HPI Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Semantic Web Technologien

Tutorial zum Forum Metadaten

Dipl.-Inf. Magnus Knuth

Hasso-Plattner-Institut für Softwaresystemtechnik

Universität Potsdam

29. Mai 2012

HPI Hasso Plattner Institut

Dipl.-Inf. Magnus Knuth

- bis 2007 Studium Informatik Uni Leipzig
- 2007-2010 Institut für Medizinische Informatik, Statistik und Epidemiologie Leipzig
- seit 2010 wissenschaftlicher Mitarbeiter / Doktorand am Hasso-Plattner-Institut, FG „Semantic Technologies & Multimedia Retrieval“

- Forschung: Semantic Web, Linked Data Cleansing, Personalisierung

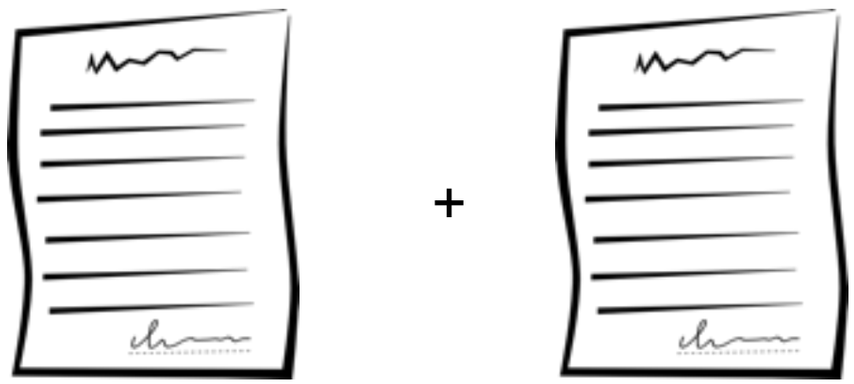
Hasso-Plattner-Institut für Softwaresystemtechnik

- das Hasso-Plattner-Institut wurde im Oktober 1998 im Rahmen einer Public-Private-Partnership gegründet
- Forschung und Lehre am HPI ist dem „*IT Systems Engineering*“ gewidmet
- 10 Professoren und ca. 100 Mitarbeiter in Forschung und Lehre
- aktuell 450 Studenten in universitären Studiengängen „IT Systems Engineering“ (BSc/MSc)
- CHE-Ranking sieht HPI seit 2010 auf Top-Rang

Semantic Web Technologien
Inhalt

XML – Extensible Markup Language

- Was XML alles kann
  - XML gestattet die Definition beliebiger Tags
    (Metasprache zur Definition neuer Markupsprachen)
  - die Definition neuer Tags erfolgt in einem speziellen Dokument,
    der Document Type Definition (DTD) / XML Schema Definition



XML – Extensible Markup Language

- Was XML alles kann

```
<h2>
  Max Mustermann
</h2>

<p>
  Sesamstr. 49a <br>
  <b>93123  Bad Sulzdetfurth </b>
</p>
```

HTML

XML – Extensible Markup Language

- Was XML alles kann

```
<h2>
  Max Mustermann
</h2>

<p>
  Sesamstr. 49a <br>
  <b>93123  Bad Sulzdetfurth </b>
</p>
```

HTML

```
<Adresse>
  <Vorname> Max </Vorname>
  <Nachname> Mustermann </Nachname>
  <Straße> Sesamstr. </Straße>
  <Hausnummer> 49a </Hausnummer>
  <PLZ> 93123 </PLZ>
  <Ort> Bad Sulzdetfurth </Ort>
</Adresse>
```

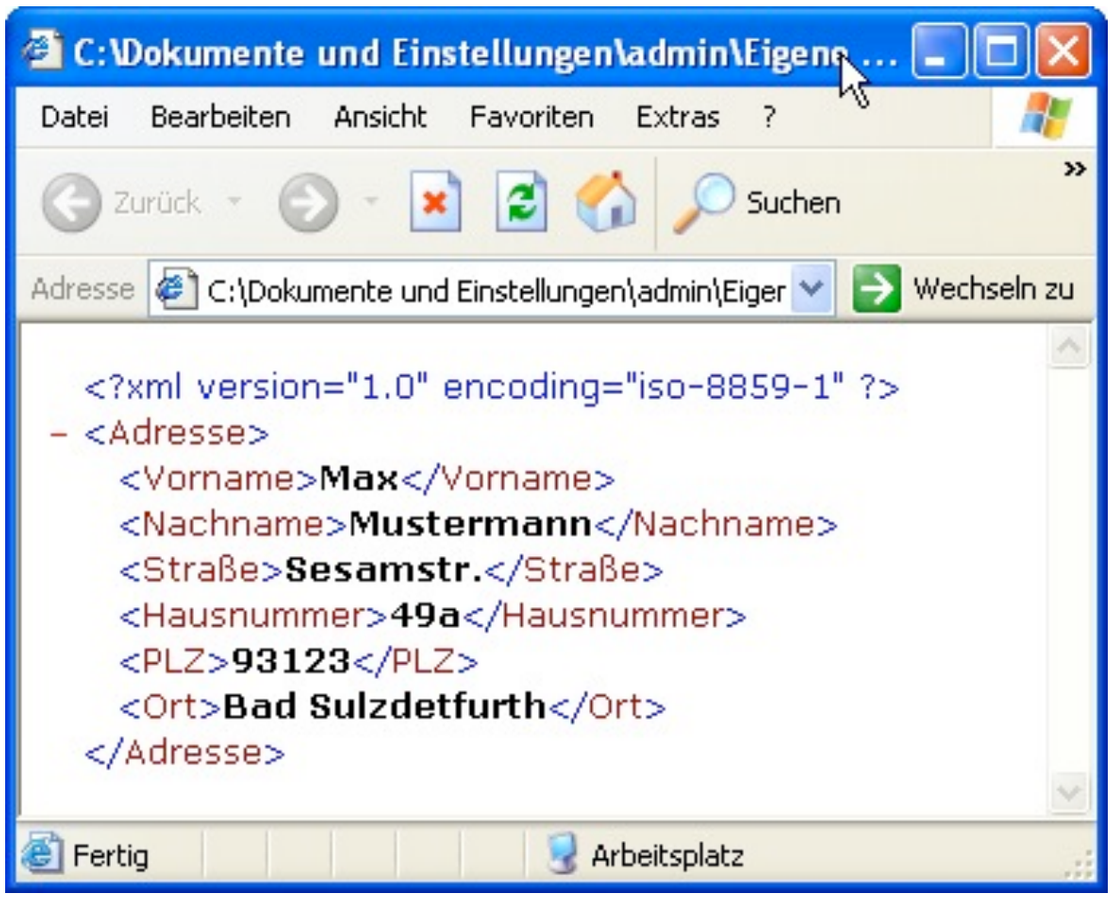
XML

XML als semi-strukturiertes Austauschdatenformat (Vokabular)
für beliebige Anwendungen

XML – Extensible Markup Language

- Was XML alles kann

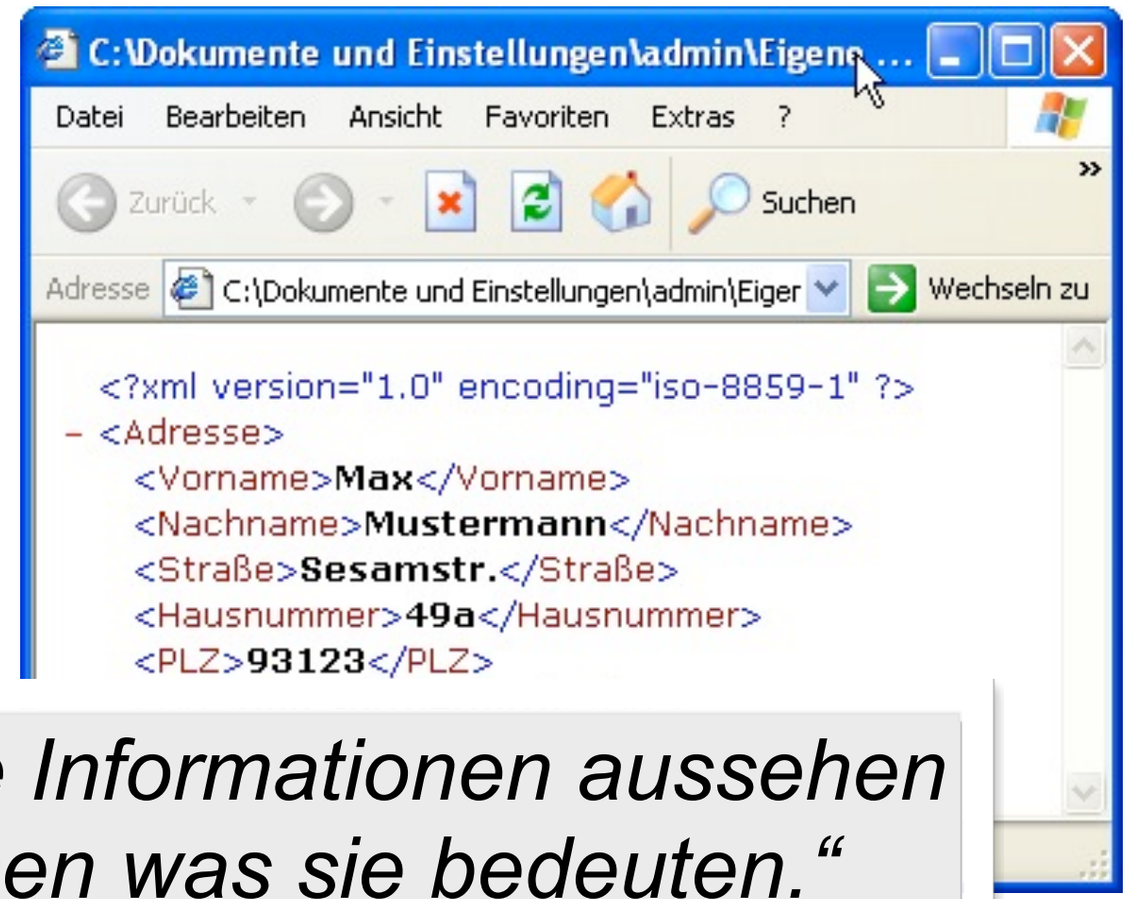
HTML



XML

XML – Extensible Markup Language

- Was XML alles kann

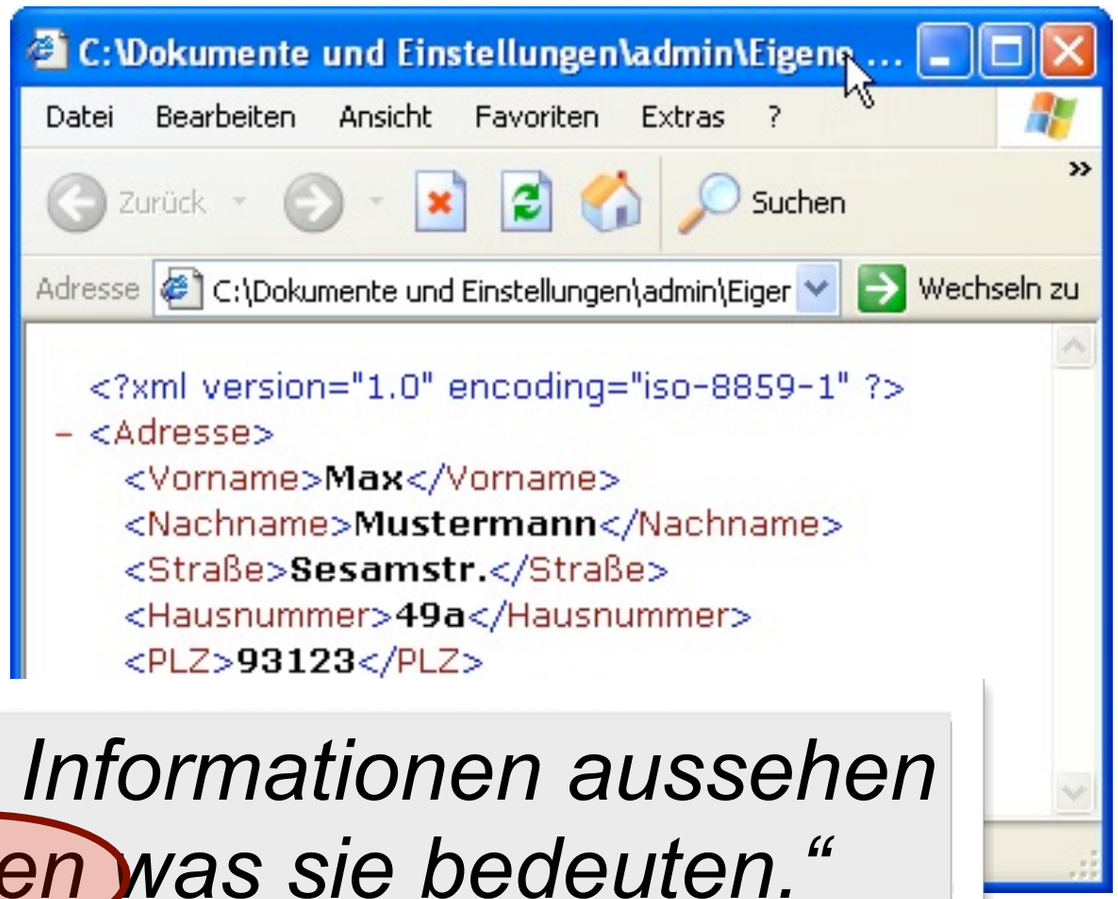


„HTML sagt Ihnen wie die Informationen aussehen sollen, aber XML sagt Ihnen was sie bedeuten.“

Quelle: Charles F. Goldfarb, „The XML Handbook“

XML – Extensible Markup Language

- Was XML alles kann



„HTML sagt Ihnen wie die Informationen aussehen sollen, aber XML sagt Ihnen was sie bedeuten.“

Quelle: Charles F. Goldfarb, „The XML Handbook“

XML – Extensible Markup Language

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE address SYSTEM "Beispiel.dtd">
<adresse>
  <vorname> Max </vorname>
  <nachname> Mustermann </nachname>
  <straße> Sesamstr. </straße>
  <hausnummer> 49a </hausnummer>
  <plz  prefix="D"> 93123 </plz>
  <ort> Bad Sulzdetfurth </ort>
</adresse>
```

XML – Extensible Markup Language

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE address SYSTEM "Beispiel.dtd">
<adresse>
  <vorname> Max </vorname>
  <nachname> Mustermann </nachname>
  <straße> Sesamstr. </straße>
  <hausnummer> 49a </hausnummer>
  <plz  prefix="D"> 93123 </plz>
  <ort> Bad Sulzdetfurth </ort>
</adresse>
```

XML Deklaration

XML – Extensible Markup Language

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE address SYSTEM "Beispiel.dtd">
<adresse>
  <vorname> Max </vorname>
  <nachname> Mustermann </nachname>
  <straße> Sesamstr. </straße>
  <hausnummer> 49a </hausnummer>
  <plz  prefix="D"> 93123 </plz>
  <ort> Bad Sulzdetfurth </ort>
</adresse>
```

XML Deklaration

XML DTD

XML – Extensible Markup Language

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE address SYSTEM "Beispiel.dtd">
<adresse>
   <vorname> Max </vorname>
   <nachname> Mustermann </nachname>
   <straße> Sesamstr. </straße>
   <hausnummer> 49a </hausnummer>
   <plz  prefix="D"> 93123 </plz>
   <ort> Bad Sulzdetfurth </ort>
</adresse>
```

XML Deklaration

XML DTD

XML Tags

XML – Extensible Markup Language

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE address SYSTEM "Beispiel.dtd">
<adresse>
   <vorname> Max </vorname>
   <nachname> Mustermann </nachname>
   <straße> Sesamstr. </straße>
   <hausnummer> 49a </hausnummer>
   <plz  prefix="D"> 93123 </plz>
   <ort> Bad Sulzdetfurth </ort>
</adresse>
```

XML Deklaration

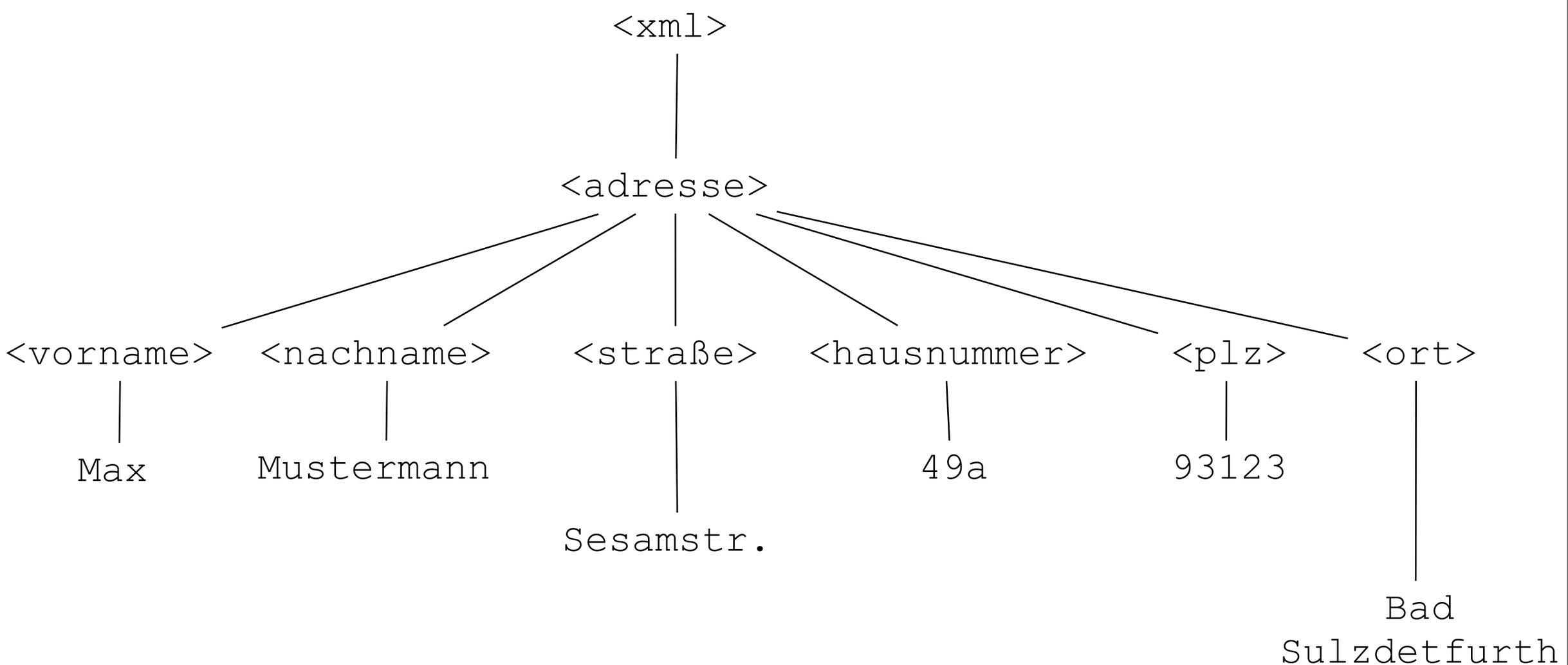
XML DTD

XML Tags

XML Attribute

XML – Extensible Markup Language

- Veranschaulichung von XML-Daten als gerichteter Graph



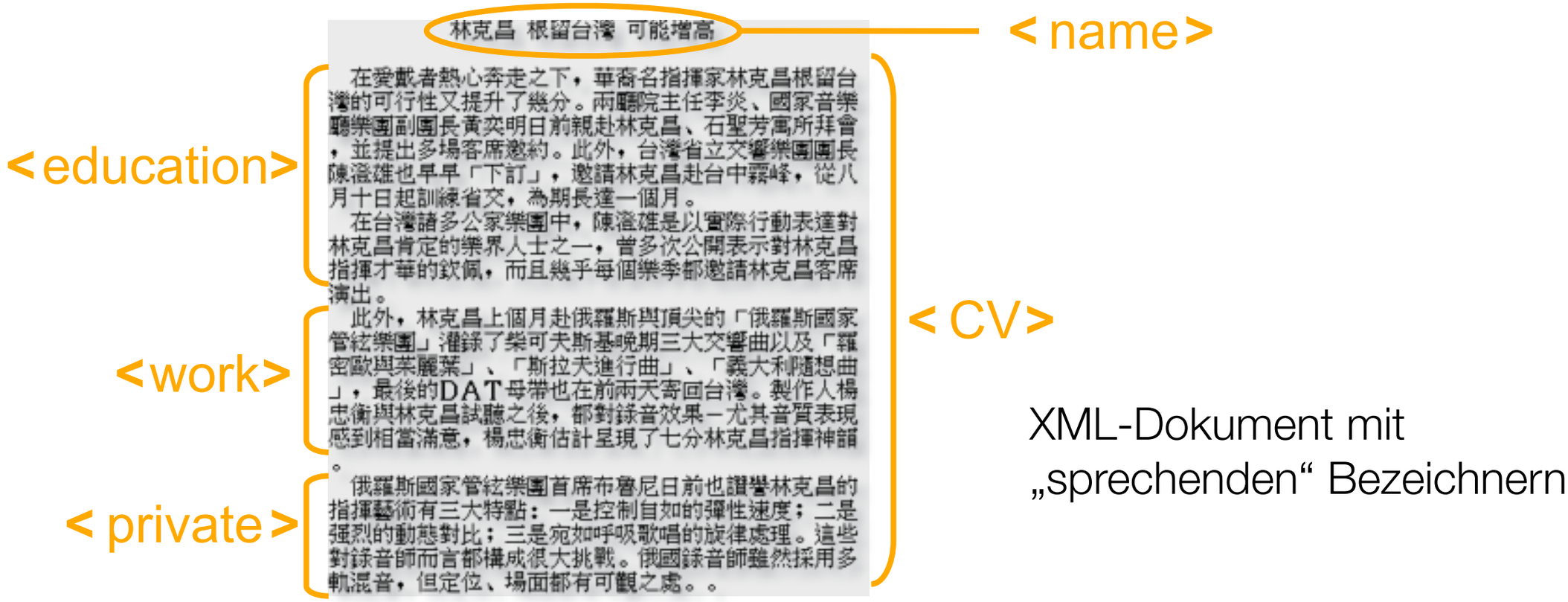
XML und Semantik

- allgemeine, erweiterbare Meta-Markup-Sprache zur **Repräsentation von semi-strukturierten Daten**

```
<adresse>
 <name>
  <vorname>Magnus</vorname>
  <nachname>Knuth</nachname>
 </name>
 <strasse>Prof.-Dr.-Helmert-Str.</strasse>
 <hausnummer>2-3</hausnummer>
 <plz>14482</plz>
 <ort>Potsdam</ort>
</adresse>
```

- **Aber**: *woher weiß man, dass <adresse> eine Adresse bezeichnet?*

Warum XML alleine noch nicht ausreicht…



XML-Dokument mit
„sprechenden“ Bezeichnern

Dokument

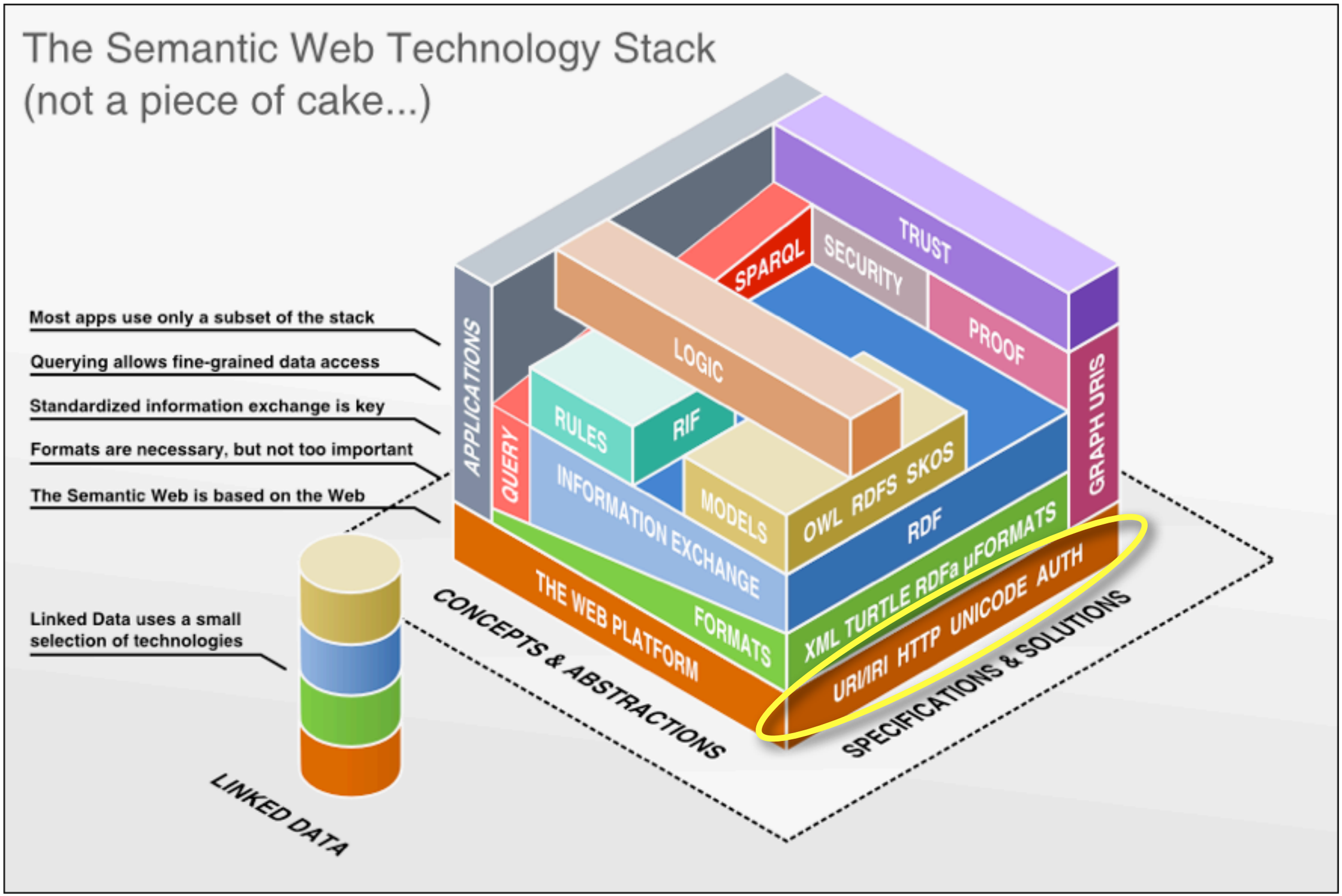
Warum XML alleine noch nicht ausreicht…

Dokument

XML-Dokument mit „sprechenden“ Bezeichnern aus der Sicht eines Programms

1. Semantic Web Basisarchitektur

1.2 URI - Uniform Resource Identifier



1. Semantic Web Basisarchitektur

1.2 URI - Uniform Resource Identifier

- Ein **Uniform Resource Identifier** (URI) definiert ein einfaches und erweiterbares Schema zur weltweit eindeutigen Identifikation von abstrakten oder physikalischen Ressourcen (RFC 3986).

- Ressource kann jedes Objekt sein, das (im Kontext der jeweiligen Anwendung) eine klare Identität besitzt,
  - also z.B. Webseiten, Bücher, Orte, Personen, Beziehungen zwischen diesen Dingen, abstrakte Konzepte usw.
- Das URI Konzept ist in verschiedenen Anwendungsbereichen bereits etabliert,
  - wie z.B. Web (URL, PRN, pURL),
  - Bücher (ISBN),
  - Digital Object Identifier (DOI)

URI, Ressource und Repräsentation

URI

http://www.tour-eiffel.fr/index.html

identifiziert

Ressource:
Eiffelturm

beschreibt

Repräsentation

Metadaten:
Content-type: text/html
Data:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
 HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Le site officiel de la Tour Eiffel</title>
 ...
</html>
```

repräsentiert

Information über den Eiffelturm

Ressource:
Eiffelturm
Metadaten

Bezeichner und Bezeichnetes

„The distinguishing characteristic of [information] resources is that all of their essential characteristics can be conveyed in a message.“

W3C: Architecture of the World Wide Web, Volume One

- eine Ressource wird durch ihre **Metadaten** beschrieben
- auch wenn eine Ressource nicht vom Web Server zurückgeliefert werden kann, ist eventuell dennoch die Repräsentation einer Ressource verfügbar, die die ursprüngliche Ressource hinreichend beschreibt

„Metadata is machine understandable information about web resources or other things.“

Tim Berners-Lee: Axioms of Web Architecture: Metadata, 1997

1. Semantic Web Basisarchitektur

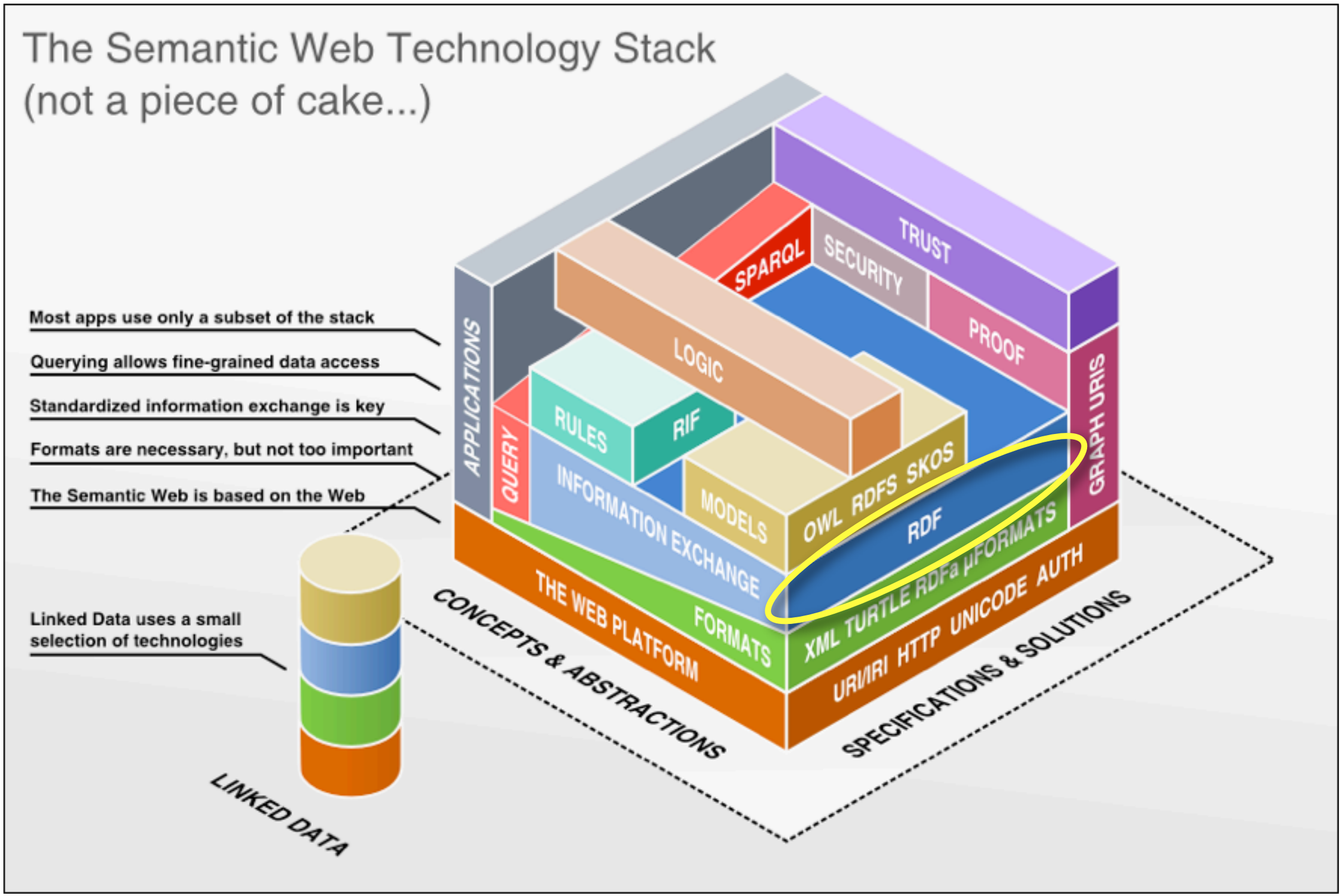
1.2 URI - Uniform Resource Identifier

URI = schema"://"[userinfo"@"]host[:port][path]["?"query]["#"fragment]

- **schema:** z.B. http, ftp, mailto,...
- **userinfo:** z.B. username:password
- **host:** z.B. Domain-Name, IPv4/IPv6-Adresse
- **port:** z.B. 80 für Standard http-Port
- **path:** z.B. Pfadangabe im WWW-Server Filesystem
- **query:** z.B. Parameter, die an Anwendung weitergegeben werden
- **fragment:** z.B. Angabe eines bestimmten Dokumententeilbereichs

1. Semantic Web Basisarchitektur

1.3 RDF - Resource Description Framework



Wissensrepräsentation - ein Beispiel

- Wie repräsentiere ich folgende Aussage?

Magnus Knuth hat die Telefonnummer ++49 331 5509570

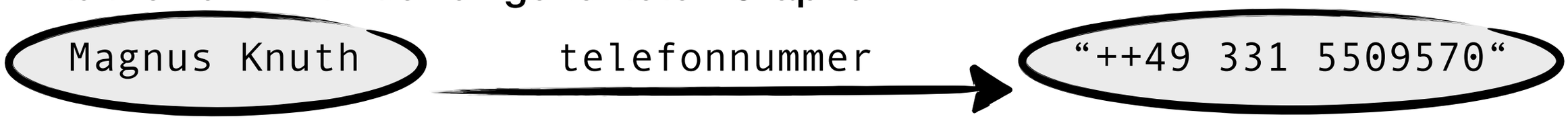
- XML:

```
<telefon>
  <inhaber>Magnus Knuth</inhaber>
  <nummer>++49 331 5509570</nummer>
</telefon>
```

```
<person name="Magnus Knuth">
  <telefonnummer>++49 331 5509570</telefonnummer>
</person>
```

```
<person name="Magnus Knuth" tel="++49 331 5509570" />
```

- intuitive Form mit Hilfe von **gerichteten Graphen**



1. Semantic Web Basisarchitektur

1.3 RDF - Resource Description Framework

»RESOURCE

»DESCRIPTION

»FRAMEWORK

- Ursprünglich entwickelt zur Angabe von Metadaten für Web-Ressourcen
  - 1995-1997 proprietäres Meta Content Framework (Netscape)
  - 1997 RDF als allgemeine Sprachdefinition für Metadaten, W3C Draft
  - 1998 erste RDF W3C Recommendation (http://www.w3c.org/RDF)
  - 2004 überarbeitete RDF W3C Recommendation
- Syntaktische Konvention eines einfachen, universellen semantischen Datenmodells
- RDF ist geeignet zur Beschreibung aller möglichen Web-Ressourcen
- mit RDF soll ein möglichst hohes Maß an Interoperabilität ermöglicht werden

1. Semantic Web Basisarchitektur

1.3 RDF - Resource Description Framework

»RESOURCE

»DESCRIPTION

»FRAMEWORK

- Wissen (bzw. Information) in RDF wird in Form einer Liste einzelner Statements ausgedrückt, wobei alle Statements dem selben einfachen Aufbau folgen

Resource (URI) **Property (URI)** **Literal / Resource (URI)**

Subject **Predicate** **Object**

1. Semantic Web Basisarchitektur

1.3 RDF - Resource Description Framework

»RESOURCE

»DESCRIPTION

»FRAMEWORK

- Bestandteile des RDF-Graphen:
  - **URI**: zur eindeutigen Referenzierung von Ressourcen und Properties
  - **Literale**: beschreiben Datenwerte, denen keine separate Existenz zukommt
    - typisierte Literale können mit Hilfe von XML Schema Datentypen ausgedrückt werden
      - » “Semantik“^^<http://www.w3.org/2001/XMLSchema#string>
    - Language-Tags geben (natürliche) Sprache des Literals an
      - » “Semantik“@de , “Semantics“@en

»RESOURCE

»DESCRIPTION

»FRAMEWORK

- Bestandteile des RDF-Graphen:
  - **leere Knoten**: erlauben Aussagen über ein Individuum ohne dieses zu benennen

1. Semantic Web Basisarchitektur

1.3 RDF - Resource Description Framework

»RESOURCE

»DESCRIPTION

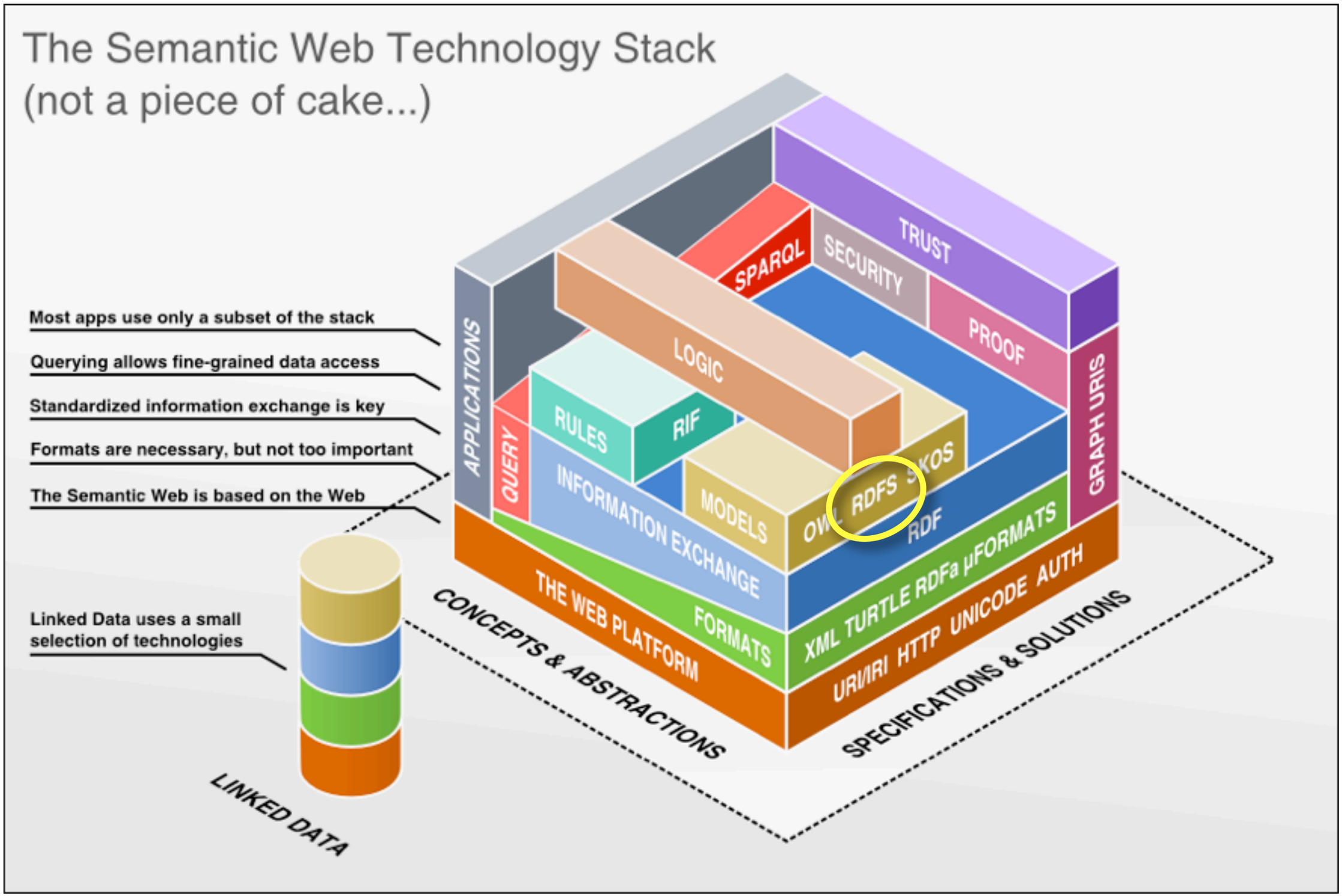
»FRAMEWORK

- RDF Darstellungsvarianten:
  - **N3 Notation**: direkte Auflistung der Triple
  - **Turtle (Terse RDF Triple Language)**: Erweiterung von N3
  - **RDF XML Serialisierung**:
  - **RDFa**: eingebettet in HTML-Code

- RDF Validator / Converter: http://www.rdfabout.com/demo/validator/

1. Semantic Web Basisarchitektur

1.4 RDFS - RDF Schema



1. Semantic Web Basisarchitektur

1.4 RDFS - RDF Schema

- RDF Schema spezifiert ein Datenmodel, über das RDF Statements entworfen werden können
  - abstrakter Datentyp (Klassen)
    - Instanziierung via <rdf:type>
  - hierarchisches Klassenmodell und Vererbung (Subklassen)
  - Eigenschaften und Restriktionen für Properties

= „Beschreibungssprache zur Definition strukturierter Vokabularien ...“

RDFS Vokabular

- Klassen
  - `rdfs:Resource`
    jede Entität in einem RDF-Modell ist Instanz dieser Klasse
  - `rdf:Property`
    Basisklasse für Eigenschaften
  - `rdfs:Class`
    Klassenkonzept, legt ein abstraktes Objekt fest und dient in Verbindung mit rdf:type zur Erzeugung von Instanzen
  - `rdfs:Literal`
    Klasse für Literalwerte, also Zeichenketten, ...

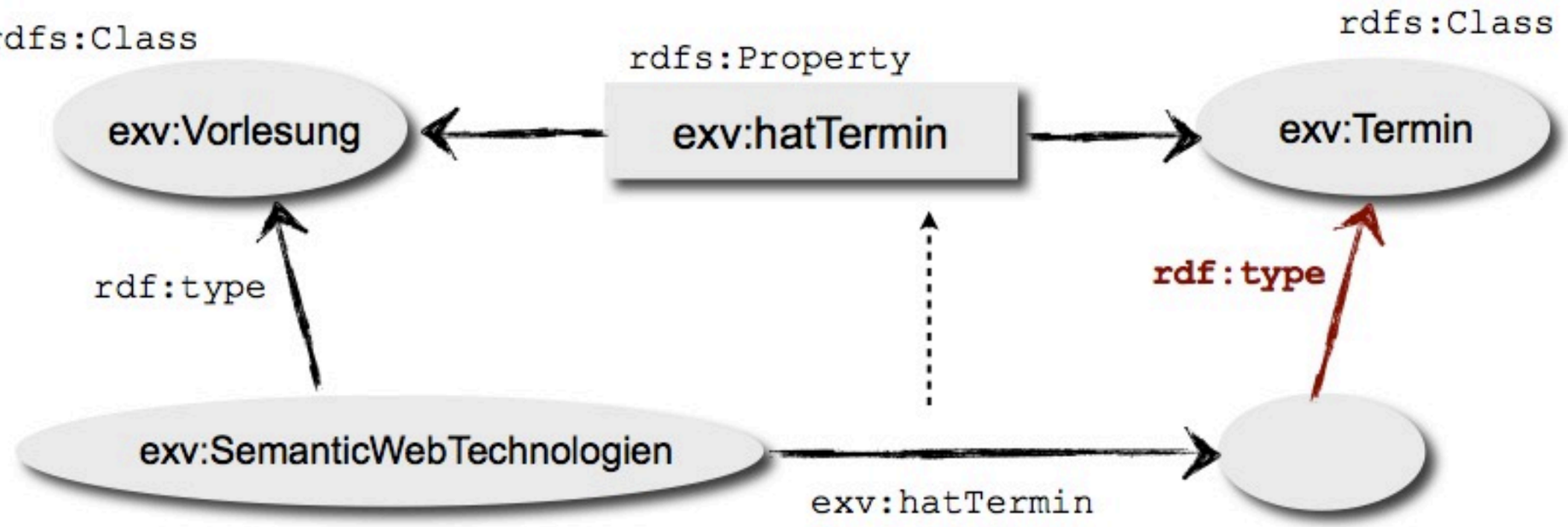
- zusätzlich noch
  `rdfs:Datatype, rdf:XMLLiteral, rdfs:Container, rdfs:ContainerMembershipProperty, ...`

RDFS Vokabular

- Properties
  - `rdfs:subClassOf`
    transitive Eigenschaft zur Festlegung von Vererbungshierarchien von Klassen
  - `rdfs:subPropertyOf`
    transitive Eigenschaft zur Festlegung von Vererbungshierarchien von Eigenschaften
  - `rdfs:domain`
    legt Anwendungsbereich einer Eigenschaft in Bezug auf eine Klasse fest
  - `rdfs:range`
    legt Wertebereich einer Eigenschaft fest

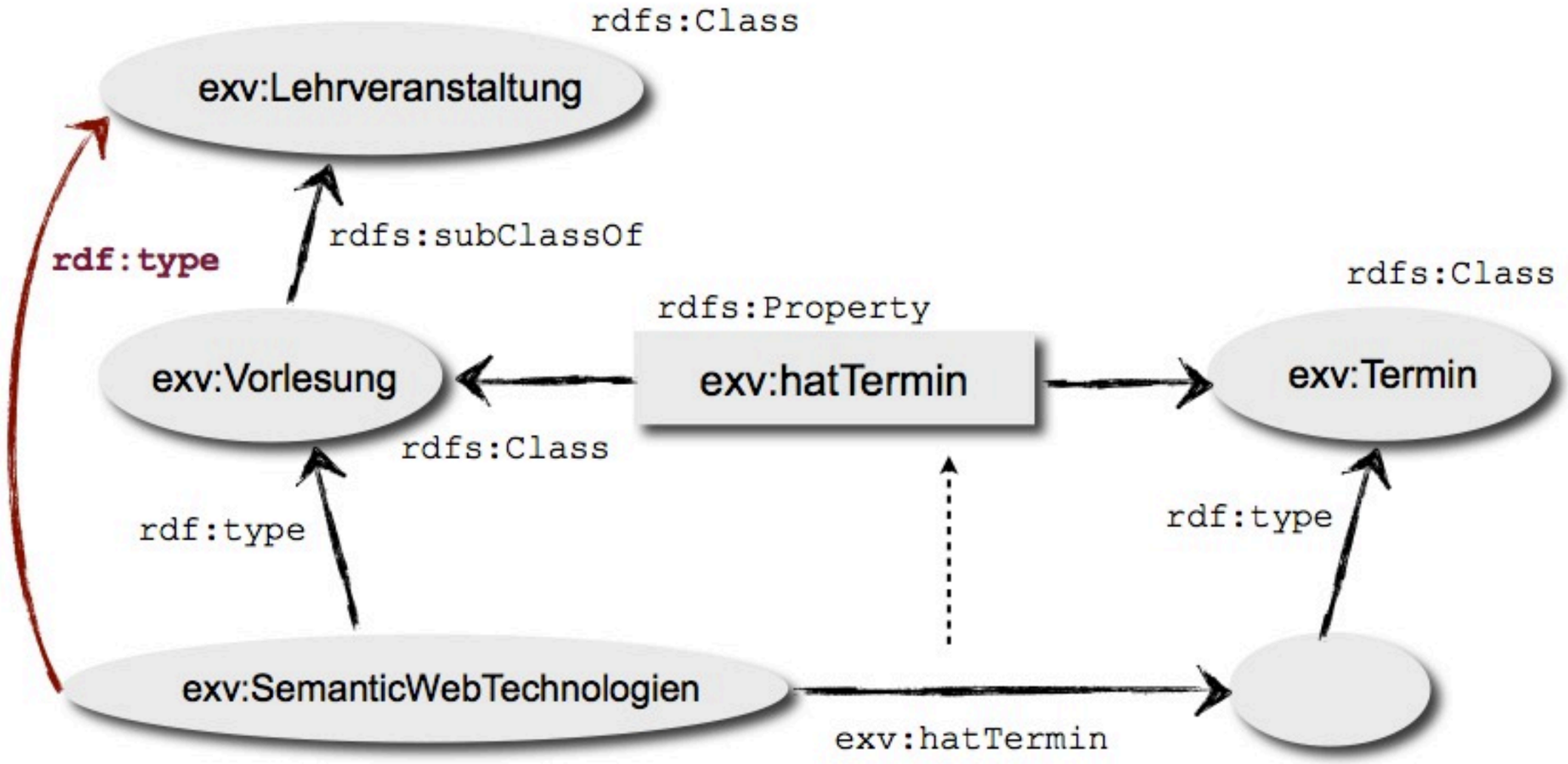
- zusätzlich noch
  `rdfs:label`, `rdfs:seeAlso`, `rdfs:comment`, `rdfs:isDefinedBy`, ...

Welche Schlussfolgerungen können wir mit RDF(S) ziehen?



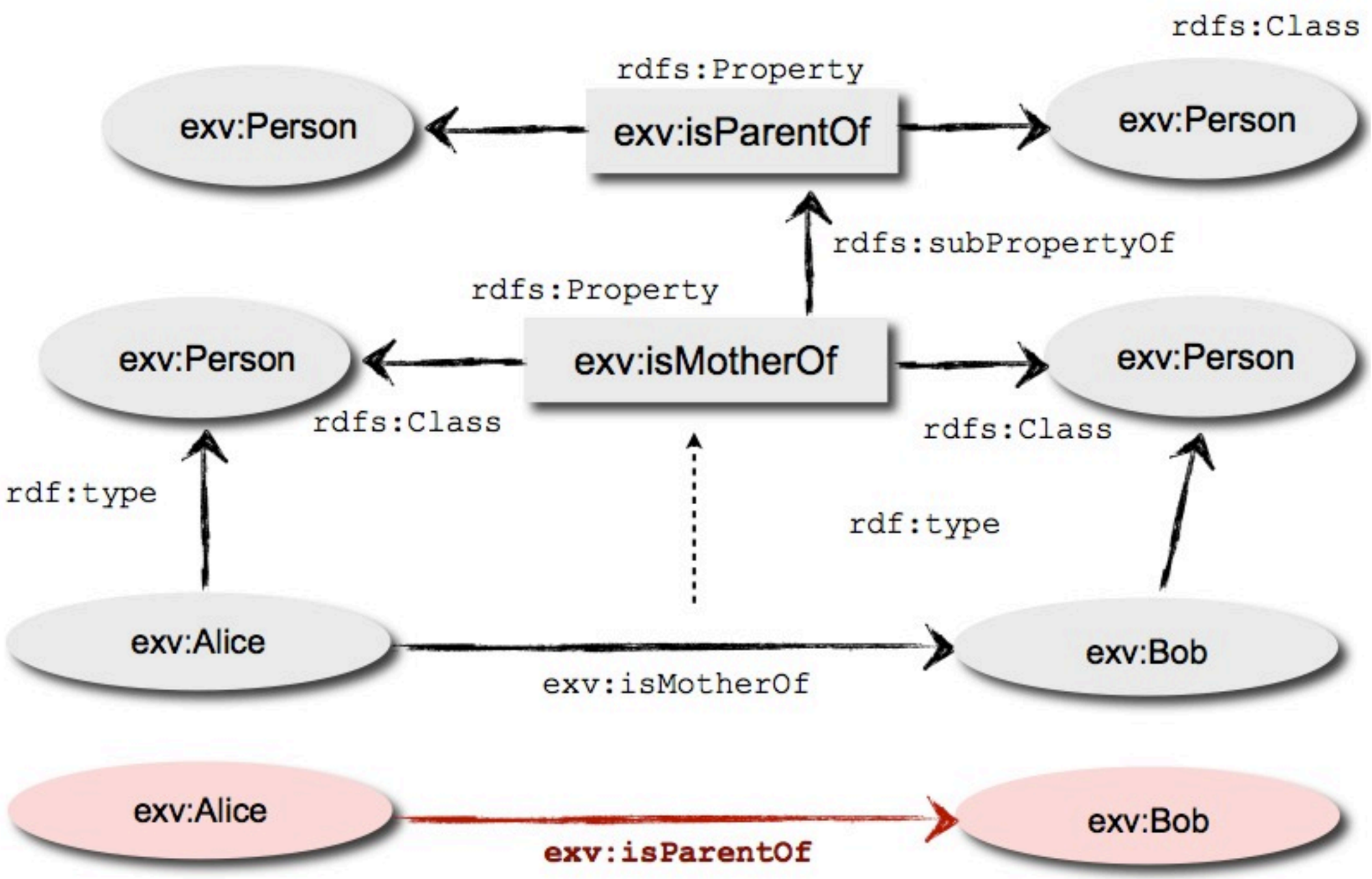
- Herleitung der Klassenzugehörigkeit einer Entität aus der Bildmenge (Range) eines Properties.

Welche Schlussfolgerungen können wir mit RDF(S) ziehen?



- Herleitung der (Super-)Klassenzugehörigkeit einer Entität aus einer Klassenhierarchie.

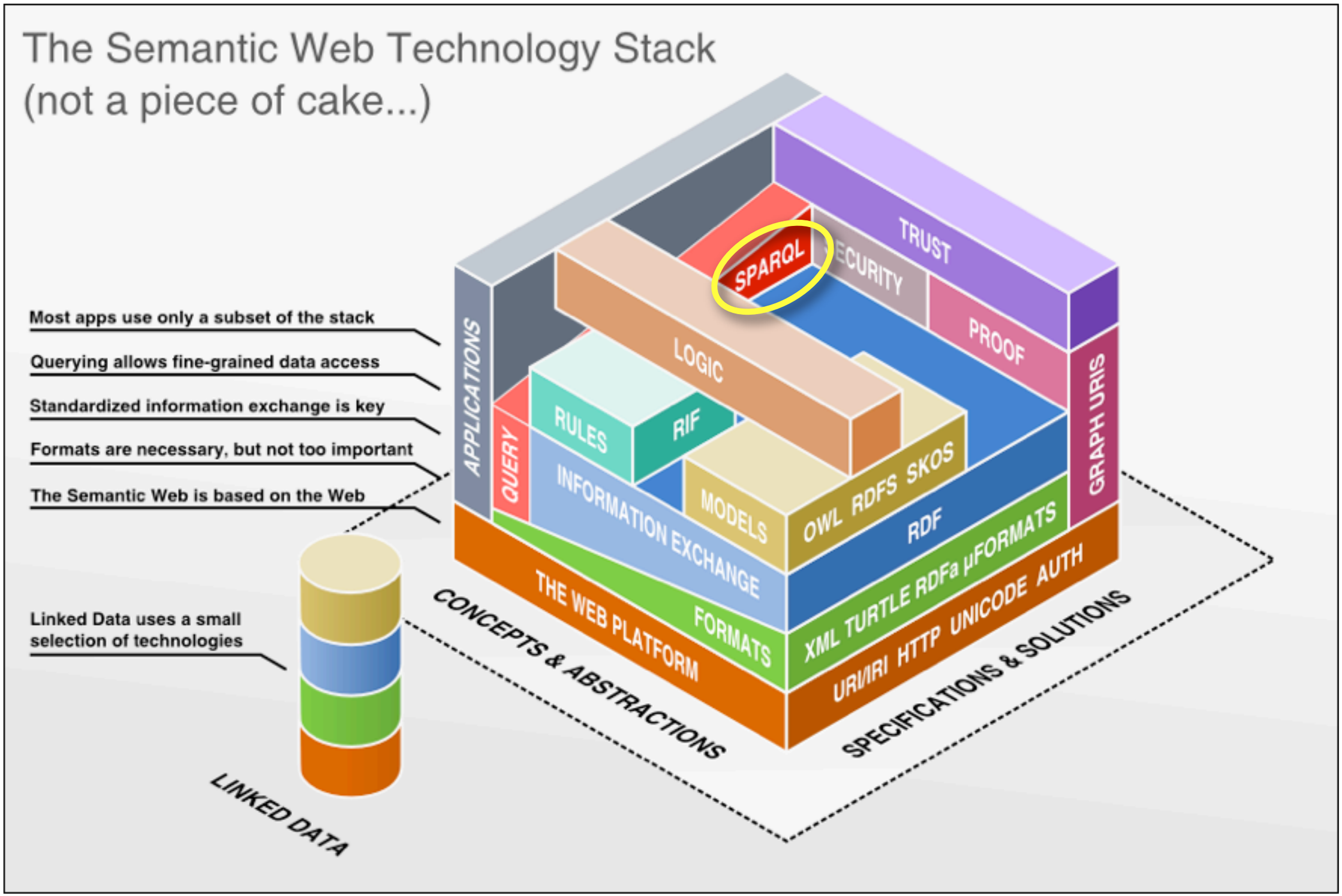
Welche Schlussfolgerungen können wir mit RDF(S) ziehen?



- Herleitung neuer Fakten aus SubPropertyBeziehungen

1. Semantic Web Basisarchitektur

1.5 SPARQL - Abfragesprache



1. Semantic Web Basisarchitektur

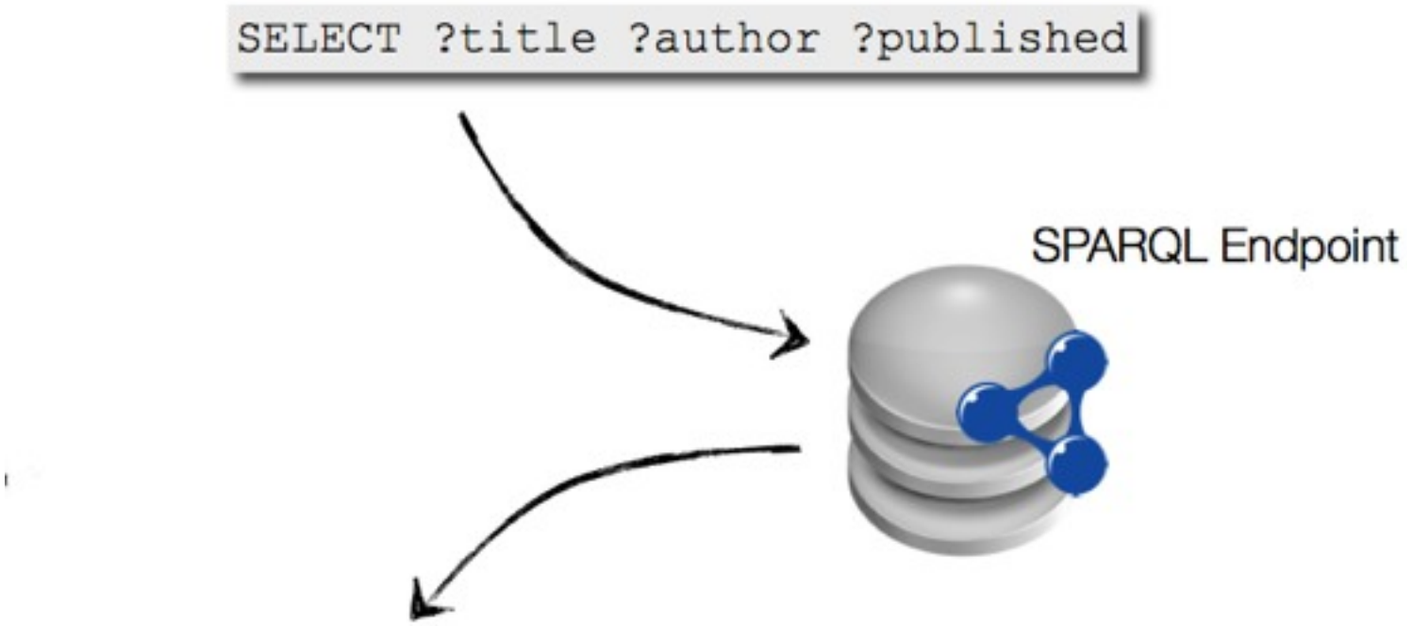
1.5 SPARQL - Abfragesprache

- **SPARQL P**rotocol **a**nd **R**DF **Q**uery **L**anguage
  - eine Abfragesprache zur Traversierung von RDF-Graphen
  - eine Protokollschicht, um SPARQL z.B. via HTTP zu nutzen
  - ein XML-Rückgabeformat
  - ein W3C Standard (seit 2008)
  - angelehnt an SQL

1. Semantic Web Basisarchitektur

1.5 SPARQL - Abfragesprache

- SPARQL **Variablen** werden an RDF-Terme gebunden
  - z.B. `?journal, ?disease, ?price`
- wie in SQL erfolgt die **Abfrage** von Variablen über ein SELECT Statement
  - z.B. `SELECT ?title ?author ?published`
- ein SELECT-Statement liefert **Abfrageergebnisse** in Form einer Tabelle



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

Triple Pattern

- ein RDF-Triple in Turtle Notation, das an beliebiger Stelle (Subject, Predicate, Object) Variablen enthalten kann
- Triple-Pattern können zu komplexen Suchanfragen kombiniert werden

- Beispiele:
  - finde Länder und ihre Hauptstädte

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

  - welche Personen tragen den Nachnamen „Schmidt“?

```
?person foaf:surname “Schmidt“ .
```

  - ausgehend von einem FOAF URI, finde den Namen der Person

```
<http://magnus.13mm.de/> foaf:name ?name .
<http://magnus.13mm.de/> foaf:knows ?friend .
?friend foaf:name ?friend_name .
```

SPARQL Abfrageformat

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

- angelehnt an SQL
- Triple im `WHERE`-Teil definieren Graph-Abfrage mit Variablen
- Abfrage liefert Tabelle mit passenden Paaren

- `PREFIX` - legt Namespaces fest
- `FROM` - gibt RDF-Quellgraphen an
- `BASE` - legt Basis-URI fest
- `ORDER BY` - legt Sortierung fest
- `LIMIT` / `OFFSET`- begrenzt Ergebnismenge

SPARQL Abfrageformat

- mit `FILTER` kann die Ergebnis-menge eingeschränkt werden

```
SELECT ?title, ?price
WHERE { ?x dc:title ?title .
        ?x ns:price ?price .
        FILTER(?price < 30) }
```

- mit `OPTIONAL` können optionale Elemente aus dem Graphen selektiert

```
SELECT ?title, ?price
WHERE { ?x dc:title ?title .
  OPTIONAL { ?x ns:price ?price . }
}
```

- mit `UNION` können mehrere Abfragen kombiniert werden (ODER-Verknüpfung)

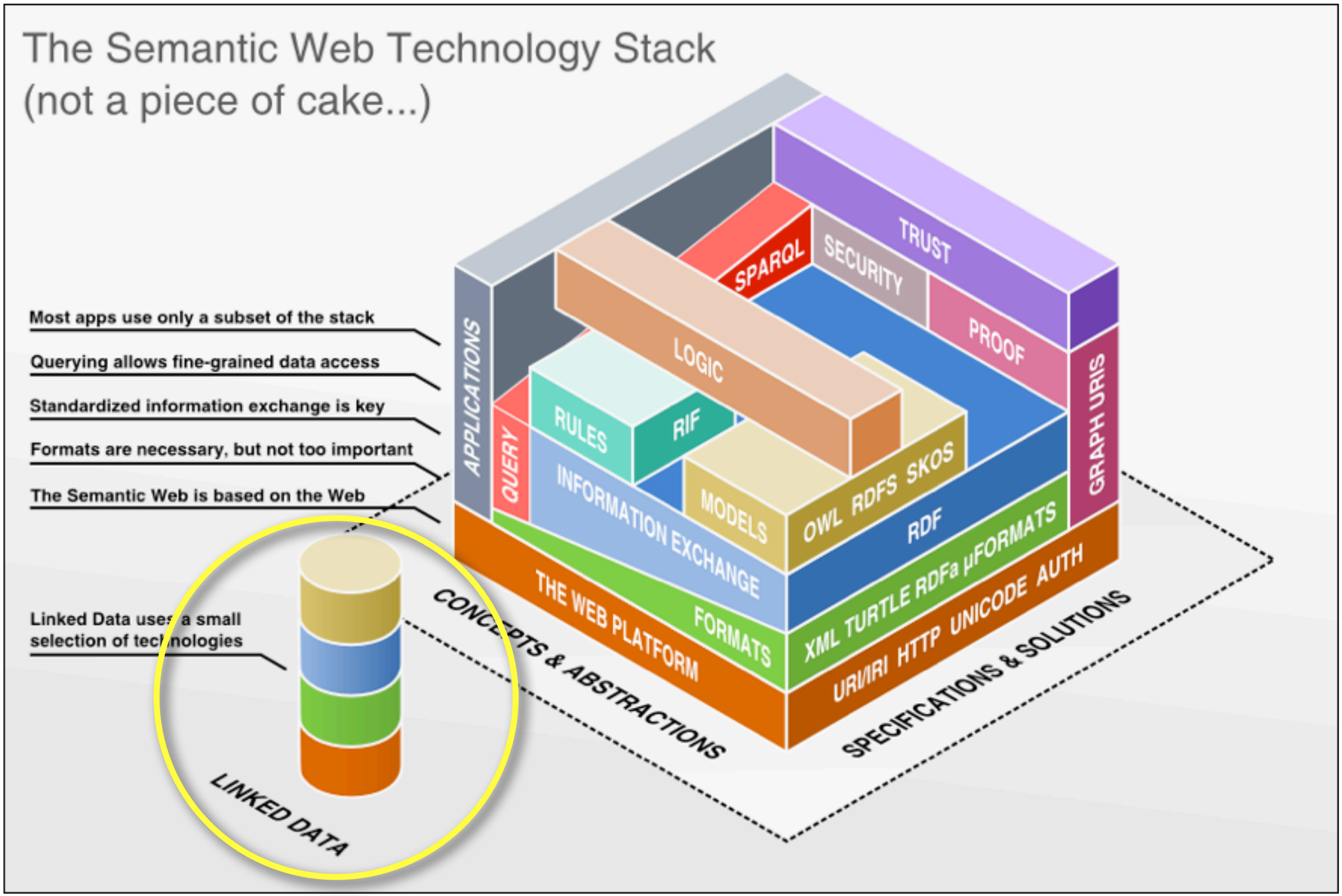
```
SELECT ?label
WHERE { { ?x dc:title ?label . }
  UNION { ?x rdfs:label ?label . }
}
```

SPARQL Abfrageformat

- neben der `SELECT`-Anfrage erlaubt SPARQL folgende Anfragen:
  - `ASK`: liefert boolesche Antwort („hat die Anfrage ein Ergebnis?“)
  - `CONSTRUCT`: liefert neue RDF-Triple zurück, gemäß vorgegebenem Triple-Pattern
  - `DESCRIBE`: liefert RDF zur Beschreibung der angefragten Ressourcen

2. Semantic Web Anwendungen

2.1 Linked Data



Linked Data and the ‘Web of Data‘

- Begriff geht auf Idee von Tim Berners-Lee zurück
  (Tim Berners-Lee, Linked Data, 2006, http://www.w3.org/DesignIssues/LinkedData.html)

- Menge von Best Practices zur Veröffentlichung und Verknüpfung von strukturierten Daten im Web
- Grundannahme: Der Wert (Nützlichkeit) von Daten im Web steigt je stärker diese mit Daten aus anderen Datenquellen verknüpft sind

Linked Data Principles

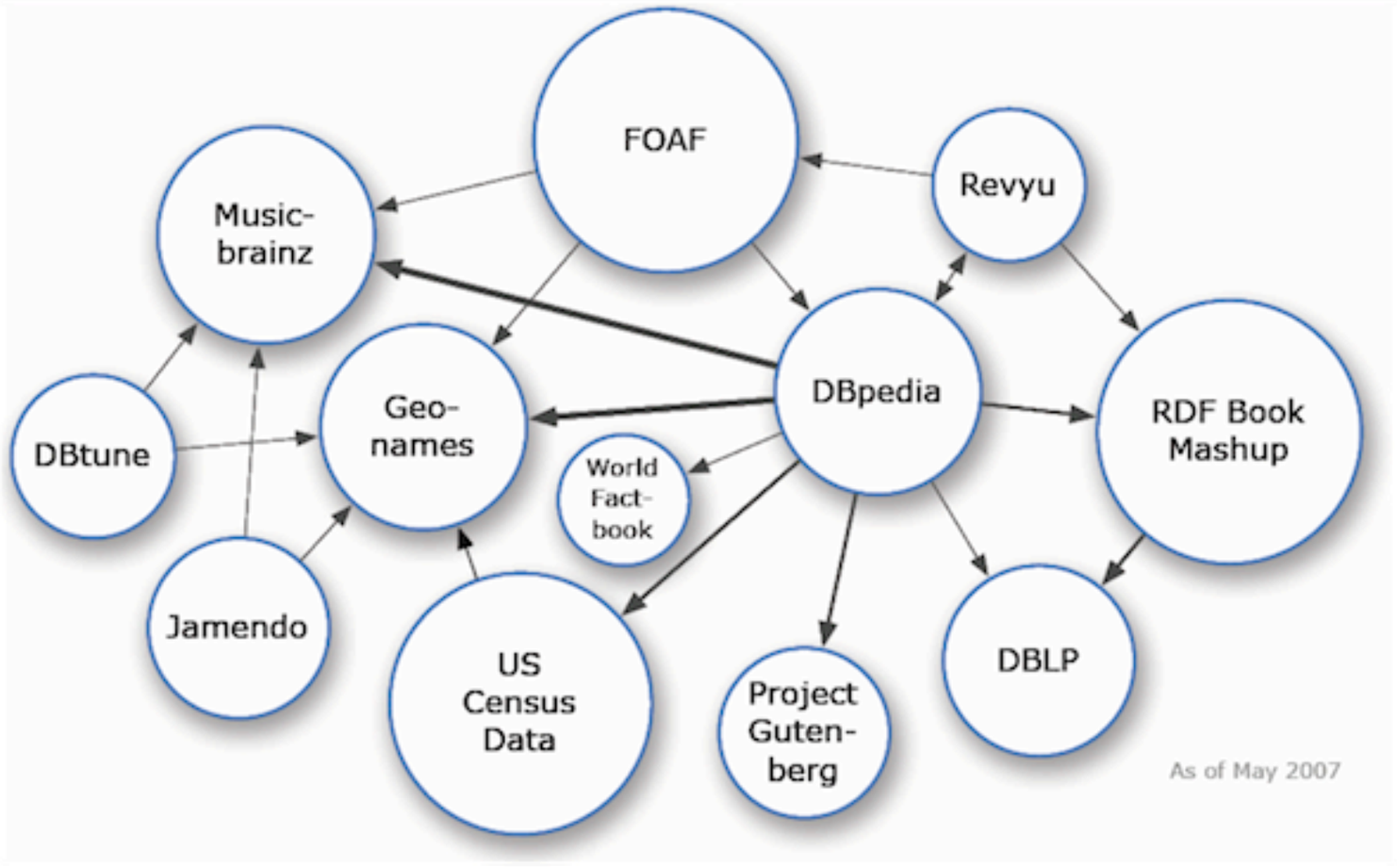
(1) Use **URIs** as names for things.

(2) Use **HTTP URIs**, so that people can **look up** those names.

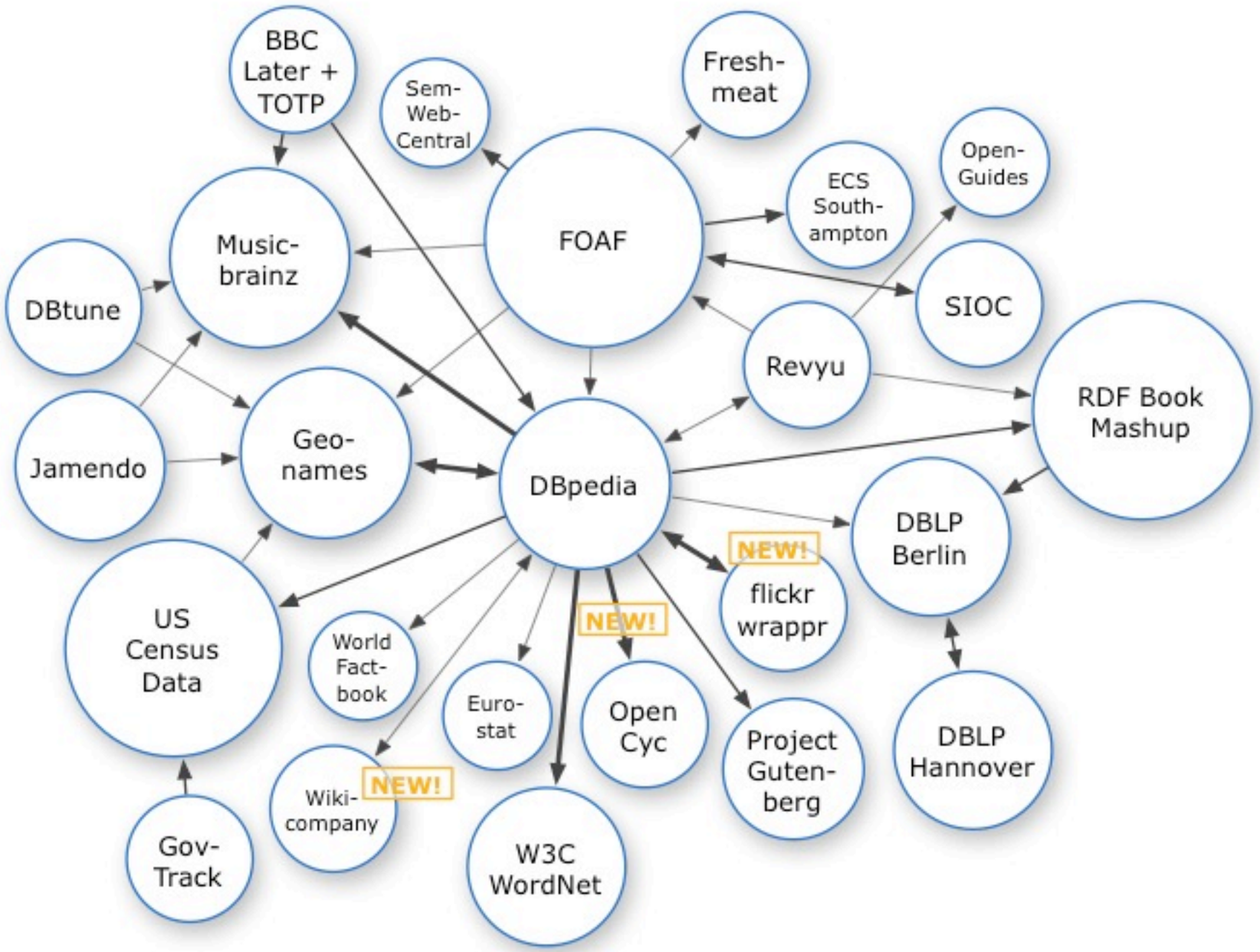
(3) When someone looks up a URI, provide **useful information**, using the **standards** (RDF, SPARQL).

(4) Include **links to other URIs**, so that they can discover more things.

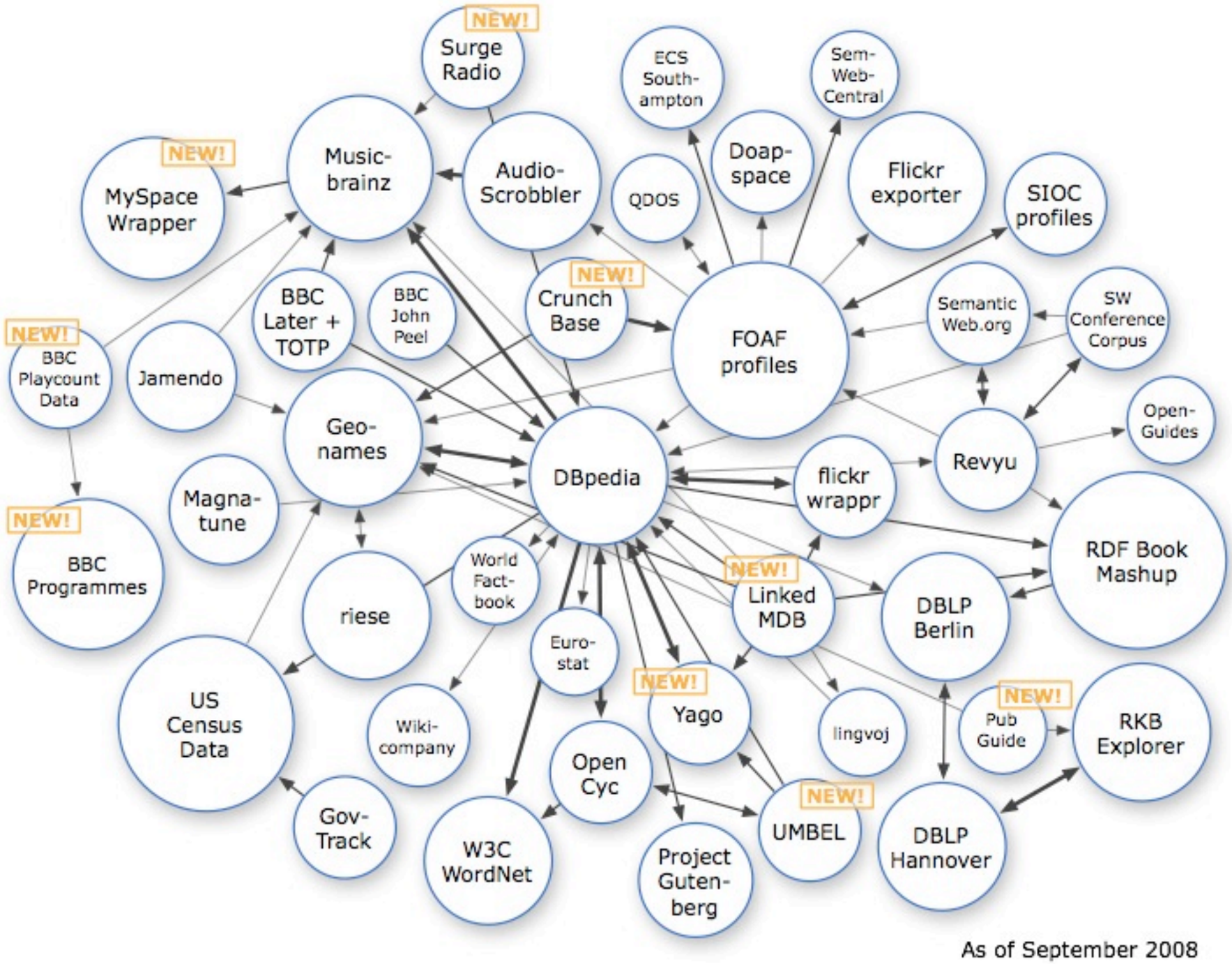
Linking Open Data Cloud



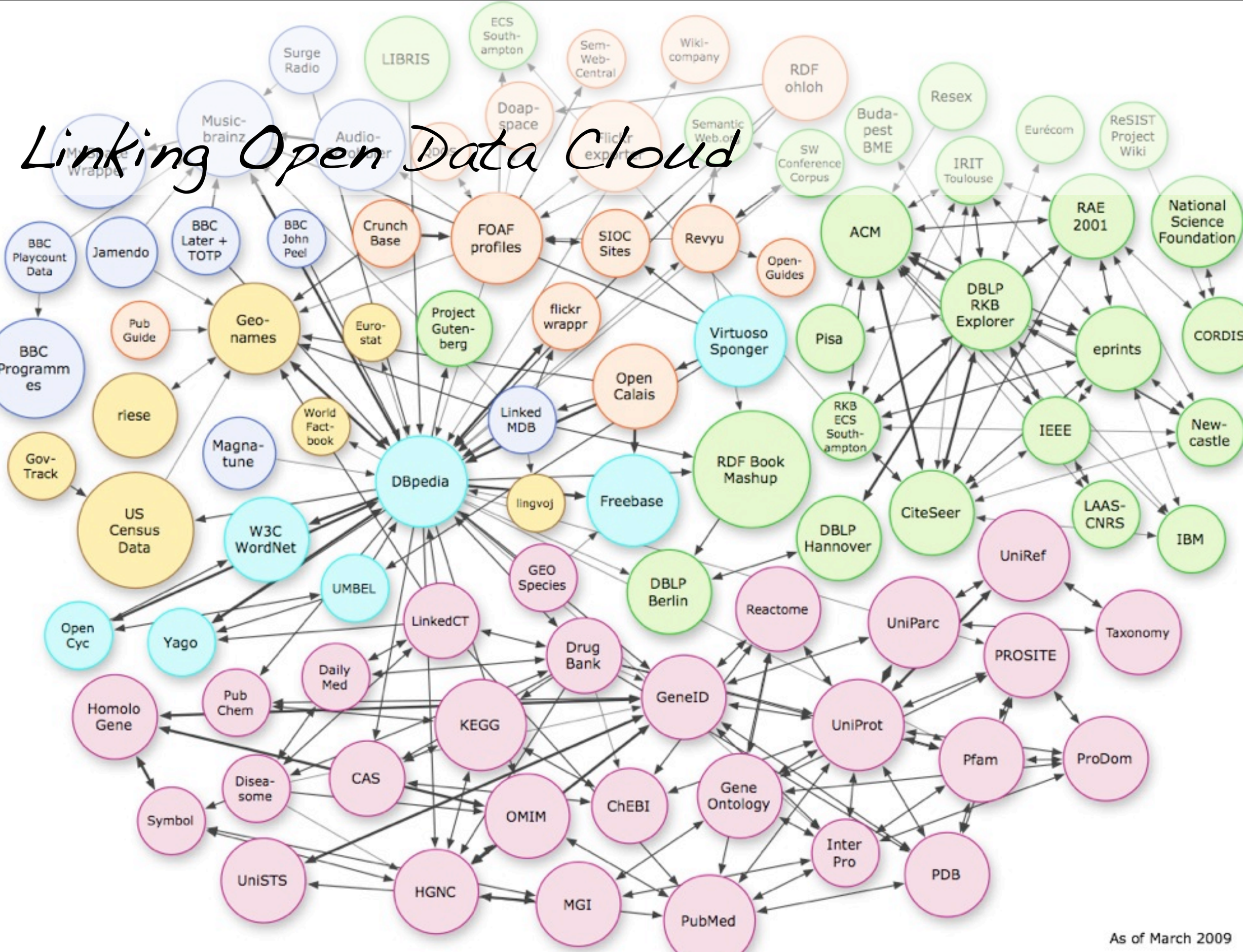
Linking Open Data Cloud



Linking Open Data Cloud



Linking Open Data Cloud

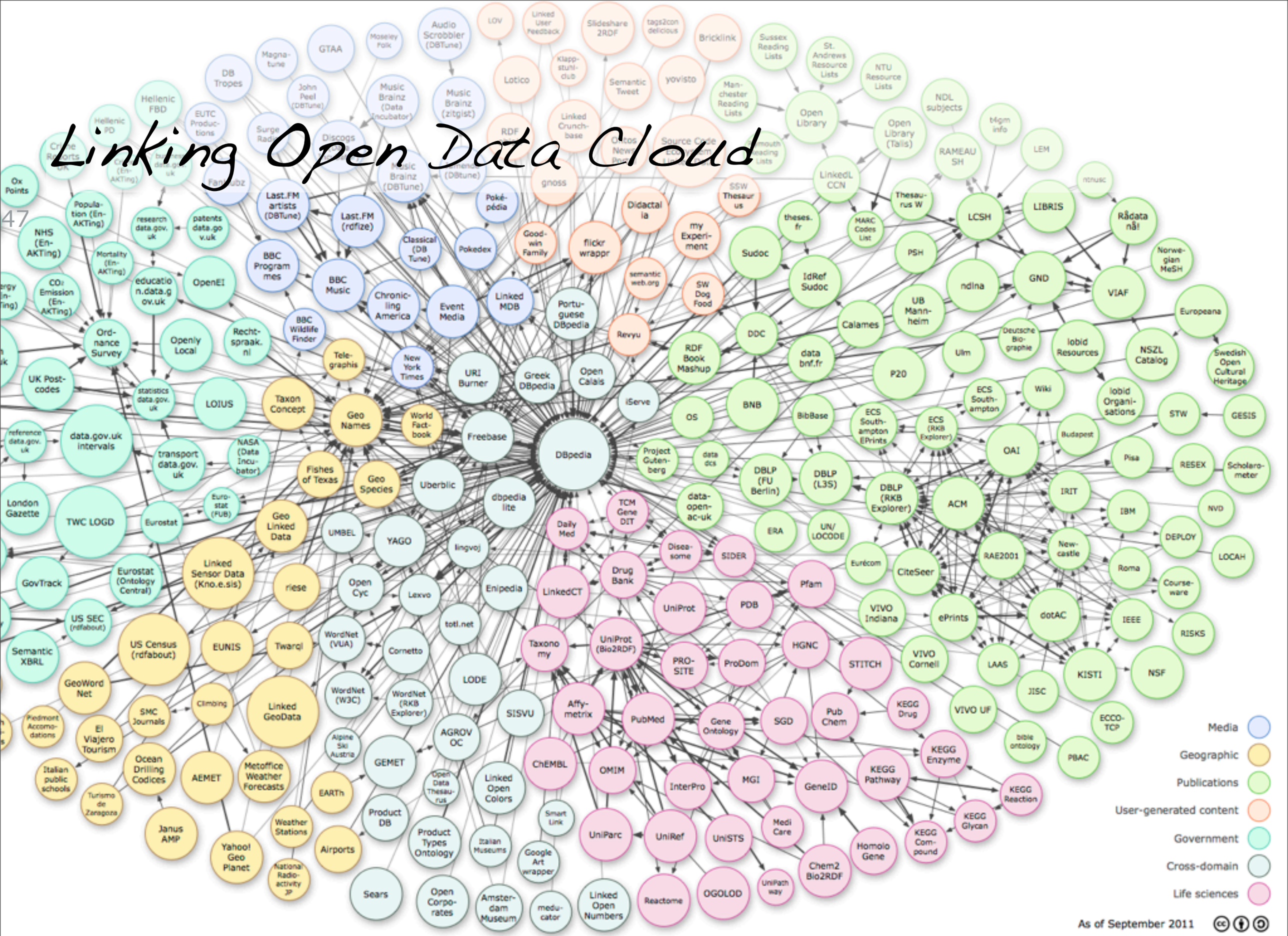


Linking Open Data Cloud

Media
Geographic
Publications
User-generated content
Government
Cross-domain
Life sciences

As of September 2010

Linking Open Data Cloud



SKOS - Simple Knowledge Organization System

- basiert auf RDF und RDFS und dient der Definition und zum Aufbau von Thesauri, sowie zum Mapping von Vokabularien und Ontologien s
  - `skos:Concept` (Klassen / Konzepte)
  - `skos:narrower`
  - `skos:broader`
  - `skos:related`
  - `skos:exactMatch, skos:narrowMatch, skos:broadMatch, skos:relatedMatch`

- notwendige Komponenten
  - lokaler RDF-Store
    - Caching von Ergebnissen
    - permanenter Speicher
  - Logik (Controller) und User Interface (-> Business Logic)
    - nicht Linked Data spezifisch
  - Daten-Integrationskomponente
    - Daten direkt aus LOD-Cloud oder
    - via Semantic Indexer (sindice, etc.) holen
  - Daten-Publishing-Komponente
    - Applikations-eigene Daten im Web of Data veröffentlichen

- Triple-Stores / SPARQL-Endpoints
  - OpenLink Virtuoso http://virtuoso.openlinksw.com/
  - Sesame http://www.openrdf.org/
  - OWLIM http://www.ontotext.com/owlim
  - Apache Jena Fuseki http://jena.apache.org/documentation/serving_data/
  - Stardog http://stardog.com/
  - 4store http://4store.org/
  - Mulgara http://www.mulgara.org/
  - viele RDBMS unterstützen RDF / SPARQL: Oracle, IBM DB2, ...
  - RDB2RDF Mappings für Legacy Daten
    - Triplify http://triplify.org/Overview
    - D2R Server http://d2rq.org/d2r-server
  - ...

- Dinge gehen einfacher mit einer entsprechenden Library:
  - ARC for SPARQL (PHP) http://arc.semsol.org/
  - RAP - RDF API für PHP http://www4.wiwiss.fu-berlin.de/bizer/rdfapi/index.html
  - Jena/ARQ (Java) http://jena.sourceforge.net/
  - Sesame (Java) http://www.openrdf.org/
  - SPARQL Wrapper (Python) http://sparql-wrapper.sourceforge.net/
  - Redland librdf (C / Perl, PHP, Python, Ruby bindings) http://librdf.org/
  - SPARQL Javascript Library http://www.thefigtrees.net/lee/blog/2006/04/sparql_calendar_demo_a_sparql.html
  - ...

Vielen Dank!

Fragen?