Knowledge Engineering with Semantic Web Technologies

Lecture 3: Semantic Web Technologies - Part 2

01: How to Query RDF(S)? - SPARQL

Dr. Harald Sack
Hasso-Plattner-Institut for IT Systems Engineering
University of Potsdam
Spring 2014

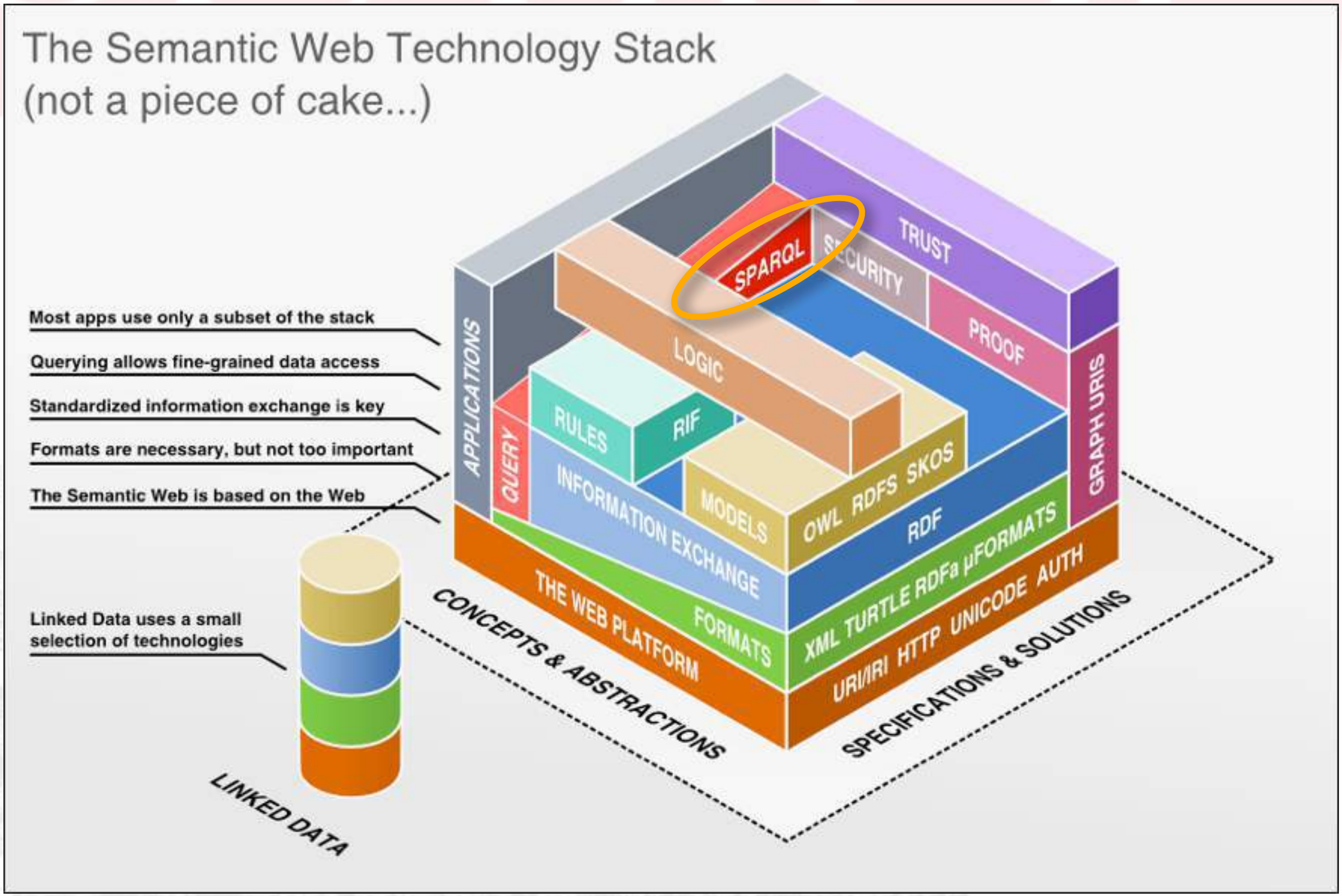
Lecture 3: Semantic Web Technologies - Pt.2

Open HPI Course: Knowledge Engineering with Semantic Web Technologies

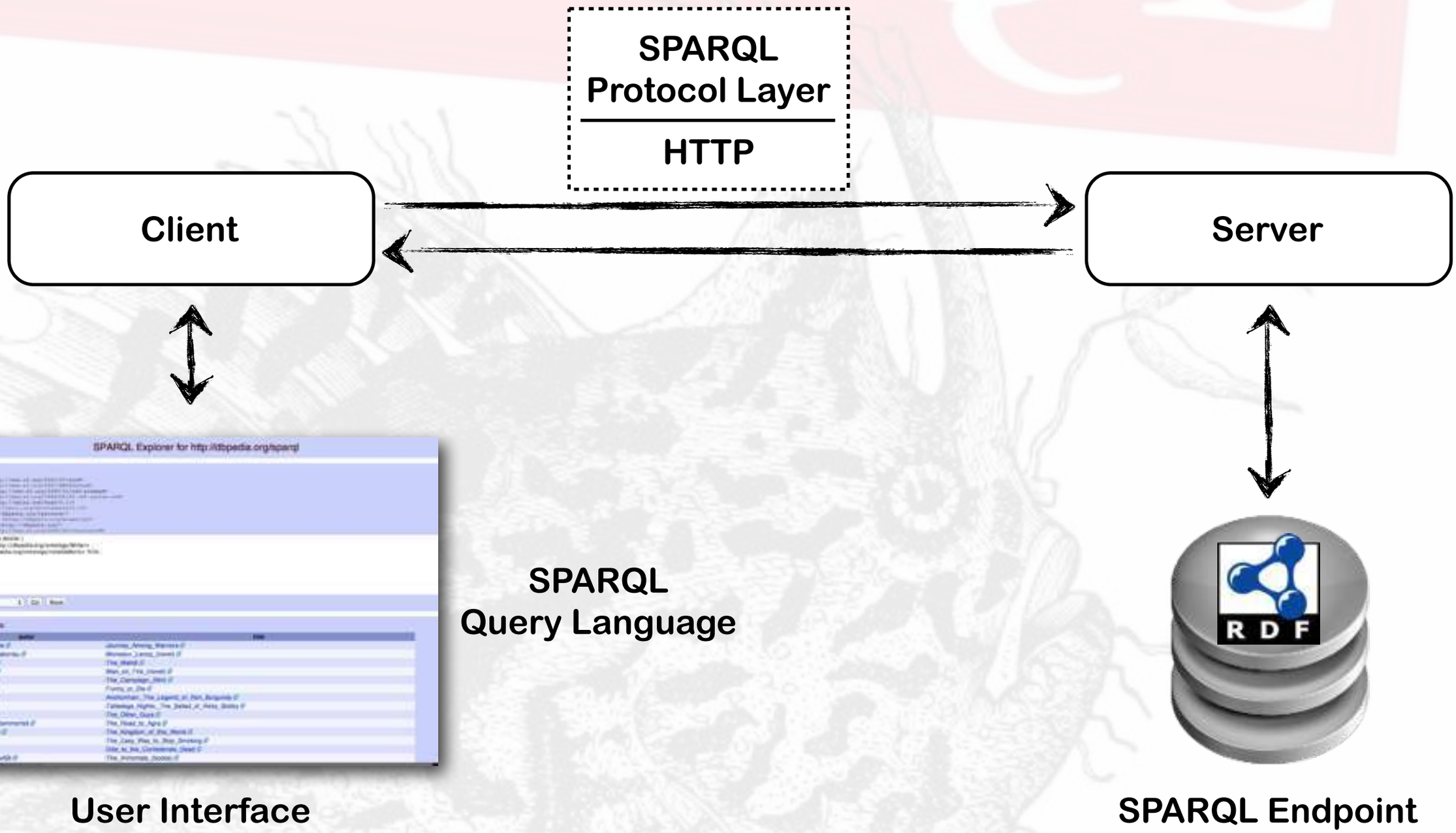
01 How to query RDF(S) ? - SPARQL

Open HPI Course: Knowledge Engineering with Semantic Web Technologies
Lecture 03: Semantic Web Technologies Pt.2

Semantic Web Technologies
Part 2



SPARQL - A Query Language for RDF



SPARQL - A Query Language for RDF

- **SPARQL P**rotocol **a**nd **R**DF **Q**uery **L**anguage is
  - a **Query Language** for RDF graph traversal (*SPARQL Query Language Specification*)
  - a **Protocol Layer**, to use SPARQL via http (*SPARQL Protocol for RDF Specification*)
  - an **XML Output Format Specification** for SPARQL queries (*SPARQL Query XML Results Format*)
  - W3C Standard (SPARQL 1.1, Mar 2013)
  - inspired by SQL

SPARQL 1.0

SPARQL 1.0 - Features

- **Extraction** of Data as
  - URIs, Blank Nodes, typed and untyped Literals.
  - RDF Subgraphs
- **Exploration** of Data via Query for unknown relations
- Execution of **complex Join Operations** on heterogeneous databases in a single query
- **Transformation** of RDF Data from one vocabulary into another
- **Construction** of new RDF Graphs based on RDF Query Graphs

For Queries we need Variables

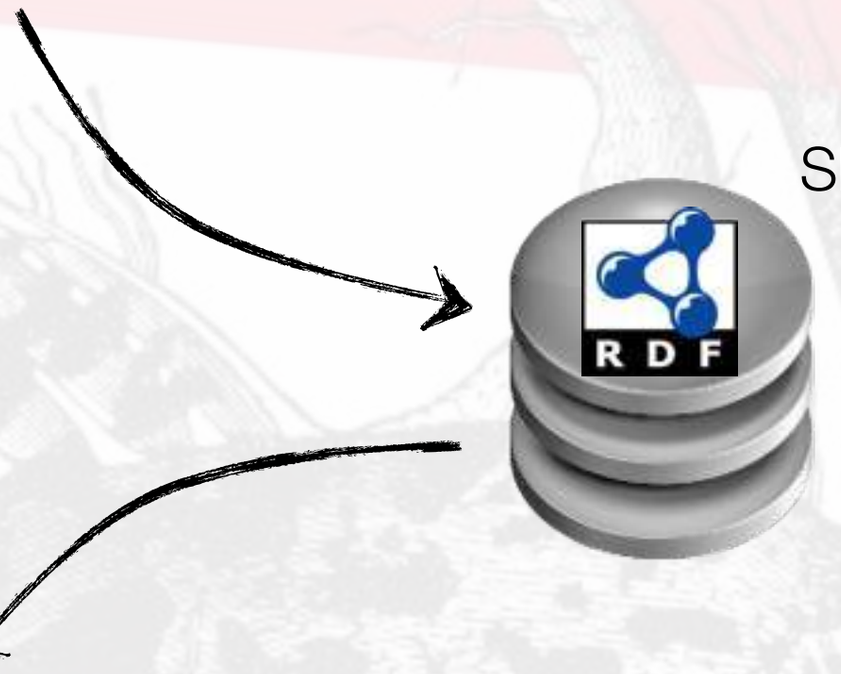
- SPARQL **Variables** are bound to RDF terms
  - e.g. **?journal, ?disease, ?price**
- In the same way as in SQL, a **Query** for variables is performed via **SELECT statement**
  - e.g. **SELECT ?title ?author ?published**
- A SELECT statement returns **Query Results** as a table

?title	?author	?published
1984	George Orwell	1948
Brave New World	Aldous Huxley	1932
Fahrenheid 451	Ray Bradbury	1953

SPARQL - Query Processing

SPARQL Query

```
SELECT ?title ?author ?published
```



SPARQL Endpoint

?title	?author	?published
1984	George Orwell	1948
Brave New World	Aldous Huxley	1932
Fahrenheid 451	Ray Bradbury	1953

SPARQL Result

SPARQL - Matching of Graph Patterns

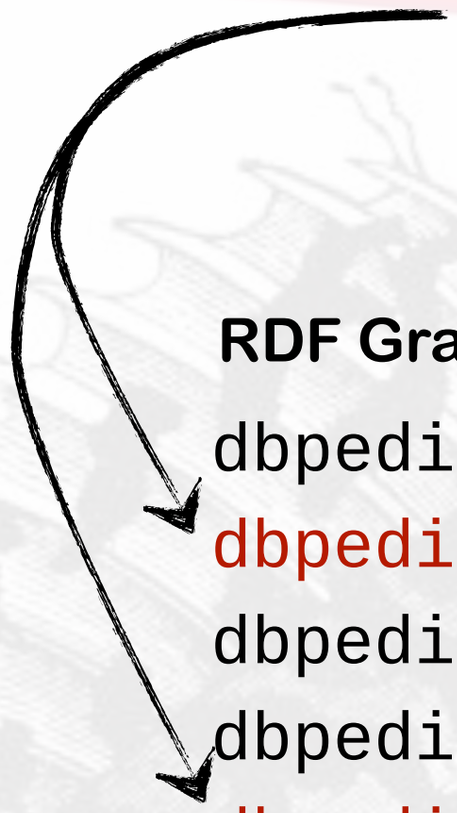
- SPARQL is based on **RDF Turtle serialization** and **basic graph pattern matching**.
- A **Graph Pattern** (Triple Pattern) is a RDF Triple that contains variables at any arbitrary place (Subject, Predicate, Object).
- **(Graph) Triple Pattern = Turtle + Variables**
- **Example:**
  - *Look for countries and their capitals:*

```
?country geo:capital ?capital .
```

SPARQL - Matching of Graph Patterns

Triple Pattern

```
?country geo:capital ?capital .
```



RDF Graph

```
dbpedia:Venezuela rdf:type dbpedia-owl:Country .
dbpedia:Venezuela geo:capital “Caracas“ .
dbpedia:Venezuela dbprop:language “Spanish“ .
dbpedia:Germany rdf:type dbpedia-owl:Country .
dbpedia:Germany geo:capital “Berlin“ .
dbpedia:Germany dbprop:language “German“ .
...
```

SPARQL - Matching of Graph Patterns

- **More Examples of (Graph) Triple Patterns:**
  - *Given a FOAF URI, find the name of a person:*

```
<http://hpi-web.de/id#haraldsack> foaf:name ?surname .
```

  - *Which persons have the family name “Schmidt“?*

```
?person pers:familyName “Schmidt“ .
```

SPARQL - Complex Query Patterns

- Graph Pattern can be combined to form **complex (conjunctive) queries** for RDF graph traversal.

  - *Find countries, their capitals and their population:*

```
?country geo:capital ?capital .
?country geo:population ?population .
```

  - *Given a FOAF URI, find the name of a person and her friends:*

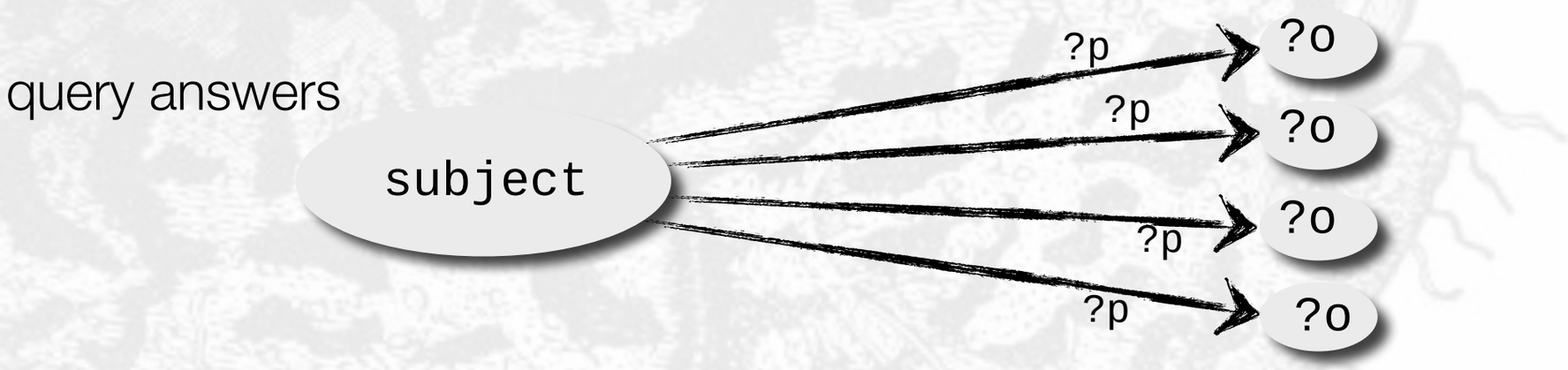
```
<http://hpi-web.de/id#haraldsack> foaf:name ?surname ;
                                  foaf:knows ?friend .
?friend foaf:name ?friend_surname .
```

SPARQL - Query Format

- inspired by SQL

```
SELECT ?p, ?o
WHERE { <subject> ?p ?o. }
```

- Triple in **`WHERE`**“ part defines graph query with variables **`?p`** and **`?o`**
- Query returns table with matching **`?p,?o`** pairs



SPARQL - General Query Format

- *Search all authors and the titles of their notable works:*

specifies namespaces

```
PREFIX :        <http://dbpedia.org/resource/>
PREFIX rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:    <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

SELECT ?author_name ?title

FROM <http://dbpedia.org/>

WHERE {
        ?author rdf:type dbpedia-owl:Writer .
        ?author rdfs:label ?author_name .
        ?author dbpedia-owl:notableWork ?work .
        ?work rdfs:label ?title .
}
```

specifies output variables

specifies graph to be queried

specifies graph pattern to be matched

SPARQL - General Query Format

- *Search all authors and the titles of their notable works **ordered by** authors in ascending order and **limit** the results to the first 100 starting the list at **offset** position 10:*

```
PREFIX :     <http://dbpedia.org/resource/>
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

SELECT ?author_name ?title
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbpedia-owl:Writer .
       ?author rdfs:label ?author_name .
       ?author dbpedia-owl:notableWork ?work .
       ?work rdfs:label ?title .
}
ORDER BY ASC (?author_name)
LIMIT 100
OFFSET 10
```

SPARQL - Blank Nodes

Blank Nodes can be used as

- **subject** or **object** of a triple pattern
- „non selectable“ **variables**

```
PREFIX hpi: <http://hpi-web.de/KnoEng14#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?name
WHERE {
        _:x rdf:type hpi:Lecture ;
            hpi:isManagedBy [rdfs:label ?name] .
}
```

- blank node identifiers might also occur in the results

SPARQL - First Hands-On

- From Wikipedia to DBpedia
  e.g. from http://en.wikipedia.org/wiki/George_Orwell to
  http://dbpedia.org/page/George_Orwell

- Browsing DBpedia with pubby
  e.g. using http://dbpedia.org/page/George_Orwell as a starting point to learn more about DBpedia structure and DBpedia ontologies

- Using DBpedia Sparql Endpoint with snorql
  http://dbpedia.org/snorql/
  and querying DBpedia via SPARQL

02 How to query RDF(S) ? – SPARQL (2)

Open HPI - Course: Semantic Web Technologies - Lecture 3: Semantic Web Basic Architecture

Knowledge Engineering with Semantic Web Technologies

Lecture 3: Semantic Web Technologies - Part 2

02: How to Query RDF(S)? - SPARQL (2)

Dr. Harald Sack
Hasso-Plattner-Institut for IT Systems Engineering
University of Potsdam
Spring 2014

Lecture 3: Semantic Web Technologies - Pt.2

Open HPI Course: Knowledge Engineering with Semantic Web Technologies

02 How to query RDF(S) ? – SPARQL (2)

Open HPI Course: Knowledge Engineering with Semantic Web Technologies
Lecture 03: Semantic Web Technologies Pt.2

SPARQL is not only a Query Language

- **SPARQL P**rotocol **a**nd **R**DF **Q**uery **L**anguage is
  - a **Query Language** for RDF Graph Traversal (*SPARQL Query Language Specification*)
  - a **Protocol Layer**, to use SPARQL via http (*SPARQL Protocol for RDF Specification*)
  - an **XML Output Format Specification** for SPARQL Queries (*SPARQL Query XML Results Format*)

SPARQL Result Format

- SPARQL results are given as well formed and valid XML documents

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  ...
</sparql>
```

- In a **<head>** element all variables of the SPARQL query are listed

```
<head>
    <variable name="x"/>
    <variable name="hpage"/>
    <variable name="name"/>
    <variable name="mbox"/>
    <variable name="blurb"/>
</head>
```

SPARQL Result Format

- For each SPARQL Query result exists a **<result>** element

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
    <variable name="x"/>
    ...
  </head>
  <results>
    <result>
      <binding name="x"> ... </binding>
      <binding name="hpage"> ... </binding>
    </result>

    <result> ... </result>
    ...
  </results>
</sparql>
```

single SPARQL query result

SPARQL Result Format

- Within a **<binding>** element a <head> variable is bound to a result

```
<result>
      <binding name="x">
        <bnode>r2</bnode>
      </binding>
      <binding name="hpage">
        <uri>http://work.example.org/bob/</uri>
      </binding>
      <binding name="name">
        <literal xml:lang="en">Bob</literal>
      </binding>
      <binding name="age">
        <literal datatype="http://www.w3.org/2001/XMLSchema#integer">
         30
        </literal>
      </binding>
      <binding name="mbox">
        <uri>mailto:bob@work.example.org</uri>
      </binding>
    </result>
```

variable bound to a result

SPARQL is not only a Query Language

- **SPARQL P**rotocol **a**nd **R**DF **Q**uery **L**anguage is
  - a **Query Language** for RDF Graph Traversal (*SPARQL Query Language Specification*)
  - a **Protocol Layer**, to use SPARQL via http (*SPARQL Protocol for RDF Specification*)
  - an **XML Output Format Specification** for SPARQL Queries (*SPARQL Query XML Results Format*)

SPARQL Protocol

- Method to query/respond of SPARQL queries via http
- A SPARQL URI consists out of 3 parts:

1. URL of a SPARQL endpoint
   (e.g. http://example.org/sparql)
2. RDF Graph(s) to be queried
   (optional, part of the query string,
   z.B. named-graph-uri=http://example.org/testrdf.rdf)
3. Query string
   (part of the query string, e.g. query=SELECT...)

```
http://example.org/sparql?named-graph-uri=http%3A%2F%2Fexample.org%2Ftestrdf&
query=SELECT+%3Freview_graph+WHERE+%7B%0D%0A++GRAPH+%3Frev
iew_graph+%7B%0D%0A+++++%3Freview+rev%3Arating+10+.%0D%0A++
%7D%0D%0A%7D
```

SPARQL Protocol - Example

- Simple SPARQL Query

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?book ?who
WHERE { ?book dc:creator ?who }
```

- HTTP Trace of the SPARQL Query

```
GET /sparql/?query=EncodedQuery&default-graph-uri=http://
www.other.example/books HTTP/1.1
Host: www.other.example
User-agent: my-sparql-client/0.1
```

SPARQL Protocol - Example

- HTTP Trace of the SPARQL Response

```
HTTP/1.1 200 OK
Date: Fri, 06 May 2008 20:55:12 GMT
Server: Apache/1.3.29 (Unix) PHP/4.3.4 DAV/1.0.3
Connection: close
Content-Type: application/sparql-results+xml
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
 <head>
   <variable name="book"/>
   <variable name="who"/>
 </head>
 <results ordered="false" distinct="false">
   <result>
     <binding name="who">
       <literal>Bob Hacker</literal>
     </binding>
     <binding name=“book">
       <literal>The Art of Hacking</literal>
     </binding> ...
</sparql>
```

SPARQL is not only a Query Language

- **SPARQL P**rotocol **a**nd **R**DF **Q**uery **L**anguage is
  - a **Query Language** for RDF Graph Traversal (*SPARQL Query Language Specification*)
  - a **Protocol Layer**, to use SPARQL via http (*SPARQL Protocol for RDF Specification*)
  - an **XML Output Format Specification** for SPARQL Queries (*SPARQL Query XML Results Format*)

SPARQL Query Language

- In addition to **SELECT** queries SPARQL allows:
  - **ASK**
    - Check whether there is at least one result
    - Result: true or false
    - Result is delivered as XML or JSON

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

ASK
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbpedia-owl:Writer .
       ?author dbpedia-owl:notableWork ?work .
}
```

SPARQL Query Language

- In addition to **SELECT** queries SPARQL allows:
  - **DESCRIBE**
    - Result: an RDF graph with data about resources
    - Result is RDF/XML or Turtle

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

DESCRIBE ?author ?work
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbpedia-owl:Writer .
       ?author dbpedia-owl:notableWork ?work .
} LIMIT 10
```

SPARQL Query Language

- In addition to **SELECT** queries SPARQL allows:
  - **CONSTRUCT**
    - Result: an RDF graph constructed from a template
    - Template: graph pattern with variables from the query pattern
    - Result is RDF/XML or Turtle

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

CONSTRUCT { ?author <http://example.org/hasWritten> ?work .}
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbpedia-owl:Writer .
       ?author dbpedia-owl:notableWork ?work .
} LIMIT 100
```

Popular SPARQL Endpoints

- http://sparql.org/sparql.html --
  General Purpose SPARQL Endpoint
- http://dbpedia.org/sparql --
  DBpedia SPARQL Endpoint
- http://dblp.rkbexplorer.com/sparql/ --
  DBLP (Computer Science Bibliographies) SPARQL Endpoint
- http://linkedmdb.org/sparql --
  Linked Movie Database SPARQL Endpoint
- http://www.w3.org/wiki/SparqlEndpoints --
  W3C supported list of active SPARQL endpoints
- ...

03 Complex queries with SPARQL

Open HPI Course: Knowledge Engineering with Semantic Web Technologies
Lecture 03: Semantic Web Technologies Pt.2

Knowledge Engineering with Semantic Web Technologies

Lecture 3: Semantic Web Technologies - Part 2

03: Complex Queries with SPARQL

Dr. Harald Sack
Hasso-Plattner-Institut for IT Systems Engineering
University of Potsdam
Spring 2014

Lecture 3: Semantic Web Technologies - Pt.2

Open HPI Course: Knowledge Engineering with Semantic Web Technologies

03 Complex Queries with SPARQL

Open HPI Course: Knowledge Engineering with Semantic Web Technologies
Lecture 03: Semantic Web Technologies Pt.2

SPARQL - Filter Constraints


```
PREFIX : <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
SELECT ?author_name ?title ?pages
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbpedia-owl:Writer .
        ?author rdfs:label ?author_name .
        ?author dbpedia-owl:notableWork ?work .
        ?work dbpedia-owl:numberOfPages ?pages
        FILTER (?pages > 500) .
        ?work rdfs:label ?title .
} LIMIT 100
```

specifies constraints for the result

- `FILTER` expressions contain operators and functions
- `FILTER` can NOT assign/create new values

SPARQL Unary Operators in Constraints

Operator	Type(A)	Result Type
!A	xsd:boolean	xsd:boolean
+A	numeric	numeric
-a	numeric	numeric
BOUND(A)	variable	xsd:boolean
isURI(A)	RDF term	xsd:boolean
isBLANK(A)	RDF term	xsd:boolean
isLITERAL(A)	RDF term	xsd:boolean
STR(A)	literal/URI	simple literal
LANG(A)	literal	simple literal
DATATYPE(A)	literal	URI

SPARQL - Filter Constraints

- Example: Filter Results only for English labels

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
SELECT ?author_name ?title ?pages
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbpedia-owl:Writer .
        ?author rdfs:label ?author_name
        FILTER (LANG(?author_name)="en").
        ?author dbpedia-owl:notableWork ?work .
        ?work dbpedia-owl:numberOfPages ?pages
        FILTER (?pages > 500) .
        ?work rdfs:label ?title
        FILTER (LANG(?title)="en") .
} LIMIT 100
```

More SPARQL Operators

- Logical connectives **&&** and **||** for xsd:boolean
- Comparison operators **=**, **!=**, **<**, **>**, **<=**, and **>=** for numeric datatypes, xsd:dateTime, xsd:string, and xsd:boolean
- Comparison operators **=** and **!=** for other datatypes
- Arithmetic operators **+**, **-**, *****, and **/** for numeric datatypes

- and in addition:
  - **REGEX**(String,Pattern) or **REGEX**(String,Pattern,Flags)
  - **sameTERM**(A,B)
  - **langMATCHES**(A,B)

SPARQL - REGEX Filter Constraints

- Example: Book titles that contain the word „love“

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
SELECT ?author_name ?title ?pages
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbpedia-owl:Writer .
        ?author rdfs:label ?author_name
        FILTER (LANG(?author_name)="en").
        ?author dbpedia-owl:notableWork ?work .
        ?work dbpedia-owl:numberOfPages ?pages .
        ?work rdfs:label ?title
        FILTER (LANG(?title)="en")
        FILTER REGEX (?title, "love", "i").
} LIMIT 100
```

string

regular expression

flags

- learn more about regular expressions at http://regexone.com/

SPARQL - Filter Constraint Evaluation

- SPARQL Filter Constraints are evaluated in 3-valued Logic
- truth values: `true`, `false`, and **`error`**

A	B	A || B	A && B
T	T	T	T
T	F	T	F
F	T	T	F
F	F	F	F
T	**E**	T	**E**
E	T	T	**E**
F	**E**	**E**	F
E	F	**E**	F
E	**E**	**E**	**E**

binary operators

A	!A
T	F
F	T
E	**E**

unary operator

SPARQL - Optional Results

- Example: Give also the German book title, if available


```
PREFIX : <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
SELECT ?author_name ?en_title ?de_title
FROM <http://dbpedia.org/>
WHERE {
      ?author rdf:type dbpedia-owl:Writer .
      ?author rdfs:label ?author_name
      FILTER (LANG(?author_name)="en").
      ?author dbpedia-owl:notableWork ?work .
      ?work rdfs:label ?en_title
      FILTER (LANG(?en_title)="en") .
      OPTIONAL {?work rdfs:label ?de_title
                FILTER (LANG(?de_title)="de") .
      }
} LIMIT 100
```

optional constraint

- The keyword **OPTIONAL** selects optional elements from the RDF graph
- complies to a Left Outer Join

SPARQL - Alternative Results via UNION

- Example: search all influencers of and people influenced by George Orwell

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
SELECT ?influencer ?influenced
FROM <http://dbpedia.org/>
WHERE {
   { :George_Orwell dbpedia-owl:influenced ?influenced . }
  UNION
   { :George_Orwell dbpedia-owl:influencedBy ?influencer . }
}
```

logical disjunction

- The keyword **UNION** allows for alternatives (logical disjunction)

SPARQL - Negation

- Example: Search for authors that don‘t have an entry for ‚notable work‘

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
SELECT ?author_name
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbpedia-owl:Writer .
        OPTIONAL {?author dbpedia-owl:notableWork ?work . }
        FILTER (!BOUND(?work)) .
} LIMIT 100
```

no variable binding

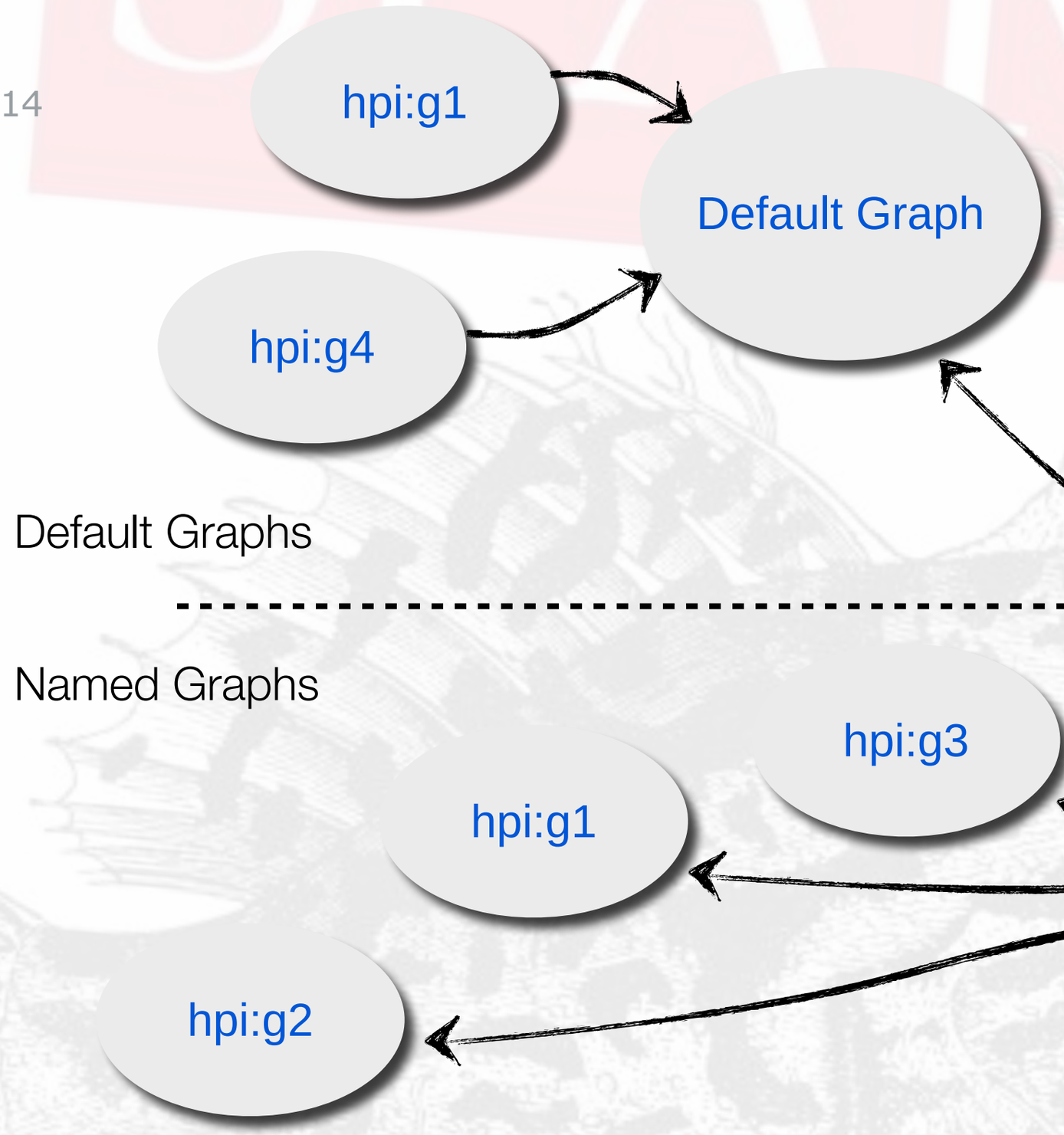
- **Negation** in SPARQL
- *(complies to ‚NOT EXISTS‘ in SQL)*

SPARQL - RDF Graphs

- SPARQL queries are executed over an RDF dataset
  - one (or more) **default RDF graph** (`FROM`)
  - zero or more **named RDF graphs** (`FROM NAMED, GRAPH`)
- **Named Graphs** can explicitely addressed via the keyword **GRAPH** and the URI of the named graph

```
SELECT ...
WHERE {
...
GRAPH <http://example.org/graph1.rdf> {
        ?x foaf:mbox ?mbox
}
...
}
```

SPARQL - RDF Graphs




```
PREFIX hpi: ...
SELECT ...
FROM hpi:g1
FROM hpi:g4
FROM NAMED hpi:g1
FROM NAMED hpi:g2
FROM NAMED hpi:g3
WHERE {
   ... A ...
   GRAPH hpi:g3 {
        ... B ...
   }
   GRAPH ?g {
     ... C ...
   }
}
```

SPARQL - RDF Graphs

- How to ask a SPARQL Endpoint which RDF graphs are available?

```
SELECT DISTINCT ?g
WHERE {
  GRAPH ?g { ?s ?p ?o . }
}
```

04 Even More SPARQL – SPARQL 1.1

Open HPI Course: Knowledge Engineering with Semantic Web Technologies
Lecture 03: Semantic Web Technologies Pt.2

Knowledge Engineering with Semantic Web Technologies

Lecture 3: Semantic Web Technologies - Part 2

04: Even more SPARQL - SPARQL 1.1

Dr. Harald Sack
Hasso-Plattner-Institut for IT Systems Engineering
University of Potsdam
Spring 2014

Lecture 3: Semantic Web Technologies - Pt.2

Open HPI Course: Knowledge Engineering with Semantic Web Technologies

04 Even More SPARQL – SPARQL 1.1

Open HPI Course: Knowledge Engineering with Semantic Web Technologies
Lecture 03: Semantic Web Technologies Pt.2

SPARQL 1.1 - New Features

- SPARQL 1.1 Query:
  - Assignments (e.g. BIND, SELECT expressions)
  - Aggregate functions (e.g. COUNT, SUM, AVG)
  - Subqueries
  - Negation (EXISTS, NOT EXISTS, MINUS)
  - Property paths
  - Basic query federation (SERVICE, BINDINGS)
- SPARQL 1.1 Update:
  - Graph update (INSERT DATA, DELETE DATA, INSERT, DELETE, DELETE WHERE, LOAD, CLEAR)
  - Graph management (CREATE, DROP, COPY, MOVE, ADD)
- SPARQL 1.1 Entailment for RDF, RDFS, OWL, RIF
- SPARQL 1.1 Service Descriptions

SPARQL 1.1 - Variable Assignments

- Example: Select all authors with their notable works and year of publication

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX dbpprop: <http://dbpedia.org/property/>

SELECT ?author ?work ?date
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbpedia-owl:Writer .
       ?author dbpedia-owl:notableWork ?work .
       ?work dbpprop:releaseDate ?date
} ORDER BY ?date
  LIMIT 100
```

author	work	date
http://dbpedia.org/resource/John_Crowley	http://dbpedia.org/resource/Little,_Big	http://dbpedia.org/resource/1981_in_literature
http://dbpedia.org/resource/Will_Self	http://dbpedia.org/resource/Umbrella_(novel)	http://dbpedia.org/resource/United_Kingdom
http://dbpedia.org/resource/Will_Self	http://dbpedia.org/resource/Umbrella_(novel)	http://dbpedia.org/resource/United_States
http://dbpedia.org/resource/Daniel_Keyes	http://dbpedia.org/resource/Flowers_for_Algernon	"April 1959"@en
http://dbpedia.org/resource/Robert_M._Pirsig	http://dbpedia.org/resource/Zen_and_the_Art_of_Motorcycle_Maintenance	"April 1974"@en
http://dbpedia.org/resource/Charlotte_Perkins_Gilman	http://dbpedia.org/resource/Herland_(novel)	"April 1979"@en
http://dbpedia.org/resource/Lloyd_Alexander	http://dbpedia.org/resource/Westmark_(novel)	"April 1981"@en
http://dbpedia.org/resource/David_Eddings	http://dbpedia.org/resource/The_Belgariad	"April 1982"@en
http://dbpedia.org/resource/Cormac_McCarthy	http://dbpedia.org/resource/Blood_Meridian	"April 1985"@en
http://dbpedia.org/resource/Daniel_Pinkwater	http://dbpedia.org/resource/Borgel	"April 1990"@en
http://dbpedia.org/resource/Patrick_McCabe_(novelist)	http://dbpedia.org/resource/The_Butcher_Boy	"April 1992"@en
http://dbpedia.org/resource/Nancy_Farmer	http://dbpedia.org/resource/The_Ear,_the_Eye_and_the_Arm	"April 1994"@en
http://dbpedia.org/resource/Michael_Brodsky	http://dbpedia.org/resource/*** (novel)	"April 1994"@en
http://dbpedia.org/resource/Margaret_Clark_(writer)	http://dbpedia.org/resource/Fat_Chance_(Margaret_Clark_novel)	"April 1996"@en
http://dbpedia.org/resource/Daniel_Handler	http://dbpedia.org/resource/The_Basic_Eight	"April 1998"@en
http://dbpedia.org/resource/Julian_Stockwin	http://dbpedia.org/resource/Kydd_(novel)	"April 2001"@en
http://dbpedia.org/resource/Dan_Brown	http://dbpedia.org/resource/The_Da_Vinci_Code	"April 2003"@en
http://dbpedia.org/resource/Brian_Keene	http://dbpedia.org/resource/The_Rising_(Keene_novel)	"April 2003"@en
http://dbpedia.org/resource/Frans_Sammut	http://dbpedia.org/resource/The_Da_Vinci_Code	"April 2003"@en
http://dbpedia.org/resource/G._B._Singh	http://dbpedia.org/resource/Gandhi_Behind_the_Mask_of_Divinity	"April 2004"@en
http://dbpedia.org/resource/Cecelia_Ahern	http://dbpedia.org/resource/PS,_I_Love_You_(novel)	"April 2004"@en
http://dbpedia.org/resource/David_Mitchell_(author)	http://dbpedia.org/resource/Black_Swan_Green	"April 2006"@en
http://dbpedia.org/resource/Timothy_Ferriss	http://dbpedia.org/resource/The_4-Hour_Workweek	"April 2007"@en
http://dbpedia.org/resource/Carole_Wilkinson	http://dbpedia.org/resource/Dragon_Moon	"April 2007"@en
http://dbpedia.org/resource/Marc_Ostrofsky	http://dbpedia.org/resource/Get_Rich_Click	"April 2011"@en
http://dbpedia.org/resource/Arthur_C._Clarke	http://dbpedia.org/resource/Rendezvous_with_Rama	"Aug 1973"@en
http://dbpedia.org/resource/Guy_Gavriel_Kay	http://dbpedia.org/resource/Tigana	"August 1990"@en
http://dbpedia.org/resource/David_Wellington_(author)	http://dbpedia.org/resource/Monster_Island_(novel)	"August 2004 April 2006"@en
http://dbpedia.org/resource/Premchand	http://dbpedia.org/resource/Bazaar-e-Husn	"Calcutta and Lahore"@en
http://dbpedia.org/resource/Henry_James	http://dbpedia.org/resource/The_Wings_of_the_Dove	"Constable: 30-Aug-1902Scribner's: 21-Aug-1902"@en
http://dbpedia.org/resource/Laurence_Sterne	http://dbpedia.org/resource/The_Life_and_Opinions_of_Tristram_Shandy,_Gentleman	"December 1759 – January 1767"@en
http://dbpedia.org/resource/Emily_Bront%C3%AB	http://dbpedia.org/resource/Wuthering_Heights	"December 1847"@en
http://dbpedia.org/resource/Henri_Barbusse	http://dbpedia.org/resource/Under_Fire_(novel)	"December 1916"@en
http://dbpedia.org/resource/David_Eddings	http://dbpedia.org/resource/The_Belgariad	"December 1984"@en
http://dbpedia.org/resource/Umera_Ahmad	http://dbpedia.org/resource/Pir-e-Kamil	"English: 2011"@en
http://dbpedia.org/resource/J._D._Salinger	http://dbpedia.org/resource/Franny_and_Zooey	"Fall 1961"@en
http://dbpedia.org/resource/Roald_Dahl	http://dbpedia.org/resource/The_Vicar_of_Nibbleswicke	"Feb 1991"@en
http://dbpedia.org/resource/James_Fenimore_Cooper	http://dbpedia.org/resource/The_Last_of_the_Mohicans	"February 1826"@en
http://dbpedia.org/resource/Joseph_Conrad	http://dbpedia.org/resource/Heart_of_Darkness	"February 1899"@en

SPARQL 1.1 - Variable Assignments

- Example: Select all authors with their notable works and year of publication

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX dbpprop: <http://dbpedia.org/property/>
PREFIX xsd:    <http://www.w3.org/2001/XMLSchema#>

SELECT ?author ?work xsd:integer(?date) AS ?year
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbpedia-owl:Writer .
        ?author dbpedia-owl:notableWork ?work .
        ?work dbpprop:releaseDate ?date .
} ORDER BY ?year
  LIMIT 100
```

new variable assignment

- SPARQL 1.1 allows the creation of new values in a query

author	work	year
http://dbpedia.org/resource/A._J._Quinnell	http://dbpedia.org/resource/The_Mahdi	
http://dbpedia.org/resource/A._J._Quinnell	http://dbpedia.org/resource/Man_on_Fire_(novel)	
http://dbpedia.org/resource/Augusten_Burroughs	http://dbpedia.org/resource/Running_with_Scissors_(memoir)	
http://dbpedia.org/resource/B._Traven	http://dbpedia.org/resource/The_Treasure_of_the_Sierra_Madre	1927
http://dbpedia.org/resource/B._Traven	http://dbpedia.org/resource/The_Death_Ship	1926
http://dbpedia.org/resource/Brian_O'Nolan	http://dbpedia.org/resource/The_Dalkey_Archive	1964
http://dbpedia.org/resource/Brian_O'Nolan	http://dbpedia.org/resource/The_Third_Policeman	1967
http://dbpedia.org/resource/Brian_O'Nolan	http://dbpedia.org/resource/At_Swim-Two-Birds	1939
http://dbpedia.org/resource/Carole_Wilkinson	http://dbpedia.org/resource/Dragonkeeper	2003
http://dbpedia.org/resource/DBC_Pierre	http://dbpedia.org/resource/Vernon_God_Little	20
http://dbpedia.org/resource/Dale_Carnegie	http://dbpedia.org/resource/How_to_Win_Friends_and_Influence_People	
http://dbpedia.org/resource/Erica_Jong	http://dbpedia.org/resource/Fear_of_Flying_(novel)	1973
http://dbpedia.org/resource/Hubert_Selby,_Jr.	http://dbpedia.org/resource/Last_Exit_to_Brooklyn	1964
http://dbpedia.org/resource/Hunter_S._Thompson	http://dbpedia.org/resource/The_Rum_Diary_(novel)	1998
http://dbpedia.org/resource/Hunter_S._Thompson	http://dbpedia.org/resource/Fear_and_Loathing_on_the_Campaign_Trail_'72	1973
http://dbpedia.org/resource/James_Frey	http://dbpedia.org/resource/A_Million_Little_Pieces	
http://dbpedia.org/resource/Jeanette_Winterson	http://dbpedia.org/resource/Oranges_Are_Not_the_Only_Fruit	21
http://dbpedia.org/resource/John_Banville	http://dbpedia.org/resource/The_Sea_(novel)	
http://dbpedia.org/resource/John_McGahern	http://dbpedia.org/resource/Amongst_Women	1990
http://dbpedia.org/resource/John_McGahern	http://dbpedia.org/resource/The_Barracks	1963
http://dbpedia.org/resource/Ken_Kesey	http://dbpedia.org/resource/Sometimes_a_Great_Notion	
http://dbpedia.org/resource/Kevin_Brooks_(writer)	http://dbpedia.org/resource/Martyn_Pig	1
http://dbpedia.org/resource/Maeve_Binchy	http://dbpedia.org/resource/Tara_Road	28
http://dbpedia.org/resource/Magnus_Mills	http://dbpedia.org/resource/The_Restraint_of_Beasts	7
http://dbpedia.org/resource/Michael_Lewis	http://dbpedia.org/resource/Liar's_Poker	
http://dbpedia.org/resource/Nancy_Mitford	http://dbpedia.org/resource/The_Pursuit_of_Love	1945
http://dbpedia.org/resource/Rebecca_Wells	http://dbpedia.org/resource/Little_Altars_Everywhere	
http://dbpedia.org/resource/Robert_Holdstock	http://dbpedia.org/resource/Mythago_Wood	1984
http://dbpedia.org/resource/Robert_M._Pirsig	http://dbpedia.org/resource/Lila:_An_Inquiry_into_Morals	1991
http://dbpedia.org/resource/Robert_M._Pirsig	http://dbpedia.org/resource/Zen_and_the_Art_of_Motorcycle_Maintenance	

SPARQL 1.1 - Variable Assignments

- Example: Select all authors with their notable works and year of publication

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX dbpprop: <http://dbpedia.org/property/>
PREFIX xsd:    <http://www.w3.org/2001/XMLSchema#>

SELECT ?author ?work (REPLACE(str(?date),"[^0-9]", "")) AS ?year
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbpedia-owl:Writer .
        ?author dbpedia-owl:notableWork ?work .
        ?work dbpprop:releaseDate ?date
        FILTER REGEX (?date, "[0-9]{4}") .
} ORDER BY ?year
  LIMIT 100
```

new variable assignment

- SPARQL 1.1 allows the creation of new values in a query

author	work	year
http://dbpedia.org/resource/A._J._Quinnell	http://dbpedia.org/resource/The_Mahdi	1982
http://dbpedia.org/resource/A._J._Quinnell	http://dbpedia.org/resource/Man_on_Fire_(novel)	1980
http://dbpedia.org/resource/Dale_Carnegie	http://dbpedia.org/resource/How_to_Win_Friends_and_Influence_People	1936
http://dbpedia.org/resource/Rebecca_Wells	http://dbpedia.org/resource/Little_Altars_Everywhere	1998
http://dbpedia.org/resource/Robert_M._Pirsig	http://dbpedia.org/resource/Zen_and_the_Art_of_Motorcycle_Maintenance	1974
http://dbpedia.org/resource/William_P._Young	http://dbpedia.org/resource/The_Shack	2007
http://dbpedia.org/resource/Beatrix_Potter	http://dbpedia.org/resource/The_Tale_of_Peter_Rabbit	1902
http://dbpedia.org/resource/Dan_Brown	http://dbpedia.org/resource/Angels_&_Demons	2000
http://dbpedia.org/resource/Daniel_Handler	http://dbpedia.org/resource/Watch_Your_Mouth	2000
http://dbpedia.org/resource/Daniel_Keyes	http://dbpedia.org/resource/Flowers_for_Algernon	1959
http://dbpedia.org/resource/Daniel_Keyes	http://dbpedia.org/resource/Flowers_for_Algernon	1966
http://dbpedia.org/resource/David_Eddings	http://dbpedia.org/resource/The_Belgariad	1984
http://dbpedia.org/resource/David_Eddings	http://dbpedia.org/resource/The_Belgariad	1982
http://dbpedia.org/resource/David_Eddings	http://dbpedia.org/resource/The_Belgariad	1984
http://dbpedia.org/resource/David_Eddings	http://dbpedia.org/resource/The_Belgariad	1983
http://dbpedia.org/resource/David_Eddings	http://dbpedia.org/resource/The_Belgariad	1982
http://dbpedia.org/resource/Larry_Niven	http://dbpedia.org/resource/Ringworld	1970
http://dbpedia.org/resource/Laurence_Sterne	http://dbpedia.org/resource/The_Life_and_Opinions_of_Tristram_Shandy,_Gentleman	17591767
http://dbpedia.org/resource/Timothy_Ferriss	http://dbpedia.org/resource/The_4-Hour_Workweek	2007
http://dbpedia.org/resource/Tucker_Max	http://dbpedia.org/resource/I_Hope_They_Serve_Beer_in_Hell	2006
http://dbpedia.org/resource/Dean_Koontz	http://dbpedia.org/resource/Phantoms_(novel)	1983
http://dbpedia.org/resource/Maxwell_Gray	http://dbpedia.org/resource/The_Silence_of_Dean_Maitland	1886
http://dbpedia.org/resource/Robert_Chambers_(publisher_born_1802)	http://dbpedia.org/resource/Vestiges_of_the_Natural_History_of_Creation	1844
http://dbpedia.org/resource/Surender_Mohan_Pathak	http://dbpedia.org/resource/Daylight_Robbery_(novel)	2010
http://dbpedia.org/resource/Joanna_Russ	http://dbpedia.org/resource/The_Female_Man	1975
http://dbpedia.org/resource/Ethel_Lilian_Voynich	http://dbpedia.org/resource/The_Gadfly	1897
http://dbpedia.org/resource/J._D._Salinger	http://dbpedia.org/resource/Franny_and_Zooey	1961
http://dbpedia.org/resource/Rudyard_Kipling	http://dbpedia.org/resource/Kim_(novel)	1901
http://dbpedia.org/resource/Samuel_Beckett	http://dbpedia.org/resource/Molloy_(novel)	19511955
http://dbpedia.org/resource/Stephen_King	http://dbpedia.org/resource/The_Stand	1990
http://dbpedia.org/resource/Stephen_King	http://dbpedia.org/resource/The_Stand	1978
http://dbpedia.org/resource/V._S._Naipaul	http://dbpedia.org/resource/A_Bend_in_the_River	1979
http://dbpedia.org/resource/Yann_Martel	http://dbpedia.org/resource/Life_of_Pi	2001

SPARQL 1.1 - Aggregate Functions

- Example: How many authors are there in DBpedia?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

SELECT (COUNT(?author)) AS ?num
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbpedia-owl:Writer .
}
```

aggregate function

- SPARQL 1.1 allows the use of aggregate functions in queries

SPARQL 1.1 - Aggregate Functions

- Example: How many **distinct** authors are there in DBpedia who have entries for notable works?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

SELECT (COUNT(DISTINCT ?author)) AS ?num
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbpedia-owl:Writer .
        ?author dbpedia-owl:notableWork ?work .
}
```

aggregate function

- SPARQL 1.1 allows the use of aggregate functions in queries

SPARQL 1.1 - Aggregate Functions

- Example: Which author wrote how many notable works?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

SELECT ?author (COUNT(?work)) AS ?num_works
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbpedia-owl:Writer .
        ?author dbpedia-owl:notableWork ?work .
} GROUP BY ?author
ORDER BY DESC (?num_works)
```

aggregate function

author	num_works
http://dbpedia.org/resource/Vince_Powell	20
http://dbpedia.org/resource/Roald_Dahl	18
http://dbpedia.org/resource/Joseph_Conrad	15
http://dbpedia.org/resource/Rexhep_Qosja	13
http://dbpedia.org/resource/Edward_Stratemeyer	12
http://dbpedia.org/resource/Julian_Stockwin	12
http://dbpedia.org/resource/Judd_Apatow	11
http://dbpedia.org/resource/Chris_Meledandri	11
http://dbpedia.org/resource/Brian_Cooke	11
http://dbpedia.org/resource/David_Mamet	11
http://dbpedia.org/resource/Laurence_Marks_(British_writer)	9
http://dbpedia.org/resource/Dino_Stamatopoulos	9
http://dbpedia.org/resource/William_Trevor	9
http://dbpedia.org/resource/Ismail_Kadare	9
http://dbpedia.org/resource/Cicero	9
http://dbpedia.org/resource/Samuel_Beckett	9
http://dbpedia.org/resource/Maurice_Gran	9
http://dbpedia.org/resource/Writings_of_Marcus_Tullius_Cicero	9
http://dbpedia.org/resource/Jim_Allen_(playwright)	9
http://dbpedia.org/resource/Johnnie_Mortimer	9
http://dbpedia.org/resource/David_Weir_(writer)	9
http://dbpedia.org/resource/Eric_Chappell	9

SPARQL 1.1 - Aggregate Functions

- Example: Which author wrote 3 notable works (acording to DBpedia)?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

SELECT ?author (COUNT(?work)) AS ?num_works
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbpedia-owl:Writer .
        ?author dbpedia-owl:notableWork ?work .
} GROUP BY ?author
HAVING COUNT(?work) = 3
ORDER BY DESC (?num_works)
```

aggregate function

author	num_works
http://dbpedia.org/resource/Fidel_Villarroel	3
http://dbpedia.org/resource/Frances_Hardinge	3
http://dbpedia.org/resource/John_Gardner_(American_writer)	3
http://dbpedia.org/resource/John_Orloff	3
http://dbpedia.org/resource/Malcolm_Lowry	3
http://dbpedia.org/resource/Monica_Edwards	3
http://dbpedia.org/resource/Peter_Bowker	3
http://dbpedia.org/resource/Richard_Russo	3
http://dbpedia.org/resource/Sajjad_Zaheer	3
http://dbpedia.org/resource/Alois_Jir%C3%A1sek	3
http://dbpedia.org/resource/Andrew_Marshall_(writer)	3
http://dbpedia.org/resource/David_Dobkin_(director)	3
http://dbpedia.org/resource/Virginia_Spencer_Carr	3
http://dbpedia.org/resource/Ernesto_Sabato	3
http://dbpedia.org/resource/Khaled_Mattawa	3
http://dbpedia.org/resource/R._A._Chandrasena	3
http://dbpedia.org/resource/Patrick_McCabe_(novelist)	3
http://dbpedia.org/resource/Chris_Valentino	3
http://dbpedia.org/resource/Carole_Wilkinson	3
http://dbpedia.org/resource/Christopher_Nolan_(author)	3
http://dbpedia.org/resource/David_Brin	3
http://dbpedia.org/resource/David_Levithan	3

SPARQL 1.1 - Aggregate Functions

- SPARQL 1.1 provides more aggregate functions
  - `SUM`
  - `AVG`
  - `MIN`
  - `MAX`
  - `SAMPLE` -- „pick“ one non-deterministically
  - `GROUP_CONCAT` -- concatenate values with a designated string separator

SPARQL 1.1 - Aggregate Functions

- Example: Compare an arbitrary title and non-english titles of notable Works by authors

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

SELECT (SAMPLE(?title) AS ?name)
       (GROUP_CONCAT(?alt_titles; SEPARATOR=", ") AS ?alt_names)
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbpedia-owl:Writer .
       ?author dbpedia-owl:notableWork ?work .
       ?work rdfs:label ?title .
       ?work rdfs:label ?alt_titles
       FILTER (LANG(?alt_titles)!="en") .
} GROUP BY ?author
```

aggregate function

name	alt_names
"Maxine Hong Kingston"@de	Maxine Hong Kingston, Maxine Hong Kingston, Maxine Hong Kingston, Maxine Hong Kingston, Maxine Hong Kingston, Maxine Hong Kingston, Maxine Hong Kingston, Maxine Hong Kingston, 汤婷婷, 汤婷婷, 汤婷婷, 汤婷婷
"The Outsider (Colin Wilson)"@en	The Outsider, The Outsider, The Occult, The Occult
"Deutschstunde"@de	Deutschstunde, Deutschstunde, Deutschstunde, Lekcja niemieckiego, Lekcja niemieckiego, Lekcja niemieckiego
"The Immortals (books)"@en	Éternels (série de romans), Éternels (série de romans), Éternels (série de romans), Os Imortais (série de livros), Os Imortais (série de livros), Os Imortais (série de livros)
"奥勃洛莫夫"@zh	奥勃洛莫夫, 奥勃洛莫夫, 奥勃洛莫夫, 奥勃洛莫夫, 奥勃洛莫夫, 奥勃洛莫夫, 奥勃洛莫夫, 奥勃洛莫夫, 奥勃洛莫夫, 奥勃洛莫夫, 奥勃洛莫夫, Oblomow, Oblomow, Oblomow, Oblomow, Oblomow, Oblomow, Oblomow, Oblomow, Oblomow, Oblomow, Oblomow, Oblómov, Oblómov, Oblómov, Oblómov, Oblómov, Oblómov, Oblómov, Oblómov, Oblómov, Oblómov, Oblomov, Oblomov, Oblomov, Oblomov, Oblomov, Oblomov, Oblomov, Oblomov, Oblomov, Oblomov, Oblómov, Oblómov, Oblómov, Oblómov, Oblómov, Oblómov, Oblómov, Oblómov, Oblómov, Oblómov, Oblómov, Oblómov, オブローモフ, オブローモフ, オブローモフ, オブローモフ, オブローモフ, オブローモフ, オブローモフ, オブローモフ, オブローモフ, オブローモフ, オブローモフ, Oblomov, Обломов, Обломов, Обломов, Обломов, Обломов, Обломов, Обломов, Обломов, Обломов, Обломов, Oblomov, Oblomov, Oblomov, Oblomov, Oblomov, Oblomov, Oblomov, Oblomov, Oblomov, Oblomov, Oblomov, Oblomov
"Labyrinth of Reflections"@en	Лабиринт отражений, Лабиринт отражений, 守夜人, 守夜人, 守夜人, 守夜人, 守夜人, 守夜人, 守夜人, 守夜人, 守夜人, Wächter-Romane, Wächter-Romane, Wächter-Romane, Wächter-Romane, Wächter-Romane, Wächter-Romane, Wächter-Romane, Wächter-Romane, Guardianes de la noche, Guardianes de la noche, Guardianes de la noche, Guardianes de la noche, Guardianes de la noche, Guardianes de la noche, Guardianes de la noche, Guardianes de la noche, Guardianes de la noche, Les Sentinelles de la nuit, Les Sentinelles de la nuit, Les Sentinelles de la nuit, Les Sentinelles de la nuit, Les Sentinelles de la nuit, Les Sentinelles de la nuit, Les Sentinelles de la nuit, Les Sentinelles de la nuit, I guardiani della notte, I guardiani della notte, I guardiani della notte, I guardiani della notte, I guardiani della notte, I guardiani della notte, I guardiani della notte, I guardiani della notte, I guardiani della notte, ナイト・ウォッチ, ナイト・ウォッチ, ナイト・ウォッチ, ナイト・ウォッチ, ナイト・ウォッチ, ナイト・ウォッチ, ナイト・ウォッチ, ナイト・ウォッチ, ナイト・ウォッチ, Nocny Patrol (powieść), Nocny Patrol (powieść), Nocny Patrol (powieść), Nocny Patrol (powieść), Nocny Patrol (powieść), Nocny Patrol (powieść), Nocny Patrol (powieść), Nocny Patrol (powieść), Ночной Дозор (роман), Ночной Дозор (роман), Ночной Дозор (роман), Ночной Дозор (роман), Ночной Дозор (роман), Ночной Дозор (роман), Ночной Дозор (роман), Ночной Дозор (роман), Ночной Дозор (роман)
"Martyn Pig"@de	Martyn Pig, Martyn Pig, Martyn Pig, Martyn Pig, Martyn Pig, Martyn Pig
"Sandokan"@en	Sandokán, Sandokán, Sandokán, Sandokán, Sandokán, Sandokan, Sandokan, Sandokan, Sandokan, Sandokan, Sandokan, Sandokan, Sandokan, Sandokan, Sandokan, Sandokan, Sandokan, Sandokan, Sandokan, Sandokan, Il Corsaro Nero (romanzo), Il Corsaro Nero (romanzo), Il Corsaro Nero (romanzo), Чёрный корсар, Чёрный корсар, Чёрный корсар
	迷霧之子, 迷霧之子, 迷霧之子, 迷霧之子, 迷霧之子, Mistborn, Mistborn, Mistborn, Mistborn, Mistborn, Mistborn, Mistborn, Mistborn, Mistborn, Mistborn, ミストボーン, ミストボーン, ミストボーン, ミストボーン, ミストボーン, Warbreaker, Warbreaker, Warbreaker, 破戰者, 破戰者, 破戰者, The Way of Kings, The Way of Kings, Elantris, Elantris, 時間之輪, 時間之輪, 時間之輪, 時間之輪, 時間之輪, 時間之輪, 時間之輪, 時間之輪, 時間之輪, 時間之輪, 時間之輪, 時間之輪, Das Rad der Zeit, Das Rad der Zeit, Das Rad der Zeit, Das Rad der Zeit, Das Rad der Zeit, Das Rad der Zeit, Das Rad der Zeit, Das Rad der Zeit, Das Rad der Zeit, Das Rad der Zeit, Das Rad der Zeit, Das Rad der Zeit, La rueda del tiempo (saga), La rueda del tiempo (saga), La rueda del tiempo (saga), La rueda del tiempo (saga), La rueda del tiempo (saga), La rueda del tiempo (saga), La rueda del tiempo (saga), La rueda del tiempo (saga), La rueda del tiempo (saga), La rueda del tiempo (saga), La rueda del tiempo (saga), La rueda del tiempo (saga), La Roue du temps, La Roue du temps, La Roue du temps, La Roue du temps, La Roue du temps, La Roue du temps, La Roue du temps, La Roue du temps, La Roue du temps, La Roue du temps, La Roue du temps, La Roue du temps, La Ruota del Tempo, La Ruota del Tempo, La Ruota del Tempo, La Ruota del

SPARQL 1.1 - Subqueries

- Example: Select all authors, who they are influenced by and all the influencers notable works

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

SELECT ?author ?influencer ?work
FROM <http://dbpedia.org/>
WHERE {
{ SELECT ?author ?influencer
  FROM <http://dbpedia.org/>
  WHERE {
       ?author rdf:type dbpedia-owl:Writer .
       ?author dbpedia-owl:influencedBy ?influencer .
  } LIMIT 10
}
 ?influencer dbpedia-owl:notableWork ?work .
}
```

subquery

- Subqueries are a way to embed SPARQL queries within other queries
- result is achieved by first evaluating the inner query

SPARQL 1.1 - Negation

- Example: Select all authors, who don‘t have a notable work entry in DBpedia

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

SELECT ?author
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbpedia-owl:Writer
        FILTER NOT EXISTS {?author dbpedia-owl:notableWork ?work .}
}
```

filter query result for existency

- Filtering of query solutions is done within a `FILTER` expression using `NOT EXISTS` and `EXISTS`.

SPARQL 1.1 - Negation

- Example: Select all authors, who don‘t have a notable work entry in DBpedia

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

SELECT ?author
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbpedia-owl:Writer
        FILTER NOT EXISTS {?author dbpedia-owl:notableWork ?work .}
}
```

filter query result for existency

- Filtering of query solutions is done within a `FILTER` expression using `NOT EXISTS` and `EXISTS`.

SPARQL 1.1 - Negation

- Example: Select all authors, who don‘t have a notable work entry in DBpedia

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

SELECT ?author
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbpedia-owl:Writer
        MINUS {?author dbpedia-owl:notableWork ?work .}
}
```

remove from query result

- Filtering of query solutions be removing possible solutions with MINUS.

SPARQL 1.1 - Property Paths

- A **property path** is a possible route through an RDF graph between two graph nodes.
  - trivial case: property path of length 1, i.e. a triple pattern
  - **alternatives**: match one or both possibilities

```
{ :book1 dc:title|rdfs:label ?displayString }
```

- **sequence**: property path of length >1

```
{ ?x foaf:mbox <mailto:alice@example> .
  ?x foaf:knows/foaf:knows/foaf:name ?name . }
```

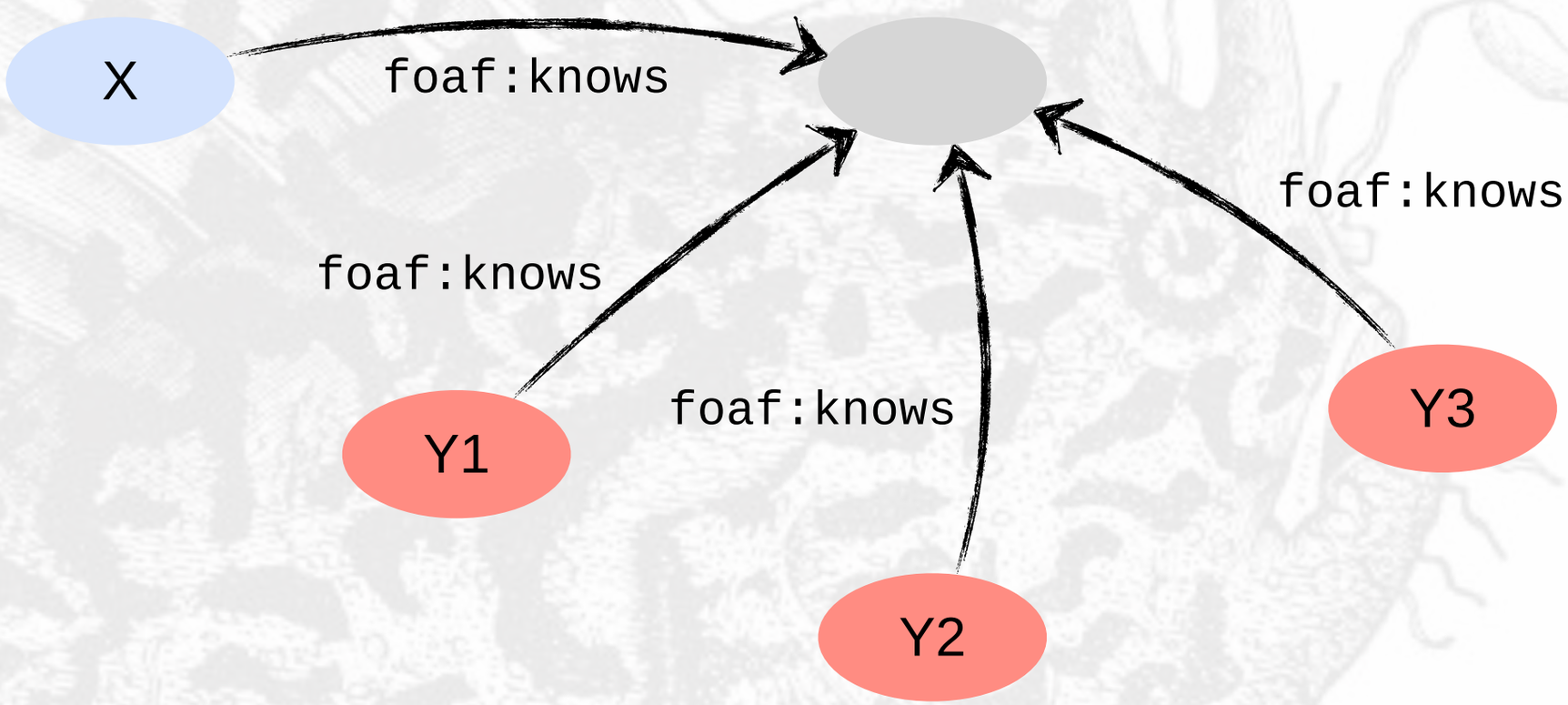
- **inverse property paths**: reversing the direction of the triple

```
{ ?x foaf:mbox <mailto:alice@example> }
                    =
{ <mailto:alice@example> ^foaf:mbox ?x }
```

SPARQL 1.1 - Property Paths

- **inverse path sequences** paths

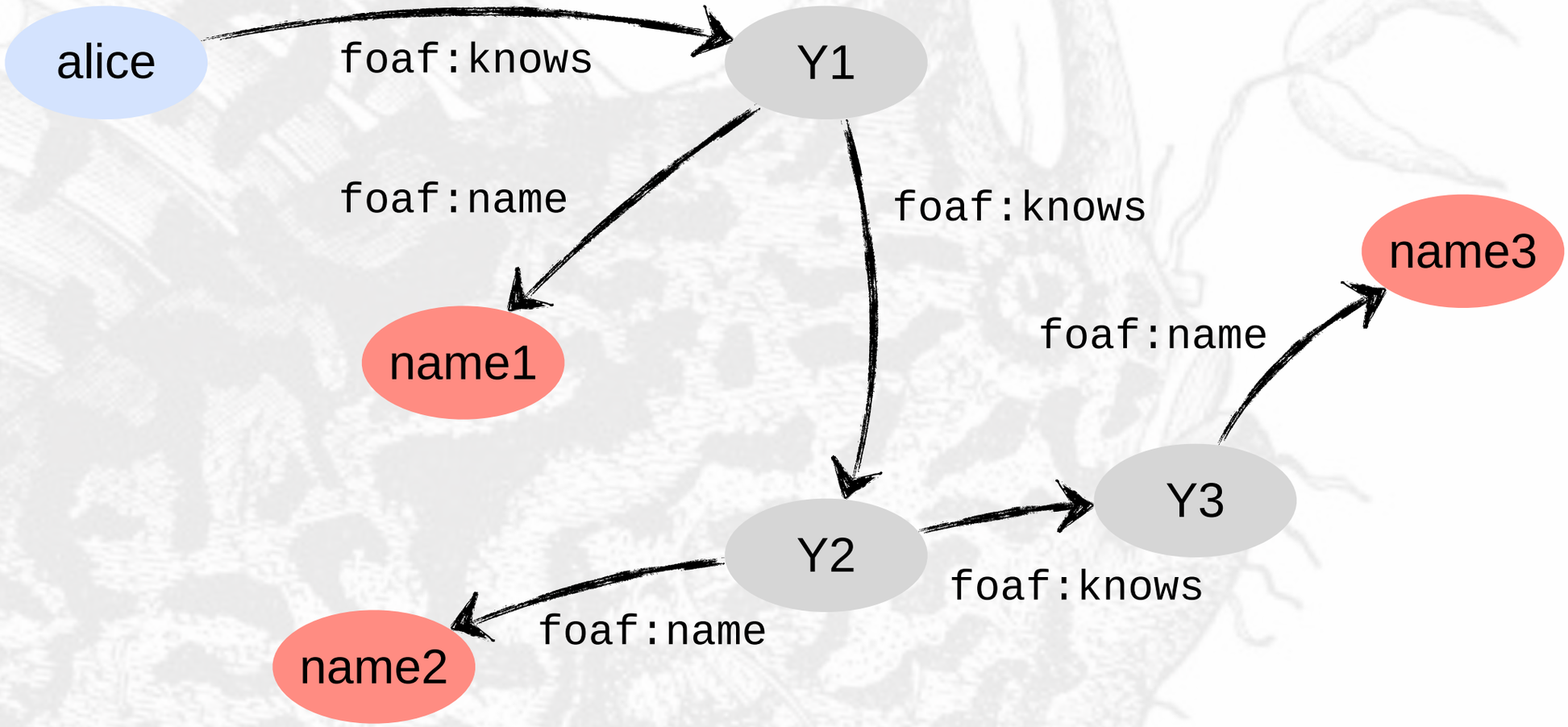
```
{ ?x foaf:knows/^foaf:knows ?y .
    FILTER(?x != ?y) }
```



SPARQL 1.1 - Property Paths

- **arbitrary length match**

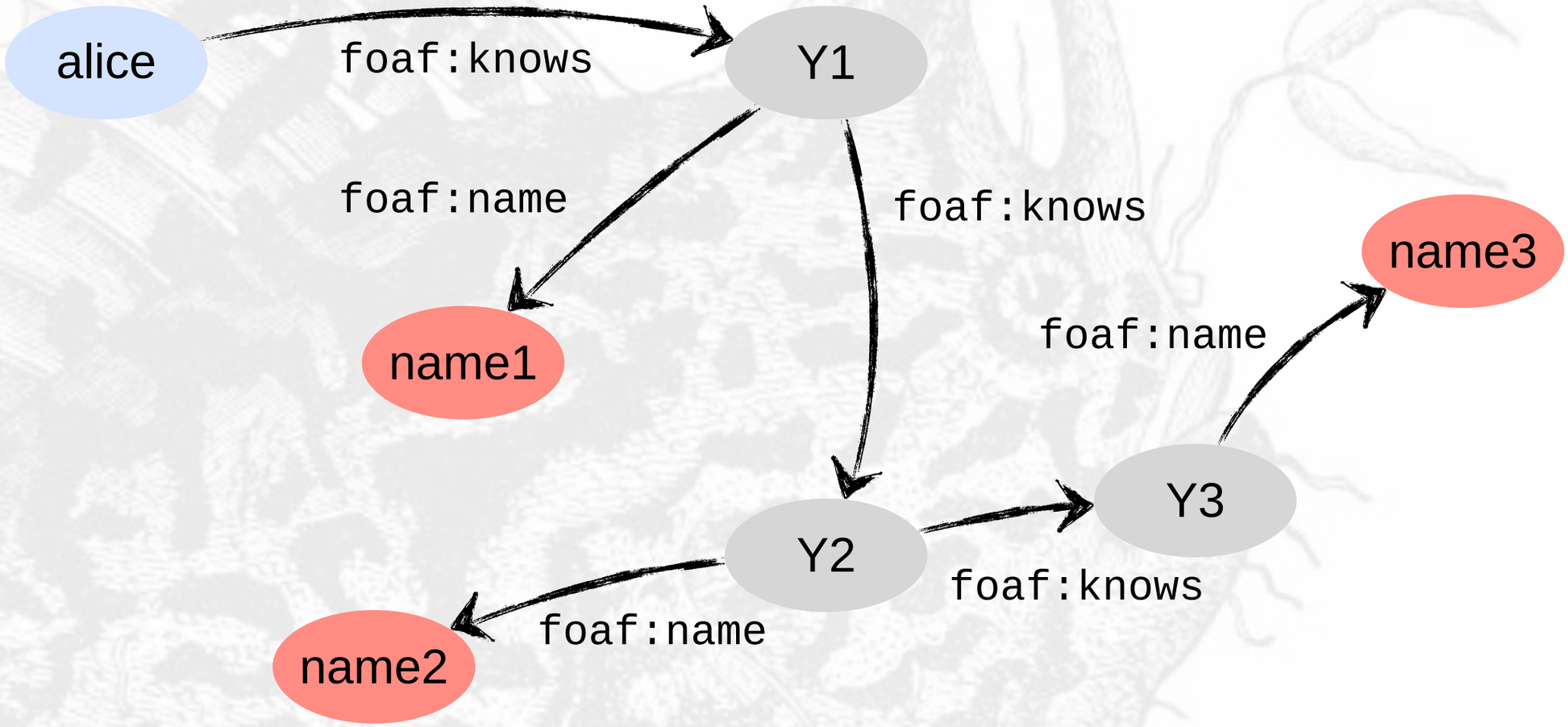
```
{ ?x foaf:mbox <mailto:alice@example> .
  ?x foaf:knows+/foaf:name ?name . }
```



SPARQL 1.1 - Property Paths

- **arbitrary length match**

```
{ ?x foaf:mbox <mailto:alice@example> .
  ?x foaf:knows+/foaf:name ?name . }
```



SPARQL 1.1 - Property Paths

- **inverse path sequences** paths

```
{ ?x foaf:knows/^foaf:knows ?y .
    FILTER(?x != ?y) }
```

- **arbitrary length match**

```
{ ?x foaf:mbox <mailto:alice@example> .
  ?x foaf:knows+/foaf:name ?name . }
```

- **negated property paths**

```
{ ?x !(rdf:type|^rdf:type) ?y }
```

SPARQL 1.1 - Property Paths

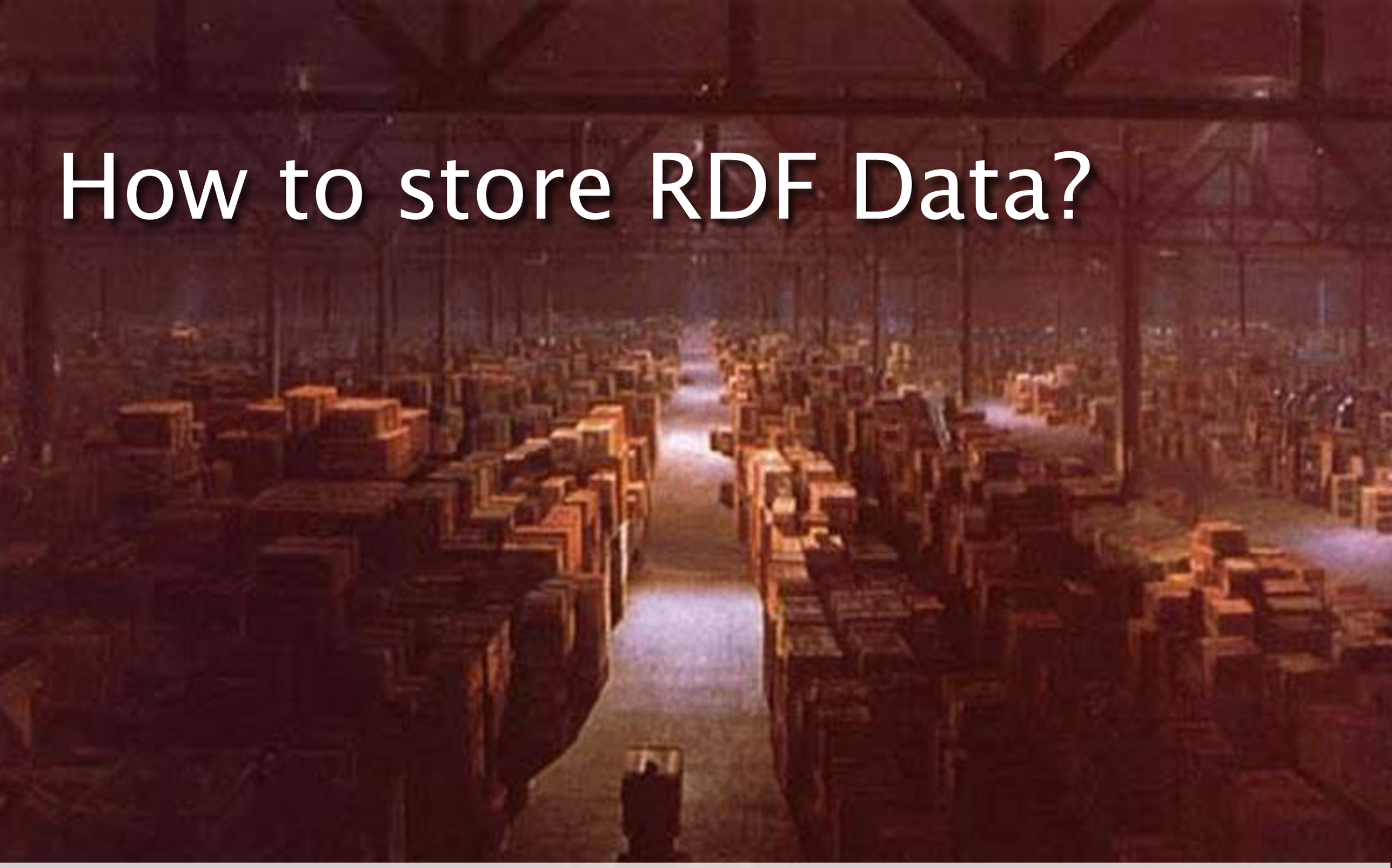
- Example: Who are the authors who were influenced by the influencers of George Orwell?

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

SELECT ?influencedByInfluencers
FROM <http://dbpedia.org/>
WHERE {
        :George_Orwell
        dbpedia-owl:influencedBy/^dbpedia-owl:influencedBy
        ?influencedByInfluencers .
}
```

path expression

How to store RDF Data?



05 How to store RDF(S) Data? - SPARQL 1.1 (5)

Open HPI Course: Knowledge Engineering with Semantic Web Technologies
Lecture 03: Semantic Web Technologies Pt.2

Knowledge Engineering with Semantic Web Technologies

Lecture 3: Semantic Web Technologies - Part 2

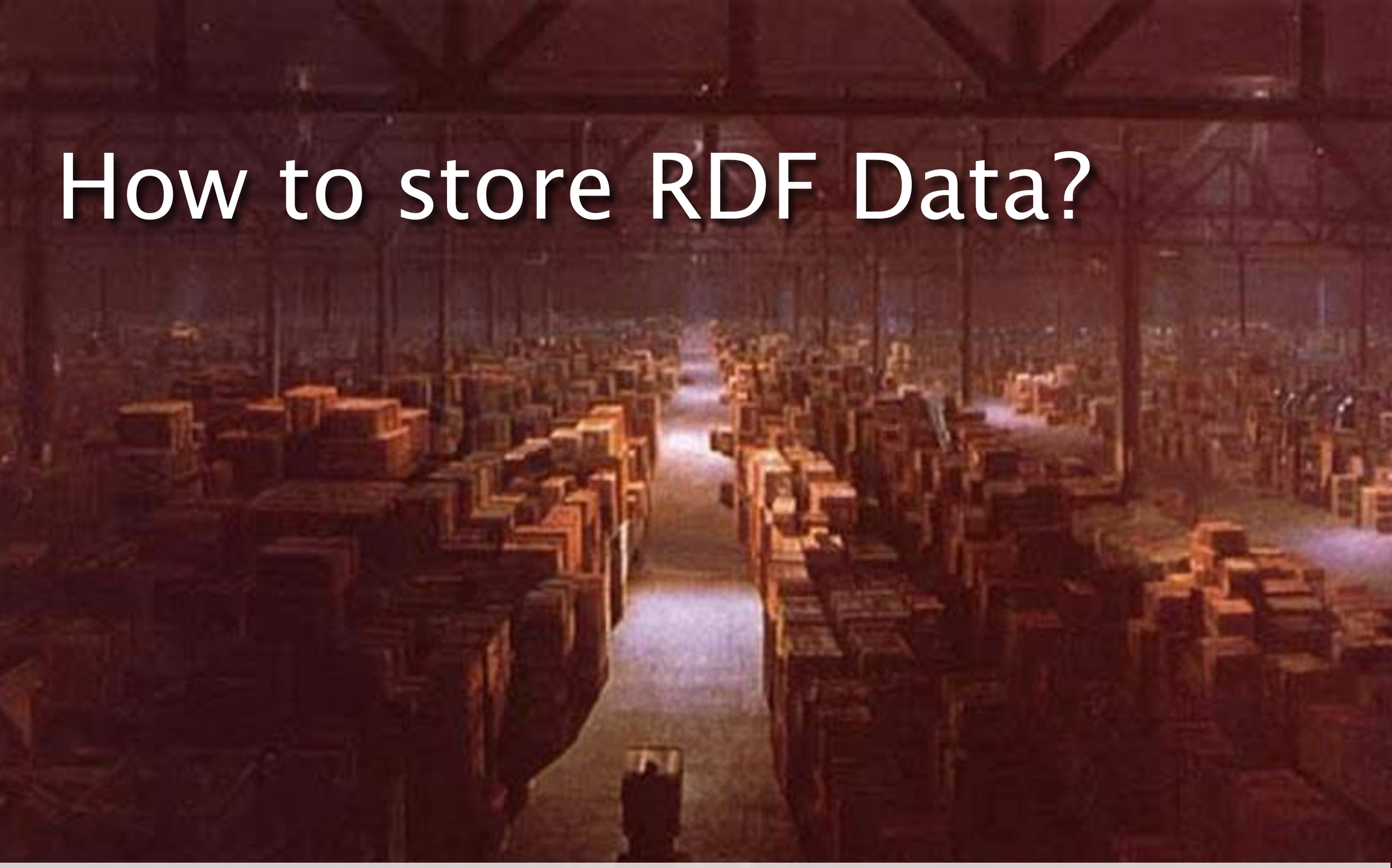
05 How to store RDF(S) Data?

Dr. Harald Sack
Hasso-Plattner-Institut for IT Systems Engineering
University of Potsdam
Spring 2014

Lecture 3: Semantic Web Technologies - Pt.2

Open HPI Course: Knowledge Engineering with Semantic Web Technologies

How to store RDF Data?



05 How to store RDF(S) Data?

Open HPI Course: Knowledge Engineering with Semantic Web Technologies
Lecture 03: Semantic Web Technologies Pt.2

RDF in RDBMS

- RDF Graph can be represented by a set of Triples (s,p,o)
- Triples can simply be stored in a **relational database management system** (RDBMS)

- **Motivation**: Use a specific relational schema for RDF data and benefit from 40 years of research in the DB community

- **3 steps for SPARQL query processing:**
  (1) Convert SPARQL query to SQL query (w.r.t. the schema)
  (2) Use RDBMS to answer SQL query
  (3) Generate SPARQL query result from SQL query result

Problem

- How to store RDF triples to carry out efficient SPARQL queries?

- 4 alternatives:
  1. Giant Triple Storage
  2. Property Tables
  3. Vertically Partitioned Tables (Binary Tables)
  4. Hexastore

Giant Triple Storage

- Basic idea:
  - Store all RDF triples in a single table
  - performance depends on efficient indexing

Subj	Property	Obj
ID1	type	FullProfessor
ID1	teacherOf	'AI'
ID1	bachelorFrom	'MIT'
ID1	mastersFrom	'Cambridge'
ID1	phdFrom	'Yale'
ID2	type	AssocProfessor
ID2	worksFor	'MIT'
ID2	teacherOf	'DataBases'
ID2	bachelorsFrom	'Yale'
ID2	phdFrom	'Stanford'
ID3	type	GradStudent
ID3	advisor	ID2
ID3	teachingAssist	'AI'
ID3	bachelorsFrom	'Stanford'
ID3	mastersFrom	'Princeton'
ID4	type	GradStudent
ID4	advisor	ID1
ID4	takesCourse	'DataBases'
ID4	bachelorsFrom	'Columbia'

Pros:

- easy to implement
- works for huge numbers of properties, if indexes are chosen with care

Cons:

- many self joins

Giant Triple Storage

- Basic idea:
  - Store all RDF triples in a single table
  - performance depends on efficient indexing

Subj	Property	Obj
ID1	type	FullProfessor
ID1	teacherOf	'AI'
ID1	bachelorFrom	'MIT'
ID1	mastersFrom	'Cambridge'
ID1	phdFrom	'Yale'
ID2	type	AssocProfessor
ID2	worksFor	'MIT'
ID2	teacherOf	'DataBases'
ID2	bachelorsFrom	'Yale'
ID2	phdFrom	'Stanford'
ID3	type	GradStudent
ID3	advisor	ID2
ID3	teachingAssist	'AI'
ID3	bachelorsFrom	'Stanford'
ID3	mastersFrom	'Princeton'
ID4	type	GradStudent
ID4	advisor	ID1
ID4	takesCourse	'DataBases'
ID4	bachelorsFrom	'Columbia'

SPARQL

```
SELECT ?university WHERE {
   ?v rdf:type :GradStudent;
     :bachelorsFrom ?university . }
```

SQL

```
SELECT t2.o AS university
FROM triples AS t1, triples AS t2
WHERE t1.p='type' AND
      t1.o='GradStudent' AND
      t2.p='bachelorsFrom' AND
      t1.s=t2.s
```

ID Based Triple Storage

- Use numerical identifier for each RDF term in the dataset
- saves space and enhances efficiency

RDF Term	ID
:ID1	1
rdf:type	2
:FullProfessor	3
:teacherOf	4
“AI“	5
:bachelorsFrom	6
“MIT“	7
:mastersFrom	8
“Cambridge	9

s	p	o
1	2	3
1	4	5
1	6	7
1	8	9
1	10	11
12	13	14
12	15	7
12	4	16

Quad Tables

- Storing **multiple** RDF graphs
- used for provenance, versioning, contexts, etc.

RDF Term	ID
:ID1	1
rdf:type	2
:FullProfessor	3
:teacherOf	4
“AI“	5
:bachelorsFrom	6
“MIT“	7
:mastersFrom	8
“Cambridge	9

g	s	p	o
100	1	2	3
100	1	4	5
101	12	16	7
100	1	8	9
102	23	21	11
100	1	6	7
102	23	22	8
101	12	4	16

Property Tables

- Combining all (or some) properties of similar subjects in n-ary tables
- Use ID based encoding for efficiency

Professors

ID	type	teacherOf	bachelorsFrom	mastersFrom	phdFrom	worksFor
ID1	FullProfessor	'AI'	'MIT'	'Cambrige'	'Yale'	NULL
ID2	AssocProfessor	'DataBases'	'Yale'	NULL	'Stanford'	'MIT'

Students

ID	type	advisor	bachelorsFrom	mastersFrom	teachingAssist	takesCourse
ID3	GradStudent	ID2	'Stanford'	'Princeton'	'AI'	NULL
ID4	GradStudent	ID1	'Columbia'	NULL	NULL	'DataBases'

Pros:

- Fewer joins
- If the data is structured, we have a relational DB

Cons:

- Potentially a lot of NULLs
- Clustering is not trivial
- Multi-value properties are complicated

Vertically Partitioned Tables (Binary Tables)

- for each unique property create a two column table
- Use ID based encoding for efficiency

type

ID1	FullProfessor
ID2	AssocProfessor
ID3	GradStudent
ID4	GradStudent

teacherOf

ID1	'AI'
ID2	'DataBases'

bachelorsFrom

ID1	'MIT'
ID2	'Yale'
ID3	'Stanford'
ID4	'Columbia'

mastersFrom

ID1	'Cambridge'
ID3	'Princeton'

phdFrom

ID1	'Yale'
ID2	'Stanford'

worksFor

ID2	'MIT'

advisor

ID3	ID2
ID4	ID1

bachelorsFrom

ID1	'MIT'
ID2	'Yale'
ID3	'Stanford'
ID4	'Columbia'

teachingAssist

ID3	'AI'

takesCourse

ID4	'DataBases'

Pros:

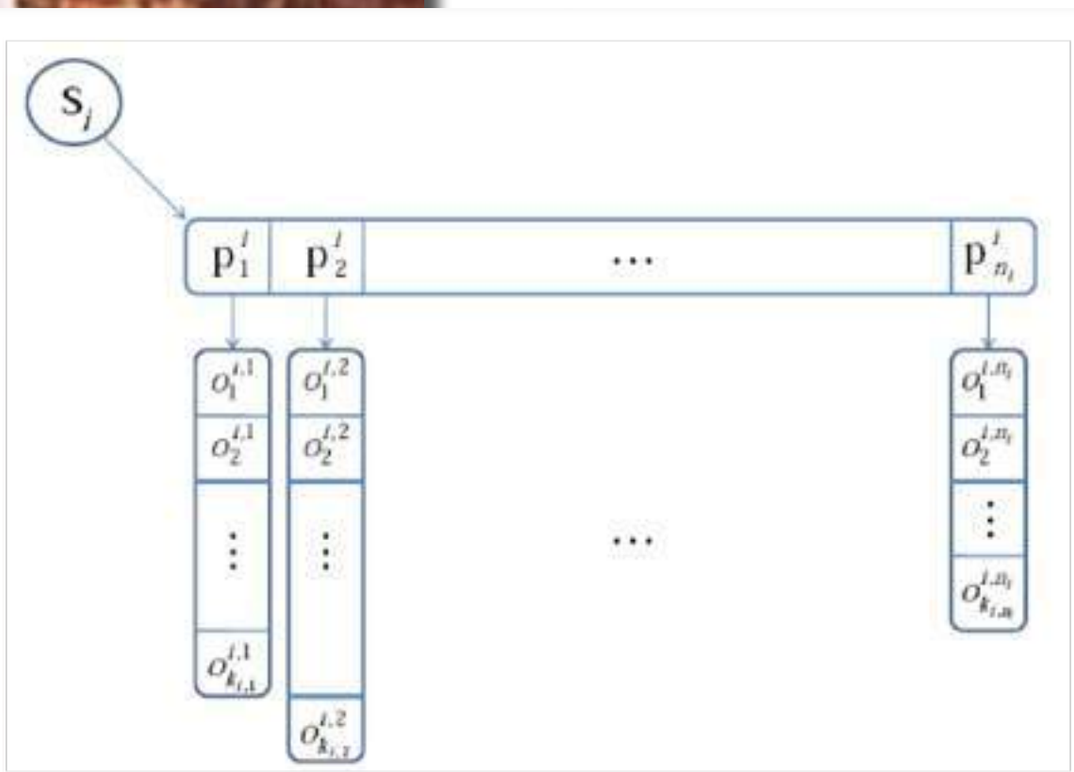
- Supports multi-value properties
- No NULLs
- Read only needed attributes (i.e. less I/O)
- No clustering
- Excellent performance (if number of properties is small, queries with bounded properties)

Cons:

- Expensive inserts
- Bad performance (large number of properties, queries with unbounded properties)

Hexastores

- Create an index for every possible combination to enable efficient processing
  - spo, pos, osp, sop, pso, ops



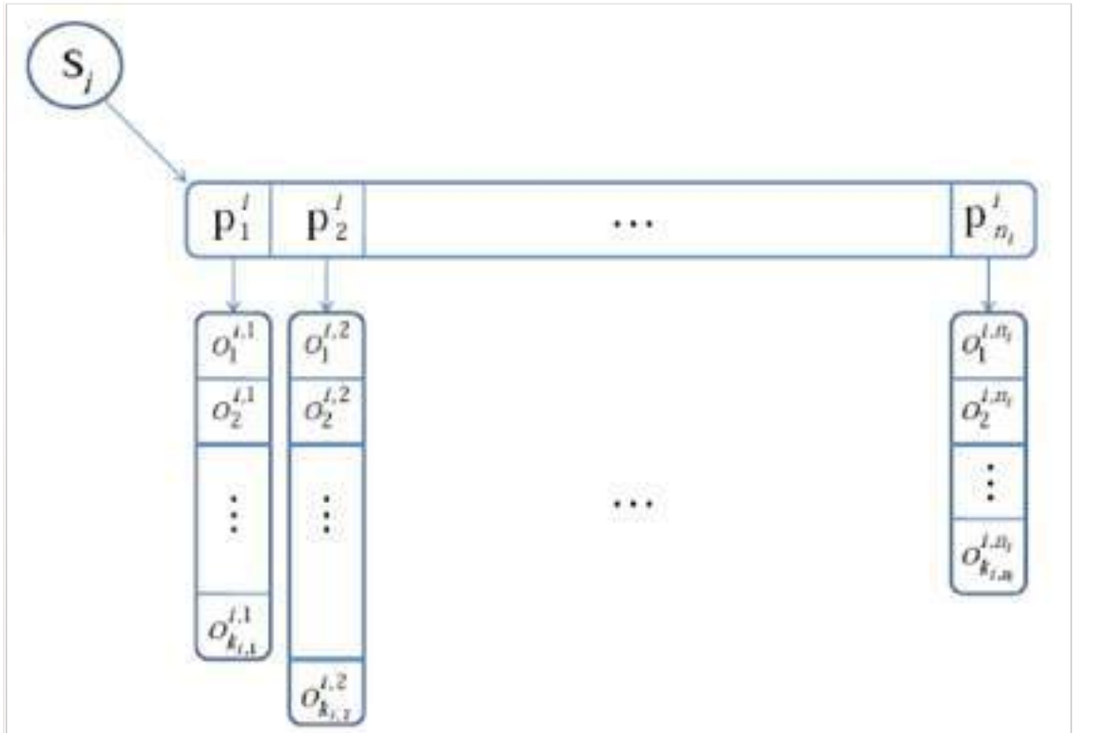
The **spo** index :

Subject key S_i points to sorted vector of n_i property keys, $\{p_1^i, p_2^i, ..., p_{n_i}^i\}$.
Each property key p_j^i is linked to a sorted list of object keys.

The object key lists are shared with the **pso** index.

Hexastores

- Create an index for every possible combination to enable efficient processing
  - spo, pos, osp, sop, pso, ops



Pros:

- fast joins (in the beginning)

Cons:

- 5 times more storage
- weak performance, when disk access is necessary

Triple Stores

- Apache Jena - http://jena.apache.org
- Open Link Virtuoso - http://virtuoso.openlinksw.com
- Sesame - http://www.openrdf.org
- SwiftOWLIM - http://www.ontotext.com/owlim
- AllegroGraph - http://www.franz.com/agraph/allegrograph/
- and many more...

- W3C maintains a list of triple stores:
  http://www.w3.org/wiki/SemanticWebTools#RDF_Triple_Store_Systems

06 Semantic Meta Data and the Web

Open HPI Course: Knowledge Engineering with Semantic Web Technologies
Lecture 03: Semantic Web Technologies Pt.2

Knowledge Engineering with Semantic Web Technologies

Lecture 3: Semantic Web Technologies - Part 2

06: Semantic Metadata and the Web

Dr. Harald Sack
Hasso-Plattner-Institut for IT Systems Engineering
University of Potsdam
Spring 2014

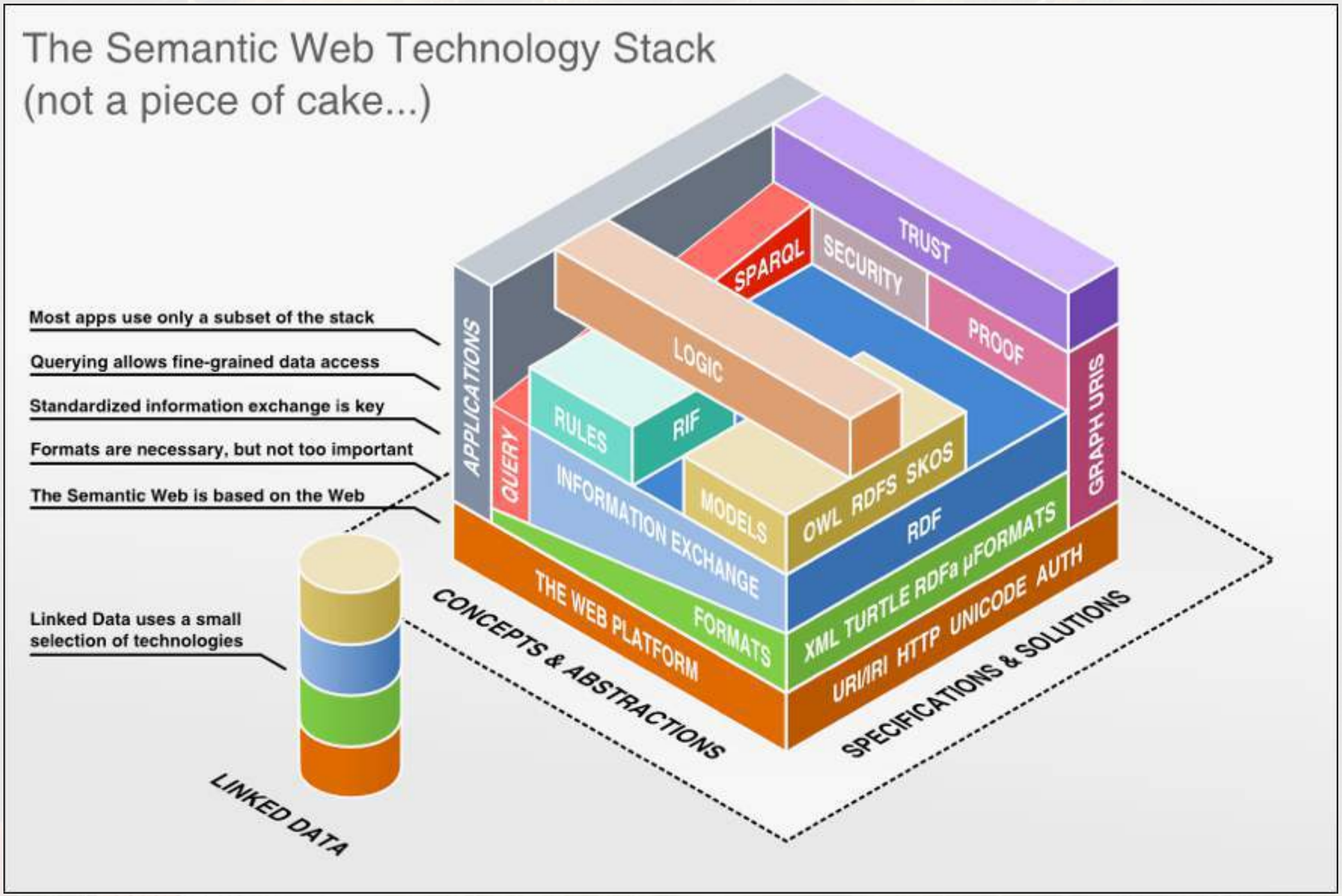
Lecture 3: Semantic Web Technologies - Pt.2

Open HPI Course: Knowledge Engineering with Semantic Web Technologies

06 Semantic Meta Data and the Web

Open HPI Course: Knowledge Engineering with Semantic Web Technologies
Lecture 03: Semantic Web Technologies Pt.2

3. Semantic Web Technologies Pt.2
Semantic Metadata and the Web



Semantic Annotation in the Web

- RDF/XML can directly be embedded in an HTML document via `<head>` and `<script>`
- RFC 3870 defines Mime-Type **application/rdf+xml**

```
<head>
 <title>My Document</title>
 <script type="application/rdf+xml">
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:dc="http://purl.org/dc/elements/1.1/">
   <rdf:Description rdf:about="http://www.w3.org/" dc:title="W3C Homepage"/>
  </rdf:RDF>
 </script>
</head>
```

Semantic Annotation in the Web

- W3C recommended linking of external RDF documents via HTML `<link>` element in HTML `<head>`

```
<link rel="meta" type="application/rdf+xml"
      href="Meta-Data-for-Web-Page.rdf"/>
```

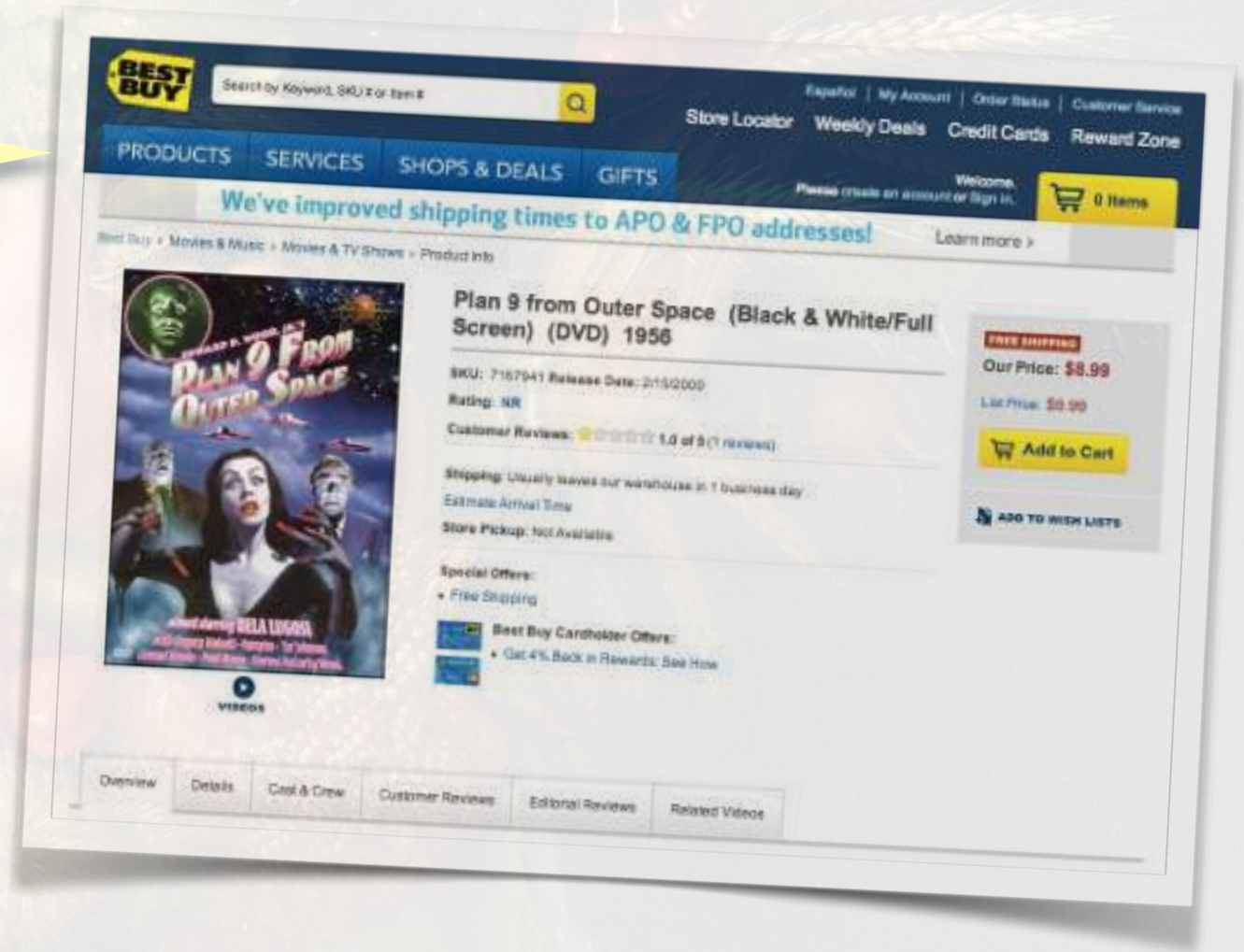
- RDF documents can also be embedded via `<a href=...>` and additional markup (e.g. RDF logo) within HTML `<body>`

Semantic Annotation in the Web

- **Problem**: HTML and RDF data are separate

wep page
HTML

RDF data
Turtle




```
<http://www.bestbuy.com/site/Plan+9+from+Outer+Space
+-+DVD/7167941.p?id=23883&skuId=7167941&st=plan
%209%20from%20outer%20space&lp=6&cp=1> <http://
ogp.me/ns#title> "Plan 9 from Outer Space - DVD" .
<http://www.bestbuy.com/site/Plan+9+from+Outer+Space
+-+DVD/7167941.p?id=23883&skuId=7167941&st=plan
%209%20from%20outer%20space&lp=6&cp=1> <http://
ogp.me/ns#type> "product" .
<http://www.bestbuy.com/site/Plan+9+from+Outer+Space
+-+DVD/7167941.p?id=23883&skuId=7167941&st=plan
%209%20from%20outer%20space&lp=6&cp=1> <http://
ogp.me/ns#url> "http://www.bestbuy.com:80/site/Plan-9-
from-Outer-Space---DVD/6429224.p?
id=23883&skuId=7167941" .
<http://www.bestbuy.com/site/Plan+9+from+Outer+Space
+-+DVD/7167941.p?id=23883&skuId=7167941&st=plan
%209%20from%20outer%20space&lp=6&cp=1> <http://
ogp.me/ns#image> "http://images.bestbuy.com:80/
BestBuy_US/images/products/6429/6429224s.jpg" .
<http://www.bestbuy.com/site/Plan+9+from+Outer+Space
+-+DVD/7167941.p?id=23883&skuId=7167941&st=plan
```

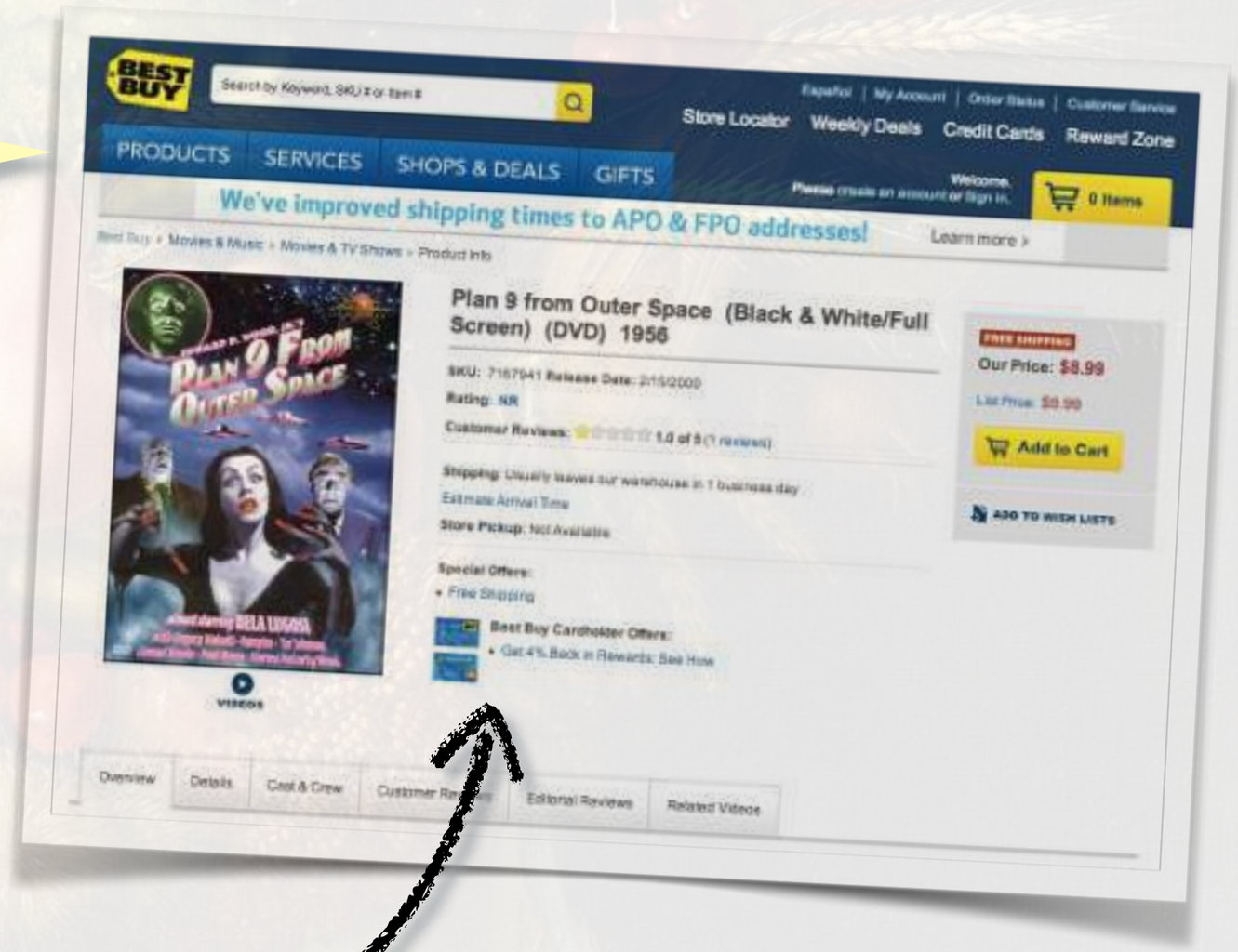
- **Maintenance**
- **Verification**
- **Consistency**

Semantic Annotation in the Web

‚Embed‘ RDF directly into HTML

wep page
HTML

RDF data
Turtle




```
<http://www.bestbuy.com/site/Plan+9+from+Outer+Space
+-+DVD/7167941.p?id=23883&skuId=7167941&st=plan
%209%20from%20outer%20space&lp=6&cp=1> <http://
ogp.me/ns#title> "Plan 9 from Outer Space - DVD" .
<http://www.bestbuy.com/site/Plan+9+from+Outer+Space
+-+DVD/7167941.p?id=23883&skuId=7167941&st=plan
%209%20from%20outer%20space&lp=6&cp=1> <http://
ogp.me/ns#type> "product" .
<http://www.bestbuy.com/site/Plan+9+from+Outer+Space
+-+DVD/7167941.p?id=23883&skuId=7167941&st=plan
%209%20from%20outer%20space&lp=6&cp=1> <http://
ogp.me/ns#url> "http://www.bestbuy.com:80/site/Plan-9-
from-Outer-Space---DVD/6429224.p?
id=23883&skuId=7167941" .
<http://www.bestbuy.com/site/Plan+9+from+Outer+Space
+-+DVD/7167941.p?id=23883&skuId=7167941&st=plan
%209%20from%20outer%20space&lp=6&cp=1> <http://
ogp.me/ns#image> "http://images.bestbuy.com:80/
BestBuy_US/images/products/6429/6429224s.jpg" .
<http://www.bestbuy.com/site/Plan+9+from+Outer+Space
+-+DVD/7167941.p?id=23883&skuId=7167941&st=plan
```

Semantic Annotation in the Web

- In principle there are three ways to embed structured data with explicit semantic annotations within HTML documents

Domain specific microformats

Generic RDFa and embedded RDF

- HTML5 Microdata including schema.org

Microformats

- Microformats (µformats) emerged about 2005
- (X)HTML Markup to express (limited) semantics in an HTML document
  - designed to solve simple, specific problems
  - designed for humans first, machines second
  - used in web pages to describe a specific type of information, as e.g. a person, an event, a product, a review, etc.
- Applications can easily extract data from HTML documents
- In general, Microformats use the `class` attribute in HTML tags (most times `<span>` or `<div>` tags)
  and assign brief and descriptive names to entities and their properties

Microformats

- Microformats reuse the following (X)HTML tag attributes:
  - `class`
  - `rel`
  - `rev`
- Predefined standard microformats:
  - hCard - personal data (vCard, RFC2426)
  - hCalender – calendars and events
  - rel-Tag – tags, keywords, categories
  - XFN – XHTML friends network
  - hReview, VoteLinks -- opinions, ratings, and reviews
  - XOXO – lists and outlines
  - ...

http://microformats.org/

Microformats

- Simple Example: HTML marked up with **hCard** microformat

```
<div class="vcard">
 <span class="fn">Joe Blow</span>
 <span class="title">Senior Manager</span>
 <span class="org">The Example Company</span>
 <span class="adr">
    <span class="street-address">Hauptstr. 123, </span>
    <span class="postal-code">14482</span>
    <span class="locality">Potsdam</span>
  </span>
 Tel.<span class="tel">604-555-1234</span>
 <img class="photo" src="www.example.com/JoeBlow.jpg" />
</div>
```

Microformats

- Microformats can **easily be transcoded to RDF** via XSLT
- **New microformat** vocabularies **first must be consolidated** by the community, while always a new XSLT stylesheet has to be developed for extraction
- By using **more than one** microformat vocabulary in a single (X)HTML document the processing **complexity increases** rapidly
- **Conflicts** with used (X)HTML attributes might be possible

Embedding RDF in HTML Attributes

RDFa

- RDFa = RDF in HTML attributes
- enables generic RDF annotation in (X)HTML documents by reusing existing (X)HTML attributes

- RDFa 1.0 based on XHTML (W3C Recommendation 2008)
- RDFa 1.1 based on HTML5 (W3C Recommendation June 2012)
  - **RDFa Lite 1.1**
  - RDFa 1.1

RDFa

- RDFa reuses existing (X)HTML attributes (e.g. `href`, `src`) and introduces new HTML attributes
  - `vocab`,
  - `typeof`,
  - `property`,
  - `resource`,
  - `prefix`

RDFa

- First we need a **vocabulary** to talk about things

```
<p vocab="http://xmlns.com/foaf/0.1/">
   My name is Harald Sack and you can
   give me a ring via 1-800-555-0527.
</p>
```

RDFa

- Then we have to define the **type of thing** we are talking about

```
<p vocab="http://xmlns.com/foaf/0.1/"
   typeof="Person">
   My name is Harald Sack and you can
   give me a ring via 1-800-555-0527.
</p>
```

RDFa

- Now we can define all **properties** of the thing we are talking about

```
<p vocab="http://xmlns.com/foaf/0.1/" typeof="Person">
   My name is
   <span property="name">Harald Sack</span>
   and you can give me a ring via
   <span property="phone">1-800-555-0527</span>.
   <img property="img"
        src="http://hpi-web.de/images/harald.jpg" />
</p>
```

RDFa

- We can create **identifiers** for the things we are talking about

```
<p vocab="http://xmlns.com/foaf/0.1/"
   resource=“#harald“ typeof="Person">
   My name is
   <span property="name">Harald Sack</span>
   and you can give me a ring via
   <span property="phone">1-800-555-0527</span>.
   <img property="img"
        src="http://hpi-web.de/images/harald.jpg" />
</p>
```

- **resource** refers to the base URI of the webpage

RDFa

- And if the vocabulary is not sufficient to describe all properties, we can use additional vocabularies by using **prefixes**

```
<p vocab="http://xmlns.com/foaf/0.1/"
   prefix="ov: http://open.vocab.org/terms/"
   resource="#harald" typeof="Person">
   My name is
   <span property="name">Harald Sack</span>
   and you can give me a ring via
   <span property="phone">1-800-555-0527</span>.
   My favorite beverage is
   <span property="ov:preferredBeverage">Green Tea
   </span>.
</p>
```

RDFa

- Distinguish two different sorts of RDF triples
  - Triple with resource as object
  - Triple with literal as object

	Subject	Property	Object
Object is Literal	resource/about	property	content or #PCDATA
Object is Resource (URI)	resource/about	property/rel	href or resource

about / rel only for compatibility with RDFa 1.0

Differences with RDFa 1.0

```
<html>
...
<div xmlns:dc="http://purl.org/dc/elements/1.1/">
  I'm currently reading
   <span about=”urn:ISBN:0-596-00027-8”>
    <span property="dc:title">
       Programming Perl
    </span>
    by
    <span property="dc:creator">Larry Wall</span>
  </span>.
</div>
...
</html>
```

Namespace

Subject

Property

Object (Literal)

Differences with RDFa 1.0

```
<?xml version="1.0" encoding="UTF-8"?>
 <html xmlns="http://www.w3.org/1999/xhtml" version="XHTML+RDFa 1.0"
    xml:lang="en">
  <head>...</head>
  <body>...</body>
 </html>
</xml>
```

XHTML

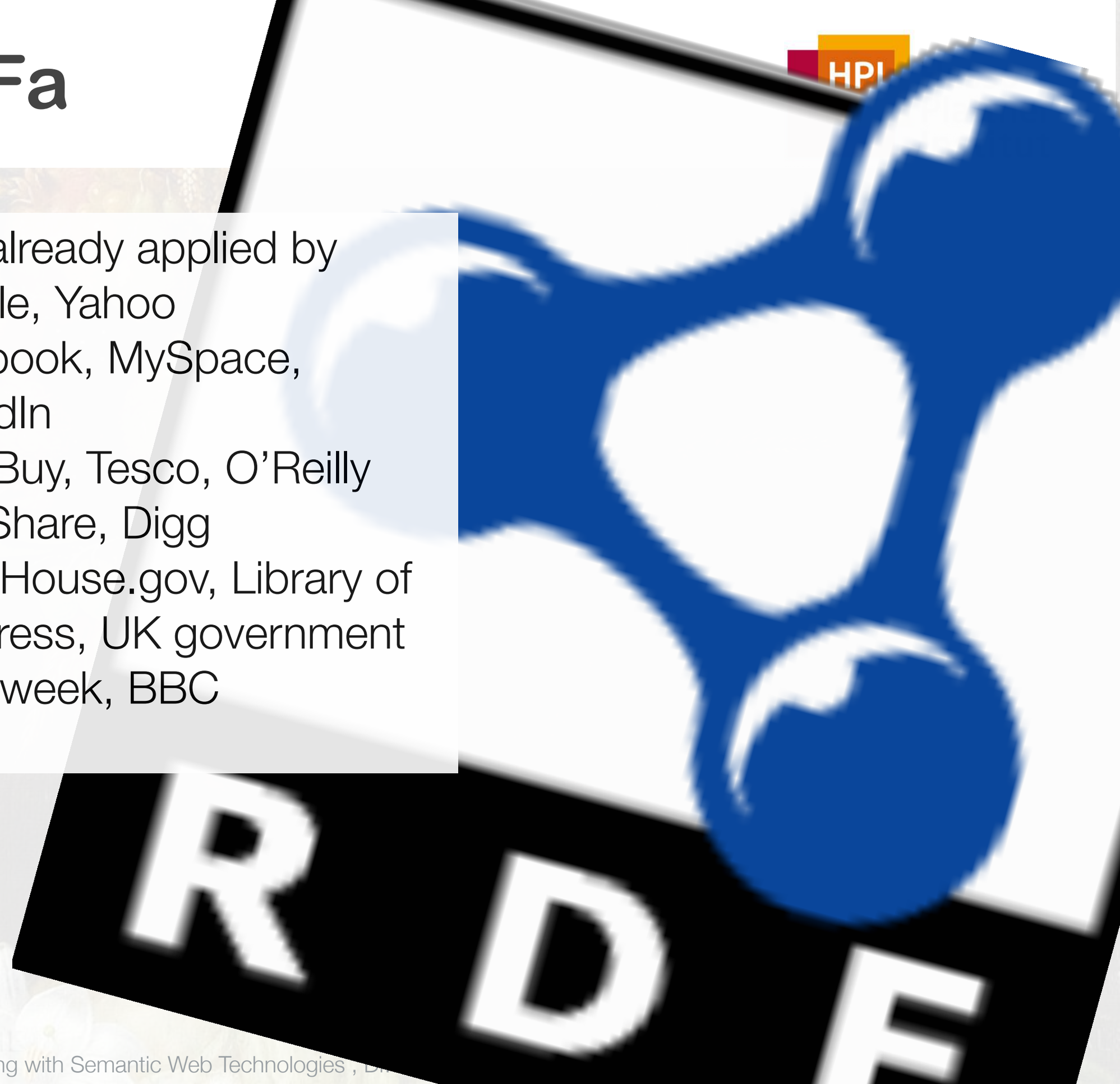
```
<!DOCTYPE html>
<html    version="HTML+RDFa 1.1" lang="en"
         xmlns="http://www.w3.org/1999/xhtml"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
         xmlns:cc="http://creativecommons.org/ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:foaf="http://xmlns.com/foaf/0.1/">
```

HTML5

RDFa

RDFa is already applied by

- Google, Yahoo
- Facebook, MySpace, LinkedIn
- Best Buy, Tesco, O'Reilly
- SlideShare, Digg
- WhiteHouse.gov, Library of Congress, UK government
- Newsweek, BBC
- ...



RDFa

...more RDFa Extractors

RDFa | Play | Documentation | Tools | Developers

Examples: Person | Social Network | Event | Place | Product | SVG

```
  xsd: http://www.w3.org/2001/XMLSchema#"
  typeof="gr:Offering">
  <div>
    <h1 property="gr:name">Canon Rebel T2i (EOS 550D)</h1>
    <div rel="foaf:depiction">
      <img style="float:left; width:20%" src="http://shop.usa.canon.com/wcsstore
    </div>
    <p property="gr:description">
      The Canon Rebel T2i (EOS 550D) is Cannon's top-of-the-line consumer digita
      It can shoot up to 18 megapixel resolution photos and features an ISO rang
    </p>
    <link rel="gr:hasBusinessFunction" href="http://purl.org/goodrelations/v1#Se
    <meta property="gr:hasEAN_UCC-13" content="013803123784" />
    Sale price:
    <span property="gr:hasPriceSpecification" typeof="gr:UnitPriceSpecification"
      <span property="gr:hasCurrency" content="USD">$</span>
      <span property="gr:hasCurrencyValue" datatype="xsd:float">899</span>
    </span>
    <link rel="gr:acceptedPaymentMethods" href="http://purl.org/goodrelations/v1
    <link rel="gr:acceptedPaymentMethods" href="http://purl.org/goodrelations/v1
    [<a rel="foaf:page" href="http://shop.usa.canon.com/">more...</a>]
  </div>
</div>
```

Canon Rebel T2i (EOS 550D)

The Canon Rebel T2i (EOS 550D) is Cannon's top-of-the-line consumer digital SLR camera. It can shoot up to 18 megapixel resolution photos and features an ISO range of 100-6400.

Sale price: $ 899 [more...]

Visualization | Raw Data

http://rdfa.info/play/

HTML5 microdata

- **HTML5 Microdata** extends microformats and addresses its shortcomings
- **schema.org** is the most prominent microdata vocabulary

schema.org

- Initiative started by Bing, Google, Yahoo! on June 2, 2011.

- **Goals:**
  - Semantic markup for web page content based on proprietary **HTML5 microdata schemata**

- **Search engines**, **web crawler,** and **Browser** recognize and understand schema.org markup

- small number of formats (extensible)

- RDF mapping available

schema.org and Rich Snippets

- **Snippets** = The few lines of text that appear under every search result
  - Designed to give users a sense for what's on the web page and why it's relevant to their query

- If Google understands the content on the web page, it can create **"Rich" Snippets**
  - **=** detailed information intended to help users with specific queries
  - You can help Google find this relevant information by adding additional microdata as HTML markup

- **Microdata** extends microformats and addresses its shortcomings
- **Microdata** annotates the DOM with scoped name/value pairs from custom vocabularies

- centered around **custom vocabularies**:
  anyone can define a microdata vocabulary and start embedding custom properties in their own web pages
- works with **name/value pairs**:
  every microdata vocabulary defines a set of named properties
- applies concept of "**scoping**":
  microdata re-use the hierarchical structure of the DOM itself to provide a way to say "all the properties within this element are taken from this vocabulary"

- HTML5 microdata specification comprises
  - **microdata Vocabularies** (`http://data-vocabulary.org/`)
    - Persons, Events, Organisations, Products, Reviews, Breadcrumbs, Ads, agreggated Reviews/Ads
  - **microdata Global Attributes**
    - `itemscope` (creates resource)
    - `itemtype` (applied vocabulary)
    - `itemid` (identifies resource)
    - `itemprop` (property of a resource)
    - `itemref` (references on other resources)

HTML5 microdata

```
<section itemscope itemtype="http://data-vocabulary.org/Person">
   Hello, my name is
   <span itemprop="name">John Doe</span>,
   I am a
   <span itemprop="title">graduate research assistant</span>
   at the
   <span itemprop="affiliation">University of Dreams</span>.
   My friends call me
   <span itemprop="nickname">Johnny</span>.
   You can visit my homepage at
   <a href="http://www.JohnnyD.com" itemprop="url">www.JohnnyD.com</a>.
   <section itemprop="address" itemscope
            itemtype="http://data-vocabulary.org/Address">
      I live at
      <span itemprop="street-address">1234 Peach Drive</span>
      <span itemprop="locality">Warner Robins</span>
      ,
      <span itemprop="region">Georgia</span>.
   </section>
</section>
```

http://yovisto.com/labs/swt1112/microdata01.html

HTML5 microdata

Extracted rich snippet data from the page

Item
Type: http://data-vocabulary.org/person
name = John Doe
title = graduate research assistant
affiliation = University of Dreams
nickname = Johnny
url
text = www.JohnnyD.com
href = http://www.johnnyd.com/
address = *Item*(1)

Item 1
Type: http://data-vocabulary.org/address
street-address = 1234 Peach Drive
locality = Warner Robins
region = Georgia

http://www.google.com/webmasters/tools/richsnippets

schema.org

- extends microdata vocabulary with proprietary **category hierarchies**:
  - Creative works:
    - CreativeWork, Book, Movie, MusicRecording, Recipe, TVSeries ...
  - Embedded non-text objects:
    - AudioObject, ImageObject, VideoObject
  - Event
  - Organization
  - Person
  - Place, LocalBusiness, Restaurant ...
  - Product, Offer, AggregateOffer
  - Review, AggregateRating
- and according **properties** (relations) with different domain and range

http://schema.org/docs/full.html

schema.org

http://schema.org/docs/full.html

Microdata Usage

HTML5 microdata

13% of all HTML pages contain structured data

Source: WebDataCommons.org, Febuary 2012

- 1.4 billion HTML pages parsed (Common Crawl corpus)
- 188 million pages contained Microformat, Microdata, RDFa data

30% of all HTML pages contain structured data

Source: Yahoo! Research, January 2012

- 3.2 billion HTML pages parsed (Bing Crawl corpus)
- 973 million pages contained Microformat, Microdata, RDFa data

Christian Bizer: LDOW2012 (04/16/2012)

Microdata Usage

HTML5 microdata

Structured Data in the Top k Alexa Websites

First x in AL	PLDs in CC #	PLDs in CC % AL	% containing structured data overall	% containing structured data RDFa	% containing structured data Microdata	% containing structured data Microformats
100	99	99.00	74.75	34.34	55.56	68.69
1k	963	96.30	62.62	40.08	31.67	46.11
10k	9,294	92.94	47.34	30.47	15.55	29.75
100k	85,058	85.01	31.94	16.46	7.20	20.07
1m	734,882	73.49	20.56	7.55	3.04	14.18

Bizer et al: Deployment of RDFa, Microdata, and Microformats on the Web – A Quantitative Analysis, (ISWC 2013)

next Lecture:

Lecture 4: Knowledge Representations I

Open HPI Course: Knowledge Engineering with Semantic Web Technologies