Semantic Web Technologies I

Lehrveranstaltung im WS12/13

PD Dr. Sebastian Rudolph
Dr. Duc Thanh Tran
M.Sc. Anees ul Mehdi
Linked Data
Dr. Duc Thanh Tran

XML und URIs
Einleitung in RDF
RDF Schema
Logik – Grundlagen
Semantik von RDF(S)
SPARQL – Syntax und Intuition
Semantik von SPARQL
Linked Data
Semantic Search
OWL – Syntax und Intuition I
OWL – Syntax und Intuition II
OWL – Semantik und Reasoning
Konjunktive Anfragen und Regelsprachen
Applications
Agenda

• Motivation
• Linked Data Principles
• Linked Data Access
MOTIVATION
Motivation

• Data volumes explode
  - More and more data available on the Web is represented in Semantic Web standards
  - Linking Open Data (LOD) initiative

• Semantic Web standards & technologies facilitate
  - Representation, exchange
  - Integration & linking of data from multiple sources
  - Consumption
Linked Data on the Web
Linked Data on the Web

2007-11
Linked Data on the Web

2008-02
Linked Data on the Web

2008-03
Linked Data on the Web

2008-09
Linked Data on the Web

As of July 2009

2009-07
Linked Data on the Web

2010-09
Linked Data on the Web
Types of Data in the Linking Open Data Cloud

http://www4.wiwiss.fu-berlin.de/lodcloud/state/ (Sept 2010)
Semantic Technologies facilitate access to data

- Q: data about Berlin?
- Q: famous people that died in Berlin?
- Q: data about Hegel?
- Q: Hegel’s publications?
- Q: data about Marlene Dietrich?
- Q: Dietrich’s songs?
DBpedia

- Linked Data version of Wikipedia
- Scripts that extract data (text, links, infoboxes) from Wikipedia
- Published as Linked Data
- Interlinking hub in the Linked Data web

- Berlin
  - http://dbpedia.org/resource/Berlin

- Hegel
  - http://dbpedia.org/resource/Georg_Wilhelm_Friedrich_Hegel

- Marlene Dietrich
  - http://dbpedia.org/resource/Marlene_Dietrich
BBC Music

- Data about BBC (radio) programmes, artists, songs…
- Combination of BBC-internal data (playlists), MusicBrainz (artists, albums), Wikipedia (artists)
- Underpinning the BBC Music website
- Data published according to Linked Data principles

- Marlene Dietrich
  - http://www.bbc.co.uk/music/artists/191cba6a-b83f-49ca-883c-02b20c7a9dd5
Virtual International Authority File (VIAF)

- Joint project of national libraries and related organisations
  - 21 institutions, among them the Deutsche Nationalbibliothek

- Provide access to “authority files”

- Matching and interlinking collections from participating institutions

- Hegel
  - http://viaf.org/viaf/89774942

- Marlene Dietrich
  - http://viaf.org/viaf/97773925
LINKED DATA PRINCIPLES
Semantic Web technologies, standardised by the W3C, are mature:

- RDF recommendation in 1999, update in 2004
- RDFa (RDF in HTML) note in 2008
- RDFS recommendation in 2004
- SPARQL recommendation in 2008
- OWL recommendation in 2004, update in 2009

Linked Data is a subset of the Semantic Web stack, including web architecture:

- HTTP
- IRI / URI
- (RDF / XML)
Linked Data Principles

1. Use URIs as names for things

2. Use HTTP URIs so that people can look up those names.

3. When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)

4. Include links to other URIs. so that they can discover more things.

http://www.w3.org/DesignIssues/LinkedData
1. Use URIs as Names for Things

- Use a unique identifier (URIs) to denote things
- Hegel, Georg Wilhelm Friedrich
  - http://dbpedia.org/resource/Georg_Wilhelm_Friedrich_Hegel
  - http://viaf.org/viaf/89774942
  - ...
Names for Things

“Now! *That* should clear up a few things around here!”
2. Use HTTP URIs

- Enables “lookup” of URIs
- Via Hypertext Transfer Protocol (HTTP)
- Piggy-backs on hierarchical Domain Name System to guarantee uniqueness of identifiers
- Uses established HTTP infrastructure
- Connects logical level (thing) with physical level (source)
- Important: distinction between name/“thing URI” and location/“source URI” („other resource“/„non-information resource“ vs. „information resource“)
Information Resources vs. Other Resources

Marlene Dietrich, the person

Name?
Creator?
Birth date?
Last change date?
License?
Copyright?
...

File containing data about Marlene Dietrich

RDF
Correspondence between thing-URI and source-URI

User Agent

HTTP
GET

Web Server

RDF

http://www.bbc.co.uk/music/artists/191cba6a-b83f-49ca-883c-02b20c7a9dd5.rdf

http://www.bbc.co.uk/music/artists/191cba6a-b83f-49ca-883c-02b20c7a9dd5#artist
3. Provide Useful Information

- When somebody looks up a URI, return data using the standards (RDF*, SPARQL)
- Resource Description Framework, a format for encoding graph-structured data (with URIs to identify nodes/vertices and links/edges)
Resource Description Framework

- Directed, labeled graph
- \texttt{triple(subject, predicate, object)}
  - \texttt{subject}: URI (or blank node)
  - \texttt{predicate}: URI
  - \texttt{object}: URI (or blank node) or RDF literal (string, integer, date…)

- RDF/XML is the most widely deployed serialisation
- Other serialisations possible (N-Triples, Turtle, Notation3…)

4. Link to Other URIs

• Enable people (and machines) to jump from server to server

• External links vs. internal links (for any predicate)

• Using external vocabularies enables linking

• Vocabularies might be interlinked, too

• Special owl:sameAs links to denote equivalence of identifiers (useful for data merging)
Equivalences via owl:sameAs

http://viaf.org/viaf/89774942
  - http://dbpedia.org/resource/Georg_Wilhelm_Friedrich_Hegel
  - http://www.idref.fr/026917467/id
  - http://libris.kb.se/resource/auth/190350
  - http://d-nb.info/gnd/118547739

http://www.bbc.co.uk/music/artists/191cba6a-b83f-49ca-883c-02b20c7a9dd5#artist
  - http://dbpedia.org/resource/Marlene_Dietrich

http://viaf.org/viaf/97773925
  - http://dbpedia.org/resource/Marlene_Dietrich
  - http://d-nb.info/gnd/118525565
  - http://libris.kb.se/resource/auth/238817
  - http://www.idref.fr/027561844/id

http://dbpedia.org/resource/Berlin
  - http://mpi.de/yago/resource/Berlin
  - http://data.nytimes.com/N50987186835223032381 - Berlin (Germany)
  - http://www4.wiwiss.fu-berlin.de/flickrwrapr/photos/Berlin
  - http://data.nytimes.com/16057429728088573361 - Gaspe Peninsula (Quebec) (?)
Benefits of Linked Data

- Explicit, simple data representation
  - Common data representation (Resource Description Framework, RDF) hides underlying technologies and systems
- Distributed System
  - Decentralised distributed ownership and control facilitates adoption and scalability
- Loose coupling with common language layer
  - Large scale systems require loose coupling, via HTTP as common access protocol
- Ease of publishing and consumption
  - Simple and easy-to-use systems and technologies to facilitate uptake
- Cross-referencing
  - Allows for linking and referencing of existing data, via reuse of URIs
- Incremental data integration
  - Start with merged RDF graphs and provide mappings as you go
Challenges

- Ramp-up cost for data conversion
  - May be alleviated by semi-automatic mappings and adequate tool support for manual conversion

- Integrated data may be messy at first
  - But can be refined as need arises

- Distributed creation and loose coordination may result in inconsistencies
  - Can be detected, diagnosed, and fixed with appropriate tools

- Not only quality but also volume is a problem
Resource Graph Explosion

Directed graph rooted in http://danbri.org/foaf.rdf

Level 0: 1

Level 1: $25^1$ (avg), $105^1$ (worst)

Level 2: $25^2$ (avg), $105^2$ (worst)

Level 3: $25^3$ (avg), $105^3$ (worst)

Level k: $n^k$
ACCESSING LINKED DATA

APPLICATIONS
CRAWLING
INDEXING
QUERY PROCESSING
Basic Application: Entity Browsing

Warehousing/Crawl-Index-Serve

Virtual Integration/Distributed Querying

SWSE, Falcons, Sindice, Watson, FactForge...

Tabulator, Disco, Zitgist…
Data Sources in SemanticSearch@AIFB Demo
[SWC Finalist 2010]

- English Wikipedia
- Data from Linked Open Data
  - DBpedia
  - YAGO
  - Many more
- Live data from Data.gov (US Government)
  - E.g. live data about earthquakes
Alice is an album by Tom Waits, released in 2002 on Epitaph Records (under the Anti sub-label). The album contains the majority of songs written for the play Alice, based on the forbidden love between Lewis Carroll and Alice Liddell, for whom he wrote the story Alice’s Adventures in Wonderland. The adaptation was directed by Robert Wilson, whom Waits had previously worked with on the play The Black Rider, and originally set up at the Thalia Theatre in Hamburg in 1992. The play has since been performed in various theatres around the world.

The album was co-released with Blood Money, containing songs from a play adapted by Robert Wilson from Georg Büchner’s Woyzeck.

It was ranked #2 in Metacritic’s Top 30 albums of 2002.[11]

The songs had been released as a bootleg in several different versions called The Alice Demos many years before its official release. The source is believed to be studio recordings taken when Waits’ car was broken into in late 1992.[21]

### Chart information

<table>
<thead>
<tr>
<th>Chart</th>
<th>Peak position</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switzerland</td>
<td>24</td>
</tr>
<tr>
<td>UK</td>
<td>20</td>
</tr>
<tr>
<td>US</td>
<td>33</td>
</tr>
</tbody>
</table>

### Track listing

All tracks written by Tom Waits and Kathleen Brennan.

1. "Alice" - 4:28
   - Tom Waits - Vocal, Piano
   - Eric Perney - Bass
   - Colin Stetson - Sax

source: [http://iwb.fluidops.com](http://iwb.fluidops.com)
Wikipedia + DBpedia

source: http://iwb.fluidops.com
Query Interpretation, Refinement and Exploration

Search Results

Initial Query
See Entire Query

?sx1
A Kind of Magic (song)
Another One Bites the Dust
Back Chat
Bicycle Race
Body Language (song)
Calling All Girls
Crazy Little Thing Called Love
Fat Bottomed Girls
Good Old-Fashioned Lover Boy
Hammer to Fall
Heaven for Everyone
I Want to Break Free
It's Late
It's a Hard Life
Keep Yourself Alive
Killer Queen
Las Palabras de Amor
Liar (Queen song)
Long Away
Mustapha

Keywords

queen single

Set searchfield to "queen single"

A (queen) is a Single
B writer A (queen)
B is a Single
A is a Single
A producer B (queen)

Semantic Completions

Syntactic Completions

Facets

producer
range: All Values (43)
type
range: All Values (43)
writer
range: All Values (42)
musical artist (42)
Brian May (13)
Frank Musiker (1)
Freddie Mercury (14)
John Deacon (7)
Roger Meddows-Taylor (7)

Result Column 1
Result Inspection, Analysis and Browsing
Querying Data Across Sources

- Data warehousing or materialisation-based approaches (MAT)

- Distributed query processing approaches (DQP) over Linked Data sources

- DQP over RDF stores (not covered)
(Linked Data) Crawling

1. Get URI from a queue
2. Open connection and fetch content
3. Process and store content
4. Extract new links and put into queue
5. At defined intervals: schedule URIs in queue

Conjunctive Queries / BGP

?x foaf:maker ?y .

- Built on triple patterns containing variables (?, ?, ?), (s, ?, ?), (?, p, ?), (?, ?, o), (s, p, ?), (?, p, o), (s, p, o)
- Variables are bound during query evaluation
- Query evaluation results in a set of variable bindings

<table>
<thead>
<tr>
<th>?x</th>
<th>?y</th>
<th>?z</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.w3.org/People/Berners-Lee/card#i">http://www.w3.org/People/Berners-Lee/card#i</a></td>
<td><a href="http://www.w3.org/DesignIssues/LinkedData.html">http://www.w3.org/DesignIssues/LinkedData.html</a></td>
<td>Linked Data - Design Issues - W3C</td>
</tr>
</tbody>
</table>
## RDF Storage (1/4)

### Horizontal table-per-class

**Class:** **Car**

<table>
<thead>
<tr>
<th>plate</th>
<th>model</th>
<th>doors</th>
<th>owner</th>
<th>series</th>
<th>img</th>
<th>seeAlso</th>
</tr>
</thead>
<tbody>
<tr>
<td>234HSJ</td>
<td>Toyota</td>
<td>5</td>
<td>Frank</td>
<td>Corolla</td>
<td>x.jpg</td>
<td>toyota.com</td>
</tr>
</tbody>
</table>

### Pros:
- Fast for certain queries, esp. “star shaped” queries
- Little redundancy in cells

### Cons:
- Becomes very sparse for larger schema
- Lots of nulls needed
- Special handling needed for multi-valued attributes

<table>
<thead>
<tr>
<th>name</th>
<th>plate</th>
<th>country</th>
<th>height</th>
<th>img</th>
<th>seeAlso</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frank</td>
<td>234HSJ</td>
<td>France</td>
<td>1.92</td>
<td>a.svg</td>
<td></td>
</tr>
<tr>
<td>Jill</td>
<td>148YUI</td>
<td>Ireland</td>
<td>1.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jim</td>
<td>923HIJ</td>
<td>US &amp; UK</td>
<td>black</td>
<td>b.svg</td>
<td></td>
</tr>
</tbody>
</table>
RDF Storage (2/4)  
Vertical triple table

<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frank</td>
<td>ownsCar</td>
<td>234HSJ</td>
</tr>
</tbody>
</table>

**Pros:**
- No more nulls needed
- Flexible for updates (even to data-schema)
- Multi-valued attributes no problem

**Cons:**
- Lot’s of self-joins
- Lot’s of redundancy in the cells

| 234HSJ | series | Corolla |
### RDF Storage (3/4)
**Vertical table per prop.**

#### Property: `model`

<table>
<thead>
<tr>
<th>subject</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>234HSJ</td>
<td>Toyota</td>
</tr>
<tr>
<td>923HIJ</td>
<td>Ford</td>
</tr>
<tr>
<td>242HFI</td>
<td>Fiat</td>
</tr>
<tr>
<td>541PJH</td>
<td>Porsche</td>
</tr>
<tr>
<td>148YUI</td>
<td>Jeep</td>
</tr>
</tbody>
</table>

#### Property: `ownsCar`

<table>
<thead>
<tr>
<th>subject</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frank</td>
<td>234HSJ</td>
</tr>
<tr>
<td>Jill</td>
<td>148YUI</td>
</tr>
<tr>
<td>Jim</td>
<td>923HIJ</td>
</tr>
<tr>
<td>Joan</td>
<td>541PJH</td>
</tr>
<tr>
<td>Mary</td>
<td>242HFI</td>
</tr>
</tbody>
</table>

#### Property: `type`

<table>
<thead>
<tr>
<th>subject</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>234HSJ</td>
<td>Car</td>
</tr>
<tr>
<td>923HIJ</td>
<td>Car</td>
</tr>
<tr>
<td>242HFI</td>
<td>Car</td>
</tr>
<tr>
<td>541PJH</td>
<td>Car</td>
</tr>
<tr>
<td>148YUI</td>
<td>Car</td>
</tr>
</tbody>
</table>

#### Pros:
- Less redundancy

#### Cons:
- Potentially many tables
- New property = new table
- Assumes predicate always known
RDF Storage (4/4)  
Hybrid

<table>
<thead>
<tr>
<th>plate</th>
<th>model</th>
<th>doors</th>
<th>owner</th>
<th>series</th>
</tr>
</thead>
<tbody>
<tr>
<td>234HSJ</td>
<td>Toyota</td>
<td>5</td>
<td>Frank</td>
<td>Corolla</td>
</tr>
<tr>
<td>923HIJ</td>
<td>Ford</td>
<td>3</td>
<td>Jim</td>
<td>Ka</td>
</tr>
<tr>
<td>242HFI</td>
<td>Fiat</td>
<td>3</td>
<td>Mary</td>
<td>Tempura</td>
</tr>
<tr>
<td>541PJH</td>
<td>Porsche</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>148YUI</td>
<td>Jeep</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Pros:**
- ~Depends

**Cons:**
- Likely to be more costly to manage

Class: **Car**  
Property: **seeAlso**

<table>
<thead>
<tr>
<th>subject</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>234HSJ</td>
<td>toyota.com</td>
</tr>
<tr>
<td>923HIJ</td>
<td>ford.com</td>
</tr>
<tr>
<td>242HFI</td>
<td>fiat.com</td>
</tr>
<tr>
<td></td>
<td>porsche.com</td>
</tr>
<tr>
<td></td>
<td>jeep.com</td>
</tr>
<tr>
<td></td>
<td>joan.com</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>ownsCar</th>
<th>nation</th>
<th>height</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frank</td>
<td>234HSJ</td>
<td>France</td>
<td>1.92</td>
</tr>
<tr>
<td>Jill</td>
<td>148YUI</td>
<td>Ireland</td>
<td>1.75</td>
</tr>
<tr>
<td>Jim</td>
<td>923HIJ</td>
<td>US &amp; UK</td>
<td>black</td>
</tr>
<tr>
<td>Joan</td>
<td>541PJH</td>
<td>India</td>
<td>1.75</td>
</tr>
<tr>
<td>Mary</td>
<td>242HFI</td>
<td>Scotland</td>
<td></td>
</tr>
</tbody>
</table>

Property: **img**

<table>
<thead>
<tr>
<th>subject</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>234HSJ</td>
<td>x.jpg</td>
</tr>
<tr>
<td>148YUI</td>
<td>i.png</td>
</tr>
<tr>
<td>Frank</td>
<td>a.svg</td>
</tr>
<tr>
<td>Jim</td>
<td>b.svg</td>
</tr>
</tbody>
</table>
1. Triple stores
   - Only service simple RDF triple patterns
     ✦ ?s rdf:type foaf:Person .
     ✦ aidan ?p galway .
     ✦ ...

2. Quad stores/SPARQL engines
   - Also service patterns involving named graphs
   - Typical for indexing data from multiple sources
   - Needed for SPARQL querying!!
     ✦ GRAPH ?g { ?s rdf:type foaf:Person }
     ✦ GRAPH foaf.rdf { aidan ?p galway }
     ✦ FROM graph1.rdf … WHERE { ?s ?p ?o . }
Building a full Quad index

- (subject, predicate, object, graph)
  - graph sometimes called context
- $2^4 = 16$ patterns to service!

<table>
<thead>
<tr>
<th>No</th>
<th>Access pattern</th>
<th>No</th>
<th>Access pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(?::?:?:?)</td>
<td>9</td>
<td>(s:::o:c)</td>
</tr>
<tr>
<td>2</td>
<td>(s:::?:?)</td>
<td>10</td>
<td>(?:::o:c)</td>
</tr>
<tr>
<td>3</td>
<td>(s:p::?:?)</td>
<td>11</td>
<td>(?:::o:)</td>
</tr>
<tr>
<td>4</td>
<td>(s:p:o:?)</td>
<td>12</td>
<td>(?:?:c)</td>
</tr>
<tr>
<td>5</td>
<td>(s:p:o:c)</td>
<td>13</td>
<td>(s:::c)</td>
</tr>
<tr>
<td>6</td>
<td>(?:p::?:?)</td>
<td>14</td>
<td>(s:p::c)</td>
</tr>
<tr>
<td>7</td>
<td>(?:p:o:?)</td>
<td>15</td>
<td>(?:p::c)</td>
</tr>
<tr>
<td>8</td>
<td>(?:p:o:c)</td>
<td>16</td>
<td>(s:::o:)</td>
</tr>
</tbody>
</table>
Index Patterns
Six prefix-indexes for quads

- Requires six indexes to service all 16 quad patterns
  - assuming prefix lookups
    - e.g., can lookup SP in SPOC index... (~indexes should be sorted)

GRAPH ?c { :ted?p ?o }
  pattern: (s:??:?)
  index: spoc

GRAPH :dblp { :ted ?p ?o }
  pattern: (s:??:c)
  index: csp

GRAPH ?c { ?s rdf:type foaf:Person }
  pattern: (?:p:o:?)
  index: poc
Index Implementation
Blocked, Sorted File

**Pros:**
- Fast! sequential reads (no OIDs)
- Fast load
- Good for simple queries, lots of results
- Configurable block sizes

**Cons:**
- Does not support updates!
- Does not support updates!!!
### Dictionary Encoding / Object IDs

#### Data Table

<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

#### Dictionary

<table>
<thead>
<tr>
<th>OID</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Frank</td>
</tr>
</tbody>
</table>

#### Pros:
- Can load more data in memory
- Faster to compute joins
- Better RISC-style processing [RDF-3X]
- Smaller on-disk footprint

#### Cons:
- Maintain a potentially massive dictionary
- Slower to externalise streaming results
- FILTERs optimisations are trickier
Query Processing (1/4)

?person foaf:name ?name .
?person foaf:based_near dbpedia:Karlsruhe .

1. get aidans friends
2. based in Karlsruhe?
3. name?

7 billion
0.3 million
2
Query Processing (2/4)
Push FILTER expressions

\[
\text{?person foaf:name ?name .}
\text{?person foaf:age ?age .}
\text{?person foaf:based_near dbpedia:Karlsruhe .}
\text{FILTER ( ?age > 110 )}
\]

- Filter intermediate results as soon as possible

1. get Karlsruhe folks
2. get age: 110+?
3. get name

7 billion  400  0.5 million
Query Processing (3/4)
Use Index, Push FILTER

?-person foaf:name ?name .
?-person foaf:age ?age .
?-person foaf:based_near dbpedia:Karlsruhe .
FILTER (?age > 110)

- If you have the right index (sorted by PO or [maybe] OP prefix with numeric order on O), seek directly to 110+ for age

get name

in Karlsruhe?

jump directly to age > 110

7 billion
0.5 million
400
...give me resources mentioning “Bonn ISWC”

- Can use FILTER/REGEX in SPARQL...

  
  FILTER ( REGEX(?o, "Bonn") && REGEX(?o2, "ISWC") )

  - ...but that sucks and is expensive
  - ...need an IR-style inverted index, not FILTER

- Most SPARQL engines support custom keyword index/syntax
  - Against literals or concatenation of literals
  - Apache Lucene! ...perfect for the job!
<table>
<thead>
<tr>
<th>Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML und URIs</td>
</tr>
<tr>
<td>Einleitung in RDF</td>
</tr>
<tr>
<td>RDF Schema</td>
</tr>
<tr>
<td>Logik – Grundlagen</td>
</tr>
<tr>
<td>Semantik von RDF(S)</td>
</tr>
<tr>
<td>SPARQL – Syntax und Intuition</td>
</tr>
<tr>
<td>Semantik von SPARQL</td>
</tr>
<tr>
<td>Linked Data</td>
</tr>
<tr>
<td><strong>Semantic Search</strong></td>
</tr>
<tr>
<td>OWL – Syntax und Intuition I</td>
</tr>
<tr>
<td>OWL – Syntax und Intuition II</td>
</tr>
<tr>
<td>OWL – Semantik und Reasoning</td>
</tr>
<tr>
<td>Konjunktive Anfragen und Regelsprachen</td>
</tr>
<tr>
<td>Applications</td>
</tr>
</tbody>
</table>
Attribution

- Slides erstellt von Andreas Harth, Aidan Hogan, Spyros Kotoulas, Jacopo Urbani für das Tutorial “Scalable Integration and Processing of Linked Data”