Projektgruppe

Seminarvortrag von Stefan Middeke

Knowledge Representation Persistence and Reasoning

4. Juni 2010

Überblick

- Motivation
- Repräsentation der Daten
- Speicherung und Abfrage von Daten
- Folgerungen aus vorhandenem Wissen
- Anforderungen für semantische Annotationssysteme
- Zusammenfassung und Einsatz für die PG

Wissen in RDBMS verwalten?

- Situation:
  - Bekannte Art der Daten
  - Bekannte Struktur der Daten
  - Keine (oder selten) Änderung an Strukturen
- Lösungsansatz:
  - Relationale Datenbank
  - Daten in Tabellen abgelegt
  - Schlüssel- / Fremdschlüsselbeziehungen
  - Abfrage in SQL

Semantisches Wissen

- Situation:
  - Vielfältige Daten
  - Struktur der Daten nicht bekannt
  - Häufig neue Strukturen und Inhalte
- Lösungsansatz 1:
  - Relationale Datenbank
  - Neue Tabelle pro Beziehung zwischen Daten erzeugen

Semantisches Wissen

- Situation:
  - Vielfältige Daten
  - Struktur der Daten nicht bekannt
  - Häufig neue Strukturen und Inhalte
- Lösungsansatz 2:
  - Wissen in abstrakten Strukturen repräsentieren und speichern
  - Flexibilität erreichen

Repräsentation von Daten

- RDF – Resource Description Framework
  - Dient zur Beschreibung von Beziehungen
  - (Subject, Predicate, Object)
  - R references (URI- Uniform Resource Identifier)
  - B blank nodes
  - L literals
  - RDF-Triple: $(S,P,O) \in (R \cup B) \times R \times (R \cup B \cup L)$

RDF Beispiel FOAF

http://de.wikipedia.org/wiki/FOAF

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:foaf="http://xmlns.com/foaf/0.1/"
     xmlns:dc="http://purl.org/dc/elements/1.1/"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<!-- Informationen zu einer Person: --> <foaf:Person xml:lang="en">

<!-- Name: -->            <foaf:name>Jimmy Wales</foaf:name>

<!-- E-Mail-Adresse :--><foaf:mbox rdf:resource="mailto:jwales@bomis.com"/>

<!-- Die Person kennt folgende andere Personen: -->

    <foaf:knows> <!-- Informationen zu einer anderen Person: -->

       <foaf:Person> <foaf:name>Angela Beesley</foaf:name> </foaf:Person>
     </foaf:knows>              </foaf:Person> </rdf:RDF>
```

Ontologie mit OWL

- OWL – Web Ontology Language
  - Beschreibt Ontologien
  - formale Zusammenhänge und Strukturen
  - Klassen, Eigenschaften (*properties*) und Instanzen

OWL - Beispiel

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" … >
<owl:Ontology rdf:about=""/>
<owl:Class rdf:ID="Gender"/>
<owl:Class rdf:ID="Person"/>
<owl:Class rdf:ID="Woman">
      <rdfs:subClassOf rdf:resource="#Person"/>
      <owl:equivalentClass>
        <owl:Restriction>
              <owl:onProperty rdf:resource="#Gender"/>
              <owl:hasValue rdf:resource="#female" rdf:type="#Gender"/>
        </owl:Restriction> </owl:equivalentClass> </owl:Class>  …
</rdf:RDF>
```

Speicherung und Abfrage der Daten

- RDF in relationaler Datenbank
- RDF native, Triple Store
  - Verbesserungsansatz: HPRD
- Graphendatenbank Neo4j
- Abfrage: SPARQL, Gremlin

SPARQL – Query Language for RDF

- Ermöglicht SQL-ähnliche Anfragen für RDF
- 4 Anfrageformen
  - SELECT
  - CONSTRUCT
  - ASK
  - DESCRIBE

SPARQL - Beispiel

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

_:a foaf:name "Johnny Lee Outlaw" .

_:a foaf:mbox <mailto:jlow@example.com> .

_:b foaf:name "Peter Goodguy" .

_:b foaf:mbox <mailto:peter@example.org> .

_:c foaf:mbox <mailto:carol@example.org> .
```

SPARQL - Beispiel

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?name ?mbox

WHERE { ?x foaf:name ?name .

        ?x foaf:mbox ?mbox }
```

Query Result:

name	mbox
"Johnny Lee Outlaw"	<mailto:jlow@example.com>
"Peter Goodguy"	<mailto:peter@example.org>

Quelle: http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/#basicpatterns

HPRD- High performance RDF database (by Baolin Hu; Bu Hu)

- Kombination verschiedener Datenbanktechniken in einen nativen RDF-Speicher
- Kombiniertes Index-Schema:
  - Ziel: verbesserte Performanz bei Anfragen
- Index Struktur für RDF
  - Grundlegender Triple Index: verwaltet alle RDF-Tripel durch Aufteilen des originalen RDF-Graphen
  - Pfad Index: wird verwendet um die Überprüfung von umfangreichen Anfrageausdrücken zu beschleunigen
  - Content Index: optionaler Index für Content-orientierte und zeit-orientierte RDF-Daten

HPRD- High performance RDF database (by Baolin Hu; Bu Hu)

- Pfad Index für häufige Anfragen
  - Vorteil: Kosten für Ausführung komplexer Anfragen kann signifikant verbessert werden

Graphendatenbank Neo4j

- hoch performant
- ist im 24/7 Produktionseinsatz
- voll ACID-transaktionale Datenbank in Java
- Datenstrukturen als Netzwerke auf dem Dateisystem
  - optimiertes Dateiformat
- als selbstständiger Server über REST oder als Embedded Server

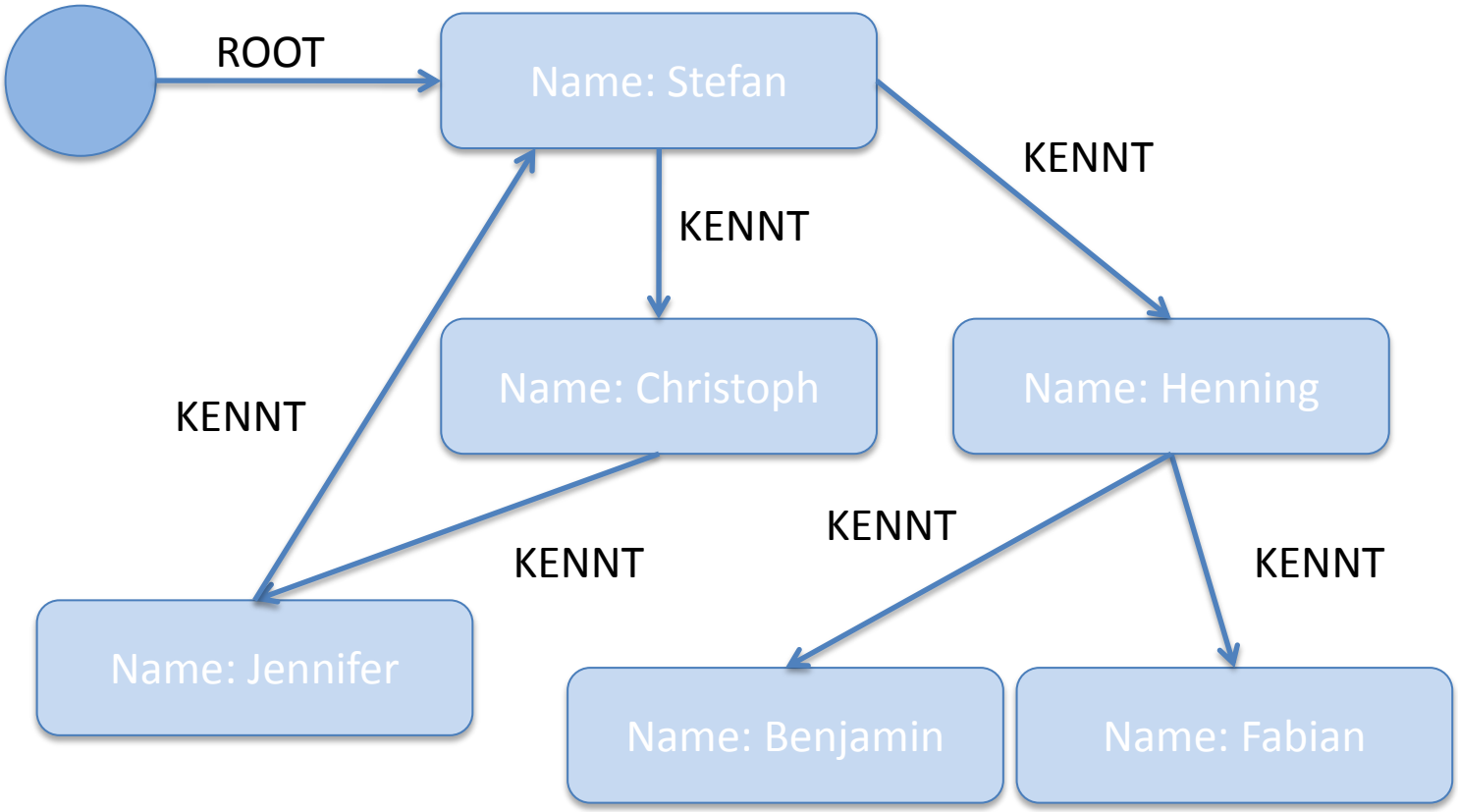
Labeled Property Graph

- Besteht aus Knoten und gerichteten Kanten
- Knoten und Kanten haben Identifizierer
- Knoten und Kanten haben verschiedene Eigenschaften

Gremlin

- Xpath-basierte Sprache für Graphen
- komplexe Anfragen elegant ausdrücken
- SPARQL – Anfragen direkt in Gremlin
- Kann mit Java (ab 1.6) eingebunden werden
- Java Universal Network/Graph Framework (JUNG)
- Verschiedene Speichertechniken
  - TinkerGraph
  - Neo4j
  - Sesame SAIL (SAIL= Storage and Inference Layer)

Beispielgraph



Beispiel: Neo4j mit Java

Öffnen oder neue Neo4j-Datenbank erstellen:

```
EmbeddedGraphDatabase neodb = new
        EmbeddedGraphDatabase(“target/stefan“);
new EmbeddedGraphDatabase(“target/stefan“);
```

Zwei Knoten mit Eigenschaft „name“ , die mit der Relation „KENNT“ verbunden sind:

```
Node stefan = neodb.createNode();
Node.setProperty(“name“, “Stefan“);
Node christoph = neodb.creatNode();
Christoph.setProperty(“name“,“Christoph“);
stefan.createRelationshipTo(christoph, KENNT);
```

Wen kennt Stefan?

```
tx = neodb.beginTx();
        for(Relationship rel:stefan.getRelationship(KENNT)){
                Node friend =rel.getOtherNode(stefan);
                System.out.println(friend.getProperty("name"));
        }
tx.success();
tx.finish();
```

Traverser

```
Traverser friends = stefan.traverse( Order.BREADTH_FIRST,
StopEvaluator.ALL_BUT_START_NODE, KENNT, DIRECTION.BOTH );

for ( Node friend : friends)  {
        System.out.println( friend.getProperty("name"));
}
```

Gremlin – Sitzung zum Beispiel

gremlin> #öffne einen Neo4j Graphen als default ($_g)

gremlin> $_g:=neo4j:open(‘tmp/stefan‘)

= => neo4jgraph[tmp/stefan]

gremlin> #setze Stefan als Startknoten ($_) über eine Volltextsuche

gremlin> $_ := g:key(‘name‘,‘Stefan‘)

= => v[1]

gremlin> #ist das auch Stefan ?

gremlin> . /@name

= => Stefan

Gremlin – Sitzung zum Beispiel

gremlin> #Wie sehen die Kanten aus?

gremlin> . /bothE

= => e[0] [1 – KENNT -> 2]

= => e[1] [1 – KENNT -> 3]

= => e[4] [4 – KENNT -> 1]

gremlin> #Die Namen von Stefan‘s Bekannten

gremlin> . / bothE[@label=‘KENNT‘] / inV / @name

= => Christoph

= => Henning

= => Jennifer

Überblick

- Motivation
- Repräsentation der Daten
- Speicherung und Abfrage von Daten
- Folgerungen aus vorhandenem Wissen
- Anforderungen für semantische Annotationssysteme
- Zusammenfassung und Einsatz für die PG

Temporal Reasoning

- Aufgaben
  - zeitliche Konsistenz sicherstellen
  - Anfragen mit zeitlichem Bezug
    - Gilt etwas zum Zeitpunkt t
    - Gelten Aussagen zur gleichen Zeit
  - Regeln bei Veränderungen
  - Aufgaben für Folgerungen aus Veränderungen
    - Beschreibung der Vergangenheit ; Zustand jetzt
    - Wie sieht die Welt in der Zukunft aus?
    - aus Zuständen für verschiedene Zeitpunkte => Regeln

Anforderungen an Semantische Annotationssysteme

7 Anforderungen nach [Uren et al.]

- Standards nutzen
- Nutzer zentriert, kooperative Nutzung
- Unterstützung bei Ontologien
- Unterstützung für verschiedene Dokumentformate
- Dokumentevolution
- Speicherung der Annotationen
- Automatisierung

Zusammenfassung

- RDF und OWL
- SPARQL
- Neo4j
- Gremlin

Ende des Vortrages

Fragen und Anmerkungen?

Referenzen

- http://www.w3.org/RDF/
- http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/
- http://wiki.github.com/tinkerpop/gremlin/
- www.neo4j.org
- Uren et. al. (2006) ‘Semantic annotation for knowledge management: Requirements and a survey of the state of the art‘, Web Semantics, Services and Agents on the World Wide Web 4(1), 14-28
- Liu, Baolin and Hu, Bo (2010) ‘HPRD: a high performance RDF database‘, International Journal of Parallel, Emergent and Distributed Systems, 25:2, 123-133 ; URL:http://dx.doi.org/10.1080/17445760802431839
- Vila, L. (1994), ‘ Survey on Temporal Reasoning in Artificial Intelligence‘