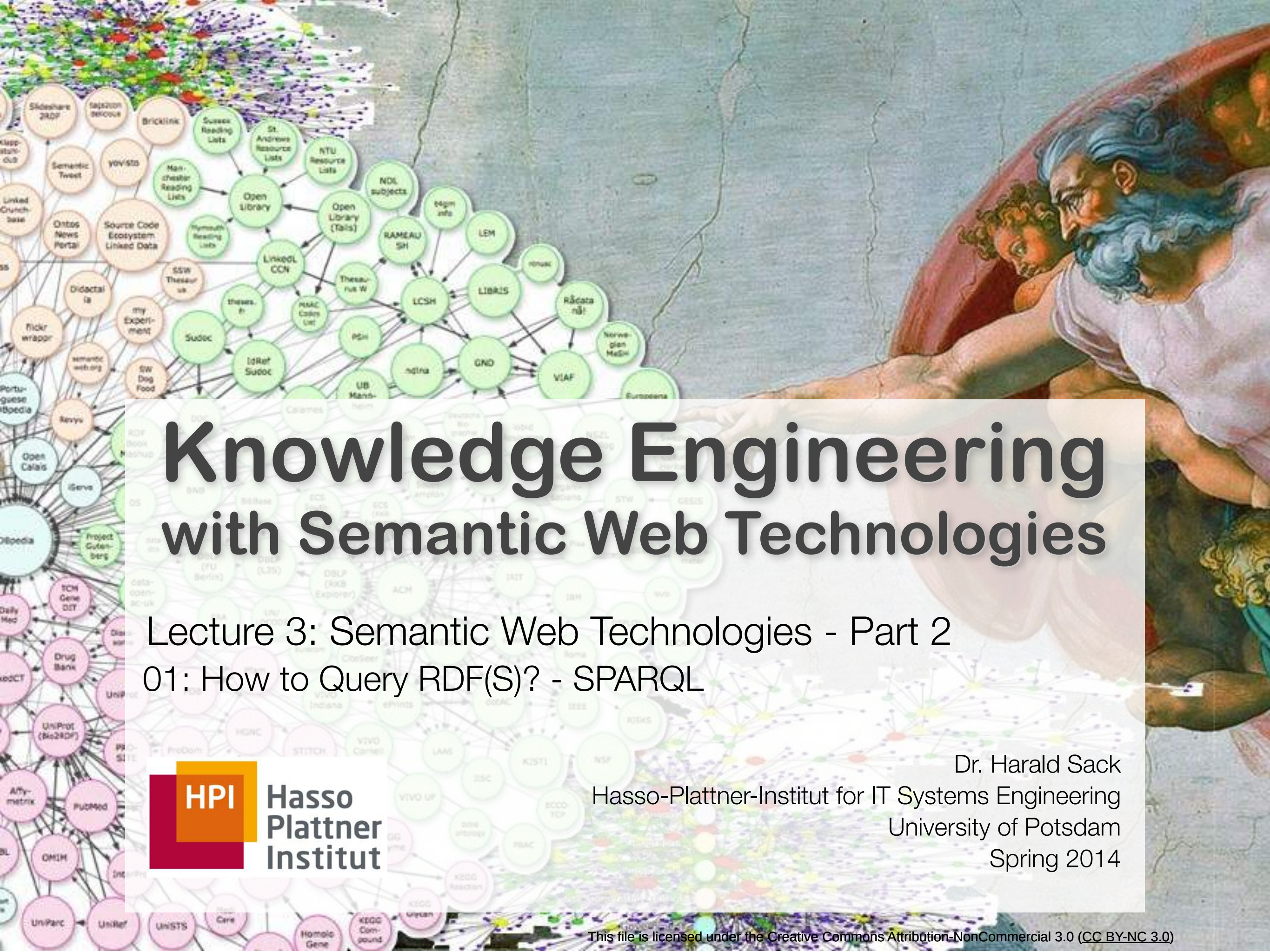


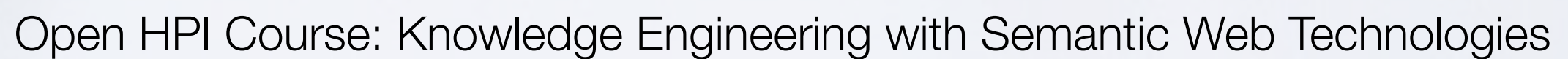


Knowledge Engineering with Semantic Web Technologies

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





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

The Semantic Web Technology Stack (not a piece of cake...)

Most apps use only a subset of the stack

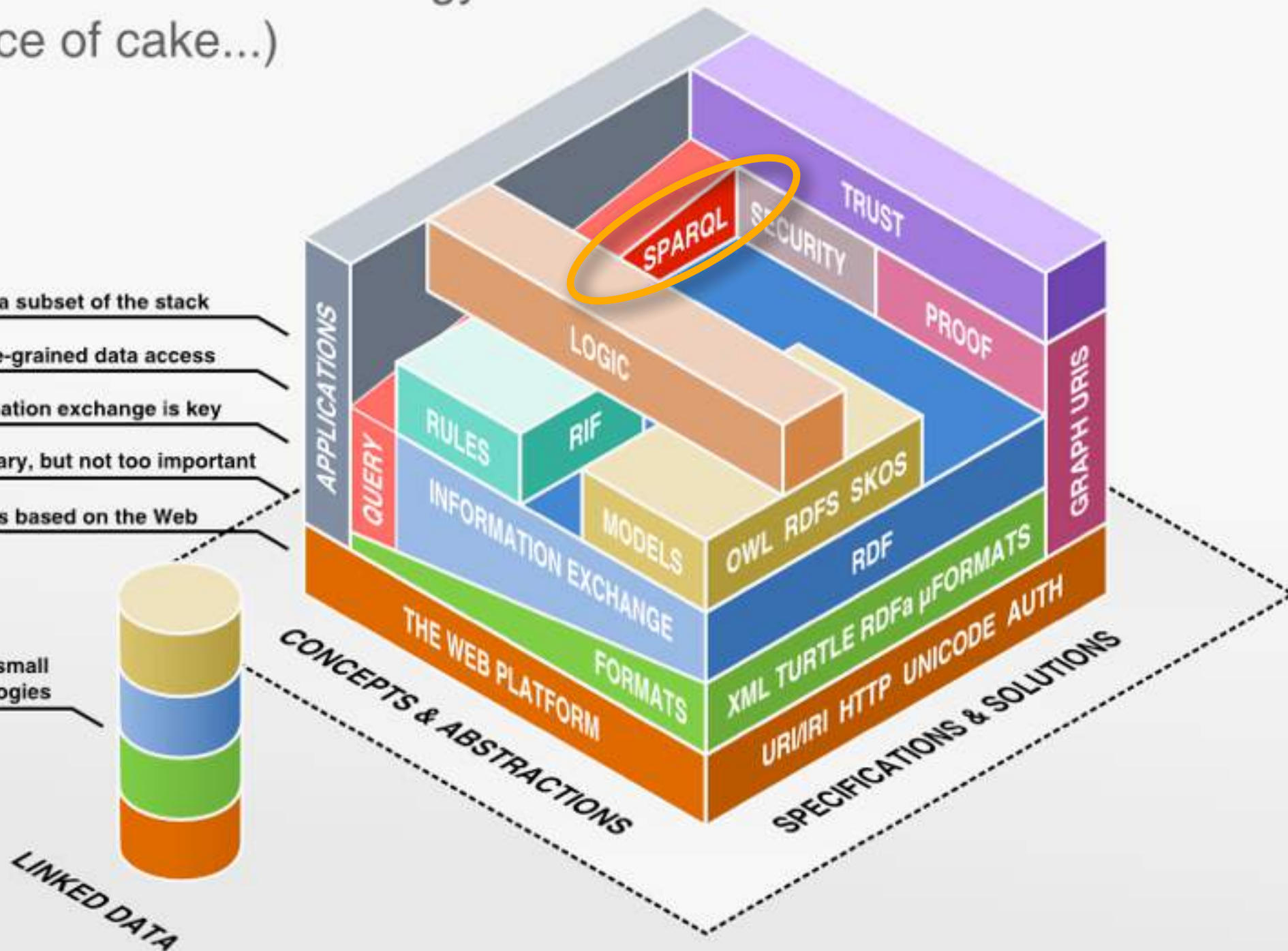
Querying allows fine-grained data access

Standardized information exchange is key

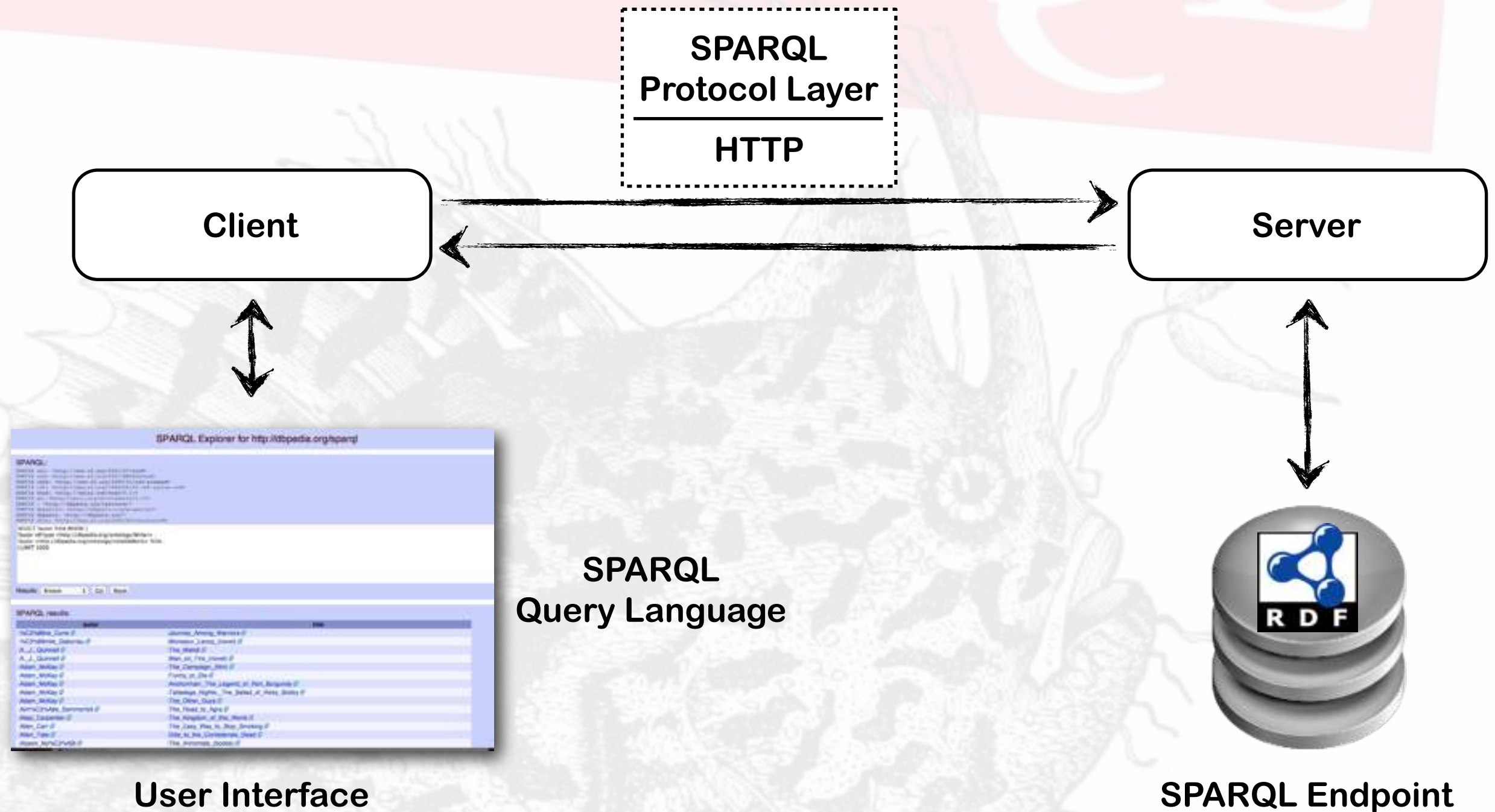
Formats are necessary, but not too important

The Semantic Web is based on the Web

Linked Data uses a small selection of technologies



SPARQL - A Query Language for RDF



- **SPARQL Protocol and RDF Query Language** 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



SPARQL 1.0

- **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
Fahrenheit 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
Fahrenheit 451	Ray Bradbury	1953

SPARQL Result

- 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** .

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" .

...

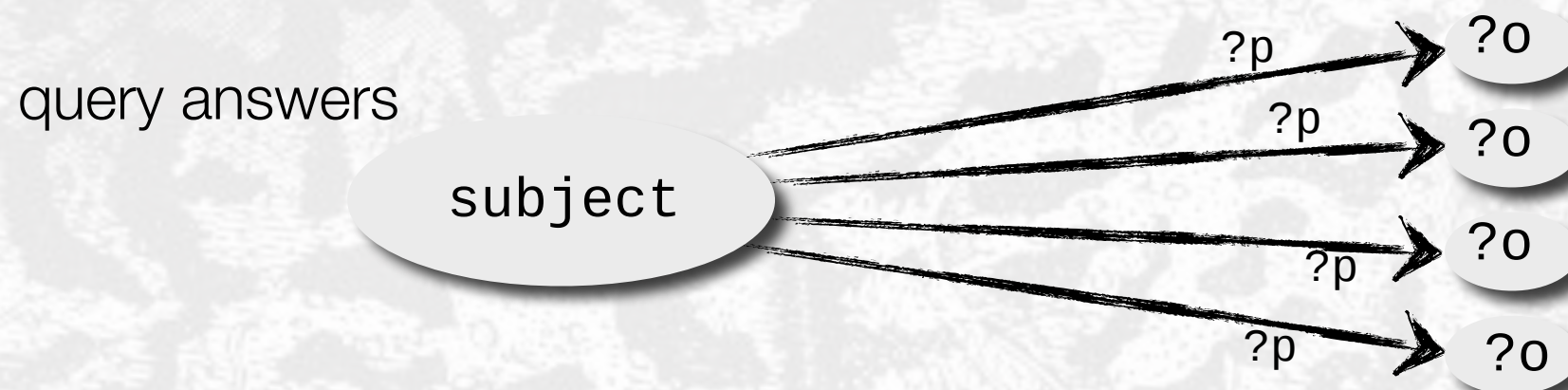
- **More Examples of (Graph) Triple Patterns:**
 - *Given a FOAF URI, find the name of a person:*
`<http://hpi-web.de/id#haraldsack> foaf:name ?surname .`
 - *Which persons have the family name “Schmidt”?*
`?person pers:familyName “Schmidt” .`

- 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 .`

- 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
```

specifies output variables

```
FROM <http://dbpedia.org/>
```

specifies graph to be queried

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

specifies graph pattern to be matched

- 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
```

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

- 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

SPARQL



02 How to query RDF(S) ? – SPARQL (2)

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