RDF Syntax: Eine Breite Wahl

Markus Stocker

30. April 2018

Rekapitulation

- Woraus besteht ein *statement*?
- Was ist ein URI und wozu dient dieser in RDF?
- Wie entstehen RDF Graphen?
- Welche Vorteile haben RDF Graphen gegenüber XML Bäume?

Übersicht

- Graph Darstellung
- Das Tripel
- Serialisierung
- RDF Syntaxen

Graph Darstellung

- In der letzten Vorlesung haben wir Graphen visuell dargestellt
- Solche Darstellung ist einfach in der kommunikation mit Menschen
- Geeignet für Vorlesungen, Folien und kleine Graphen
- Allerdings klar ungeeignet für Maschinenverarbeitung
- Auch nicht geeignet für grosse Graphen

Graph Darstellung

- Für maschinelle Verarbeitung benötigen wir eine Serialisierung
- Transformation der Graph Datenstruktur in eine Zeichenfolge
- Letztlich ein Text Dokument
- Ähnlich wie Serialisierung der XML Baumstruktur als Text Dokument

Von Graphen zu Tripelmengen

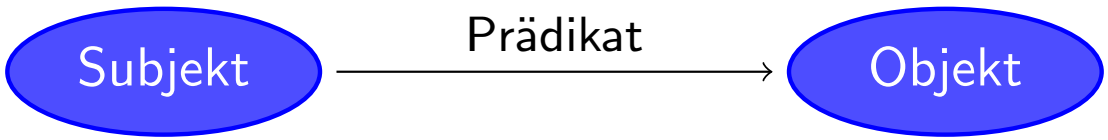
- Graph kann man anhand der Kanten zerlegen
- Jede Kante mit verknüpften Knoten ist eindeutig
- Zwei über eine Kante verknüpfte Knoten ist Elementareinheit
- In RDF wir diese Einheit Tripel (*triple*) genannt
- Ein RDF Graph ist somit eine Tripelmenge
- Und zwar ungeordnete Menge

Das Tripel

- Ein Triple besteht aus Subjekt, Prädikat, und Objekt

```
Triple := <Subject> <Predicate> <Object>
```

- Prädikate entsprechen Graph Kanten
- Üblicherweise mit `p` gekürzt
- Prädikate zeigen immer vom Subjekt zum Objekt



- Subjekt und Objekt entsprechen Graph Knoten
- Üblicherweise jeweils mit `s` und `o` gekürzt

```
Triple := (s, p, o)
```

Das Tripel

- Prädikate sind mittels URI immer benannt
- Subjekt und Objekt können unbenannt sein (*blank node*)
- Subjekt und Objekt können mittels URI benannt sein
- Objekt kann ein Literal sein

Tripel Syntaxen

- Es gibt mehrere Syntaxen für Tripelmengen
- N-Triples, Turtle, und RDF/XML wohl die bekanntesten
- Es handelt sich um drei Serialisierungen
- Für Tripelmengen (RDF Graphen) nach RDF (Text) Dokumente
- Jede Syntax hat Vor- und Nachteile
- Es ist möglich zwischen Syntaxen zu konvertieren
- Konvertierung ohne Informationsverlust

N-Triples

- Zeilenorientierte Serialisierung von Tripel
- Einfach für Software zu lesen und schreiben
- Ideal für *streaming* Anwendungen
- Ein Tripel endet mit einem Punkt
- Jedes Tripel meist auf eigener Zeile (nicht zwingend)
- Leerzeichen und Umbrüche ausserhalb URI und Literal ignoriert
- URIs zwischen spitze Klammern
- Literale zwischen Anführungszeichen

N-Triples: Beispiel

```
<http://example.org#Earth> <http://www.w3.org/2000/01/rdf-schema#label> "Earth" .
<http://example.org#Earth> <http://www.w3.org/2000/01/rdf-schema#label> "Erde" .
<http://example.org#Earth> <http://www.w3.org/2000/01/rdf-schema#label> "Terra" .
<http://example.org#Earth> <http://example.org#radius> "6371.0" .
```

N-Triples: *Blank Node*

```
<http://example.org#Earth> <http://example.org#radius> _:b .
_:b <http://example.org#value> "6371.0" .
_:b <http://example.org#unit> <http://units.org#km> .
```

Turtle

- Eine Obermenge von N-Triples
- Führt Präfixe ein um URIs zu kürzen
- Bei gleichem Subjekt (und Prädikat) kann URI entfallen
- Einfach für Menschen zu lesen, kompakt
- Für Maschinen wird entsprechender Parser benötigt

Turtle: Beispiel

```
@prefix ex: <http://example.org#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

ex:Earth rdf:type ex:Planet ;
         rdfs:label "Earth", "Erde", "Terra" ;
         ex:radius "6371.0" .
```

N-Turtle: *Blank Node*

```
@prefix ex: <http://example.org#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix unit: <http://units.org#> .

ex:Earth rdf:type ex:Planet ;
         rdfs:label "Earth", "Erde", "Terra" ;
         ex:radius [
           ex:value "6371.0" ;
           ex:unit unit:km
         ] .
```

RDF/XML

- XML basierte RDF Serialisierung
- Weit verbreitet, XML Parser in vielen Programmiersprachen
- Kodierung der Tripel muss hierarchisch sein
- Triple Gruppierung anhand Subjekte
- Wurzelknoten ist `rdf:RDF` mit globalen Namensräumen

RDF/XML: Beispiel

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:ex="http://example.org#">

  <rdf:Description rdf:about="http://example.org#Earth">
    <rdf:type rdf:resource="http://example.org#Planet"/>
    <rdfs:label>Earth</rdfs:label>
    <rdfs:label>Erde</rdfs:label>
    <rdfs:label>Terra</rdfs:label>
    <ex:radius>6371.0</ex:radius>
  </rdf:Description>

</rdf:RDF>
```

RDF/XML: *Blank Node* (1)

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:ex="http://example.org#">

  <rdf:Description rdf:about="http://example.org#Earth">
    <rdf:type rdf:resource="http://example.org#Planet"/>
    <ex:radius>
      <rdf:Description>
        <ex:value>6371.0</ex:value>
        <ex:unit rdf:resource="http://units.org#km"/>
      </rdf:Description>
    </ex:radius>
  </rdf:Description>

  </rdf:RDF>
```

RDF/XML: *Blank Node* (2)

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:ex="http://example.org#">

  <rdf:Description rdf:about="http://example.org#Earth">
    <rdf:type rdf:resource="http://example.org#Planet"/>
    <ex:radius rdf:nodeID="_:b"/>
  </rdf:Description>

  <rdf:Description rdf:nodeID="_:b">
    <ex:value>6371.0</ex:value>
    <ex:unit rdf:resource="http://units.org#km"/>
  </rdf:Description>

</rdf:RDF>
```

RDF/XML: `rdf:ID` und `rdf:about`

- Anstatt `rdf:about` kann auch `rdf:ID` benutzt werden
- Allerdings darf `rdf:ID` für ein URI nur einmal verwendet werden

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ex="http://example.org#"
         xml:base="http://example.org">

  <ex:Planet rdf:ID="Earth"/>

  <ex:Planet rdf:about="#Mars"/>

</rdf:RDF>
```

RDF/XML: Bemerkungen

- Namensräume sind zwingend nötig, keine optionale Eigenschaft
- Dies weil URI als Elementname nicht möglich (: nicht erlaubt)
- Namensräume können aber nicht in Attributwerte verwendet werden
- Lösung über XML Entitäten oder mit *base namespace*
- Bindestriche sind nicht erlaubt nach Doppelpunkt in XML Tag
- Prozentzeichen ebenfalls nicht erlaubt in XML Tag
- Dies obwohl beide Zeichen in URIs erlaubt sind

RDF/XML: Entitäten

```
<!DOCTYPE rdf:RDF[
  <!ENTITY ex 'http://example.org#'>
]>

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:ex="http://example.org#">

  <rdf:Description rdf:about="&ex;Earth">
    <rdf:type rdf:resource="&ex;Planet"/>
    <rdfs:label>Earth</rdfs:label>
    <rdfs:label>Erde</rdfs:label>
    <rdfs:label>Terra</rdfs:label>
    <ex:radius>6371.0</ex:radius>
  </rdf:Description>

</rdf:RDF>
```

RDF/XML: *Base Namespace*

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:ex="http://example.org#"
         xml:base="http://example.org">

  <rdf:Description rdf:about="#Earth">
    <rdf:type rdf:resource="#Planet"/>
    <rdfs:label>Earth</rdfs:label>
    <rdfs:label>Erde</rdfs:label>
    <rdfs:label>Terra</rdfs:label>
    <ex:radius>6371.0</ex:radius>
  </rdf:Description>

</rdf:RDF>
```

RDF Lesen, Schreiben, Verarbeiten

- Viele Programmiersprachen unterstützen RDF Lesen und Schreiben
- Nicht nur als RDF/XML zu einem XML Dokument
- Sondern als Triplemenge und RDF Graph
- RDF Daten werden also von der Sprache als Tripelmenge behandelt
- Meist wird auch die programmatische Verarbeitung unterstützt

RDF Lesen und Schreiben: Python

```
from rdflib import Graph

g = Graph()

g.parse(data=rdf, format='nt')

g.serialize(format='turtle')
```

RDF Verarbeiten: Python

```
from rdflib import Graph, URIRef, Literal
from rdflib.namespace import RDF, RDFS

g = Graph()

g.add((URIRef('http://example.org#Earth'),
       RDF.type,
       URIRef('http://example.org#Planet')))

g.add((URIRef('http://example.org#Earth'),
       URIRef('http://example.org#radius'),
       Literal('6371.0')))
```

Zusammenfassung

- Von Graphvisualisierungen zu Tripelmengen
- Das Tripel bestendend aus Subject, Prädikat, Object
- Verschiedene Syntaxen (N-Triples, Turtle, RDF/XML)
- Beispiele für Serialisierungen von Tripelmengen
- RDF programmatisch Lesen und Schreiben: Python