DH2022 Tokyo

DH2022

DIGITAL HUMANITIES 2022

Linked Data, Semantic Web, SPARQL

Christian Chiarcos

University of Cologne, Germany

University of Cologne

Nexus Linguarum

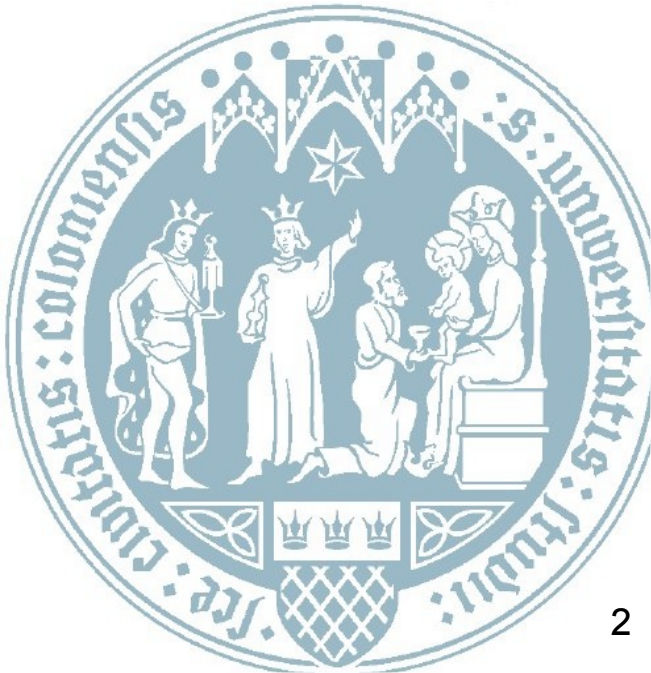
LiODi Linked Open Dictionaries

Prêt-à-LLOD

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Technical Foundations

- RDF & Linked Data: Introduction by Example
- Semantic Web: Basics
- SPARQL: Using it!



Linking Data for DH: Motivation

Many resources in humanities exist in isolation

- Disconnected from other resources (silos)
- Proprietary and heterogeneous formats
- Different representation schemas,

  different communities (CIDOC, Web Annotation, TEI, ...)
- Non-standard access means (APIs)
- Different access levels (from “write me an email” to web services)
- Several repositories with different metadata and schemas

Lack of interoperability across datasets that are potentially complementary & that could be combined together

img source https://www.flickr.com/photos/docsearls/5500714140, CC-BY 2.0

Linked Data

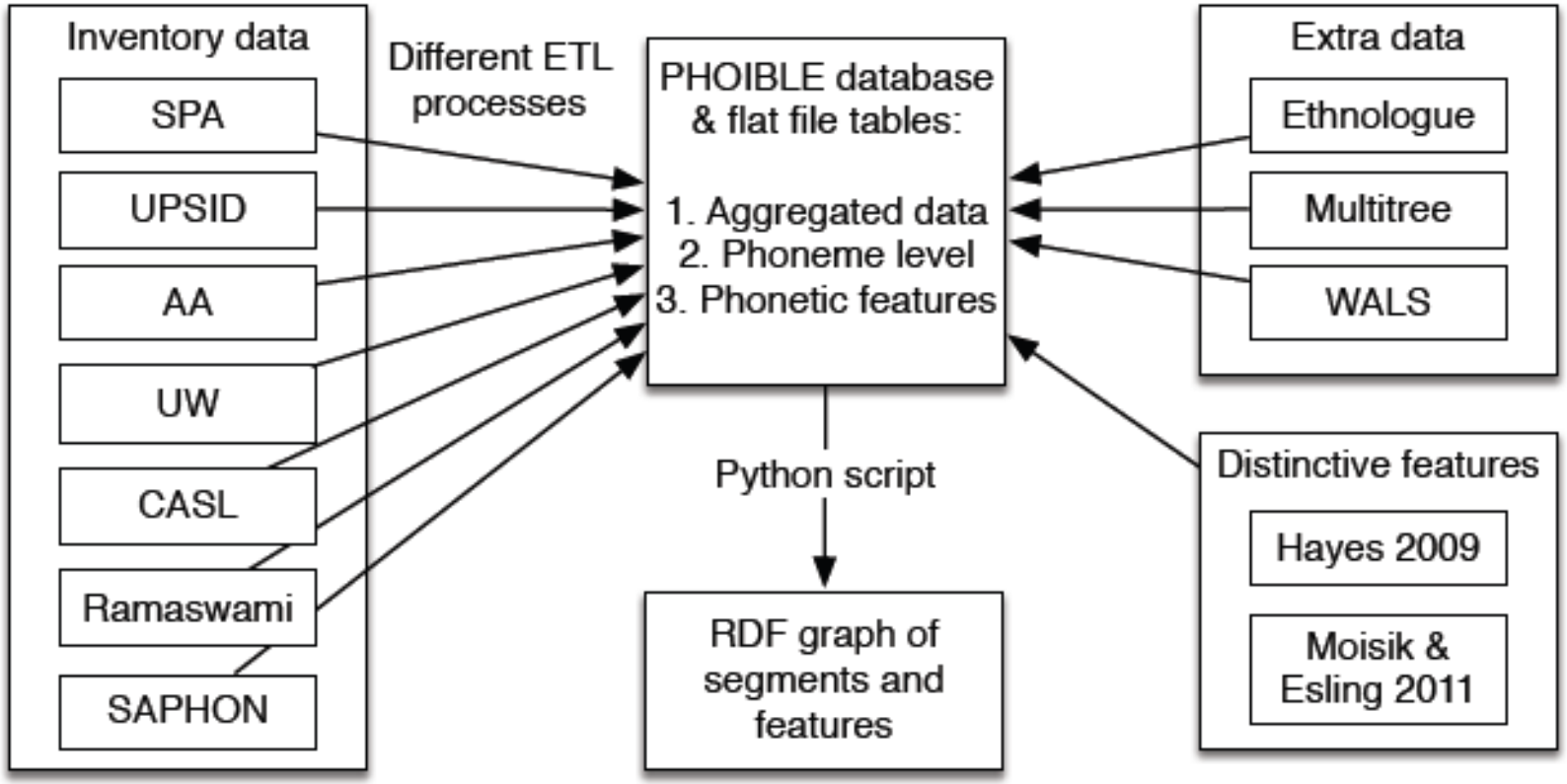
Introduction by Example

From Tables to RDF ...

- PHOnetics Information Base and LExicon (PHOIBLE)
  - Moran, S. (2012). Using Linked Data to Create a Typological Knowledge Base. In Chiarcos, C., Nordhoff, S., and Hellmann, S. (eds), *Linked Data in Linguistics: Representing and Connecting Language Data and Language Metadata*. Springer, Heidelberg.
- Phoneme inventories and phonological features
  - Covers ~20% of the world's spoken languages
  - Compiled from various sources, at first as a flat table (list)

(Example courtesy of Steven Moran, University of Neuchâtel)

From Tables to RDF ...



(Example courtesy of Steven Moran, University of Neuchâtel)

From Tables to RDF ...

Source	id	ISO639-3	trump	root	wals_genus	population	latitude	longitude	phoneme_id	glyph_id	glyph	class	comb	num
SPA	1	kor	1	asis	Korean	42,000,000	37:30	128:0	1	1	t͡ʃʰ	cons	c-d-c-c	4
SPA	3	lbe	1	ncau	Lak-Dargwa	157,000	42:0	47:0	124	1	t͡ʃʰ	cons	c-d-c-c	4
SPA	5	kat	1	kart	Kartvelian	3,900,000	42:0	44:0	203	1	t͡ʃʰ	cons	c-d-c-c	4
SPA	6	bsk	1	asis	Burushaski	87,000	36:30	74:30	240	1	t͡ʃʰ	cons	c-d-c-c	4
SPA	14	khm	1	ausa	Khmer	12,300,000	12:30	105:0	632	19	uː	vowel	v-d	2
SPA	27	tha	1	taik	Kam-Tai	20,200,000	15:00	100:40	1150	19	uː	vowel	v-d	2

(Example courtesy of Steven Moran, University of Neuchâtel)

From Tables to RDF ...

Source	id	ISO639-3	trump	root	wals_genus	population	latitude	longitude	phoneme_id	glyph_id	glyph	class	comb	num
SPA	1	kor	1	asis	Korean	42,000,000	37:30	128:0	1	1	$t\underline{\int}^h$	cons	c-d-c-c	4
SPA	3	lbe	1	ncau	Lak-Dargwa	157,000	42:0	47:0	124	1	$t\underline{\int}^h$	cons	c-d-c-c	4
SPA	5	kat	1	kart	Kartvelian	3,900,000	42:0	44:0	203	1	$t\underline{\int}^h$	cons	c-d-c-c	4
SPA	6	bsk	1	asis	Burushaski	87,000	36:30	74:30	240	1	$t\underline{\int}^h$	cons	c-d-c-c	4
SPA	14	khm	1	ausa	Khmer	12,300,000	12:30	105:0	632	19	u:	vowel	v-d	2
SPA	27	tha	1	taik	Kam-Tai	20,200,000	15:00	100:40	1150	19	u:	vowel	v-d	2

Subject
(primary key)

(Example courtesy of Steven Moran, University of Neuchâtel)

From Tables to RDF ...

Property
(„Relation“)

Source	id	ISO639-3	trump	root	wals_genus	population	latitude	longitude	phoneme_id	glyph_id	glyph	class	comb	num
SPA	1	kor	1	asis	Korean	42,000,000	37:30	128:0	1	1	$t\underset{\sim}{ʃ}^{h}$	cons	c-d-c-c	4
SPA	3	lbe	1	ncau	Lak-Dargwa	157,000	42:0	47:0	124	1	$t\underset{\sim}{ʃ}^{h}$	cons	c-d-c-c	4
SPA	5	kat	1	kart	Kartvelian	3,900,000	42:0	44:0	203	1	$t\underset{\sim}{ʃ}^{h}$	cons	c-d-c-c	4
SPA	6	bsk	1	asis	Burushaski	87,000	36:30	74:30	240	1	$t\underset{\sim}{ʃ}^{h}$	cons	c-d-c-c	4
SPA	14	khm	1	ausa	Khmer	12,300,000	12:30	105:0	632	19	u:	vowel	v-d	2
SPA	27	tha	1	taik	Kam-Tai	20,200,000	15:00	100:40	1150	19	u:	vowel	v-d	2

Subject

(Example courtesy of Steven Moran, University of Neuchâtel)

From Tables to RDF ...

Property
(„Relation“)

Source	id	ISO639-3	trump	root	wals_genus	population	latitude	longitude	phoneme_id	glyph_id	glyph	class	comb	num
SPA	1	kor	1	asis	Korean	42,000,000	37:30	128:0	1	1	tʃʰ	cons	c-d-c-c	4
SPA	3	lbe	1	ncau	Lak-Dargwa	157,000	42:0	47:0	124	1	tʃʰ	cons	c-d-c-c	4
SPA	5	kat	1	kart	Kartvelian	3,900,000	42:0	44:0	203	1	tʃʰ	cons	c-d-c-c	4
SPA	6	bsk	1	asis	Burushaski	87,000	36:30	74:30	240	1	tʃʰ	cons	c-d-c-c	4
SPA	14	khm	1	ausa	Khmer	12,300,000	12:30	105:0	632	19	u:	vowel	v-d	2
SPA	27	tha	1	taik	Kam-Tai	20,200,000	15:00	100:40	1150	19	u:	vowel	v-d	2

Subject

Object

(Example courtesy of Steven Moran, University of Neuchâtel)

From Tables to RDF ...

Property
(„Relation“)

Source	id	ISO639-3	trump	root	wals_genus	population	latitude	longitude	phoneme_id	glyph_id	glyph	class	comb	num
SPA	1	kor	1	asis	Korean	42,000,000	37:30	128:0	1	1	t͡ʃʰ	cons	c-d-c-c	4
SPA	3	lbe	1	ncau	Lak-Dargwa	157,000	42:0	47:0	124	1	t͡ʃʰ	cons	c-d-c-c	4
SPA	5	kat	1	kart	Kartvelian	3,900,000	42:0	44:0	203	1	t͡ʃʰ	cons	c-d-c-c	4
SPA	6	bsk	1	asis	Burushaski	87,000	36:30	74:30	240	1	t͡ʃʰ	cons	c-d-c-c	4
SPA	14	khm	1	ausa	Khmer	12,300,000	12:30	105:0	632	19	u:	vowel	v-d	2
SPA	27	tha	1	taik	Kam-Tai	20,200,000	15:00	100:40	1150	19	u:	vowel	v-d	2

Subject

Object

1. Decompose tables into RDF triples, i.e.,
   - entity attribute value resp.
   - Subject Property Object

(Example courtesy of Steven Moran, University of Neuchâtel)

From Tables to RDF ...

Property („Relation“)

Source	id	ISO639-3	trump	root	wals_genus	population	latitude	longitude	phoneme_id	glyph_id	glyph	class	comb	num
SPA	1	kor	1	asis	Korean	42,000,000	37:30	128:0	1	1	t͡ʃʰ	cons	c-d-c-c	4
SPA	3	lbe	1	ncau	Lak-Dargwa	157,000	42:0	47:0	124	1	t͡ʃʰ	cons	c-d-c-c	4
SPA	5	kat	1	kart	Kartvelian	3,900,000	42:0	44:0	203	1	t͡ʃʰ	cons	c-d-c-c	4
SPA	6	bsk	1	asis	Burushaski	87,000	36:30	74:30	240	1	t͡ʃʰ	cons	c-d-c-c	4
SPA	14	khm	1	ausa	Khmer	12,300,000	12:30	105:0	632	19	u:	vowel	v-d	2
SPA	27	tha	1	taik	Kam-Tai	20,200,000	15:00	100:40	1150	19	u:	vowel	v-d	2

Subject

Object

1. Decompose tables into RDF triples, i.e.,
   - entity attribute value resp.
   - Subject Property Object

:tha —phoible:hasSegment→ :u_

From Tables to RDF ...

Property
(„Relation“)

Source	id	ISO639-3	trump	root	wals_genus	population	latitude	longitude	phoneme_id	glyph_id	glyph	class	comb	num
SPA	1	kor	1	asis	Korean	42,000,000	37:30	128:0	1	1	$t\underline{ʃ}^h$	cons	c-d-c-c	4
SPA	3	lbe	1	ncau	Lak-Dargwa	157,000	42:0	47:0	124	1	$t\underline{ʃ}^h$	cons	c-d-c-c	4
SPA	5	kat	1	kart	Kartvelian	3,900,000	42:0	44:0	203	1	$t\underline{ʃ}^h$	cons	c-d-c-c	4
SPA	6	bsk	1	asis	Burushaski	87,000	36:30	74:30	240	1	$t\underline{ʃ}^h$	cons	c-d-c-c	4
SPA	14	khm	1	ausa	Khmer	12,300,000	12:30	105:0	632	19	u:	vowel	v-d	2
SPA	27	tha	1	taik	Kam-Tai	20,200,000	15:00	100:40	1150	19	u:	vowel	v-d	2

Subject

Object

1. Decompose tables into RDF triples, i.e.,
   - entity attribute value resp.
   - Subject Property Object

```
:tha    phoible:hasSegment    :u_    .
```

Turtle format
triples separated by .

From Tables to RDF ...

Source	id	ISO639-3	trump	root	wals_genus	population	latitude	longitude	phoneme_id	glyph_id	glyph	class	comb	num
SPA	1	kor	1	asis	Korean	42,000,000	37:30	128:0	1	1	tʃʰ	cons	c-d-c-c	4
SPA	3	lbe	1	ncau	Lak-Dargwa	157,000	42:0	47:0	124	1	tʃʰ	cons	c-d-c-c	4
SPA	5	kat	1	kart	Kartvelian	3,900,000	42:0	44:0	203	1	tʃʰ	cons	c-d-c-c	4
SPA	6	bsk	1	asis	Burushaski	87,000	36:30	74:30	240	1	tʃʰ	cons	c-d-c-c	4
SPA	14	khm	1	ausa	Khmer	12,300,000	12:30	105:0	632	19		vowel	v-d	2
SPA	27	tha	1	tail	Kam-Tai	15:00	100:40	1150	19		u:	vowel	v-d	2

hasSegment

Subject

Object

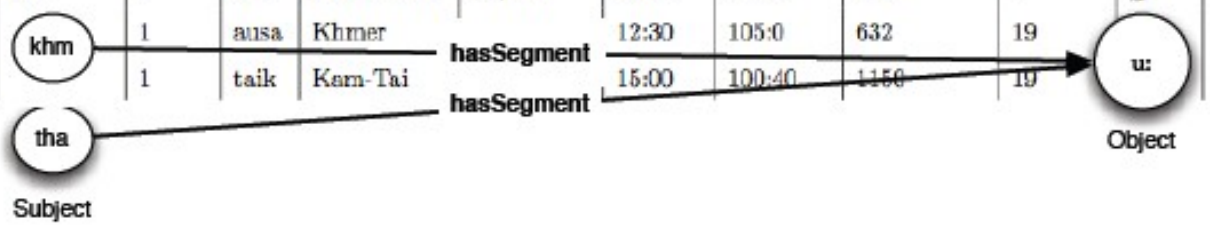
1. Decompose tables into RDF triples, i.e.,
   - entity attribute value resp.
   - Subject Property Object

```
:tha    phoible:hasSegment    :u_    .
```

Turtle format
triples separated by .

From Tables to RDF ...

Source	id	ISO639-3	trump	root	wals_genus	population	latitude	longitude	phoneme_id	glyph_id	glyph	class	comb	num
SPA	1	kor	1	asis	Korean	42,000,000	37:30	128:0	1	1	t_j^h	cons	c-d-c-c	4
SPA	3	lbe	1	ncau	Lak-Dargwa	157,000	42:0	47:0	124	1	t_j^h	cons	c-d-c-c	4
SPA	5	kat	1	kart	Kartvelian	3,900,000	42:0	44:0	203	1	t_j^h	cons	c-d-c-c	4
SPA	6	bsk	1	asis	Burushaski	87,000	36:30	74:30	240	1	t_j^h	cons	c-d-c-c	4
SPA	14	khm	1	ausa	Khmer		12:30	105:0	632	19	u:	vowel	v-d	2
SPA	27	tha	1	taik	Kam-Tai		15:00	100:40	1150	19	u:	vowel	v-d	2

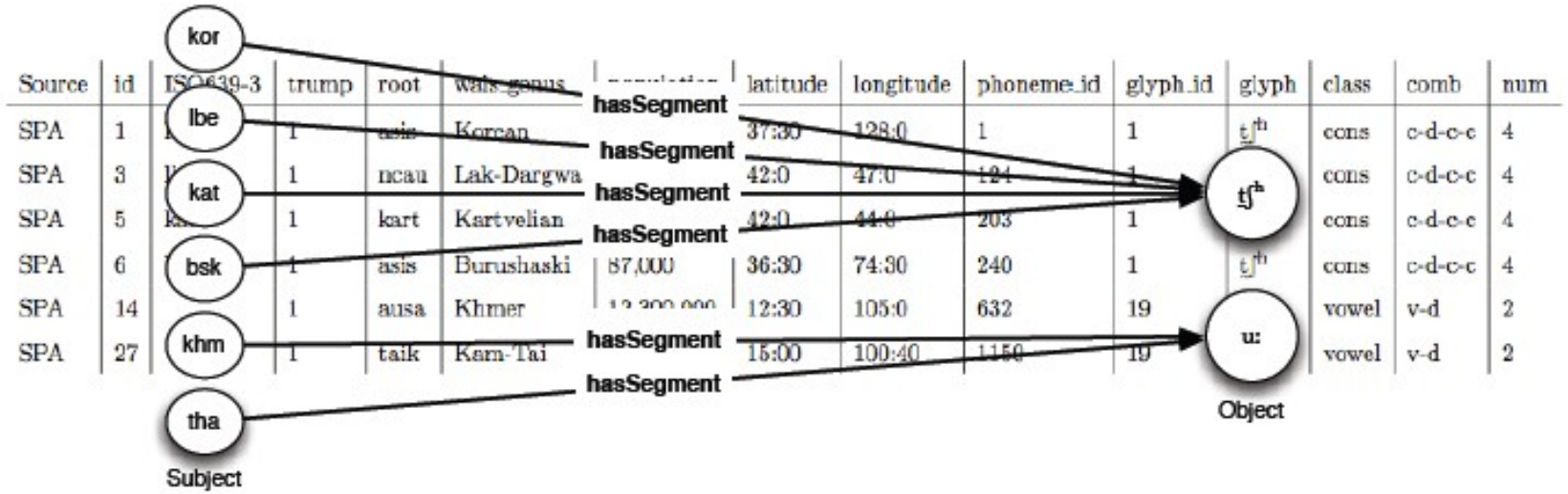


1. Decompose tables into RDF triples
2. Multiple triples constitute an RDF graph

```
:khm    phoible:hasSegment    :u_    .
:tha    phoible:hasSegment    :u_    .
```

Turtle format
triples separated by .

From Tables to RDF ...



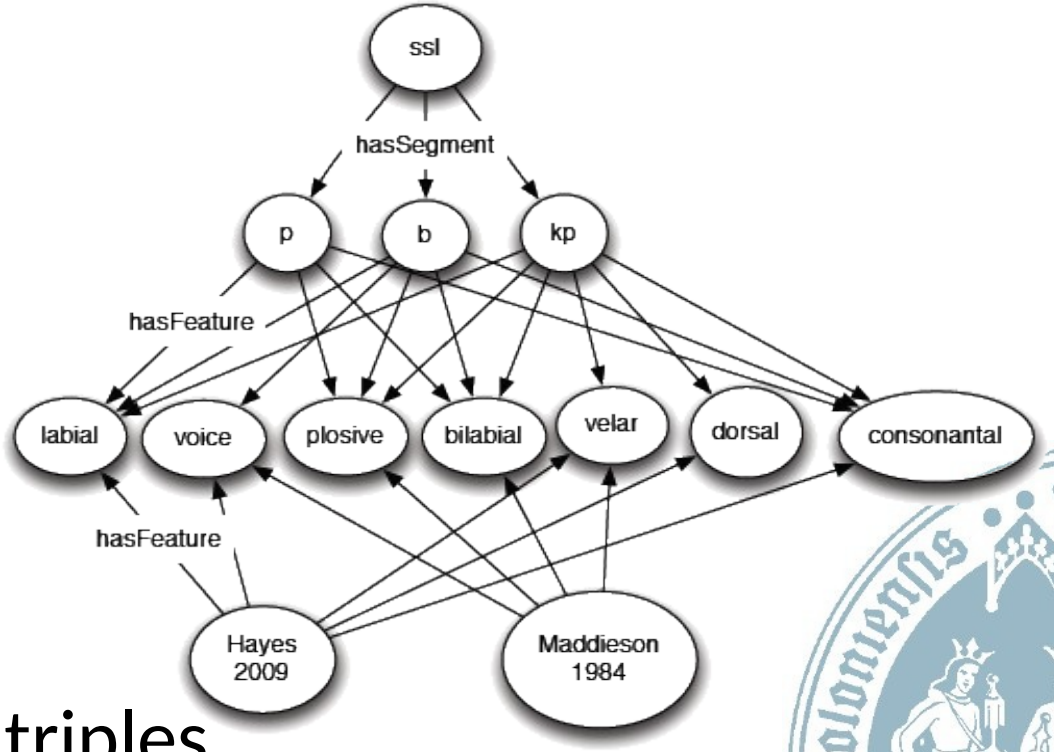
Source	id	ISO 639-3	trump	root	Wals genus	latitude	longitude	phoneme_id	glyph_id	glyph	class	comb	num
SPA	1		1	asis	Korean	37:30	128:0	1	1	tʃʰ	cons	c-d-c-c	4
SPA	3		1	ncau	Lak-Dargwa	42:0	47:0	124	1	tʃʰ	cons	c-d-c-c	4
SPA	5		1	kart	Kartvelian	42:0	44:0	203	1	tʃʰ	cons	c-d-c-c	4
SPA	6		1	asis	Burushaski	87,000	36:30	74:30	240	1	tʃʰ	cons	c-d-c-c	4
SPA	14		1	ausa	Khmer	12,300,000	12:30	105:0	632	19	u:	vowel	v-d	2
SPA	27		1	taik	Kam-Tai	15:00	100:40	1150	19	u:	vowel	v-d	2

1. Decompose tables into RDF triples
2. Multiple triples constitute an RDF graph

```
:khm    phoible:hasSegment    :u_    .
:tha    phoible:hasSegment    :u_    .
```

Turtle format
triples separated by .

From Tables to RDF ...

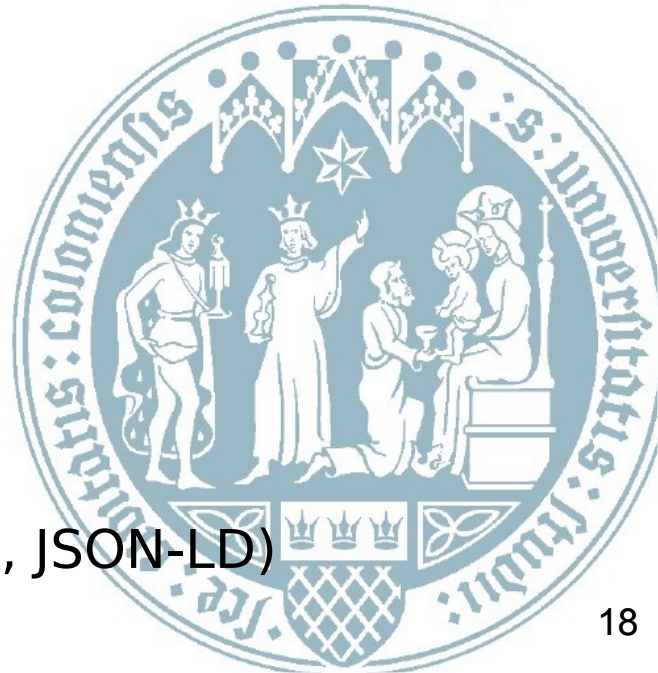


1. Decompose tables into RDF triples
2. Multiple triples constitute an RDF graph
3. A graph can aggregate triples from other sources, as well

From Tables to RDF ...

RDF is a conceptual model for graphs ...
While graphs can be represented in other ways, too, RDF tech allows us to

- Provide explicit semantics (RDF Schema, external knowledge graphs)
- Validate consistency (SHACL, ShEx; ontology languages)
- Infer implicit information (RDFS, OWL)
- Merge (not only syntactically, but semantically)
- Query in a standardized, platform-independent way
- Link (enrich with external data)
- Wrap or enrich non-RDF data sources (CSV2RDF, GRDDL; RDFa, JSON-LD)



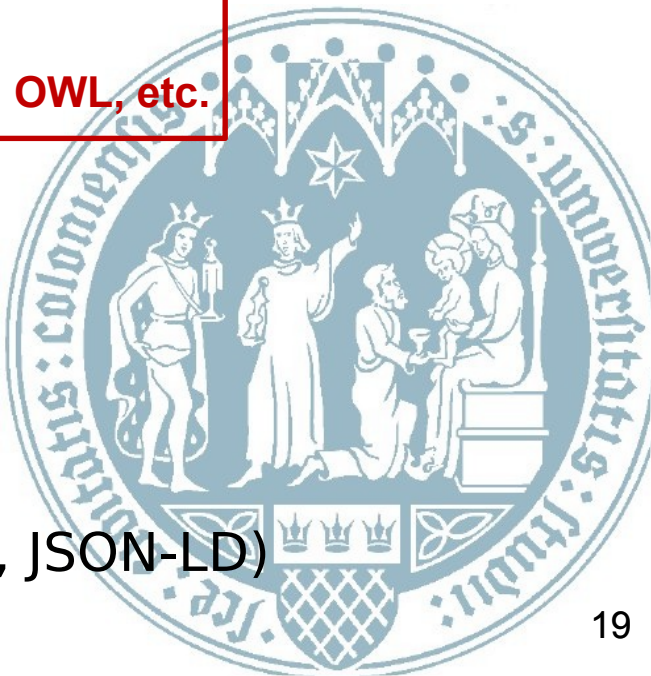
RDF tech builds on a pool of standards

RDF is a conceptual model for graphs ...
While graphs can be represented in other ways, too, RDF tech allows us to

- Provide explicit semantics (RDF Schema, external knowledge graphs)
- Validate consistency (SHACL, ShEx; ontology languages)
- Infer implicit information (RDFS, OWL)

RDFS, OWL, etc.

- Merge (not only syntactically, but semantically)
- Query in a standardized, platform-independent way
- Link (enrich with external data)
- Wrap or enrich non-RDF data sources (CSV2RDF, GRDDL; RDFa, JSON-LD)



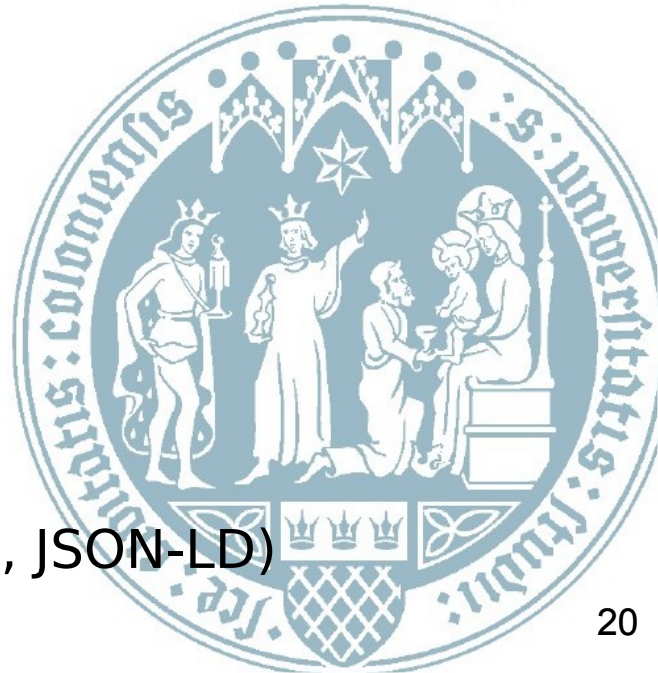
RDF tech builds on a pool of standards

RDF is a conceptual model for graphs ...
While graphs can be represented in other ways, too, RDF tech allows us to

- Provide explicit semantics (RDF Schema, external knowledge graphs)
- Validate consistency (SHACL, ShEx; ontology languages)
- Infer implicit information (RDFS, OWL)
- Merge (not only syntactically, but semantically)
- Query in a standardized, platform-independent way
- Link (enrich with external data)

URIs & SPARQL

- Wrap or enrich non-RDF data sources (CSV2RDF, GRDDL; RDFa, JSON-LD)



Uniform Resource Identifiers (URIs)

- Agree on a common vocabulary and names for entities
- **URIs** provide globally unique identifiers

"hasSegment"

string, not unambiguous

vs.

<http://mlode.nlp2rdf.org/resource/phoible/hasSegment>

URIs

vs.

@prefix phoible: <http://mlode.nlp2rdf.org/resource/phoible/>

... phoible:hasSegment ...

SPARQL

Merge data and query it using the W3C standard SPARQL (SPARQL Protocol and Query Language)

"the SQL of the Semantic Web"

```
SELECT DISTINCT ?language
WHERE {
    ?language phoible:hasSegment ?segment .
    ?segment phoible:hasFeature phoible:delayed_release
}
```

SPARQL complements a standard RDF syntax (Turtle) with variables and query operators …

Linked Data

- use URIs as names for things (1)
  - links to external URIs (links) allow us to retrieve more information from these sites
- **if** they can be resolved via HTTP (2)
- **and** provide information via RDF/SPARQL (3)
- **and** they include links to other URIs (4)

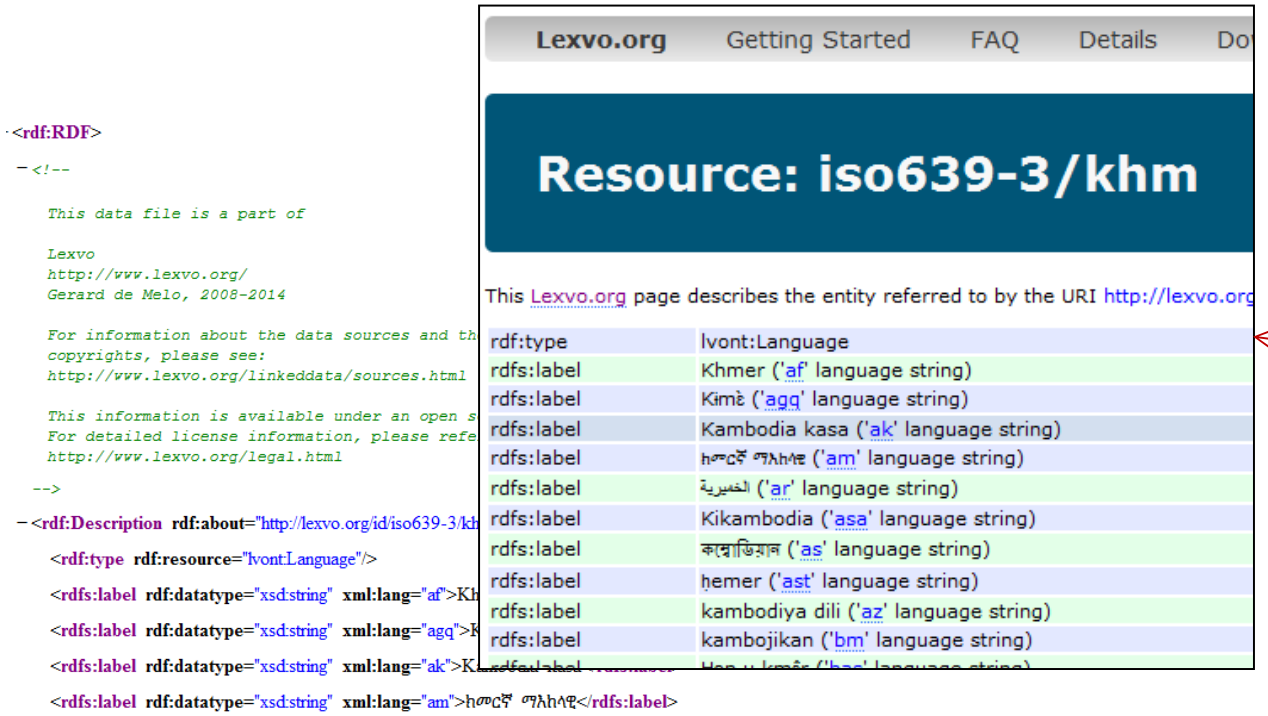
⇒ then, this is Linked Data (informal)

```
@prefix phoible: <http://mlode.nlp2rdf.org/resource/phoible/>
phoible:khm phoible:hasSegment "u:".
phoible:khm owl:sameAs <http://lexvo.org/id/iso639-3/khm>.
```

Turtle notation

http://www.w3.org/DesignIssues/LinkedData.html

From Tables to RDF to Linked Data



The resulting data can then be queried with an RDF data base
And exposed via a SPARQL end point

```
@prefix phoible: <http://mlode.nlp2rdf.org/resource/phoible/>
phoible:khm phoible:hasSegment "u:".
phoible:khm owl:sameAs <http://lexvo.org/id/iso639-3/khm>.
```

Turtle notation

Linked Open Data: The 5 star plan

★ Make your data available on the Web under an open license

★★ Make it available as structured data

(Excel sheet instead of image scan of a table)

★★★ Use a non-proprietary format

(CSV file instead of an Excel sheet)

★★★★ Use Linked Data format

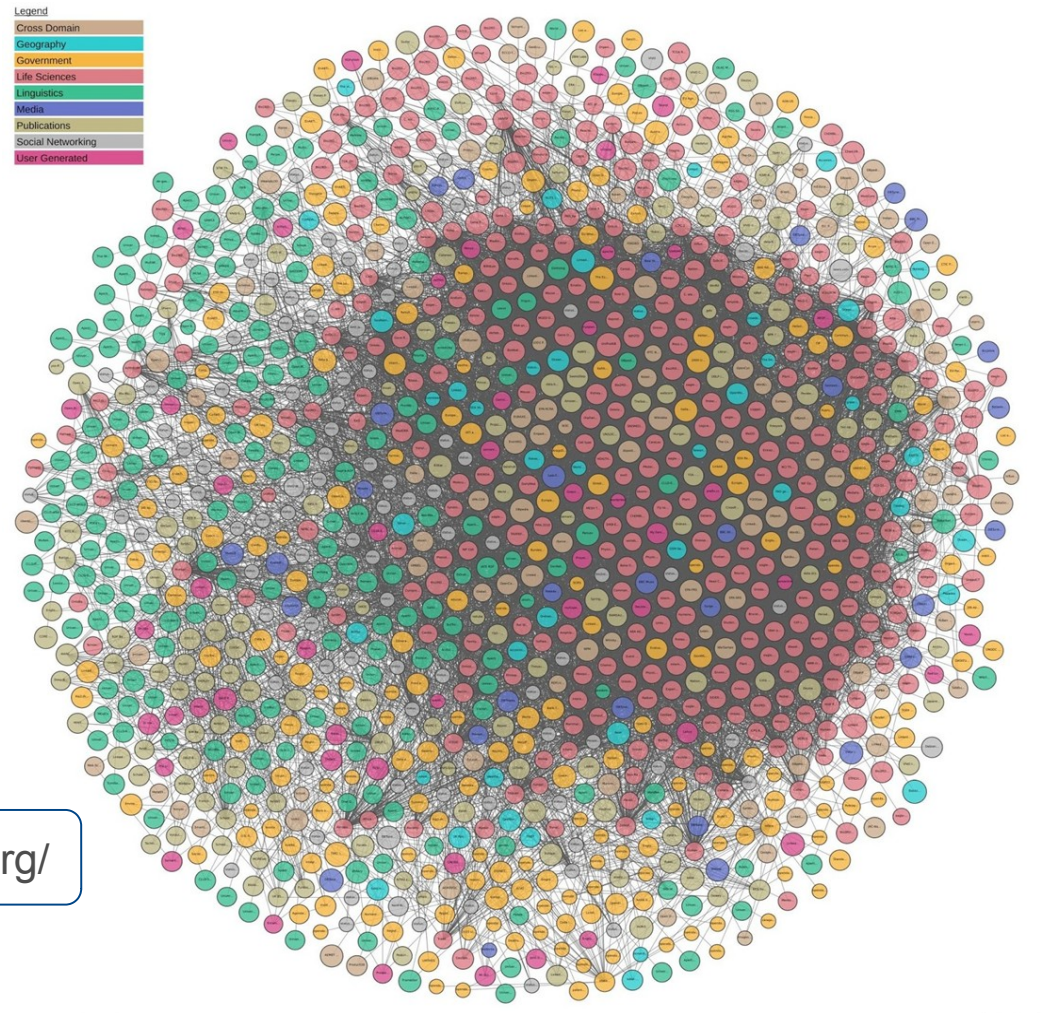
(URIs to identify things, RDF to represent data)

★★★★★ Link your data to other people's data to provide context

http://www.w3.org/DesignIssues/LinkedData.html (2006 addition)

LOD Cloud Today (lod-cloud.net)

“Linguistics” in green



http://linguistic-lod.org/

Semantic Web

Ontologies & Reasoning

Ontology (knowledge representation)

- technical formalization of a particular domain
  - normally comprised of two components
    - terms (“TBox”)
      - classes

        Tree, *Fruit*
      - properties

        bearsFruit
      - axioms (e.g., domain and range of a property)

        bearsFruit: *Tree* -> *Fruit*

Ontology & RDF

- There is a collection of vocabularies to model ontologies in RDF
  - RDF Schema (RDFS), Web Ontology Language (OWL)

    => can thus be represented in RDF, e.g., Turtle
    - classes

      Tree, Fruit

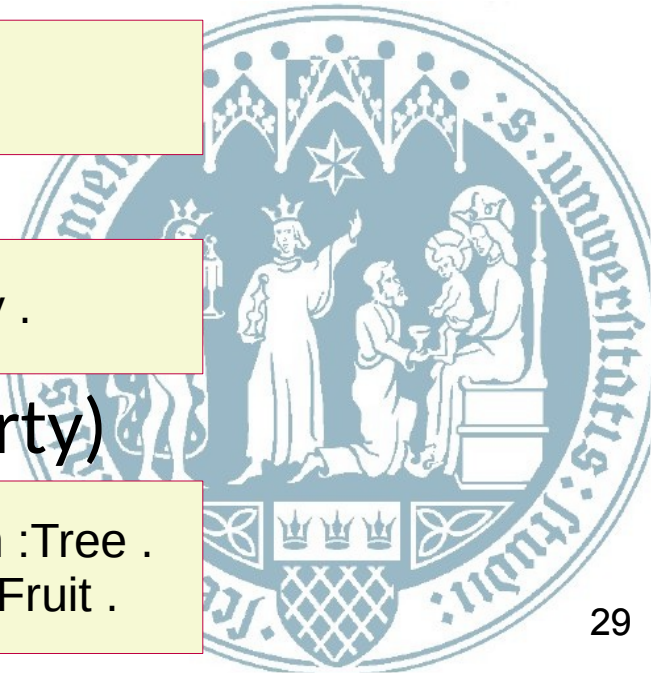
      ```
      :Tree rdf:type rdfs:Class .
      :Fruit rdf:type rdfs:Class .
      ```
    - properties

      bearsFruit

      ```
      :bearsFruit rdf:type owl:ObjectProperty .
      ```
    - axioms (e.g., domain and range of a property)

      bearsFruit: Tree -> Fruit

      ```
      :bearsFruit rdfs:domain :Tree .
      :bearsFruit rdfs:range :Fruit .
      ```



Ontology & RDF

- technical formalization of a particular domain
  - normally comprised of two components
    - statements (“atoms”, hence “ABox”)
      - instances

        Apple, *Appletree*
      - relations

        An Appletree bears an Apple

Ontology & RDF

- technical formalization of a particular domain
  - normally comprised of two components

    statements in Turtle

    - instances

      Apple, *Appletree*

    - relations

      An Appletree bears an Apple

```
:apple rdf:type rdfs:Resource .
:appletree rdf:type rdfs:Resource .
```

```
:appletree :bearsFruit :apple .
```

Inference

- deriving implicit information automatically (with a reasoner)
- by combining
  - an entailment regime (a specific type of semantics, e.g., RDFS)

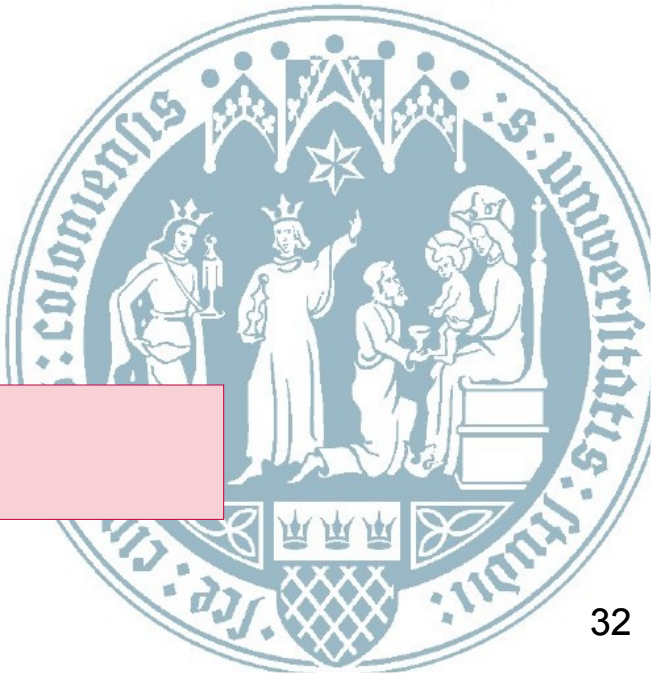
if *R* rdfs:domain *A* **and** we know that *x R y* **then** infer *x rdf:type A .*

if *R rdfs:range B* **and** we know that *x R y* **then** infer *y rdf:type B .*

  - TBox axioms and ABox statements

```
:bearsFruit rdfs:domain :Tree .
:bearsFruit rdfs:range :Fruit .
```

```
:appletree :bearsFruit :apple .
```



Inference

- deriving implicit information automatically (with a reasoner)
- by combining
  - an entailment regime (a specific type of semantics, e.g., RDFS)

if *R* rdfs:domain *A* **and** we know that *x R y* **then** infer *x rdf:type A .*

if *R rdfs:range B* **and** we know that *x R y* **then** infer *y rdf:type B .*

:appletree rdf:type :Tree .

:apple rdf:type :Fruit .

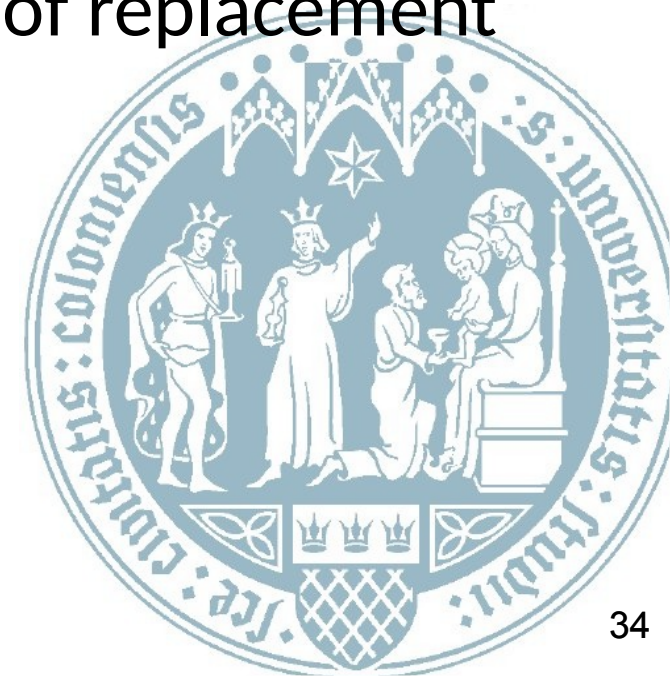
  - TBox axioms and ABox statements

:bearsFruit rdfs:domain :Tree .
:bearsFruit rdfs:range :Fruit .

:appletree :bearsFruit :apple .

Inference

- deriving implicit information automatically (with a reasoner)
- by combining
  - an entailment regime (a specific type of semantics, e.g., RDFS)
  - RDFS semantics can be implemented by means of replacement rules
    - some limits on expressivity
    - but very fast
  - OWL supports more advanced semantics
    - disjunction, negation, cardinality constraints



SW tech in the Linked Data era

- The original vision of the Semantic Web was automated web-scale reasoning
- This is still far away, but SW vocabularies and technology are widely used for
  - knowledge graphs
  - data modelling for RDF data
  - development of Linked Data vocabularies
  - validation and inference over Linked Data data structures

Basic SPARQL

SPARQL Query Language

A Little Excercise in Reading RDF

See below a fragment of real-world RDF/Turtle Data (JRC Names)

```
ns1:Aad_Stoop   rdf:type    dbpedia-owl:Person ;
    rdfs:label  "Ad Stoop" ,
        "Aad Stoop" ;
    dcterms:provenance  "The original data was retrieved from http
    did the RDF transformation, please refer to the original site
    skos:prefLabel  "Aad Stoop" ;
    ns1:hasId   "634034" ;
    dc:license  <http://langtech.jrc.it/JRC-Names.html> .
```

Can you draw a diagram?

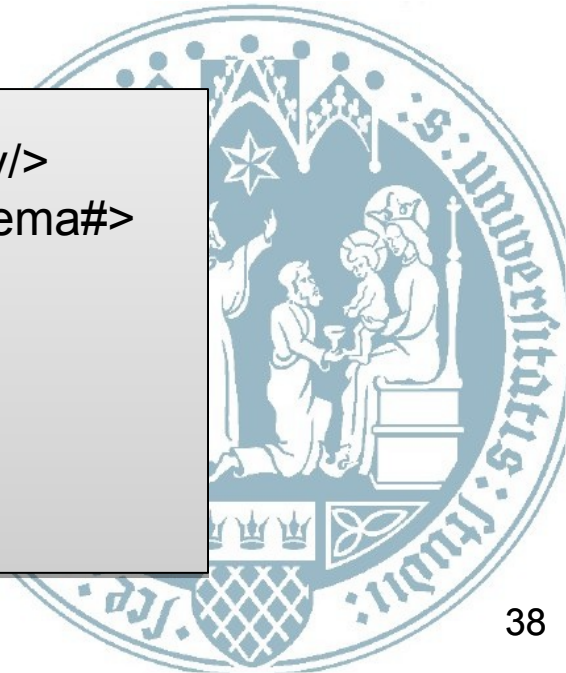
hints:

. separates triples
; separates triples with the same subject
, separates triples with the same subject and predictate (list of objects)

SPARQL

- „SQL meets Turtle“
  - extends Turtle-like triple syntax with
    - variables (marked with *?name)*, and
    - specification of return values

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?a ?l
WHERE {
      ?a a dbpedia-owl:Person .
      ?a rdfs:label ?l .
}
```

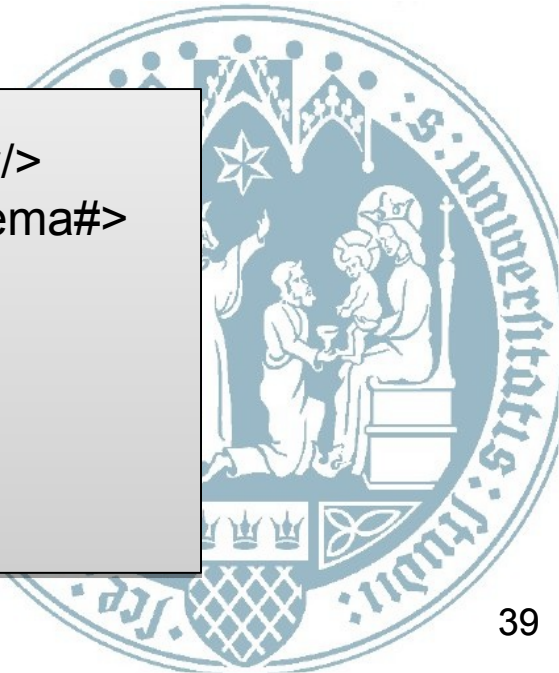


SPARQL

- PREFIX
  - namespace declaration (cf. Turtle)
- SELECT
  - specifies return values: variable binding
- WHERE
  - query
- triples
  - with
    variables

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?a ?l
WHERE {
      ?a a dbpedia-owl:Person .
      ?a rdfs:label ?l .
}
```

Variables



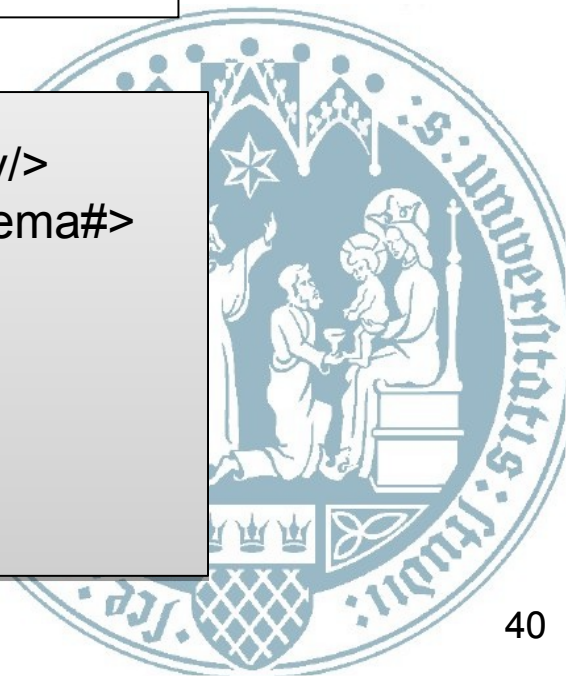
Example

- data

```
ns1:Aad_Stoop   rdf:type    dbpedia-owl:Person ;
    rdfs:label  "Ad Stoop" ,
        "Aad Stoop" ;
    dcterms:provenance  "The original data was retrieved from http
    did the RDF transformation, please refer to the original site
    skos:prefLabel  "Aad Stoop" ;
    ns1:hasId   "634034" ;
    dc:license  <http://langtech.jrc.it/JRC-Names.html> .
```

- query

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?a ?l
WHERE {
      ?a a dbpedia-owl:Person .
      ?a rdfs:label ?l .
}
```



Example

- data

```
ns1:Aad_Stoop   rdf:type    dbpedia-owl:Person ;
    rdfs:label  "Ad Stoop" ,
        "Aad Stoop" ;
    dcterms:provenance  "The original data was retrieved from http
    did the RDF transformation, please refer to the original site
    skos:prefLabel  "Aad Stoop" ;
    ns1:hasId   "634034" ;
    dc:license  <http://langtech.jrc.it/JRC-Names.html> .
```

- results*

?a	?l
ns1:Aad_Stoop	„Ad Stoop“
ns1:Aad_Stoop	„Aad Stoop“
…	…

* output format can be specified, can be table, triples, html, etc.

Links

- http://www.sparql.org/
  - links to authorative information
  - Online validator and processor
    - query public data without a local endpoint ;)
    - loads data from FROM clause

- If you prefer prose
  - https://en.wikibooks.org/wiki/SPARQL is quite usable

Advanced SPARQL

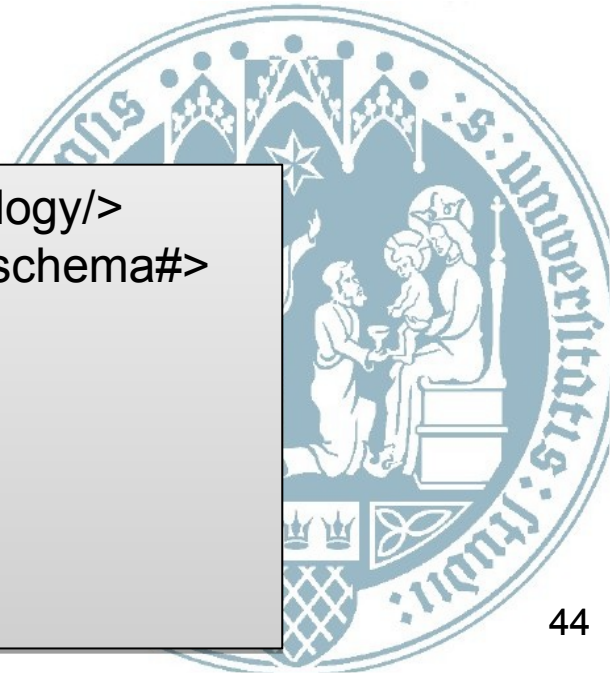
FILTER
BIND
functions
OPTIONAL, UNION, MINUS
SERVICE
UPDATE

Use these slides as reference when designing your own queries ;)

Restricting a result set

- *after* a result set is initialized with a series of statements, it can be filtered
- FILTER(...)
  - filter conditions aren‘t triples, but functions over variable values

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?a ?l
WHERE {
      ?a a dbpedia-owl:Person .
      ?a rdfs:label ?l .
      FILTER(strstarts(str(?l), “Peter”)
}
```



SPARQL Functions (Selection)

general

DATATYPE
STR
IRI
LANG
BOUND
IN
NOT IN
isBLANK

comparison

=
<
>
!=

boolean

&&
||
!

Test string equality with FILTER

```
SELECT ?p
WHERE {
        ?p rdfs:label ?label
        FILTER(str(?label) = “Georga W Busha”)
}
```

Test string equality with FILTER

```
SELECT ?p
WHERE {
      ?p rdfs:label ?label
      FILTER(str(?label) = “Georga W Busha”)
}
```

... or with BIND

```
SELECT ?p
WHERE {
      ?p rdfs:label ?label.
      BIND(str(?label) as ?plainLabel)
      FILTER(?plainlabel = “Georga W Busha”)
}
```

Modifiers: ORDER BY

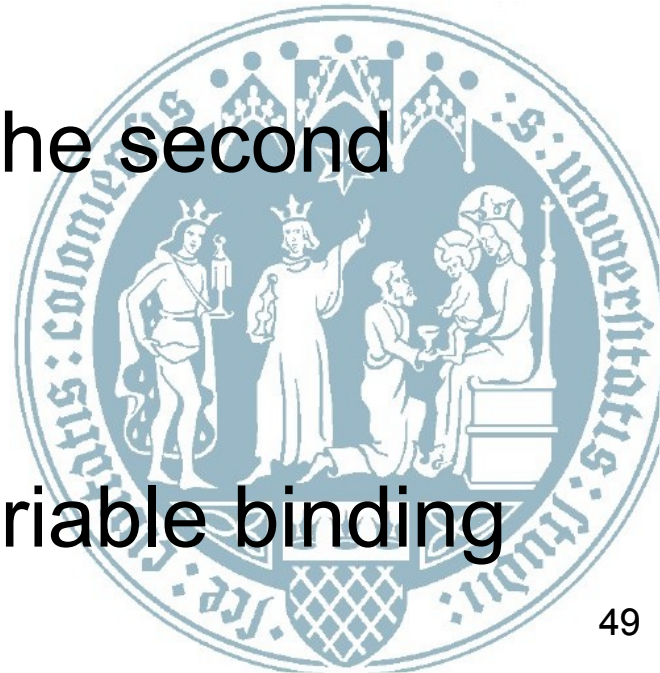
- sort the results of SELECT

```
SELECT ?p
WHERE {
      ?p a dbpedia-owl:Person.
      ?p rdfs:label ?l
}
ORDER BY ?l
```

p
http://mlode.nlp2rdf.org/resource/jrc-names/A_Flod
http://mlode.nlp2rdf.org/resource/jrc-names/Abdul_Kalam
http://mlode.nlp2rdf.org/resource/jrc-names/A_Petersen
http://mlode.nlp2rdf.org/resource/jrc-names/A_Thorbjørnsen
http://mlode.nlp2rdf.org/resource/jrc-names/Abdul_Kalam
http://mlode.nlp2rdf.org/resource/jrc-names/Aabid_Hussain_Khan
http://mlode.nlp2rdf.org/resource/jrc-names/Aad_de_Mos
http://mlode.nlp2rdf.org/resource/jrc-names/Aad_Goudriaan

Query operators

- OPTIONAL { ... }
  - if no variable binding for the sub-query in {...} can be found, return *null* values
- { ... } UNION { ... }
  - return the variable bindings of the first or the second sub-query
- MINUS { ... }
  - return no results if the sub-query has a variable binding



SPARQL Update

- SPARQL can not only be used for querying, but also for manipulating an RDF graph
- instead of SELECT, use
  - INSERT
  - DELETE

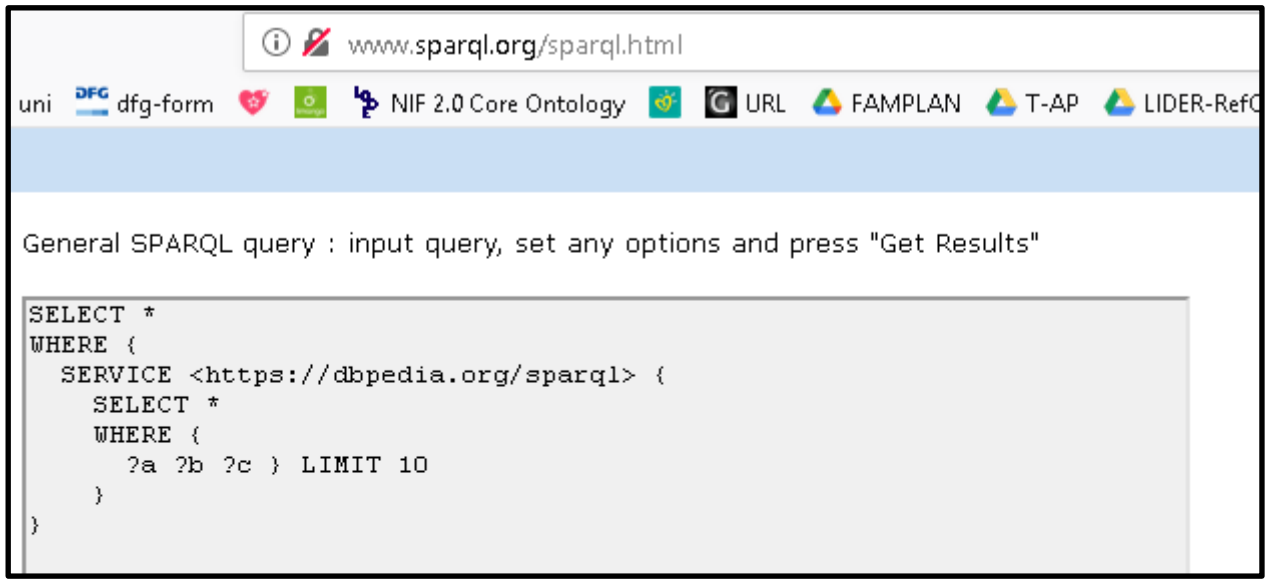
  Followed by triples in { ... }, these will be added or removed from the RDF graph

  (if authorized)

SERVICE

Federation (querying remote data)

- With the keyword SERVICE, SPARQL can be instructed to read an external web service (if authorized)



SERVICE

Federation (querying remote data)

- With the keyword SERVICE, SPARQL can be instructed to read an external web service (if authorized)
- External data dumps can also be read with a slightly different syntax
  - FROM: fetch data source (if authorized)
- Both functions can be tested online with sparql.org

Have fun!

Nicht sicher | sparql.org/sparql.html

SPARQLer - General purpose processor

General SPARQL query : input query, set any options and press "Get Results"

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ontolex: <http://www.w3.org/ns/lemon/ontolex#>
PREFIX vartrans: <http://www.w3.org/ns/lemon/vartrans#>
SELECT DISTINCT ?translation
WHERE {
        SERVICE <http://kaiko.getalp.org//sparql?default-graph-uri> {
                ?entry_fun ontolex:canonicalForm/ontolex:writtenRep "fun"@en.
                ?t (vartrans:translatableAs|^vartrans:translatableAs) ?entry_fun.
                ?t ontolex:canonicalForm/ontolex:writtenRep ?translation.
        }
        BIND(lang(?translation) as ?lang)

} ORDER BY ?lang ?translation LIMIT 20
```

Target graph URI (or use FROM in the query)

If no dataset is provided, the query will execute agains an empty one.

The query can contain use VALUES to set some variables.

Output: JSON

XSLT style sheet (blank for none): /xml-to-html.xsl

Force the accept header to text/plain regardless

Get Results

SPARQLer Query Results

translation
"веселие" @bg
"забава" @bg
"смешен" @bg
"Amüsement" @de
"Gaudi" @de
"Spaß" @de
"Vergnügen" @de
"lustig" @de
"spaßig" @de
"verlustieren" @de
"κέφι" @el
"diversión" @es
"divertido" @es
"embullarse" @es
"gracia" @es
"placer" @es
"hauska" @fi
"hauskanpito" @fi
"hauskuus" @fi
"huvi" @fi