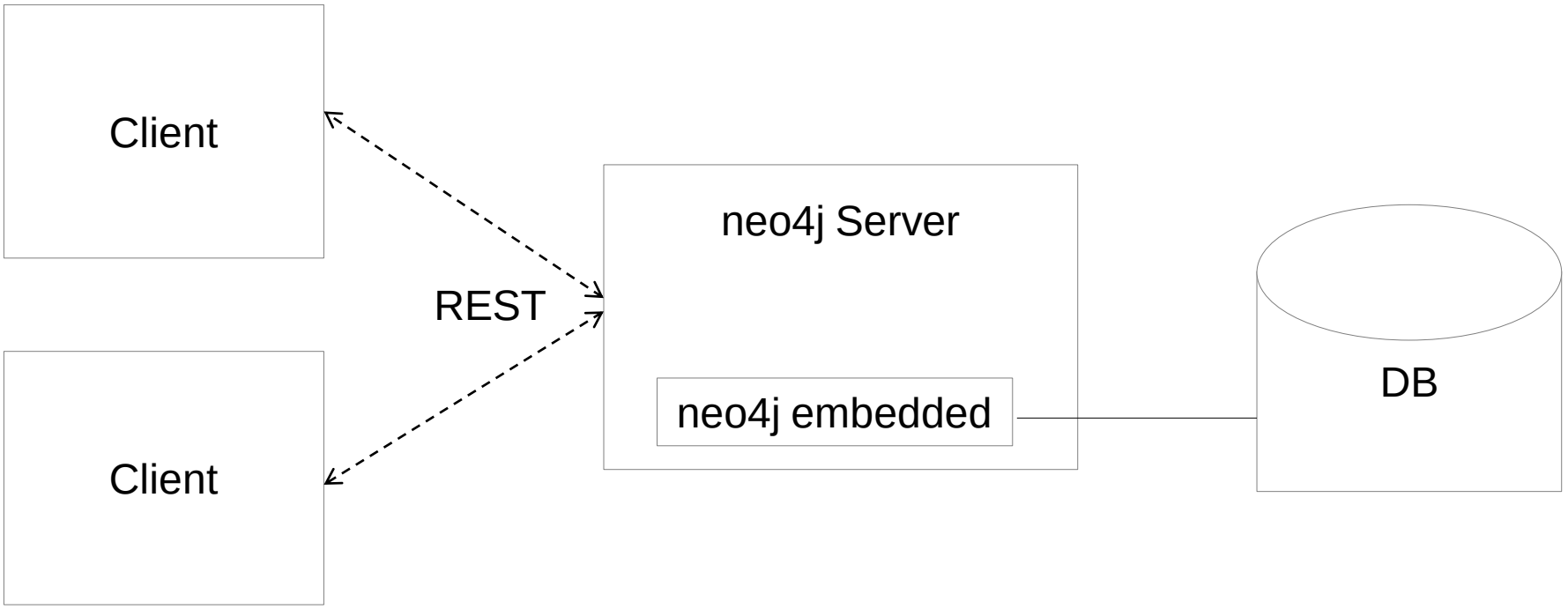
"Gib mir alle Freunde der Freunde von Fred."

```
Iterator<Path> paths =

Traversal.description()
.expand(new OrderedByTypeExpander().add(RelType.FRIENDS))
.order(CommonBranchOrdering.PREORDER_BREADTH_FIRST)
.uniqueness(Uniqueness.NODE_GLOBAL)
.evaluator(Evaluators.atDepth(2))
.traverse(fred)

.iterator();
```

neoj4: Server



neoj4: Server

REST-API Client Libraries

Sprache	Client Library
Java	Java-Rest-Binding
.NET	Neo4jClient
Python	py2Neo
PHP	Neo4jPHP
Ruby	neography
JavaScript (node.js)	Node-neo4j
Closure	Neocons

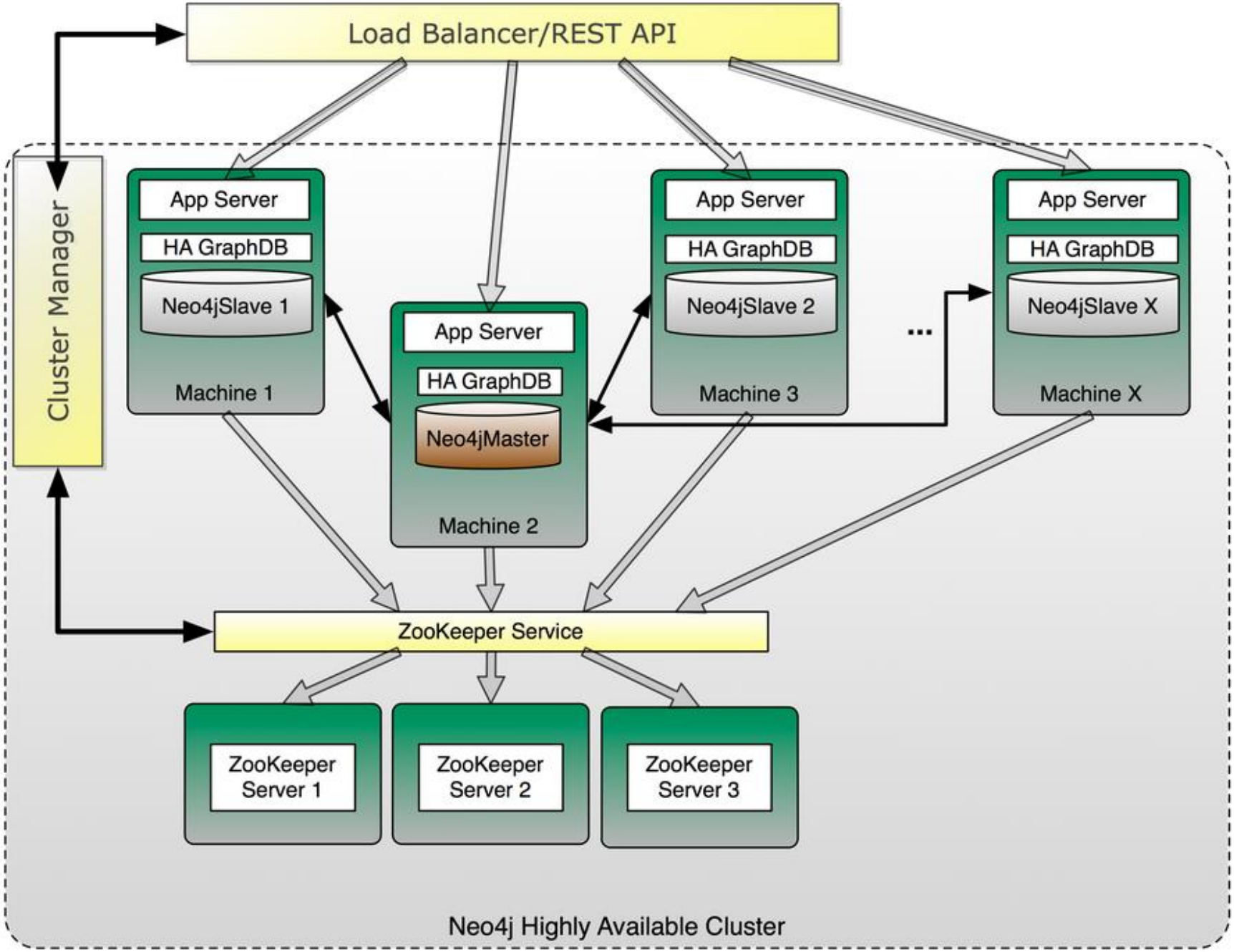
neoj4: Cypher Query Language

RDBMS - SQL

- SELECT
- INSERT
- UPDATE
- DELETE

neo4j - Cypher

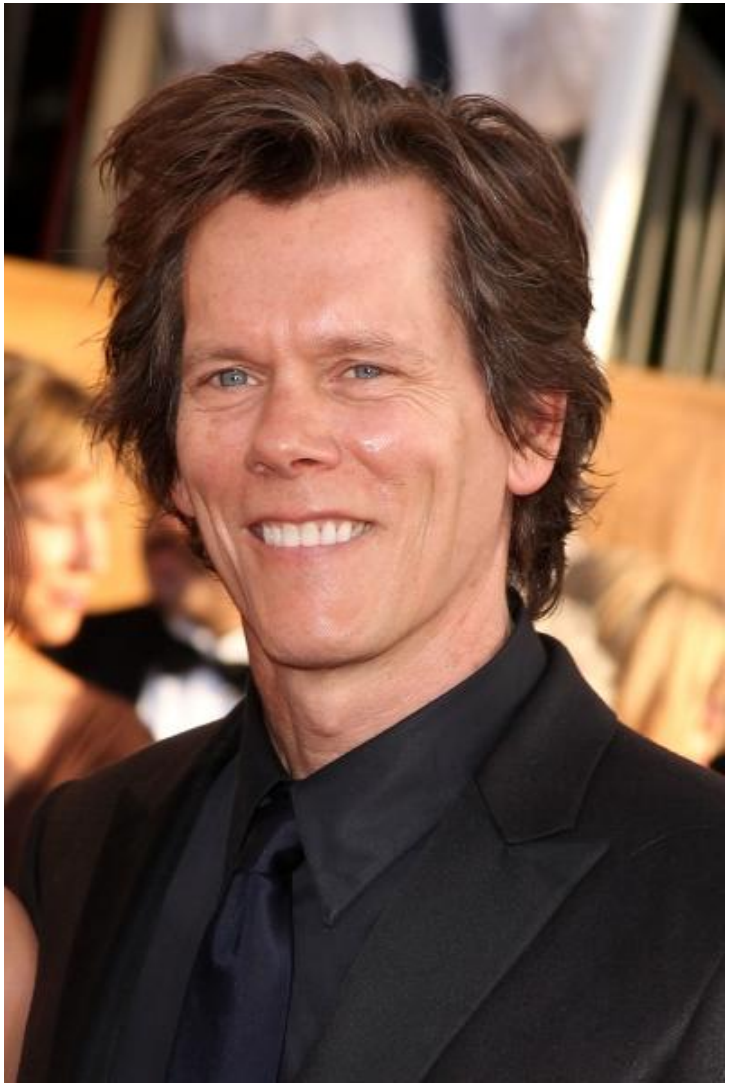
- START … RETURN
- CREATE
- SET
- DELETE


Load Balancer/REST API
Cluster Manager
App Server
HA GraphDB
Neo4jSlave 1
Machine 1
App Server
HA GraphDB
Neo4jMaster
Machine 2
App Server
HA GraphDB
Neo4jSlave 2
Machine 3
...
App Server
HA GraphDB
Neo4jSlave X
Machine X
ZooKeeper Service
ZooKeeper Server 1
ZooKeeper Server 2
ZooKeeper Server 3
Neo4j Highly Available Cluster

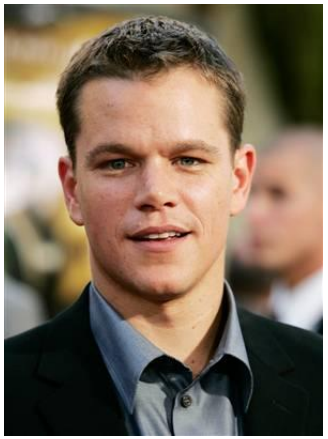
Beispiel + Demo

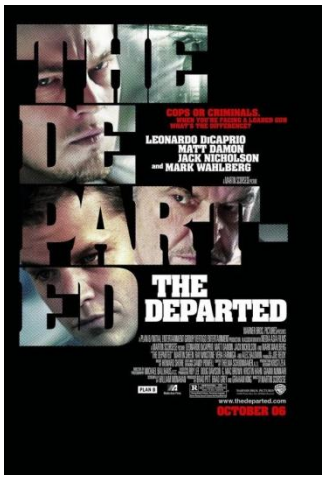
Beispiel + Demo: Problemstellung

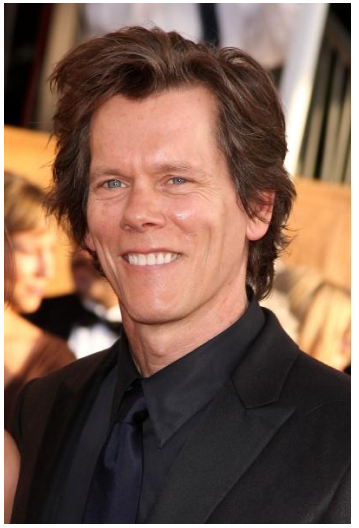
Six Degrees of ~~Separation~~ Kevin Bacon



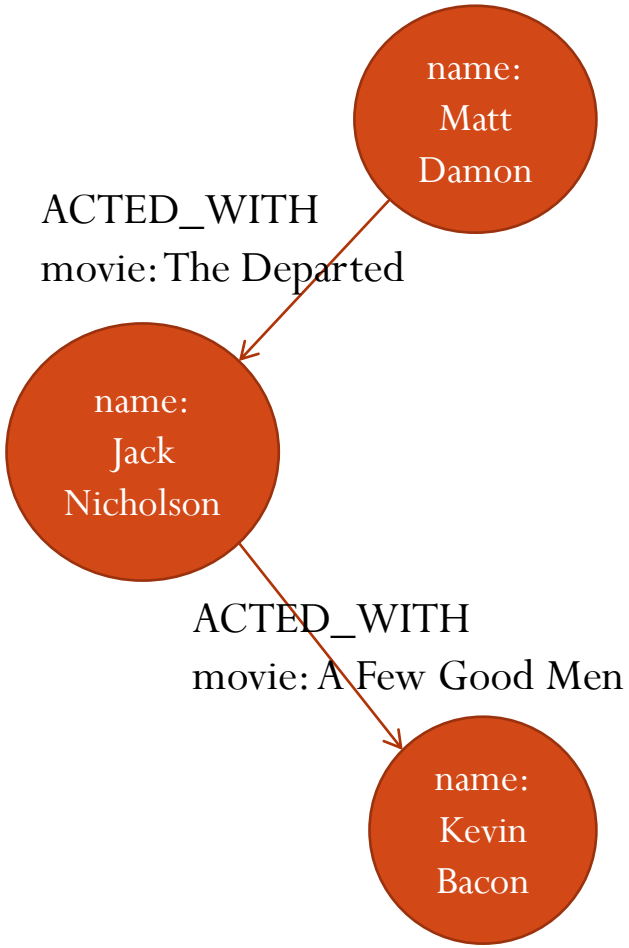
Beispiel + Demo: Problemstellung

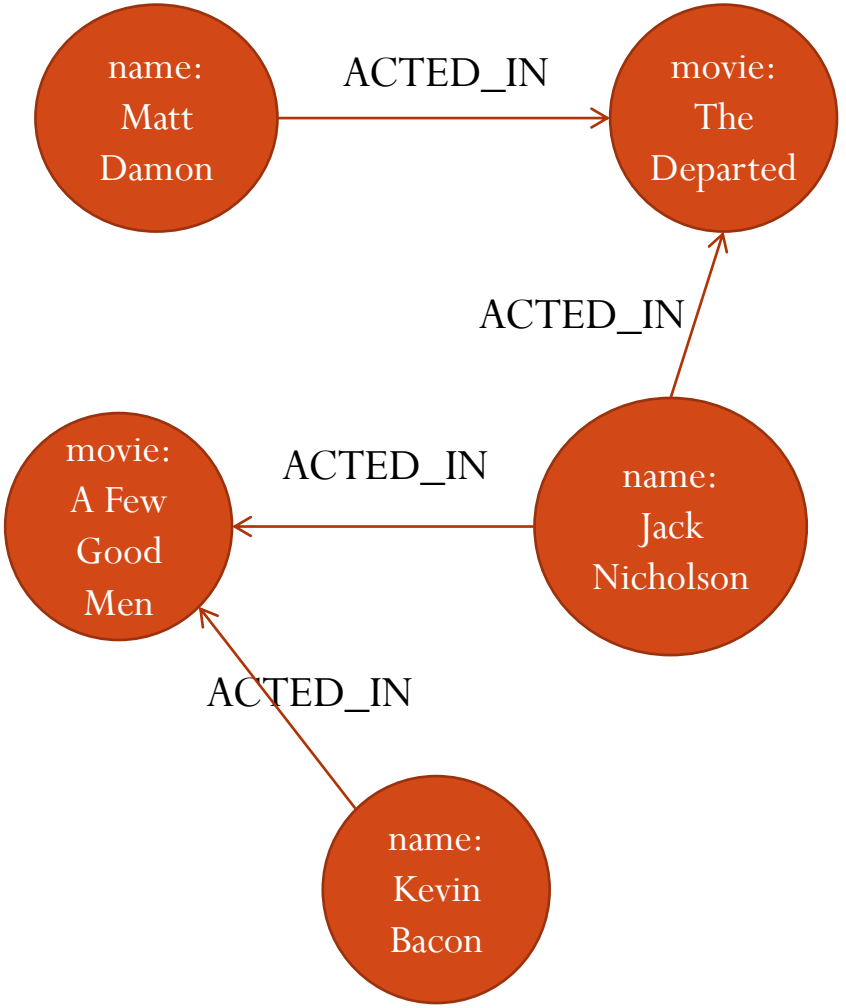






Beispiel + Demo: Datenmodell





Beispiel + Demo: Daten


```
Bacon, Ken (III)  "Weird, True & Freaky" (2008) {Bizarre
Bacon, Kenneth L. "Weird, True & Freaky" (2008) {Swallowe
Bacon, Kevin (I)  15th Annual Critics' Choice Movie Awards (2
                  17th Annual Screen Actors Guild Awards (201
                  18th Annual Screen Actors Guild Awards (201
                  1997 VH1 Fashion Awards (1997) (TV)  [Himse
                  1998 Blockbuster Entertainment Awards (1998
                  19th Annual GLAAD Media Awards (2008) (TV)
                  2000 Blockbuster Entertainment Awards (2000
                  8 (2012) (V)  [Charles Cooper]
                  A Few Good Men (1992)  [Capt. Jack Ross]  <
                  A Few Good Men: From Stage to Screen (2001)
                  A Little Vicious (1991)  [Narrator]
                  A Look Behind the Scenes: Super (2011)  [Hi
                  AFI's 100 Years... 100 Stars: America's Gre
```

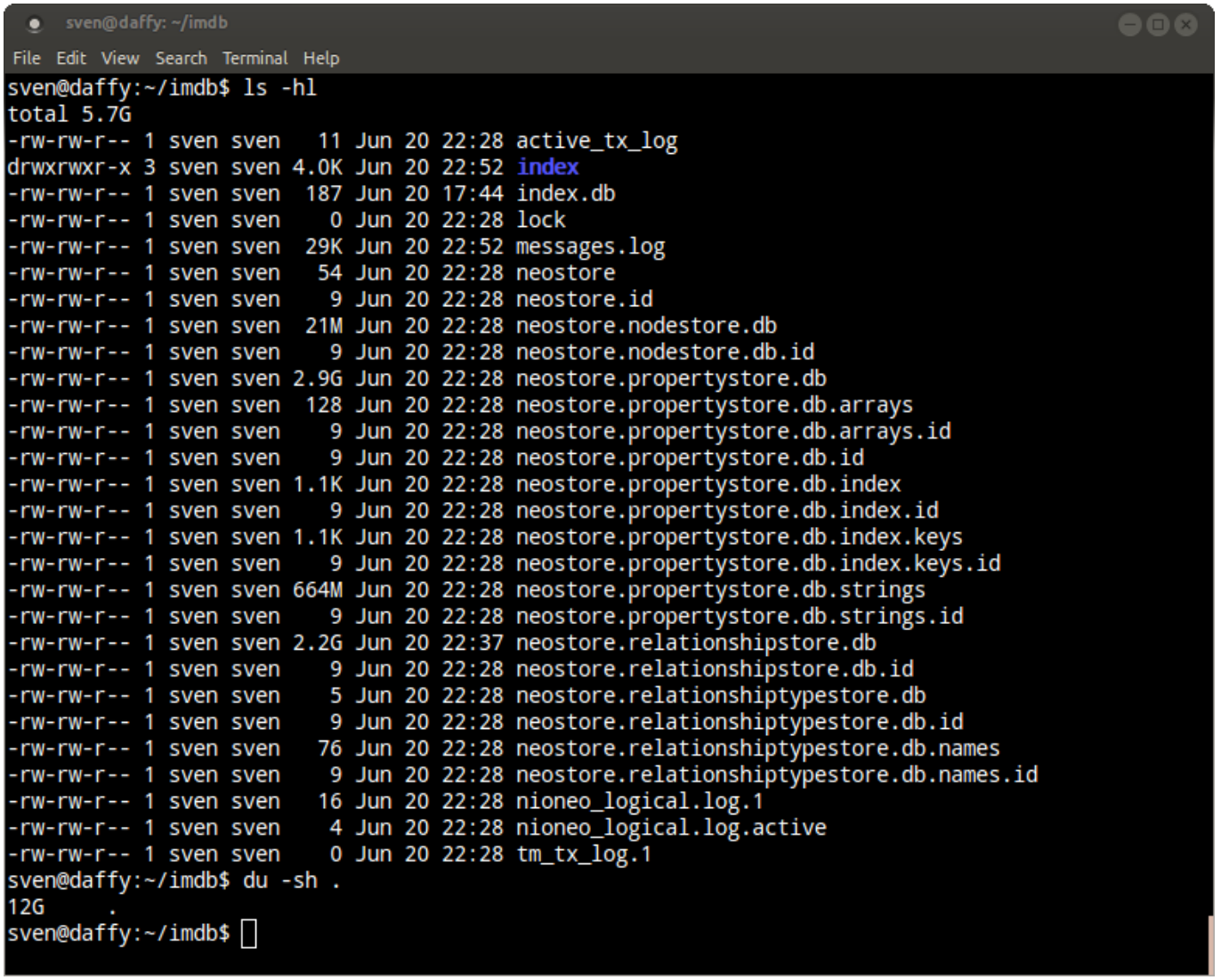
Dateigröße : ca. 1GB

Beispiel + Demo: Datenimport

- Problem: Transaktionen langsam (Two-Phase Commit)
- Lösung: **`org.neo4j.unsafe.batchinsert.BatchInserter`**!
  - Keine Transaktionen
  - Keine Thread-Safety

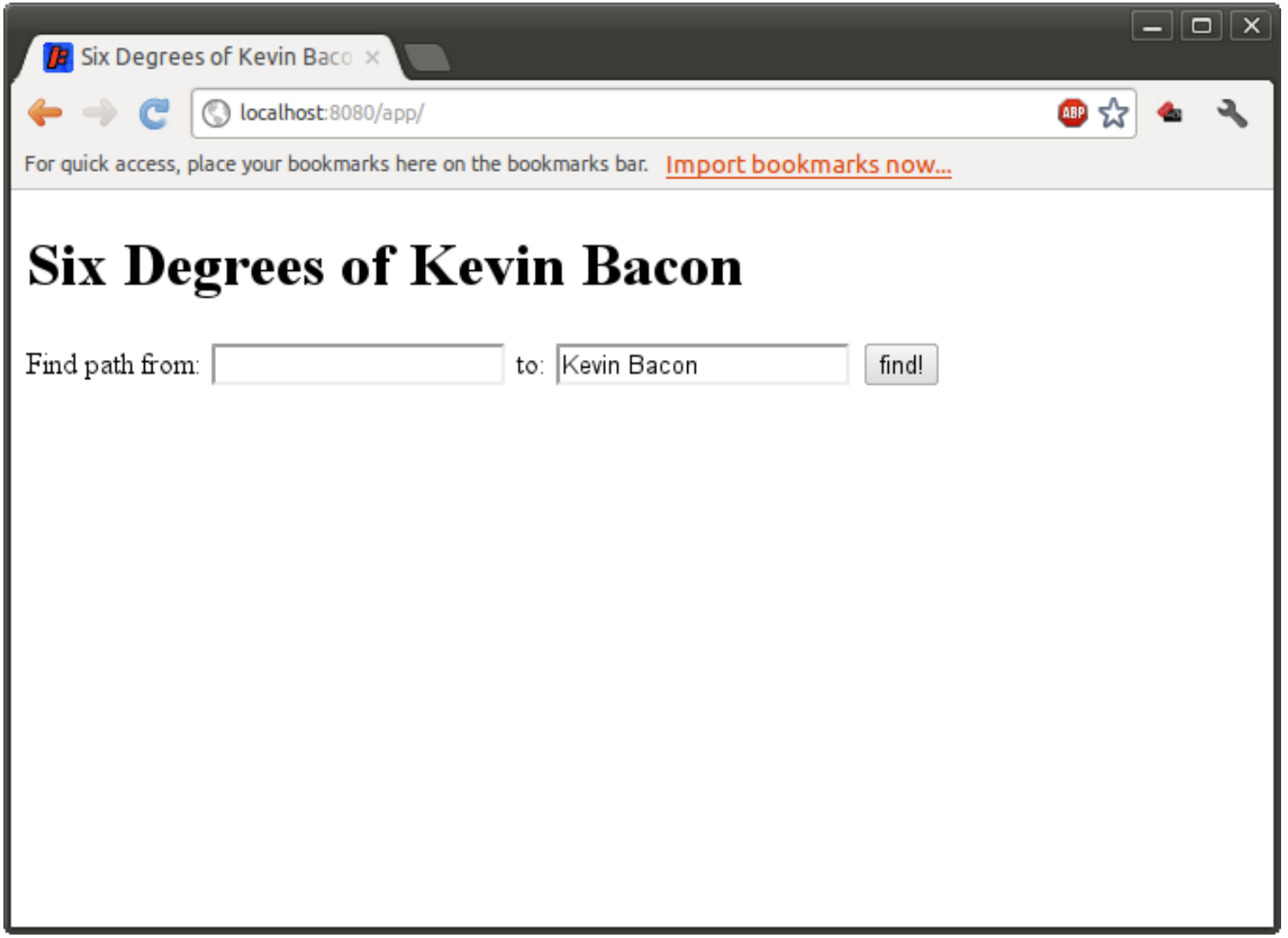
```
//Boilerplate init
BatchInserter batch = BatchInserters.inserter("/home/sven/imdb");
BatchInserterIndexProvider prov = new LuceneBatchInserterIndexProvider(ins);
BatchInserterIndex actors = prov.nodeIndex("actors", MapUtil.stringMap("type",
   "exact"));
BatchInserterIndex movies = prov.relationshipIndex("movies",
   MapUtil.stringMap("type", "exact"));
Map<String, Object> props = new HashMap<String, Object>();
//Kevin Bacon Knoten anlegen
props.put("name", "Kevin Bacon");
long kevin = batch.createNode(props);
actors.add(kevin, props);
//Jack Nicholson Knoten anlegen
props.put("name", "Jack Nicholson");
long jack = batch.createNode(props);
actors.add(jack, props);
//Film-Kante "A Few Good Men" anlegen
RelationshipType actedWith = DynamicRelationshipType.withName("ACTED_WITH");
props.clear();
props.put("movie", "A Few Good Men");
long fewGoodMen = batch.createRelationship(kevin, jack, actedWith, props);
movies.add(fewGoodMen, props);
//Boilerplate shutdown
prov.shutdown();
batch.shutdown();
```

Beispiel + Demo: Datenbank



Dauer des Import: ca. 3-4h

Beispiel + Demo: Webapplikation



Beispiel + Demo: Cypher Query

```
start
  n1=node:actors(name={fromActor}),
  n2=node:actors(name={toActor})
match
  p=shortestpath(
     n1-[:actedWith*..{max}]-n2
  )
return p
limit 1
```

Vielen Dank für die Aufmerksamkeit

Noch Fragen?