Tutorial:

Einführung in Daten-Web-Technologien

Dr. Jens Lehmann, Dr. Sören Auer

AKSW, Universität Leipzig

Agenda

Teil 1: Einführung in Daten-Web-Technologien (ab 13 Uhr)

13.00 - 13.45 Uhr Semantic Web Vision, Grundlagen RDF, RDF-Schema

13:45 - 14:45 OWL + Demo Protégé, SPARQL + Triple Stores

14:45 - 15.30 Daten-Web-Beispiele: DBpedia und LinkedGeoData

(15 Minuten Kaffeepause)

Teil 2: GoodRelations (ab 15:45 Uhr bis 18 Uhr)

Agenda

Teil 1: Einführung in Daten-Web-Technologien (ab 13 Uhr)

13.00 - 13.45 Uhr Semantic Web Vision, Grundlagen RDF, RDF-Schema

13:45 - 14:45 OWL + Demo Protégé, SPARQL + Triple Stores

14:45 - 15.30 Daten-Web-Beispiele: DBpedia und LinkedGeoData

(15 Minuten Kaffeepause)

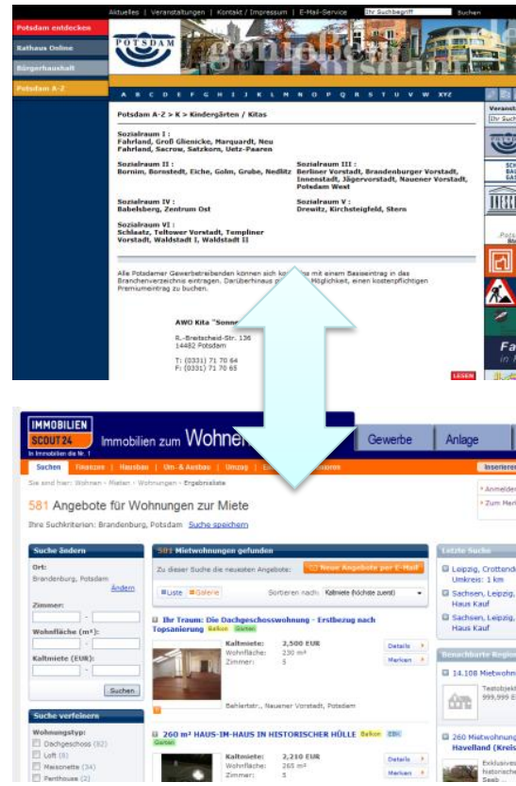
Teil 2: GoodRelations (ab 15:45 Uhr bis 18 Uhr)

Warum Semantic Web?

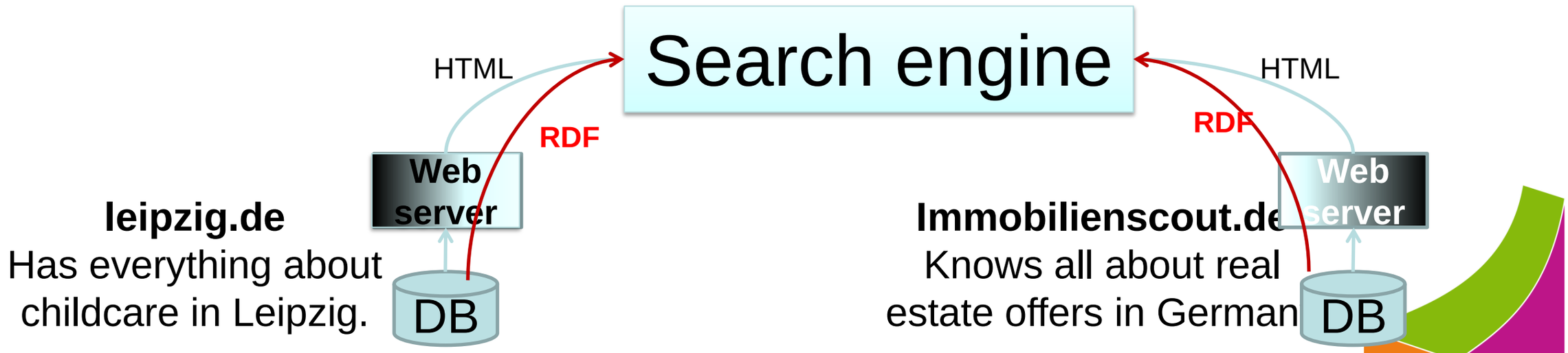
Problem: Try to search for these things on the current Web:

- Apartments near German-Russian bilingual childcare in Leipzig.
- ERP service providers with offices in Vienna and London.
- Researchers working on multimedia topics in Eastern Europe.

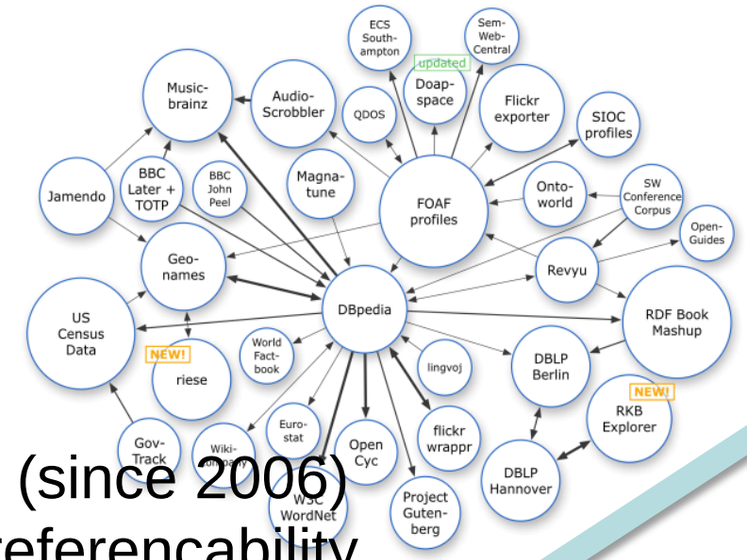
Information is available on the Web, **but opaque to current Web search**.



Solution: complement text on Web pages with structured linked open data & intelligently combine/integrate such structured information from different sources:



Vom Web der Dokumente zum ***Semantic Data Web***



Data Web (since 2006)
- URI de-referencability
- Web Data integration
- RDF serializations

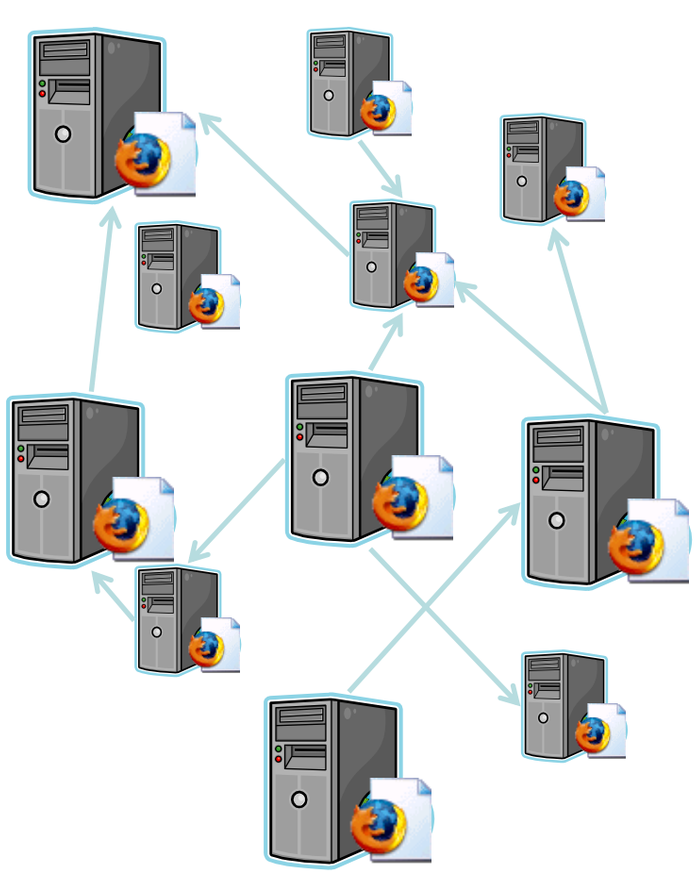
Semantic Web
(Vision 1998, starting ???)
- Reasoning
- Logic, Rules
- Trust

Social Web (since 2003)
- Folksonomies/Tagging
- Reputation, sharing
- Groups, relationships

Web (since 1992)
- HTTP
- HTML/CSS/JavaScript

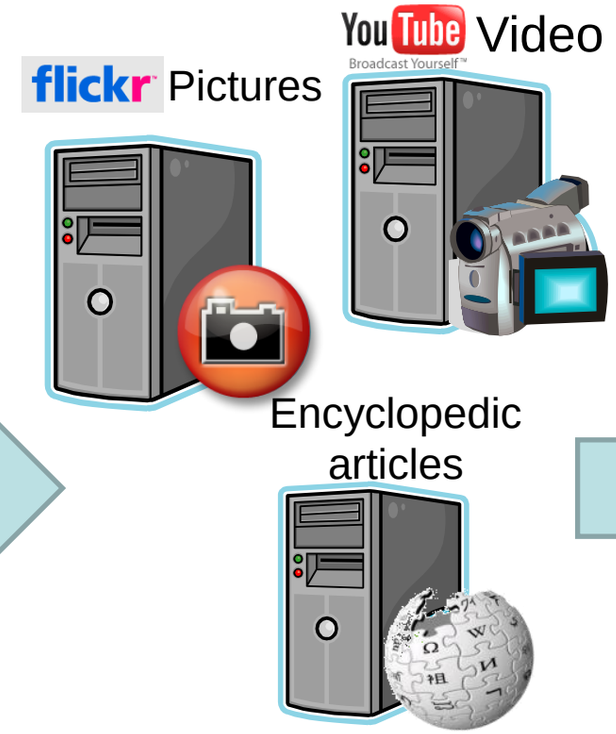
Web 1.0

Many Web sites containing unstructured, textual content



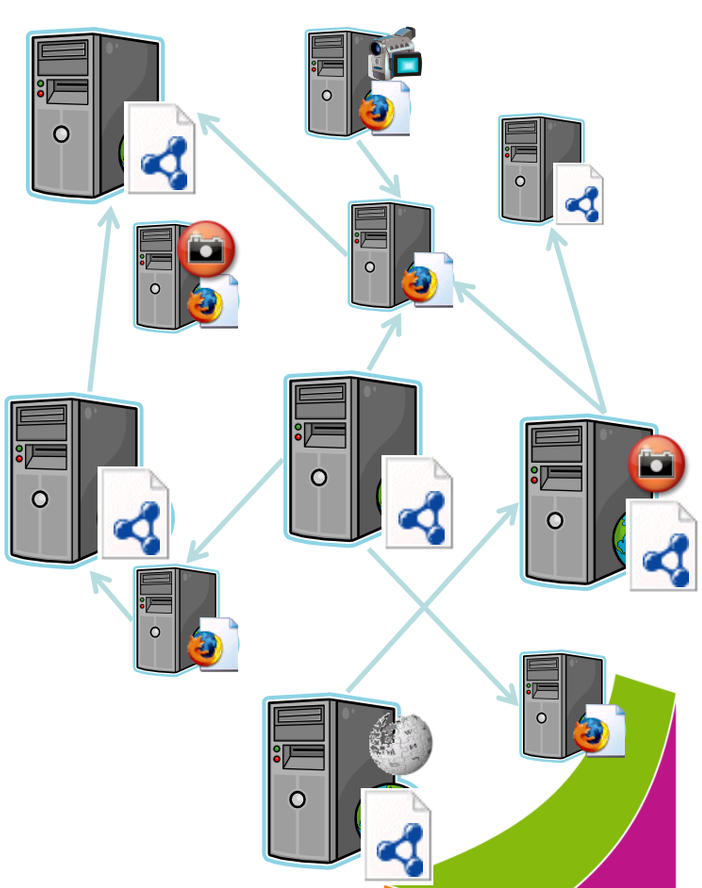
Web 2.0

Few large Web sites are specialized on specific content types

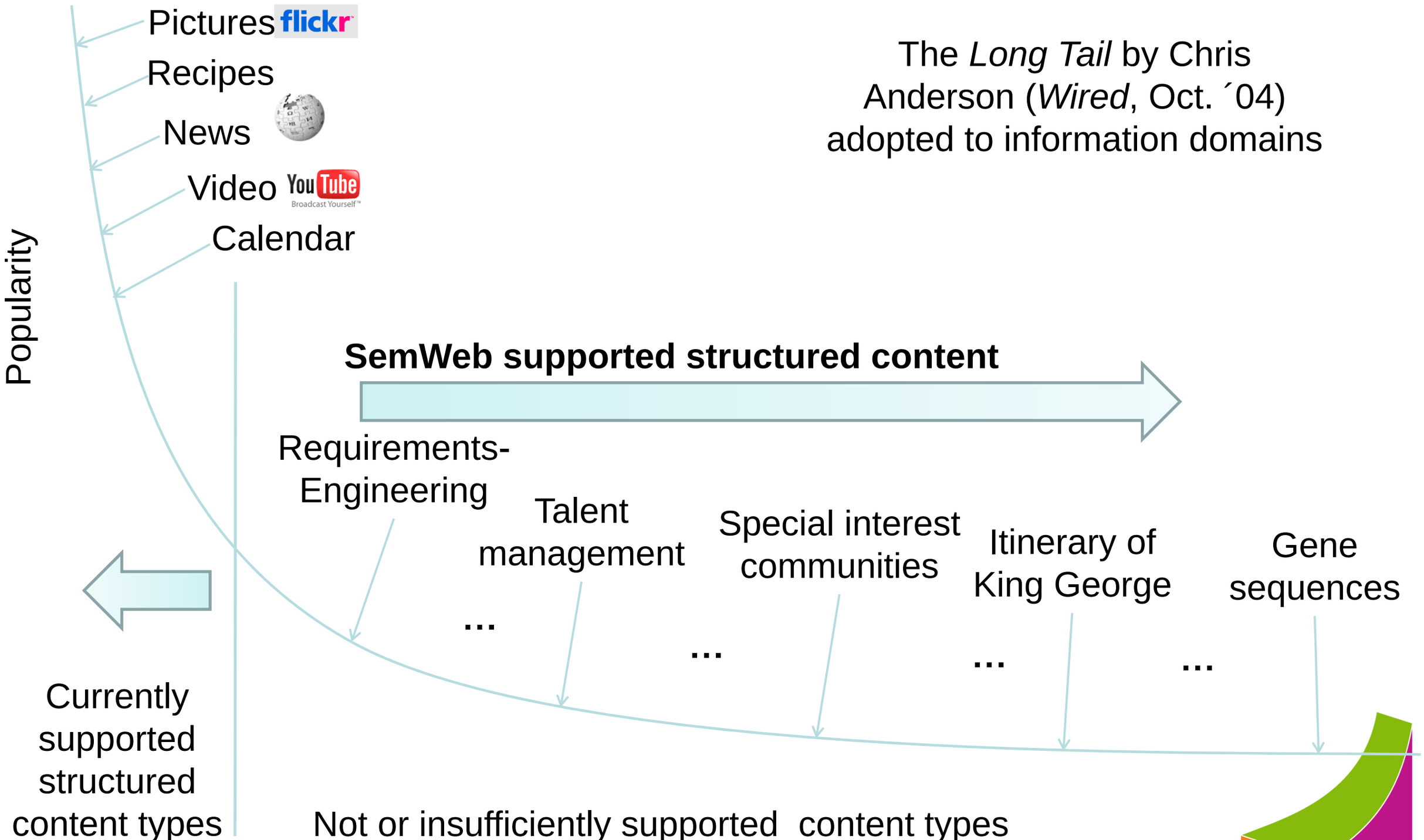


Web 3.0

Many Web sites containing & semantically syndicating arbitrarily structured content

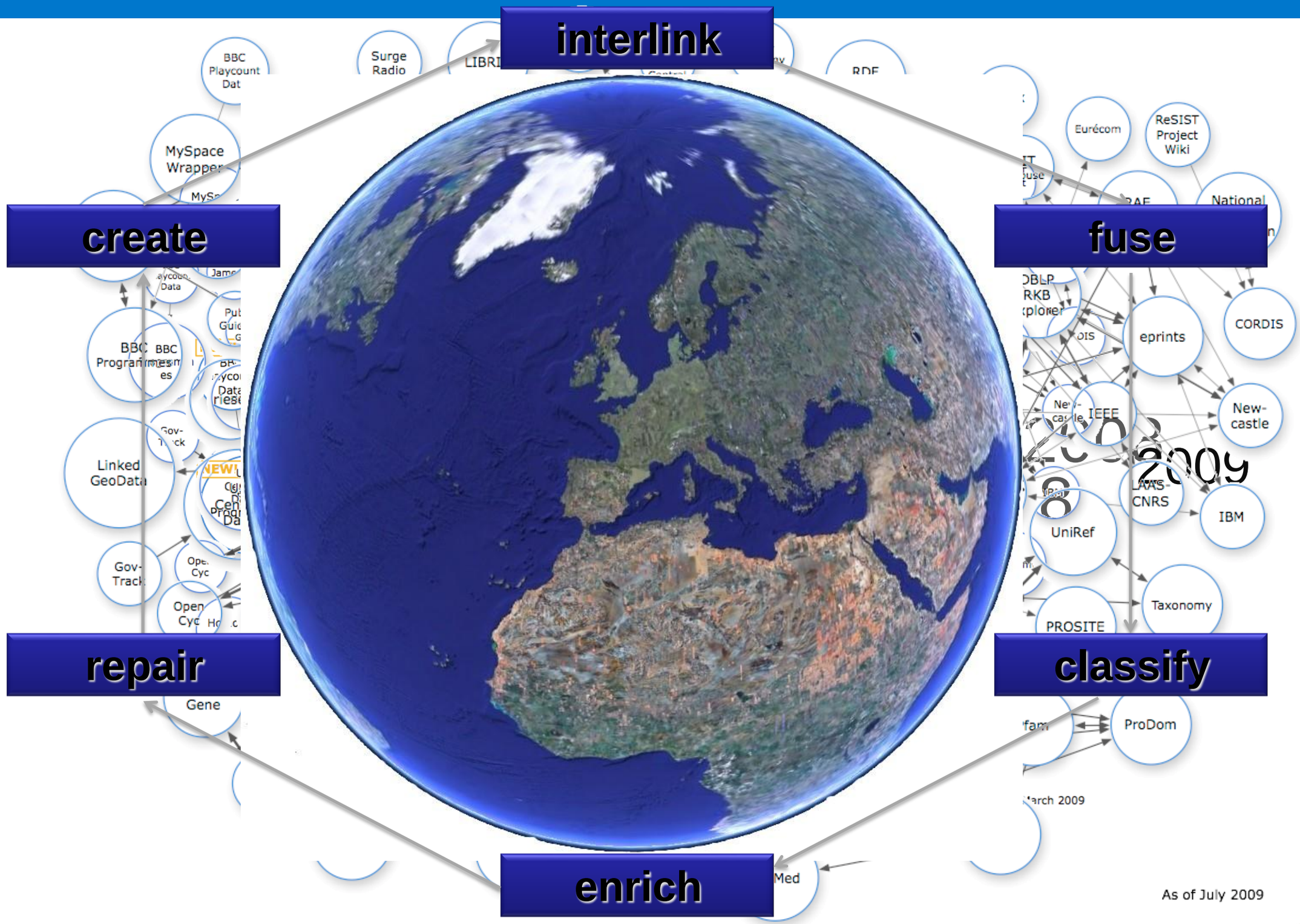


The Long Tail of Information Domains



The *Long Tail* by Chris Anderson (*Wired*, Oct. ´04) adopted to information domains

Die Vision: ein Web Vernetzter Daten



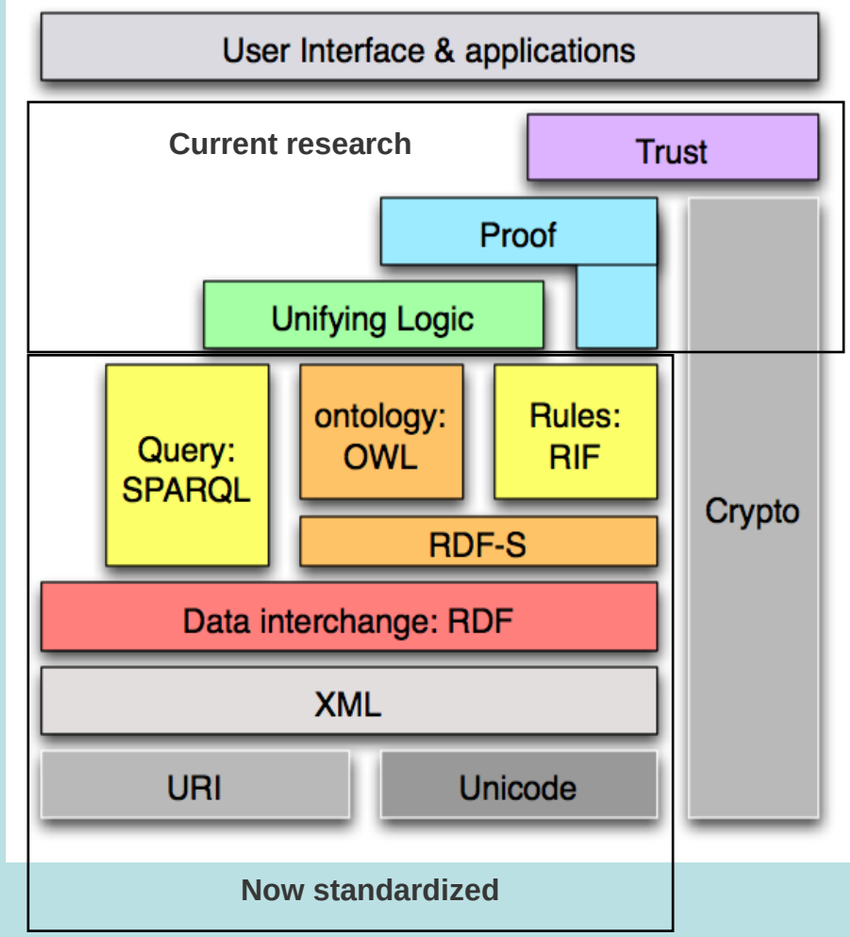
2009

Conceptual Level
Data Access and Integration

Data Integration	**Enterprise Information Integration** sets of heterogeneous data sources appear as a single, homogeneous data source	**Data Warehousing** • Based on extract, transform load (ETL) • Global-As-View (GAV)	**Research** Mediators Ontology-based P2P Web service-based	**Data Web** • URIs as entity identifiers • HTTP as data access protoc • Local-As-View (LAV)
Data Access	**Object-relational mappings (ORM)** • NeXT's EOF / WebObjects • ADO.NET Entity Framework • Hibernate	**Procedural APIs** • ODBC • JDBC	**Linked Data** • de-referencable URIs • RDF serialization formats	**Query Languages** • Datalog, SQL • **SPARQL** • XPATH/XQuery
Data Models	**RDBMS** • Organize data in relations, rows, cells • Oracle, DB2, MS-SQL	**Column-oriented DBMS** • Collocates column values rather than row values • Vertica, C-Store, MonetDB	**Triple/Quad Stores** • RDF data model • Virtuoso, Oracle, Sesame **Entity-attribute-value (EAV)** • HELP medical record system, TrialDB	**Others** • XML, hierachical, tree, graph-oriented DBMS

Semantic Web - Standards

Semantic Web Architecture



Standardization Semantic Web

1994	• First public presentation of the Semantic Web idea
1998	• Start of standardization of data model (RDF) and a first ontology languages (RDFS) at W3C
2000	• Start of large research projects about ontologies in the US and Europe (DAML & Ontoknowledge)
2002	• Start of standardization of a new ontology language (OWL) based on research results
2004	• Finalization of the standard for data (RDF) and ontology (OWL)
2006	• Standardization of a quer y language (SPARQL, 6. April 2006) • Ongoing work on rule languages (SWRL, DL-safe rules, RIF) • Extension of OWL to OWL 1.1 / 2.0 • Ontology language of OMG based on UML (ODM)
2008	•RDFa
2009	•OWL2

W3C Semantic Web

Agenda

Teil 1: Einführung in Daten-Web-Technologien (ab 13 Uhr)

13.00 - 13.45 Uhr Semantic Web Vision, Grundlagen RDF, RDF-Schema

13:45 - 14:45 OWL + Demo Protégé, SPARQL + Triple Stores

14:45 - 15.30 Daten-Web-Beispiele: DBpedia und LinkedGeoData

(15 Minuten Kaffeepause)

Teil 2: GoodRelations (ab 15:45 Uhr bis 18 Uhr)

URIs - Idee

- **URI** = Uniform Resource Identifier
- dienen zur weltweit eindeutigen Bezeichnung von Ressourcen
- Ressource kann jedes Objekt sein, was (im Kontext der gegebenen Anwendung) eine klare Identität besitzt (z.B. Bücher, Orte, Menschen, Verlage, Beziehungen zwischen diesen Dingen, abstrakte Konzepte usw.)
- in bestimmten Domänen ähnliches bereits realisiert: ISBN für Bücher

http://aksw.org/Events/2011/LeipzigerSemanticWebDay

URIs - Syntax

- Erweiterung des URL-Konzeptes; nicht jede URI bezeichnet aber ein Webdokument (umgekehrt wird als URI für Webdokumente häufig deren URL verwendet)
- Beginnt mit dem sogenannten URI-Schema das durch ":" vom nachfolgenden Teil getrennt ist (z.B.: http, ftp, mailto)
- häufig hierarchisch aufgebaut

`http://aksw.org/Events/2011/LeipzigerSemanticWebDay`

Selbstdefinierte URIs

- nötig, wenn für eine Ressource (noch) keine URI existiert (bzw. bekannt ist)
- Strategie zur Vermeidung von (ungewollten) Überschneidungen: Nutzung von http-URIs einer eigenen Domäne/Webseite
- ermöglicht auch Ablegen einer Dokumentation zur URI an dieser Stelle

Beschreibendes vs. Beschriebenes

- Trennung von URI für Ressource und deren Dokumentation durch URI-Referenzen (durch "#" angehängte Fragmente) oder content negotiation
- z.B.: als URI für Shakespeares "Othello"
  http://de.wikipedia.org/wiki/Othello
  nicht geeignet, besser
  http://de.wikipedia.org/wiki/Othello#URI

Bestandteile von RDF-Graphen

URIs

- zur eindeutigen Referenzierung von Ressourcen
- bereits im Rahmen von XML behandelt

Literale

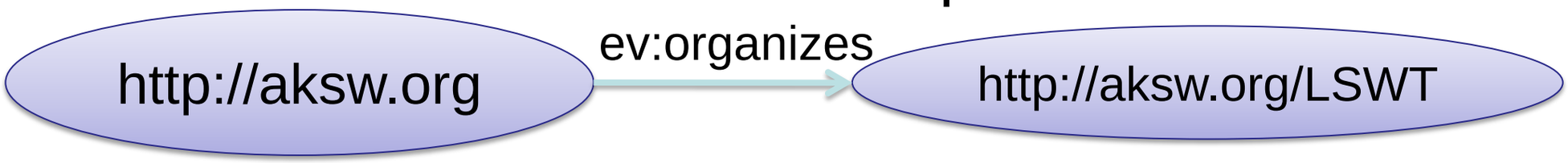
- beschreiben Datenwerte denen keine separate Existenz zukommt

leere Knoten (blank nodes)

- erlauben Existenzaussagen über ein Individuum mit gewissen Eigenschaften ohne es zu benennen

RDF-Tripel

- Bestandteile eines RDF-Tripels

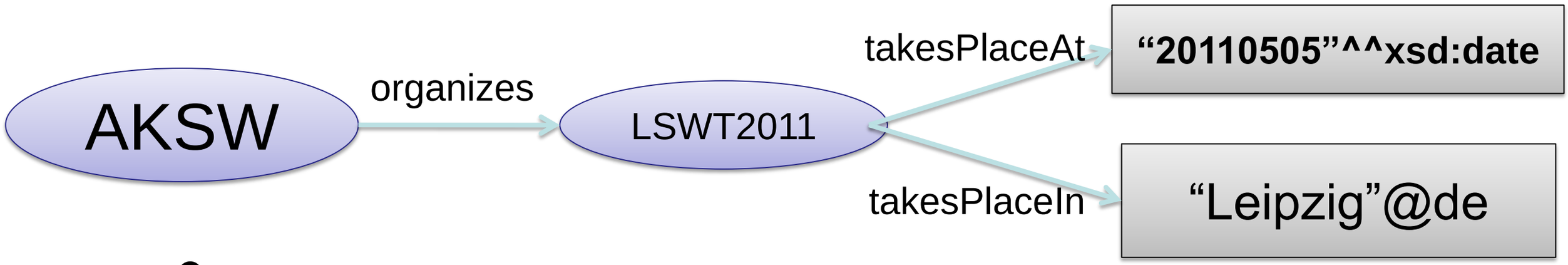


- angelehnt an linguistische Kategorien, aber nicht immer stimmig

- erlaubte Belegungen:
  Subjekt : URI oder leerer Knoten
  Prädikat: URI (auch Propertys genannt)
  Objekt : URI oder leerer Knoten oder Literal

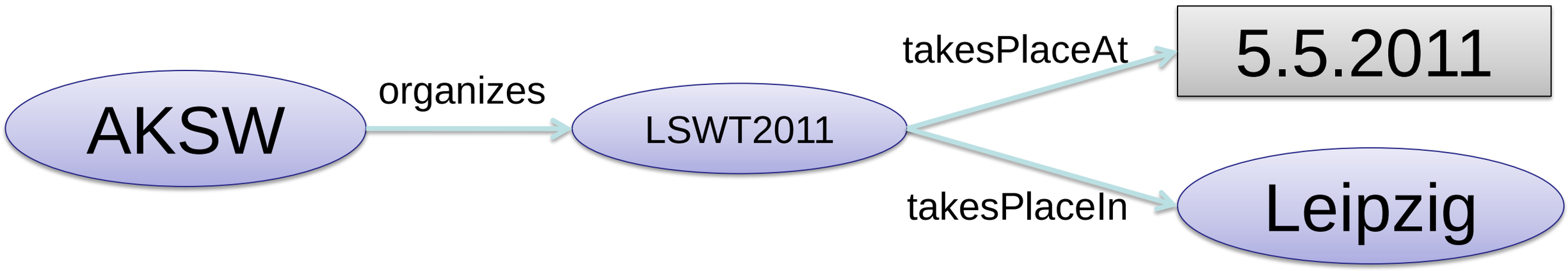
- Knoten- und Kantenbezeichner eindeutig, daher ursprünglicher Graph aus Tripel-Liste rekonstruierbar

Literale



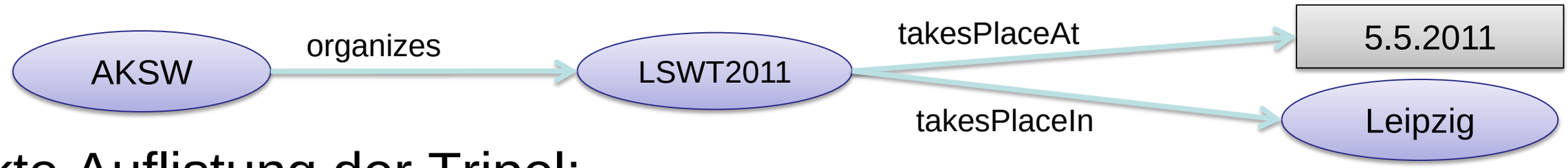
- zur Repräsentation von Datenwerten
- Darstellung als Zeichenketten
- Interpretation erfolgt durch *Datentyp*
- Literale ohne Datentyp
  - werden wie Zeichenketten behandelt
  - können optional mit Sprachtag versehen werden

Graph als Menge von Tripeln



- verschiedene Darstellungsmöglichkeiten für Graphen
- hier verwendet: Liste von (Knoten-Kante-Knoten)-Tripeln

Turtle: Einfache Syntax für RDF



Direkte Auflistung der Tripel:

```
<http://aksw.org>      <http://aksw.org/event/organizes>     <http://aksw.org/LSWT> .
<http://aksw.org/LSWT> <http://aksw.org/event/takesPlaceAt>  "20110505"^^xsd:date .
<http://aksw.org/LSWT> <http://aksw.org/event/takesPlaceIn>  <http://dbpedia.org/Leipzig> .
```

Syntax in Turtle:

- URIs in spitzen Klammern
- Literale in Anführungszeichen
- Tripel durch Punkt abgeschlossen
- Leerzeichen und Zeilenumbrüche außerhalb von Bezeichenern werden ignoriert

```
@prefix ev:  <http://aksw.org/event/>
@prefix dbp: <http://dbpedia.org/resource/>
<http://aksw.org>      ev:organizes      <http://aksw.org/LSWT> .
<http://aksw.org/LSWT> ev:takesPlaceAt   "20110505"^^xsd:date .
<http://aksw.org/LSWT> ev:takesPlaceIn   dbp:Leipzig .
```

Klassen und Instanzen

```
ex:LSWT2011 rdf:type  ex:Veranstaltung .
```

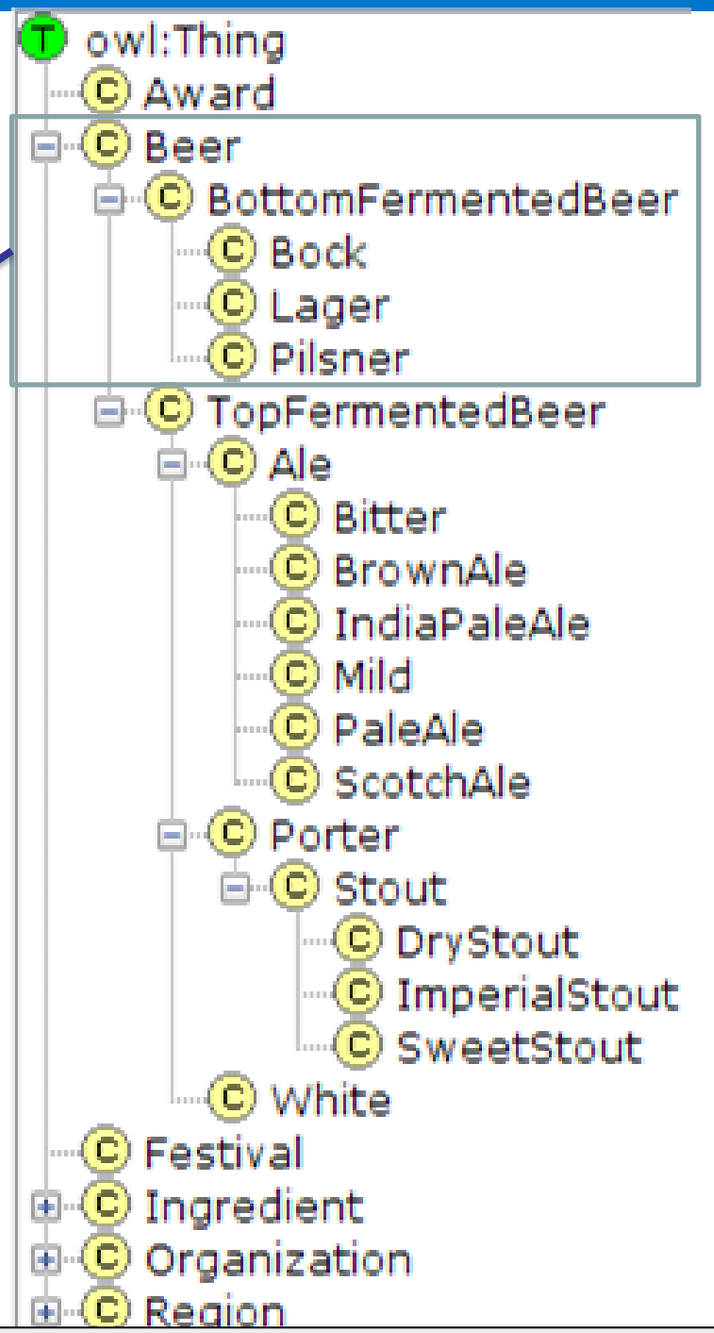
- charakterisiert „LSWT2010“ als Instanz der (neu definierten) Klasse „Veranstaltung“
- Klassenzugehörigkeit ist nicht exklusiv, z.B. mit o.g. Tripel gleichzeitig möglich:

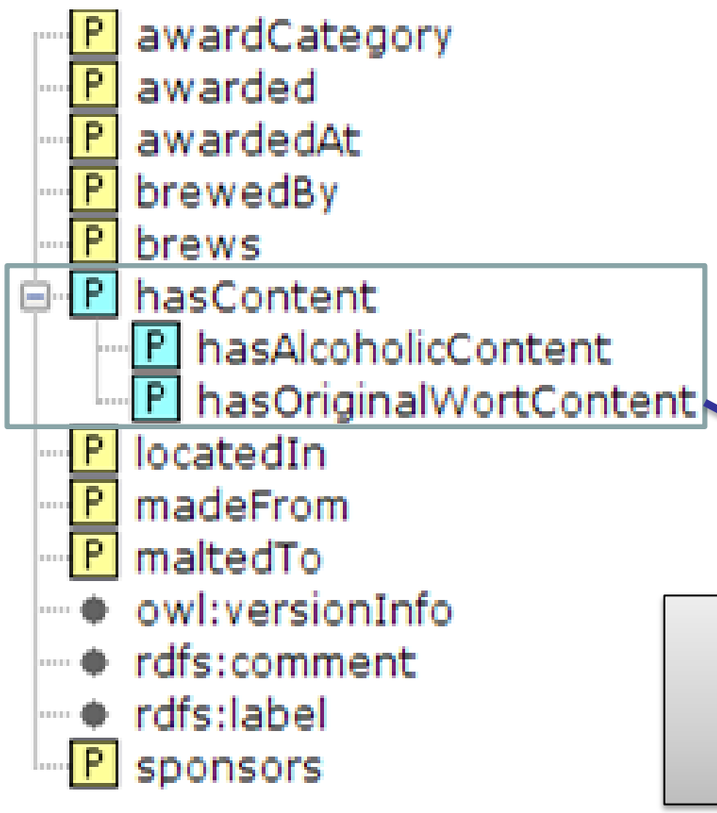
```
ex:LSWT2011  rdf:type  ex:Unterhaltsam .
```

- allgemein: a priori syntaktisch keine eindeutige Unterscheidung zwischen Individuen- und Klassenbezeichnern möglich
- auch in der Realität Charakterisierung manchmal schwierig, beispielsweise für http://www.un.org/#URI

RDF Vokabulare:
Klassen & Eigenschaften Hierarchien

Beer	rdf:type	rdfs:Class
BottomFermentedBeer	rdfs:subClassOf	Beer
Bock	rdfs:subClassOf	BottomFermentedBeer
Lager	rdfs:subClassOf	BottomFermentedBeer
Pilsner	rdfs:subClassOf	BottomFermentedBeer

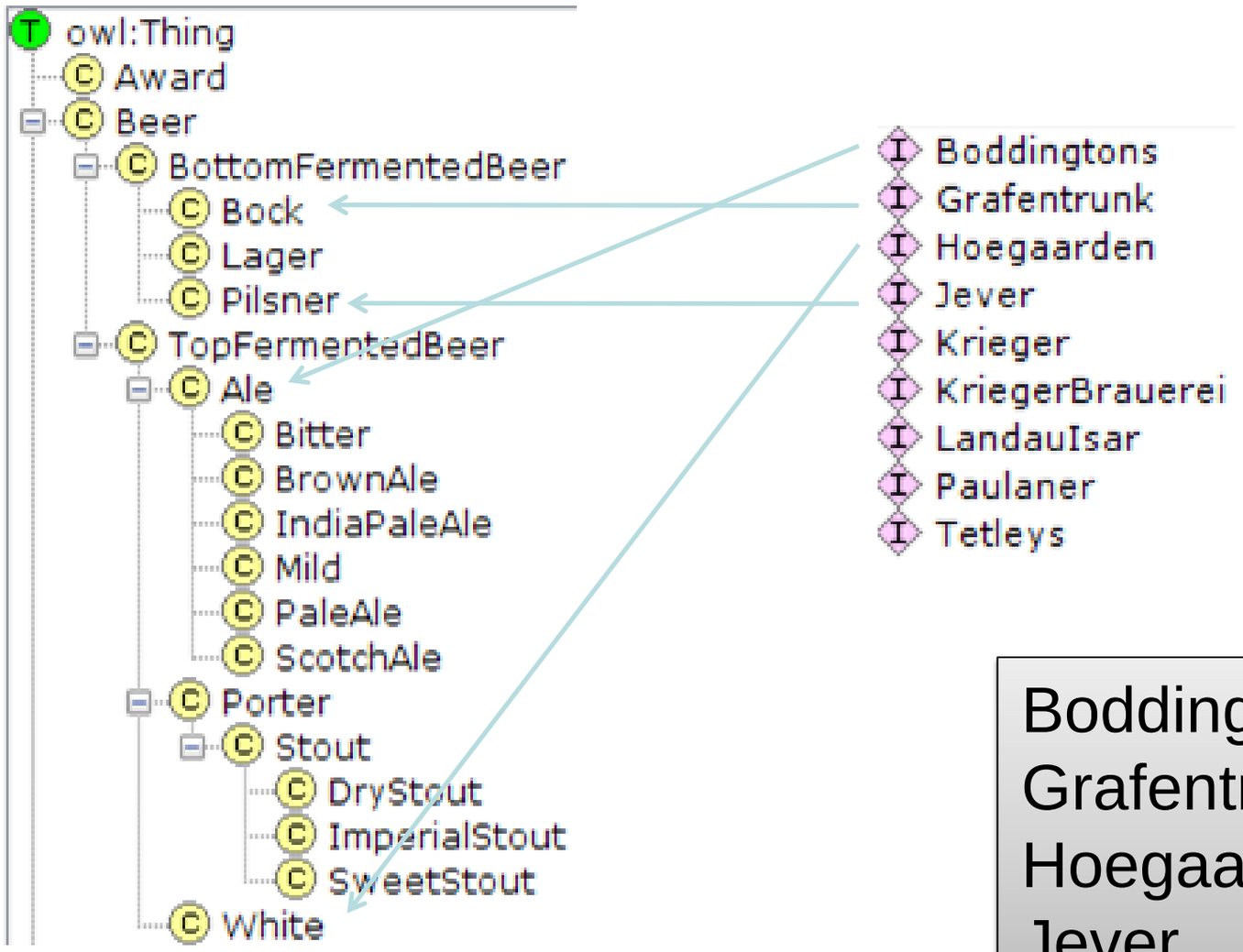




hasContent	rdf:type	rdfs:Property
hasAlcoholicContent	rdfs:subPropertyOf	hasContent
hasOriginalWortContent	rdfs:subPropertyOf	hasContent

RDF-S Instanzen

- Instanzen sind einer oder mehreren Klassen zugeordnet:



Boddingtons	rdf:type	Ale
Grafentrunk	rdf:type	Bock
Hoegaarden	rdf:type	White
Jever	rdf:type	Pilsner

Propertys

- andere Bezeichnungen: Relationen, Beziehungen
- Achtung: Propertys sind in RDF(S) nicht (wie in OOP) speziellen Klassen zugeodnet
- Property-Bezeichner in Tripeln üblicherweise an Prädikatsstelle
- charakterisieren, auf welche Art zwei Ressourcen zueinander in Beziehung stehen
- mathematisch oft dargestellt als Menge von Paaren: verheiratet_mit = {(Adam,Eva),(Brad,Angelina),...}
- URI wird als Property-Bezeichner gekennzeichnet durch entsprechende Typung:

```
ex:hasContent   rdf:type   rdf:Property   .
```

Unterpropertys

- ähnlich zu Unter-/Oberklassen auch Unter-/Oberpropertys denkbar und sinnvoll
- Darstellung in RDFS mittels rdfs:subPropertyOf z.B.:

```
ex:glücklichVerheiratetMit  rdf:subPropertyOf rdf:verheiratetMit .
```

- erlaubt, aus dem Tripel

  ```
  ex:Markus   ex:glücklichVerheiratetMit   ex:Anja   .
  ```

  zu schlussfolgern, dass

  ```
  ex:Markus   ex:verheiratetMit   ex:Anja   .
  ```

Domain und Range

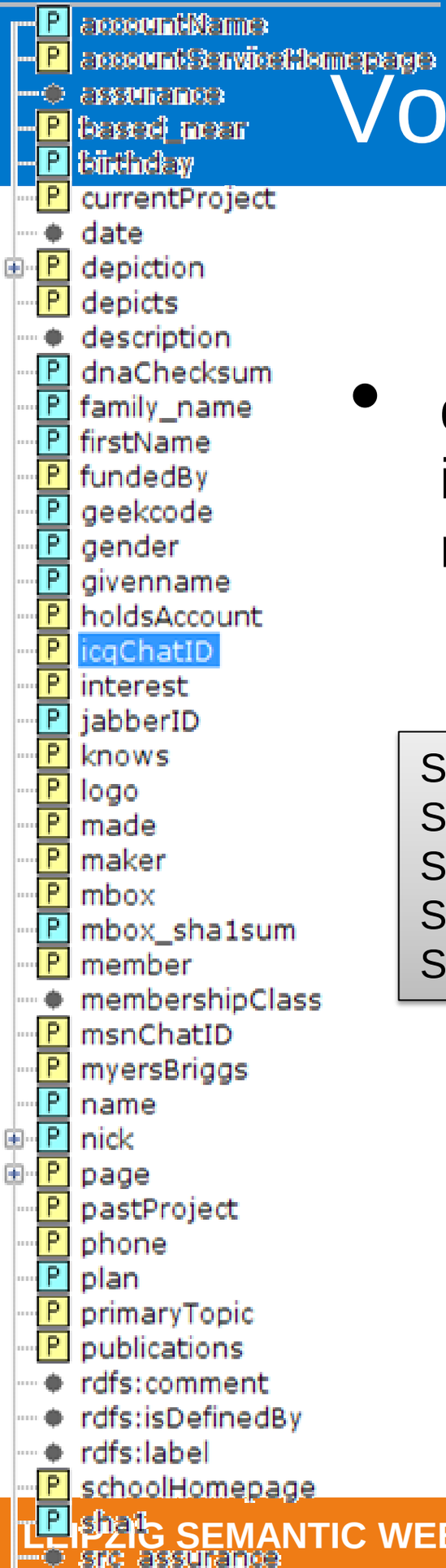
- Properties werden unabhängig von Klassen definiert

hasAlcoholicContent	rdfs:domain	Beer
hasAlcoholicContent	rdfs:range	xsd:float
hasAlcoholicContent	rdfs:subPropertyOf	hasContent
brews	rdf:type	owl:ObjectProperty
brews	rdfs:domain	Brewery
brews	rdfs:range	Beer

- **Domain:** Verknüpfung mit einer oder mehreren Klassen

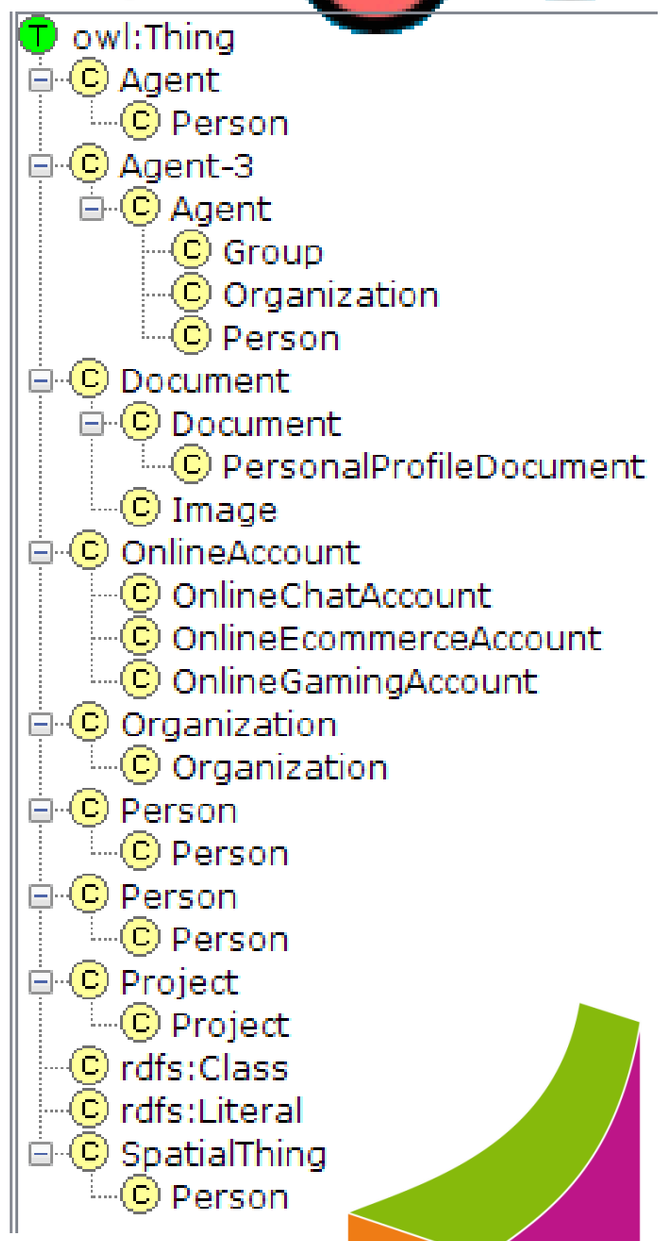
- **Range:** definiert Werte, welche die Property annehmen kann
  - Instanzen einer bestimmten Klasse
  - Literale, z.B. typisiert nach XML-Schema-Datentypen

Vokabulare: Friend-of-a-Friend (FOAF)



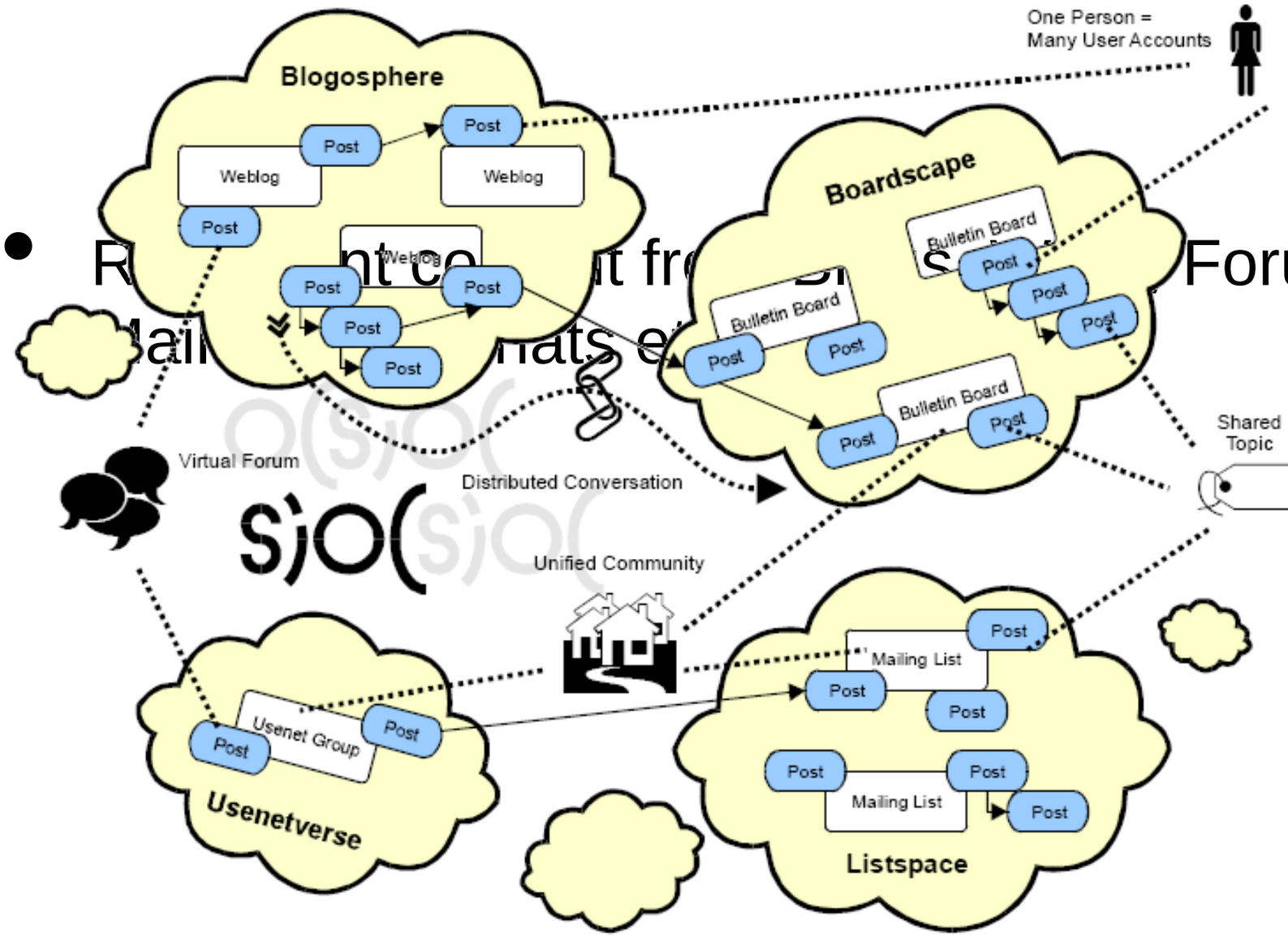
- defines classes and properties for representing information about people and their relationships

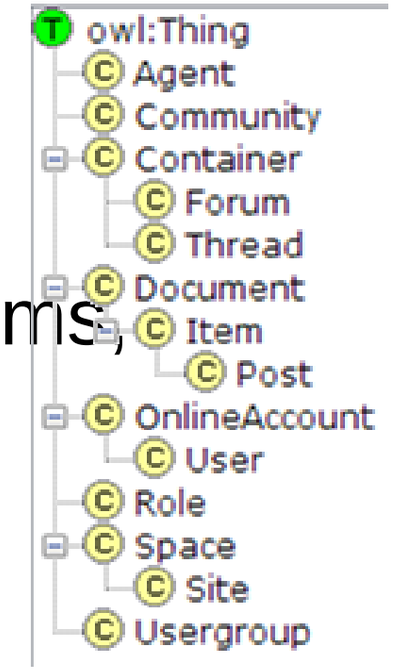
Soeren	rdf:type	foaf:Person
Soeren	currentProject	http://OntoWiki.net
Soeren	foaf:homepage	http://aksw.org/Soeren
Soeren	foaf:knows	http://sembase.at/Tassilo
Soeren	foaf:mbox_sha1sum	09ac456515dee



Vokabulare: Semantically Interlinked Online Communities.

- about:
- account_of
- administrator_of
- attachment:
- avatar
- container_of
- content
- content_encoded
- created_at
- creator_of
- description
- description
- email
- email_sha1
- feed
- first_name
- function_of
- group_of
- has_administrator
- has_container
- has_creator
- has_function
- has_group
- has_host
- has_member
- has_moderator
- has_modifier
- has_owner
- has_parent
- has_part
- has_reply
- has_scope
- has_space
- has_subscriber
- has_usergroup
- host_of
- id
- ip_address
- last_name
- link
- links_to
- member_of
- moderator_of
- modified_at
- modifier_of

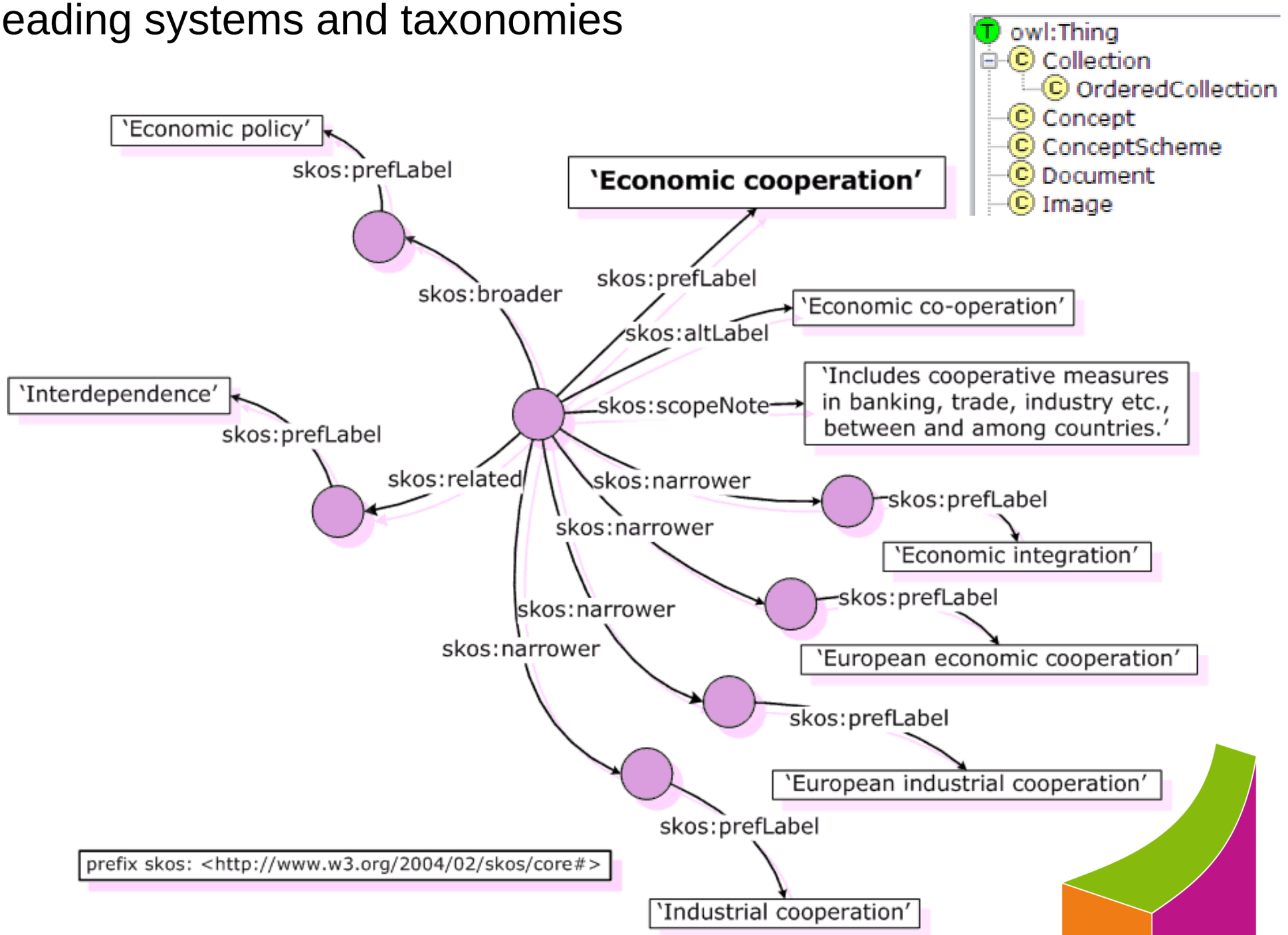




Vokabulare: Simple Knowledge Organization System (SKOS)

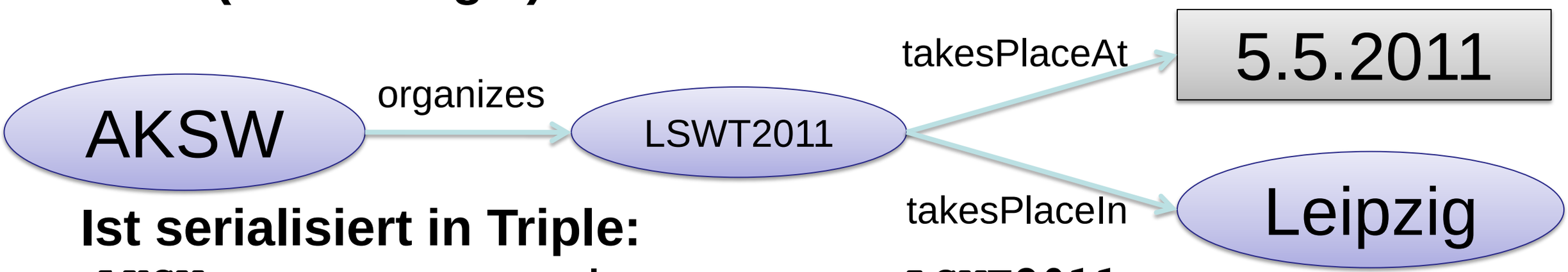
- broader
  - broaderGeneric
  - broaderPartitive
- broaderInstantive
- changeNote
- contributor
- creator
- date
- definition
- description
- example
- hasTopConcept
- hasVersion
- homepage
- inScheme
- isPrimarySubjectOf
- isReplacedBy
- isSubjectOf
- issued
- isVersionOf
- member
- memberList
- modified
- narrower
  - narrowerGeneric
  - narrowerPartitive
- narrowerInstantive
- owl:versionInfo
- primarySubject
- rdf:value
- rdfs:comment
- rdfs:isDefinedBy
- rdfs:label
- rdfs:seeAlso
- related
  - relatedHasPart
  - relatedPartOf
- replaces
- semanticRelation
- subject
- subjectIndicator
- symbol
- term_status
- title

- support the use of thesauri, classification schemes, subject heading systems and taxonomies



Linked Data Web

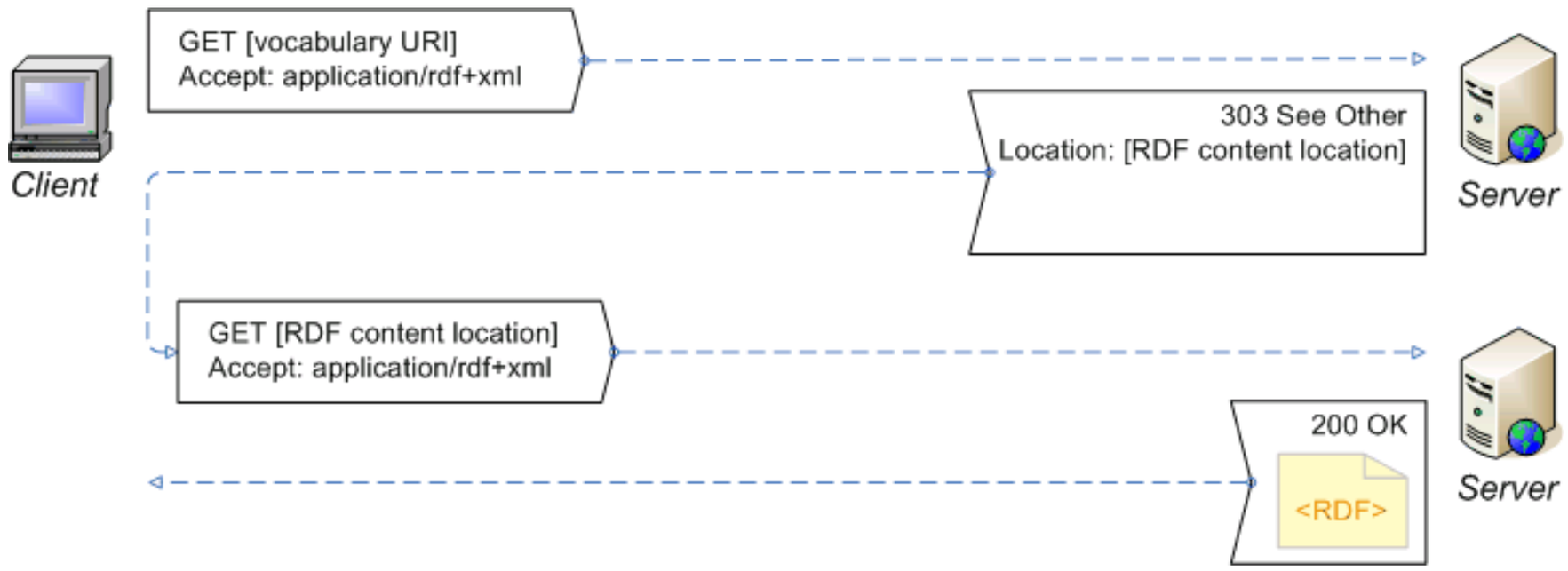
1. Nutzt (in der Regel) RDF als Datenmodel



Ist serialisiert in Triple:

```
AKSW          organizes      LSWT2011
LSWT2011      takesPlaceAt   "20110505"^^xsd:date
LSWT2011      takesPlaceIn   Leipzig
```

2. Nutzt dereferenzierbare HTTP-URIs, z.B. Content-negotiation



3. Links auf Resourcen (in anderen Wissensbasen)

Agenda

Teil 1: Einführung in Daten-Web-Technologien (ab 13 Uhr)

13.00 - 13.45 Uhr Semantic Web Vision, Grundlagen RDF, RDF-Schema

13:45 - 14:45 OWL + Demo Protégé, SPARQL + Triple Stores

14:45 - 15.30 Daten-Web-Beispiele: DBpedia und LinkedGeoData

(15 Minuten Kaffeepause)

Teil 2: GoodRelations (ab 15:45 Uhr bis 18 Uhr)

Ontologie –philosophisch

- Begriff existiert nur in der Einzahl (es gibt also keine „Ontologien“)
- bezeichnet die *„Lehre vom Sein“*

- zu finden bei Aristoteles (Sokrates), Thomas von Aquin, Descartes, Kant, Hegel, Wittgenstein, Heidegger, Quine, …

Ontologie – informatisch

Gruber (1993):

„An Ontology is a

formal specification

of a shared

conceptualization

of a domain of interest“

- maschinell interpretierbar
- beruht auf Konsens
- beschreibt Begrifflichkeiten
- bezogen auf ein „Thema“ (Gegenstandsbereich)

Ontologie – praktisch

- Instanziierung von Klassen durch Individuen
- Begriffshierarchien (Taxonomien, „Vererbung“): Klassen, Begriffe
- binäre Relationen zwischen Individuen: Properties, Roles
- Eigenschaften von Relationen (z.B. range, transitive)
- Datentypen (z.B. Zahlen): concrete domains
- logische Ausdrucksmittel
- klare Semantik!

Object Properties

```
Zugehoerigkeit rdf:type owl:ObjectProperty
```

Domain und Range abstrakter Rollen

```
Zugehoerigkeit rdfs:domain Person

Zugehoerigkeit rdfs:range Organisation
```

Datatype Properties

Datatype Properties haben Datentypen im Range

```
Vorname   rdf:type    owl:DatatypeProperty
```

Domain und Range konkreter Rollen

```
Vorname   rdfs:domain Person
```

```
Vorname   rdfs:range   xsd:string
```

XML Datentypen (und selbst definierte) können verwendet werden

Propertyeigenschaften

- Domain
- Range
- Transitivität, d.h.
  r(a,b) und r(b,c) impliziert r(a,c)
- Symmetrie, d.h.
  r(a,b) impliziert r(b,a)
- Funktionalität
  r(a,b) und r(a,c) impliziert b=c
- Inverse Funktionalität
  r(a,b) und r(c,b) impliziert a=c

Einfache Klassenbeziehungen

Disjunkt:

```
Mann     owl:disjointWith     Frau
```

Äquivalenz:

```
Auto     owl:equivalentClass     Automobil
```

Logische Klassenkonstruktoren

- logisches Und (Konjunktion):
  `owl:intersectionOf`
- logisches Oder (Disjunktion):
  `owl:unionOf`
- logisches Nicht (Negation):
  `owl:complementOf`
- Werden verwendet, um komplexe Klassen aus einfachen Klassen zu konstruieren.

Beziehungen zwischen Individuen

Gleichheit von Individuen:

`SebastianDietzold owl:sameAs SebastianTramp`

Verschiedenheit von Individuen:

`SoerenAuer owl:differentFrom SebastianTramp`

Propertyeinschränkungen (allValuesFrom)

dienen der Definition komplexer Klassen durch Properties:

```
Wissenschaftler   rdf:type              owl:Restriction

Wissenschaftler   owl:onProperty        authorOf

Wissenschaftler   owl:allValuesFrom     WissenschaftlichePubl
```

Rolleneinschränkungen (someValuesFrom)

```
BestSellerAuthor   rdf:type            owl:Restriction
BestSellerAuthor   owl:onProperty      authorOf
BestSellerAuthor   owl:someValuesFrom  BestSeller
```

Rolleneinschränkungen (Kardinalitäten)

Ehemann	**rdf:type**	**owl:Restriction**
Ehemann	**owl:onProperty**	**verheiratetMit**
Ehemann	**owl:Cardinality**	**“1”^^xsd:int**

Kinderreich	**rdf:type**	**owl:Restriction**
Kinderreich	**owl:onProperty**	**hatKind**
Kinderreich	**owl:MinCardinality**	**“3”^^xsd:int**

Rolleneinschränkungen (hasValue)

```
Leipziger    rdf:type          owl:Restriction

Leipziger    owl:onProperty    hatWohnOrt

Leipziger    owl:hasValue      Leipzig
```

OWL Werkzeuge

Editoren

- Protegé, http://protege.stanford.edu
- SWOOP, http://www.mindswap.org/2004/SWOOP/
- OWL Tools, http://owltools.ontoware.org/

Inferenzmaschinen

- Pellet, http://www.mindswap.org/2003/pellet/index.shtml
- KAON2, http://kaon2.semanticweb.org
- FACT++, http://owl.man.ac.uk/factplusplus/
- Racer, http://www.racer-systems.com/
- Cerebra, http://www.cerebra.com/index.html

Agenda

Teil 1: Einführung in Daten-Web-Technologien (ab 13 Uhr)

13.00 - 13.45 Uhr Semantic Web Vision, Grundlagen RDF, RDF-Schema

13:45 - 14:45 OWL + Demo Protégé, SPARQL + Triple Stores

14:45 - 15.30 Daten-Web-Beispiele: DBpedia und LinkedGeoData

(15 Minuten Kaffeepause)

Teil 2: GoodRelations (ab 15:45 Uhr bis 18 Uhr)

Triple Stores

Aufgaben

- Laden und Serialisieren von RDF
- Persistente Speicherung von RDF Wissensbasen – oft als Quad (G,S,P,O)
- Performantes Abfragen (via SPARQL)
- Teilweise Inferenz

Arten von Triple Stores

Native

- Speziell optimiert für das RDF-Datenmodell
- Vertreter: OWLIM, YARS

Datenbankbasierte

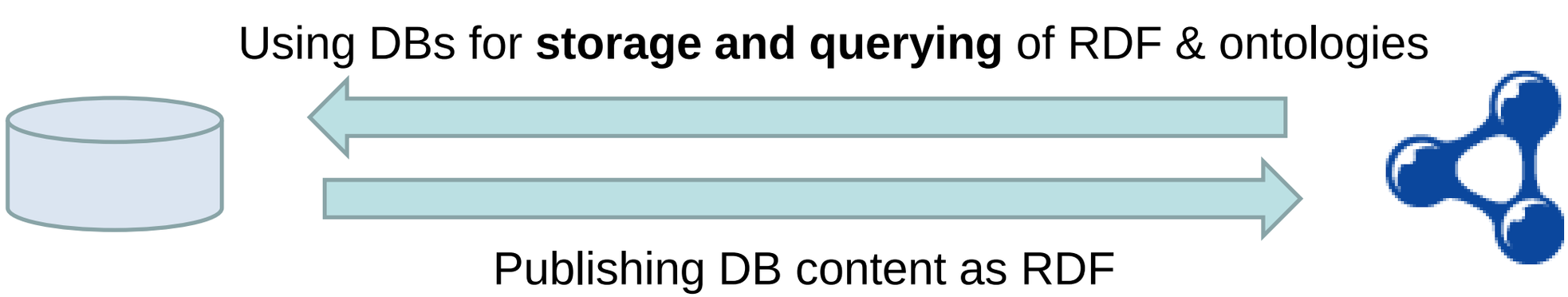
- Relationales DB-Schema zum Speichern
- Vertreter: Sesame/SDB, OntoWiki/Erfurt

Mischformen

- Relationale Datenbanken erweitert um spezielle Datentypen und Indizees
- Vertreter: Oracle, Virtuoso

Vergleich DB vs. RDF/Ontologien

	Relational Databases	RDF & Ontologies
Data Model	Relational (tables, columns, rows)	Triples (subject, predicate, object)
Schema and data separation	☑	☐
Implicit information	☐	☑
Scalability	☑	☐
Schema flexibility	☐	☑
Web data integration readiness	☐	☑



Relationales Schema für RDF

Viele Varianten möglich:

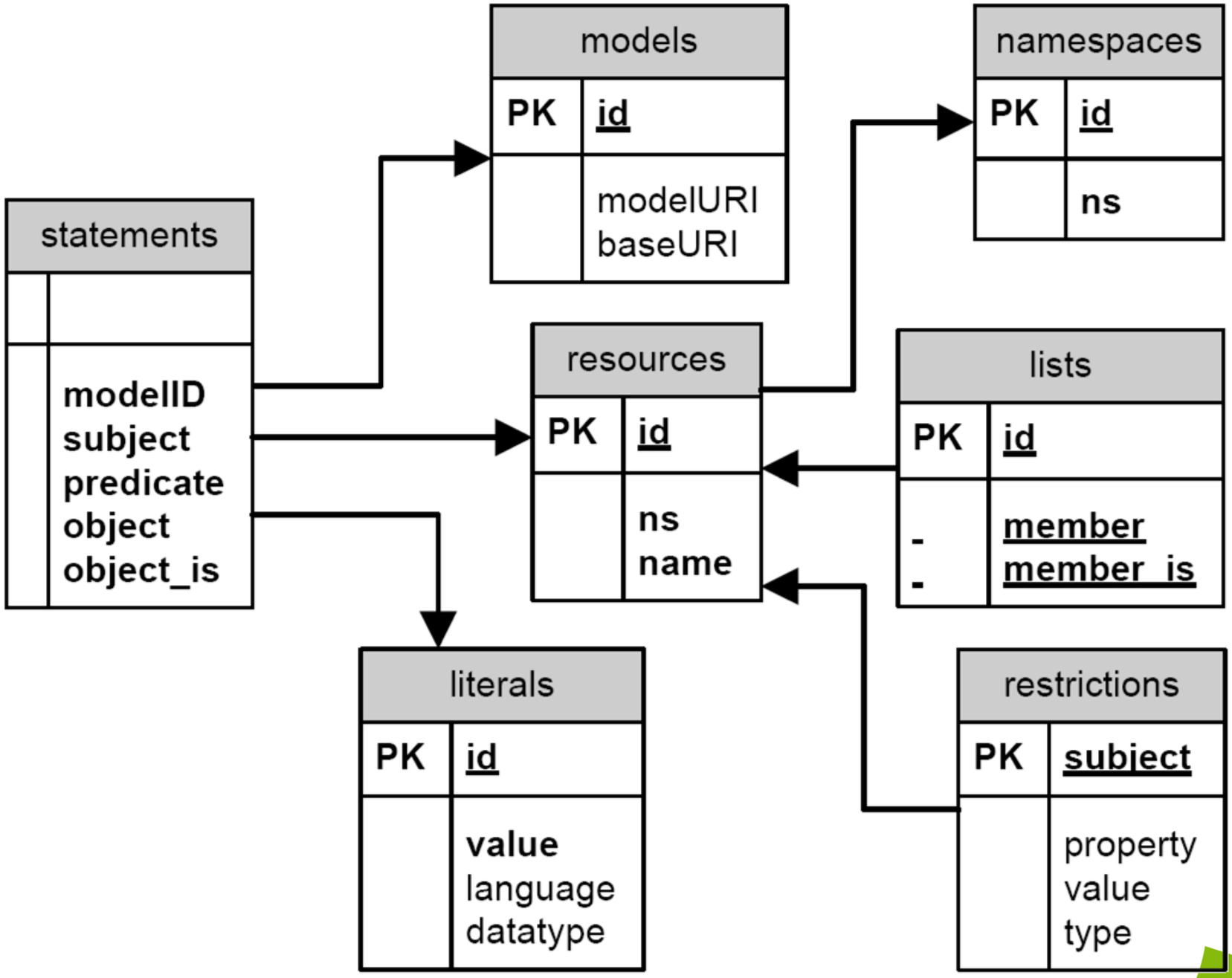
- Normalisiert
- Unnormalisiert
- Mischformen

Hilfstabellen – materialisierte Views

- für bestimmte Properties, Listen, OWL-Kostrukte

Verschiedene Indizierungen

- z.B. GSP, GPO, GOS etc.



Typische Vertreter

Open Source:

- **Virtuoso** - komplete Middleware (inkl. DB, Web-Server, Web Services …)
- **Jena TDB** – Persistenzschicht für Jena API, pure-Java, memory mapped I/O, angepasste B+Tree-Implementierung, optimierte range filter für XSD

Proprietär:

- **YARS2**: A Federated Repository for Querying Graph Structured Data from the Web distributed architecture, scalability experiments up to 7bn LUBM(50000)
- **BigOWLIM** – supports forward-chaining inference, 3bn LUBM(25000)
- **AllegroGraph**
- **Oracle 11G**

Mehr unter: http://esw.w3.org/topic/LargeTripleStores

SPARQL

SPARQL (sprich engl. sparkle) steht für

- **SPARQL Protocol And RDF Query Language**

W3C-Spezifikation (seit 15. Januar 2008)

Anfragsprache zur Abfrage von Instanzen aus RDF-Dokumenten

große praktische Bedeutung

Teile der SPARQL-Spezifikation

- Anfragesprache: Thema dieser Vorlesung
- Ergebnisformat: Darstellung von Ergebnissen in XML
- Anfrageprotokoll: Übermittlung von Anfragen und Ergebnissen

Einfache Beispielanfrage

```
PREFIX ex: <http://example.org/>
SELECT ?titel ?autor WHERE {
  ?buch ex:VerlegtBei  <http://springer.com/Verlag> .
  ?buch ex:Titel       ?titel .
  ?buch ex:Autor       ?autor .
}
```

Hauptbestandteil ist ein Anfragemuster (WHERE)

- Anfragemuster verwenden die Turtle-Syntax für RDF
- Muster dürfen Variablen enthalten (?variable)

Kurzschreibweisen für URIs möglich (PREFIX)

Anfrageergebnis durch Auswahl von Variablen (SELECT)

Graph-Muster

RDF Graph:

```
…
JensLehmann       rdf:type          foaf:person
JensLehmann       foaf:knows        SoerenAuer
JensLehmann       foaf:birthDay     “10.4.1981”
SebastianTramp    rdf:type          foaf:person
SebastianTramp    foaf:knows        SoerenAuer
…
```

Graph pattern (SPARQL core):

```
?person   rdf:type       foaf:person
?person   foaf:knows     http://aksw.org/SoerenAuer
```

OPTIONAL

Das Schlüsselwort OPTIONAL erlaubt die Angabe optionaler Teile eines Musters.

Beispiel:

```
{ ?buch ex:VerlegtBei
   <http://springer.com/Verlag> .
  OPTIONAL { ?buch ex:Titel ?titel . }
  OPTIONAL { ?buch ex:Autor ?autor . }
}
```

Teile eines Anfrageergebnisses können ungebunden sein:

?buch	?titel	?autor
http://example.org/buch1	“Semantic Web for Dummies”	http://example.org/autor1
http://example.org/buch2	“Faust 1”	
http://example.org/buch3		Schiller

UNION

Das Schlüsselwort UNION erlaubt die Angabe alternativer Teile eines Musters.

Beispiel:

```
{ ?buch ex:VerlegtBei <http://springer.com/Verlag> .

{ ?buch ex:Autor      ?autor . } UNION

{ ?buch ex:Verfasser  ?autor . }

}
```

Ergebnis entspricht Vereinigung der Ergebnisse mit einer der beiden Bedingungen

Gleiche Variablennamen in beiden Teilen von UNION beeinflussen sich nicht

Filter

Viele Anfragen sind auch mit komplexen Graph-Mustern nicht möglich:

- „Welche Personen sind zwischen 18 und 23 Jahre alt?“
- „Der Nachname welcher Personen enthält einen Bindestrich?“
- „Welche Texte in deutscher Sprache sind in der Ontologie angegeben?“

Filter als allgemeiner Mechanismus für solche Ausdrucksmittel

Filter in SPARQL

Beispiel:

```
PREFIX ex: <http://example.org/>
SELECT ?buch WHERE
{ ?buch  ex:VerlegtBei  <http://springer.com/Verlag> .
  ?buch  ex:Preis       ?preis
FILTER (?preis < 35)
}
```

Schlüsselwort FILTER, gefolgt von Filterausdruck in Klammern

Filterbedingungen liefern Wahrheitswerte (und ev. auch Fehler)

Viele Filterfunktionen nicht durch RDF spezifiziert / Funktionen teils aus XQuery/XPath-Standard für XML übernommen

Filterfunktionen: Vergleiche und Arithmetik

Vergleichoperatoren: <, =, >, <=, >=, !=

- Vergleich von Datenliteralen gemäß der jeweils natürlichen Reihenfolge
- Unterstützung für numerische Datentypen, xsd:dateTime, xsd:string (alphabetische Ordnung), xsd:Boolean (1 > 0)
- für andere Typen und sonstige RDF-Elemente nur = und != verfügbar
- kein Vergleich von Literalen inkompatibler Typen (z.B. xsd:string und xsd:integer)

Arithmetische Operatoren: +, -, *, /

- Unterstützung für numerische Datentypen
- Verwendung zur Kombination von Werten in Filterbedingungen

Bsp.: FILTER(?gewicht/(?groesse * ?groesse) >= 25)

SPARQL Ausgabeformate

Bisher waren alle Ergebnisse Tabellen: Ausgabeformat SELECT

Kodierung von Ergebnissen in RDF-Graphen: Ausgabeformat CONSTRUCT

```
CONSTRUCT { ?person  ex:mailbox  ?email .
            ?person  ex:telefon  ?telefon . }
WHERE     { ?person  ex:email    ?email .
            ?person  ex:tel      ?telefon . }
```

SPARQL unterstützt zwei weitere Ausgabeformate:

- ASK prüft nur, ob es Ergebnisse gibt, keine Parameter
- DESCRIBE (informativ) liefert zu jeder gefundenen URI eine RDF-Beschreibung (anwendungsabhängig)

Ergebnisse sortieren

Sortierung von Ergebnissen mit Schlüsselwort ORDER BY

```
SELECT ?buch, ?preis
WHERE { ?buch <http://example.org/Preis> ?preis . }
ORDER BY ?preis
```

- Sortierung wie bei Filter-Vergleichoperatoren,
- Sortierung von URIs alphabetisch als Zeichenketten
- Reihenfolge zwischen unterschiedlichen Arten von Elementen: Ungebundene Variable < leere Knoten < URIs < RDF-Literale
- nicht jede Möglichkeit durch Spezifikation definitert

Weitere mögliche Angaben:

- ORDER BY DESC(?preis): absteigend
- ORDER BY ASC(?preis): aufsteigend, Voreinstellung
- ORDER BY DESC(?preis), ?titel: hierarchische Ordnungskriterien

LIMIT, OFFSET und DISTINCT

Einschränkung der Ergebnismenge:

- LIMIT: maximale Anzahl von Ergebnissen (Tabellenzeilen)
- OFFSET: Position des ersten gelieferten Ergebnisses
- SELECT DISTINCT: Entfernung von doppelten Tabellenzeilen

```
SELECT DISTINCT ?buch, ?preis
WHERE { ?buch <http://example.org/Preis> ?preis . }
ORDER BY ?preis LIMIT 5 OFFSET 25
```

LIMIT und OFFSET nur mit ORDER BY sinnvoll!

SPARQL Beschränkungen

- Keine Daten Manipulation, d.h. Insert, Update, Delete Anfragen SPARUL Erweiterung - http://jena.hpl.hp.com/~afs/SPARQL-Update.html
- Keine Daten- und Zugriffs-Kontrolle, d.h. GRANT, REVOKE etc.
- Keine Anfrageintrospektion (ala SQL DESCRIBE etc.)
- Keine Subqueries
- Keine Anfrageparametrisierung (Prepare)
- Keine Cursors
- Keine Volltextsuche (bif:contains Erweiterung)
- Keine Aggregatfunktionen

Agenda

Teil 1: Einführung in Daten-Web-Technologien (ab 13 Uhr)

13.00 - 13.45 Uhr Semantic Web Vision, Grundlagen RDF, RDF-Schema

13:45 - 14:45 OWL + Demo Protégé, SPARQL + Triple Stores

14:45 - 15.30 Daten-Web-Beispiele: DBpedia und LinkedGeoData

(15 Minuten Kaffeepause)

Teil 2: GoodRelations (ab 15:45 Uhr bis 18 Uhr)