

# Previously on Foundations of Linked Data ...

- We have learned how to explain and to match basic graph patterns to graphs.
- We translated a given SPARQL WHERE clause to a SPARQL algebra expression.
- Given a query, we have learned how RDF datasets are handled with FROM and FROM NAMED clauses in conjunction with GRAPH.
- We evaluated a SPARQL algebra expression on a given RDF dataset and specified the solutions of the entire algebra expression and also of partial expressions.
- We have learned how to generate the results to a SPARQL abstract query, considering the solution sequence of the graph pattern algebra expression and the query form.

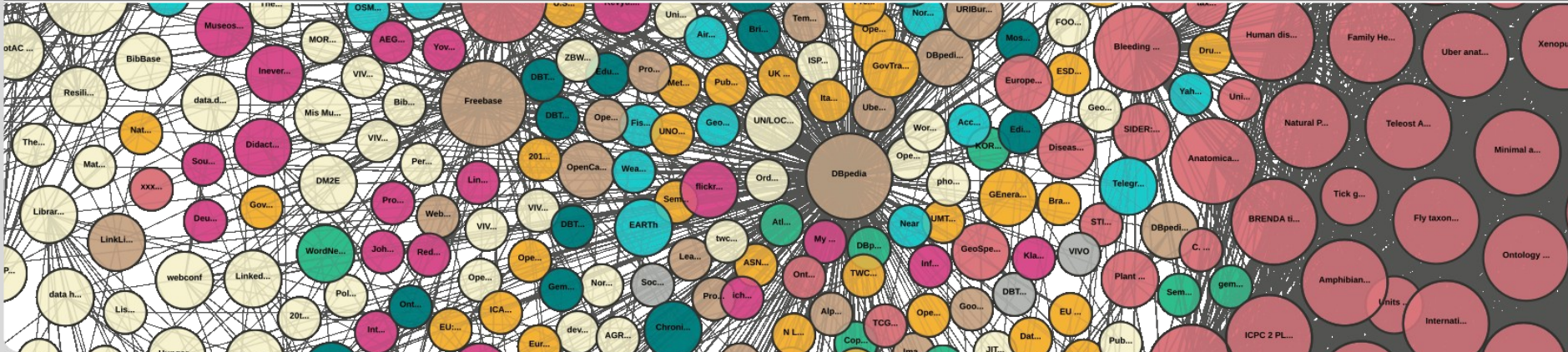
# Foundations of Linked Data - Summer 2022

## C06 – Publishing and Consuming Linked Data

**Version: 2022-05-31**

**Lecturer: Prof. Dr. Andreas Harth**

CHAIR OF TECHNICAL INFORMATION SYSTEMS



# CC - Creative Commons Licensing

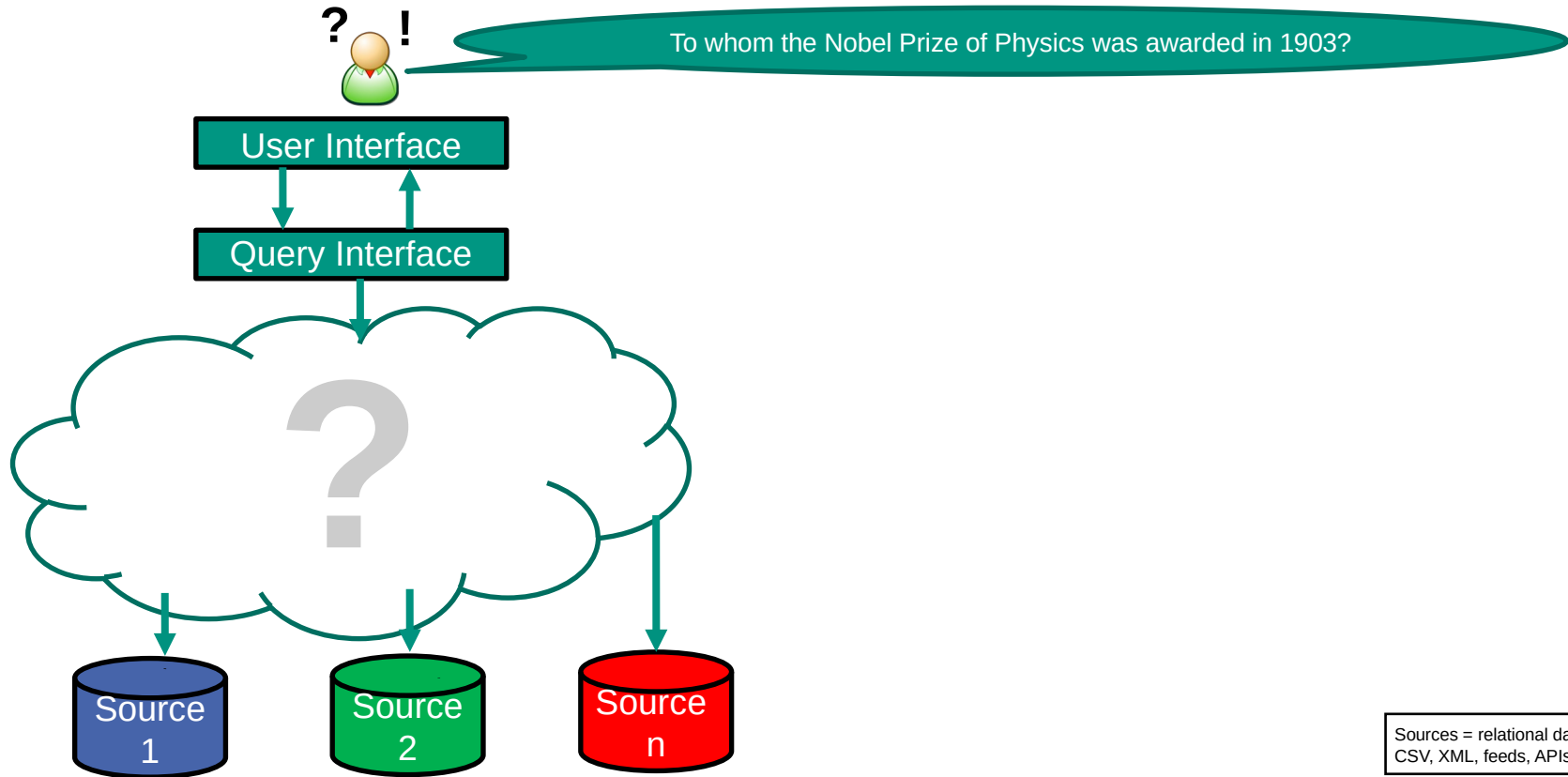
- This set of slides is part of the lecture „Foundations of Linked Data“ held at Friedrich-Alexander-Universität Erlangen-Nürnberg.
- The content of the lecture was prepared by Prof. Dr. Andreas Harth based on his book „Introduction to Linked Data“.
- The initial slides were prepared by Lars Heling, Maribel Acosta, Tobias Käfer, Christian Fleiner and Andreas Harth.
- **This content is licensed under a Creative Commons Attribution 4.0 International license (CC BY 4.0):** <http://creativecommons.org/licenses/by/4.0/>



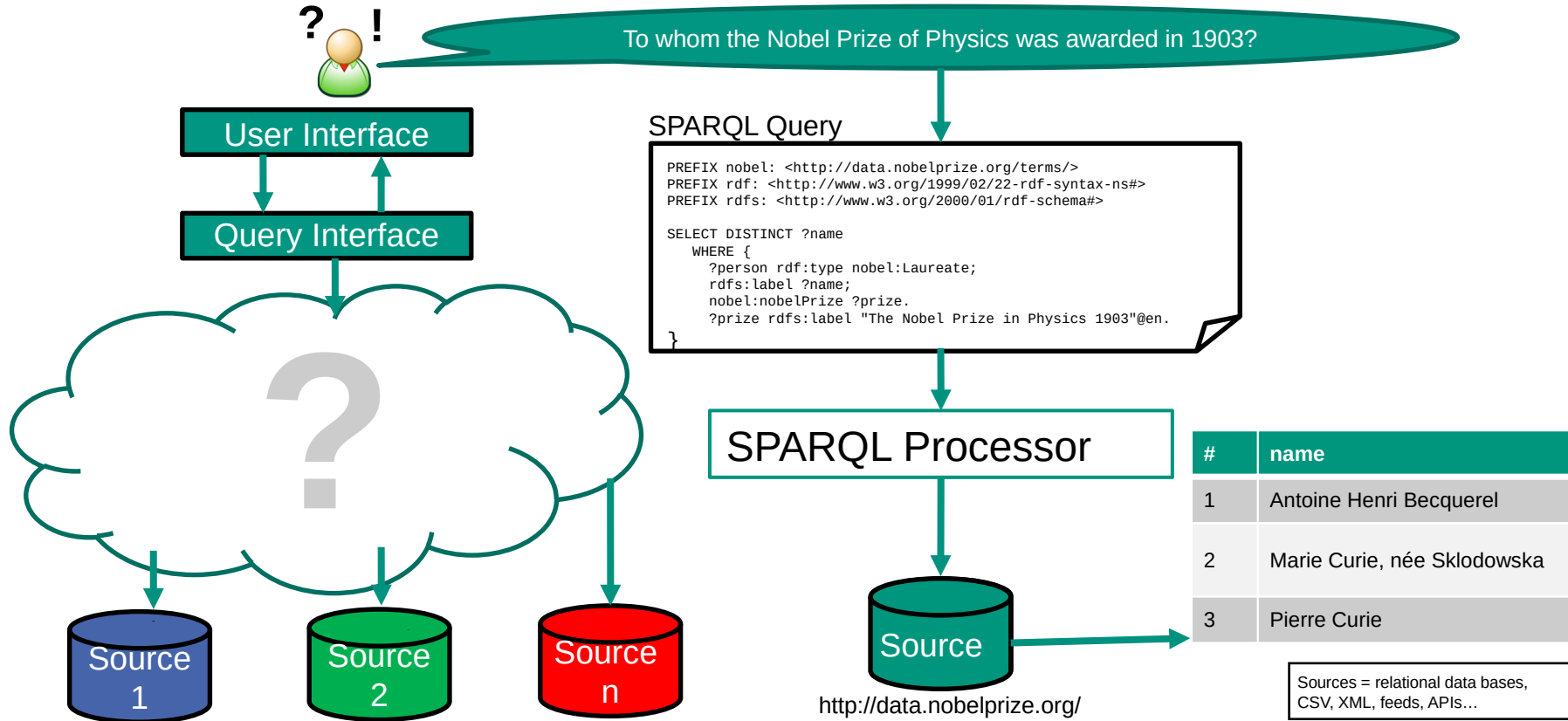
# Outline

- **Linked Data Systems Architectures**
- Web Architecture Revisited
- Linked Data User Interfaces
- Linked Data Querying Processors
- Linked Data Platform (LDP)
- Raising Legacy Systems to Linked Data

# Linked Data Integration System Architecture (1)



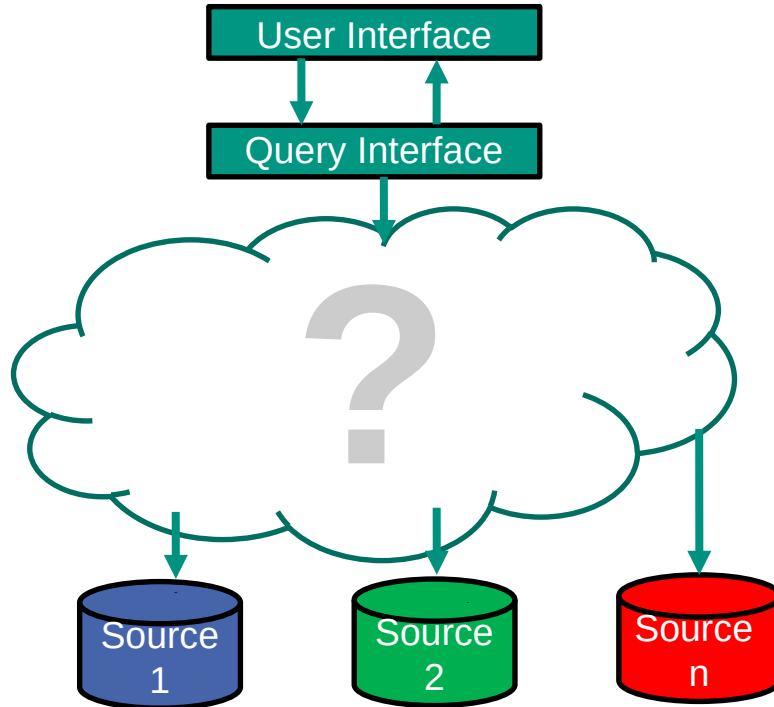
# Linked Data Integration System Architecture (2)



# Linked Data Integration System Architecture (3)



Show me all paintings of John Singer Sargent!

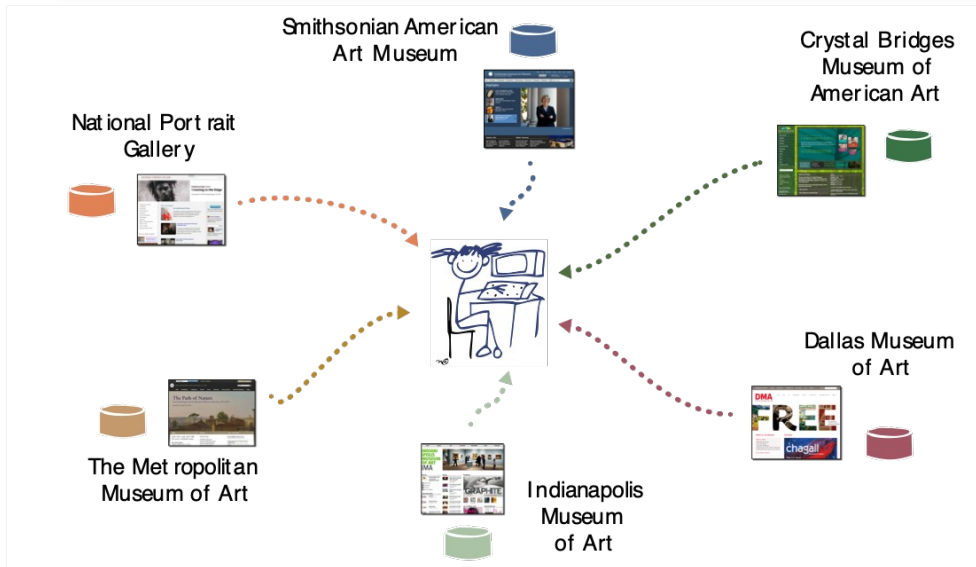


Sources = relational data bases,  
CSV, XML, feeds, APIs...

# Cultural Heritage Data: Get Info from Web Pages



## Paintings of John Singer Sargent



### ■ Problem

- Web pages are machine processable, but not machine understandable
- Impractical for building applications using the data

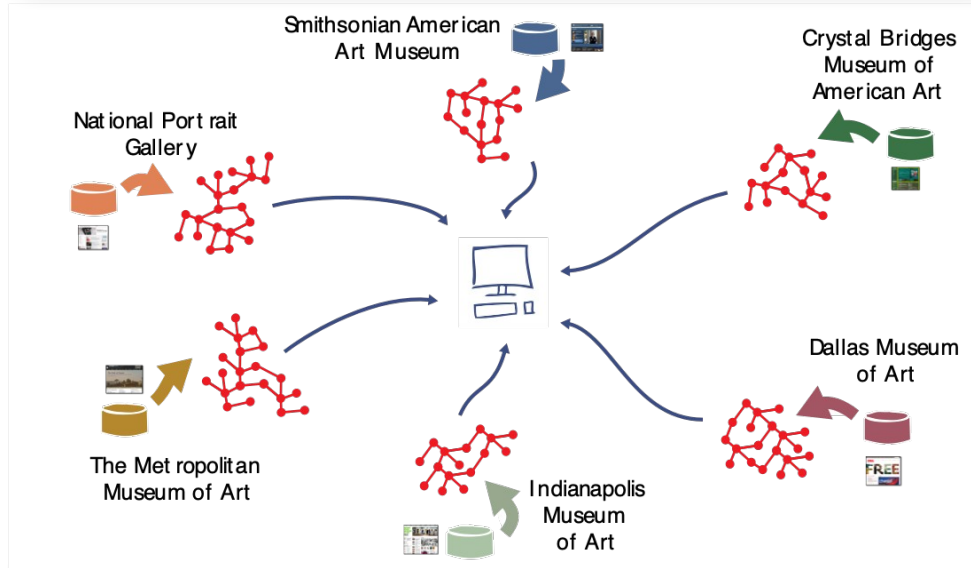
### ■ Solution: publish the data as Linked Data!



# Step 1: Provide Data as RDF (Over HTTP)



## Paintings of John Singer Sargent



### Positive

- Data is public
- Data is in a common format

### Negative

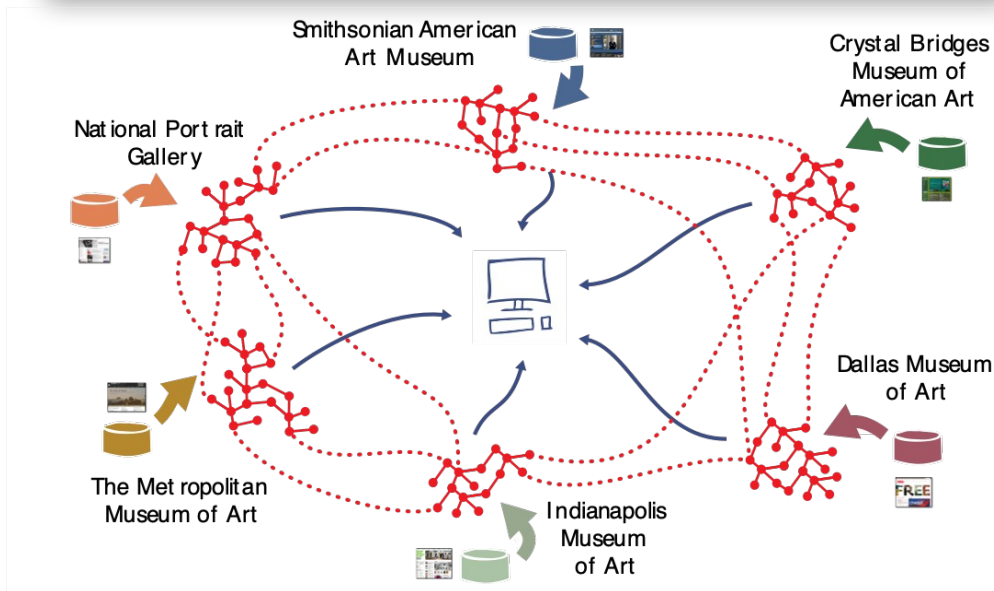
- We only have islands of data

## Step 2: Map Identifiers (URIs) Between Sources



Paintings of John Singer Sargent

■ Mapped Identifiers

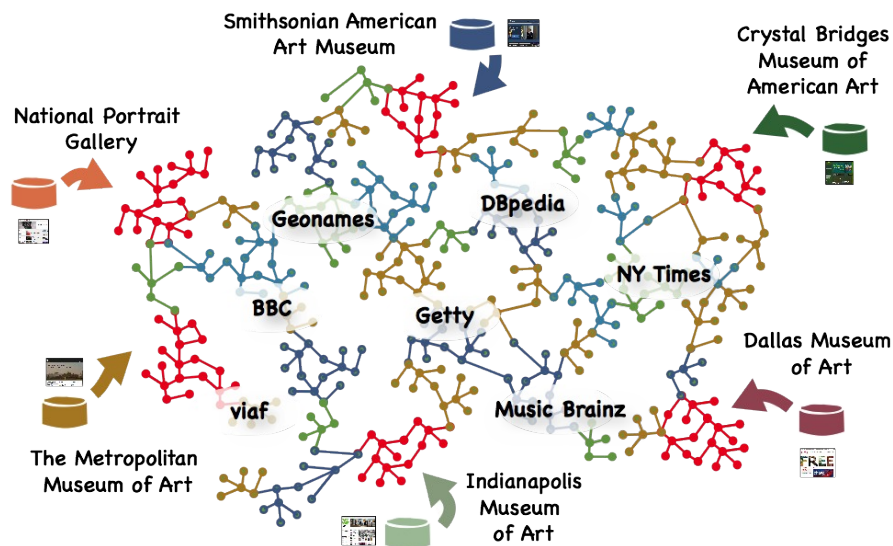


# Step 3: Linked Open Data



Paintings of John Singer Sargent

■ Data with Mapped Schemas



# Linked Data Principles

- **Use URIs to name things.**
  - Things are not only documents, but also people, locations, concepts, etc.
- **Use HTTP URIs so that users can look up those names.**
  - Users refer to humans and machine agents alike.
- **When someone looks up a URI, provide useful information, using the standards (RDF, RDFS, SPARQL).**
  - What “useful” means depends on the data publisher (but the data publisher should return the “useful” data in RDF).
- **Include links to other URIs, so that they can discover more things.**
- See [official documentation](#)

# Data Integration (Read) and System Interoperation (Read / Write)

- **Linked Data** provides a uniform network interface and a graph-structured data representation
- **Linked Data technologies** are useful for data publishing, exchange and integration.
  - Consisting of ways to encode, access, and manipulate information
  - Consisting of ways to represent, access, and manipulate knowledge
- **Data integration** denotes the combination of data from multiple sources.
- **Data integration** can be simply achieved by merging graphs. However, this action might require object consolidation and entity linking,
- **System interoperation** denotes the combination of functionalities from multiple systems and applications.

# Application of Linked Data

- The application of Linked Data is recommended in scenarios where:
  - Many people control data and want to retain control.
  - Existing Linked Data should be reused
  - You have already gained experience with the technologies and want to use existing Linked Data tools.
- How to store (publish) and access (consume) a huge RDF dataset in a way that allows for querying?
  - Manually collect all data and load the data into a SPARQL endpoint?
  - Automatically follow links and load the collected data into a SPARQL endpoint?
  - Automatically load data into SPARQL endpoint by using queries?

□ Use a federated query engine

# Integration of Graph Data

- Ideally, users are querying the Web without the requirement to know where the data was published.
- Realistically, data is published at different servers resulting into an RDF dataset.
- Hyperlinks can be used to connect data and systems in a decentralized manner.

# Outline

- Linked Data Systems Architectures
- **Web Architecture Revisited**
- Linked Data User Interfaces
- Linked Data Querying Processors
- Linked Data Platform (LDP)
- Raising Legacy Systems to Linked Data



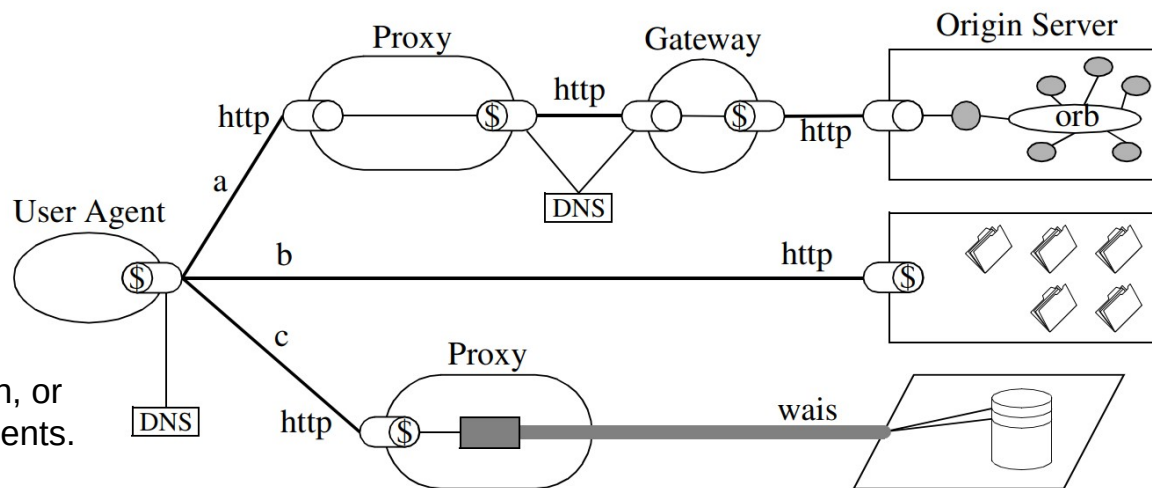
# Components: User Agent vs. Origin Server

## ■ According to Fielding [1]:

### ■ Components

- Origin Server
- Gateway
- Proxy
- User Agent

- A **connector** is an abstract mechanism that mediates communication, coordination, or cooperation among components.



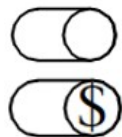
Client Connector: ○    Client+Cache: ○\$    Server Connector: ○○    Server+Cache: ○\$

[1] Fielding, R. T. (2000). REST: architectural styles and the design of network-based software architectures. Doctoral dissertation, University of California. Figure 5-10

# Components: User Agent vs. Origin Server

## ■ According to Fielding [1]:

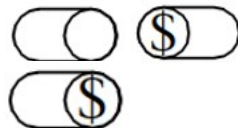
- An **origin server** uses a server connector to govern the namespace for a requested resource. It is the definitive source for representations of its resources and must be the ultimate recipient of any request that intends to modify the value of its resources.



- A **gateway** (a.k.a., reverse proxy) component is an intermediary imposed by the network or origin server to provide an interface encapsulation of other services, for data translation, performance enhancement, or security enforcement.







- A **proxy** component is an intermediary selected by a client to provide interface encapsulation of other services, data translation, performance enhancement, or security protection.



- A **user agent** uses a client connector to initiate a request and becomes the ultimate recipient



Client Connector:  Client+Cache:  Server Connector:  Server+Cache:  access to n needs.

[1] Source: Fielding, R. T. (2000). REST: architectural styles and the design of network-based software architectures. Doctoral dissertation, University of California. Figure 5-10

# REST: REpresentational State Transfer

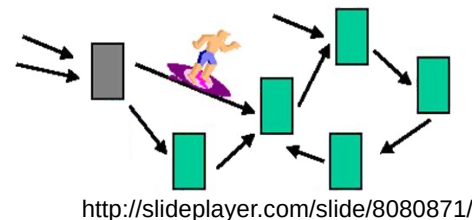
- A widely used architecture style for HTTP APIs that require read and write access to resources is REpresentational State Transfer (REST).
- A service that follows the formal constraints of REST is called RESTful.
- Those constraints ensure several desirable properties:
  - Performance
  - Scalability
  - Simplicity
  - Modifiability
  - Visibility
  - Portability
  - Reliability

# REST- Formal Constraints

- Client-Server Architecture: Ensured by using HTTP.
- Statelessness: Ensured by using HTTP.
- Cacheability: Ensured by using HTTP (GET and POST).
- Layered System: Proxies or load balancers placed between client and server may not affect their communication. Security can be another layer separating business logic and security logic.
- Code On Demand (optional): Possibility to download code from the server. Has never played a big role.
- Uniform Interface: Allows decoupling of architecture and application. Has four sub-constraints:
  - Resource Identification in Requests: Individual resources are identified, e. g. using URIs. Identifier and representation are separated.
  - Resource Manipulation Through Representation: When a client holds a representation of a resource it has everything needed to modify or delete the resource.
  - Self-Descriptive Messages: Messages include enough information to understand how to process it.
  - Hypermedia As The Engine Of Application State (HATEOAS): given an entry point (resource) responses should include links (or other hypermedia controls) to discover all resources it needs.

# General User Agent Model

- Characteristics of a generic user agent on the web (e.g., Web browser):



1. The user agent starts its interaction based on a specific seed URI

2. The user agent performs HTTP requests on URIs and parses the response

3. Based on the response the user agent has one or multiple choices as to which interaction to perform next

4. The user agent decides which link to follow and initiates a new request

# Server-driven Content Negotiation

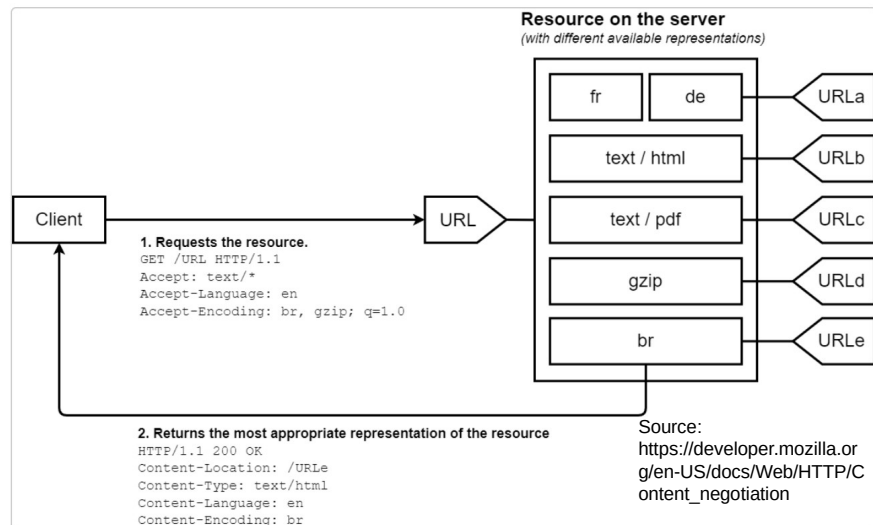
- Servers might host multiple resource representations. Some of them might not even be processable by a user agents.
- In order to return the most fitting resource representation for the user agent, content negotiation is performed.
  - User agent (client) sends several HTTP headers along with the URL.
  - The server uses the header information as basis to decide which resource to return.

## USER AGENT LOOP

```
do
  emit HTTP request on URI
  parse and process response
  user reads response and \
  selects URI
while not bored
```

## SERVER LOOP

```
forever:
  wait for incoming HTTP request
  parse HTTP request
  if (GET):
    decide on response
    serialize HTTP response
```



# When Resource State is (Not) Sent/Received?

## – HTTP Message Semantics [RFC7231]

HTTP Request Method	HTTP Request or Response Code	HTTP Message Semantics: The HTTP Message Body Contains...
GET	Request	Nothing
PUT	Request	State of the resource
POST	Request	Arbitrary data or state of resource
DELETE	Request	Nothing
any	Non-2xx	State of the request
GET	2xx	State of the resource
PUT	2xx	State of the resource or empty
POST	2xx	State of the request (referring to new resource)
DELETE	2xx	State of the request or empty

# Two Perspectives on the Linked Data Principles

## Origin Server (Publisher)

- Coin URIs to name things.
- Use a HTTP server to provide access to documents.
- Upon receiving a request for a URI, the server returns useful information (about the URI in the request) in RDF.
- The “useful information” the server returns in the RDF document includes links to other URIs (on other servers).

## User Agent (Consumer)

- Assume URIs as names for things.
- User agents look up HTTP URIs.
- User agents parse RDF documents with useful information and provide the ability to process SPARQL queries.
- User agents can discover more things via accessing links to other URIs.



# Outline

- Linked Data Systems Architectures
- Web Architecture Revisited
- **Linked Data User Interfaces**
- Linked Data Querying Processors
- Linked Data Platform (LDP)
- Raising Legacy Systems to Linked Data

# Linked Data User Interfaces

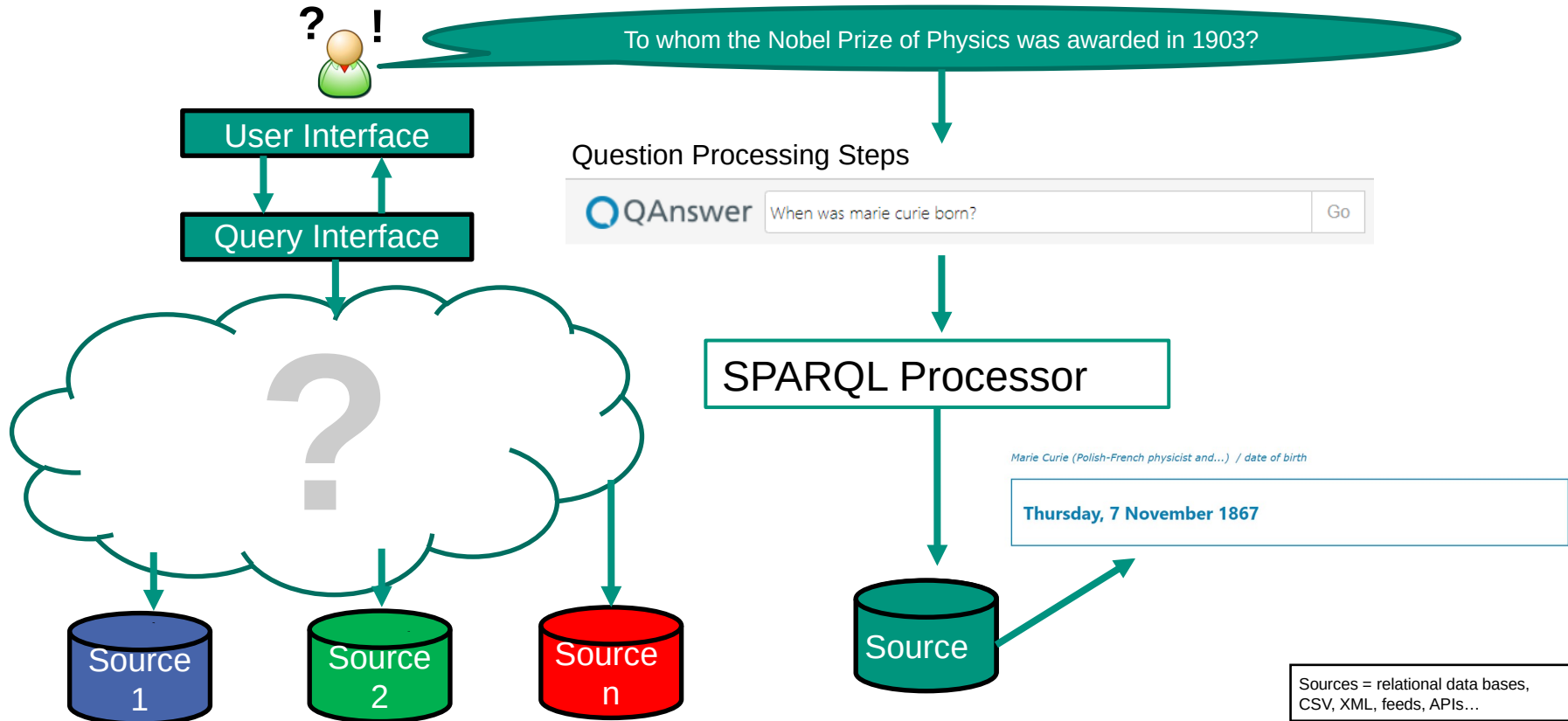
- We consider user interfaces on Linked Data for search, browsing, question answering, and visualizations for displaying results
- Virtual Assistants / Semantic Web Agents
  - A first vision was published by Tim Berners-Lee, James Hendler, and Ora Lassila in 2001:
    - The Semantic Web: A New Form of Web Content That is Meaningful to Computers Will Unleash a Revolution of New Possibilities. [ScientificAmerican.com](https://www.scientificamerican.com/article/semantic-web/).
- Mixed Reality Interfaces
  - Harnessing semantically annotated digital twins
- Question Answering over Linked Data (QALD)

# QALD Systems

- According to the Tutorial “Constructing Question Answering Systems over Knowledge Graphs” that was conducted as part of the ESWC’21 [1]:
  - Knowledge Graphs are designed to be easily consumed by machines, but they are not easily accessible by end-users. Question Answering (QA) over Knowledge Graphs (KGs) is seen as a technology able to bridge the gap between end-users and Knowledge Graphs.
- Question processing steps with QAnswer (a Question Answering Query Engine):
  1. Expansion
  2. Query generation
  3. Query ranking
  4. Answer decision
- Try out [Demo](#) or see [official slides](#) with example.

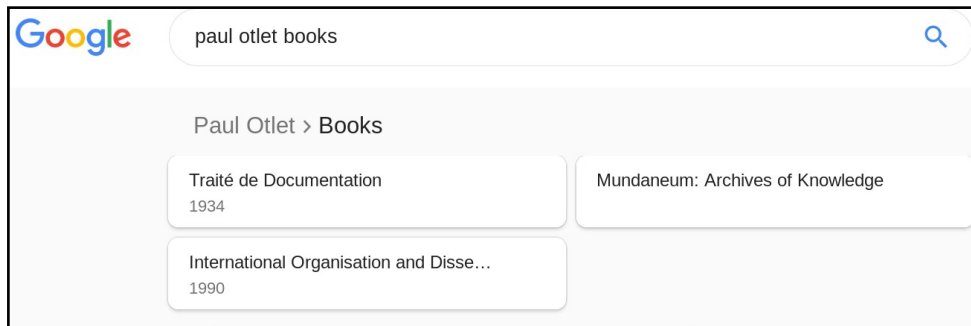
[1] QA Tutorial @ ESWC 2021 · Constructing Question Answering Systems over Knowledge Graphs ([qanswer.github.io](https://qanswer.github.io))

# Linked Data Integration System Architecture (QALD)



# User Interaction with Data: Querying Linked Data

- Entity Search: e.g., “paul otlet”, “paul otlet books”



- Natural Language Question Answering (e.g, QALD challenge)
  - “How many children does Eddie Murphy have?”
  - “Which museum in New York has the most visitors?”
  - “Is Lake Baikal bigger than the Great Bear Lake?”
  - “How many companies were founded in the same year as Google?”
- Speech Interfaces (e.g., Siri, “what is the national anthem of Bulgaria?”)

# Simple Node Browser - Graph

- Simple Node Browser works on plain Linked Data and makes the browsable graph visible.
- The user starts from an initial URI/entry point and decides which link to follow.
- The visit history is not recorded.
- Google's Knowledge Graph and Pubby are examples of simple node browsers.
- Also works with Linked Data on the web (RDF dataset Web)
- Modern Webrowsers like Google Chrome oder Mozilla Firefox classify as hypermedia user agents.

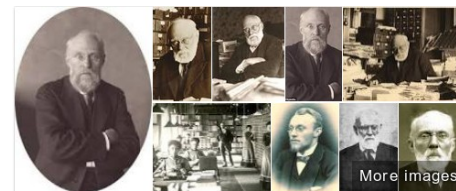
# User Interaction with Data: Simple Node Browser

- One resource is in focus, and the description of the resource is rendered in HTML
- Use **rdfs:label** as human-readable label
- Use **rdfs:comment** for short description
- Use **rdf:type** to show the classes the resource is an instance of
- Use **foaf:img** to show an image of the resource

About: **Paul Otlet**  
An Entity of Type `agent`, from Named Graph `http://dbpedia.org`, within Data Space `dbpedia.org`

Paul Marie Ghislain Otlet was a French-speaking Belgian author, entrepreneur, visionary, lawyer and peace activist; he is one of several people who have been considered the father of information science, a field he called "documentation". Otlet created the Universal Decimal Classification, one of the most prominent examples of faceted classification.

Property	Value
<code>dbpedia-owl:abstract</code>	<ul style="list-style-type: none"><li>Paul Marie Ghislain Otlet was a French-speaking Belgian author, entrepreneur, visionary, lawyer and peace activist; he is one of several people who have been considered the father of information science, a field he called "documentation". Otlet created the Universal Decimal Classification, one of the most prominent examples of faceted classification. Otlet was responsible for the widespread adoption in Europe of the standard American 3x5 inch index card used until recently in most library catalogs around the world (by now largely displaced by the advent of online public access catalogs). Otlet wrote numerous essays on how to collect and organize the world's knowledge, culminating in two books, the <i>Traité de Documentation</i> (1934) and <i>Monde: Essai d'universalisme</i> (1935). In 1907, following a huge international conference, Henri La Fontaine and Otlet created the Central Office of International Associations, which was renamed to the Union of International Associations in 1910, and which is still located in Brussels. They also created a great international center called at first Palais Mondial (World Palace), later, the Mandarum to house the collections and activities of their various organizations and institutes. Otlet was also an Idealist and peace activist, pushing internationalist political ideas that were embodied in the League of Nations and its International Institute of Intellectual Cooperation, working alongside his colleague Henri La Fontaine, who won the Nobel Peace Prize in 1913, to achieve their ideas of a new world polity that they saw arising from the global diffusion of information and the creation of new kinds of international organization.</li></ul>
<code>dbpedia-owl:almaMater</code>	<ul style="list-style-type: none"><li><code>dbpedia:Université libre de Bruxelles</code></li><li><code>dbpedia:Catholic University of Leuven (1834-1968)</code></li></ul>
<code>dbpedia-owl:birthDate</code>	<ul style="list-style-type: none"><li>1868-08-23 (xsd:date)</li></ul>
<code>dbpedia-owl:birthPlace</code>	<ul style="list-style-type: none"><li><code>dbpedia:Belgium</code></li><li><code>dbpedia:Brussels</code></li><li><code>dbpedia:Belgium</code></li></ul>
<code>dbpedia-owl:citizenship</code>	<ul style="list-style-type: none"><li>1944-12-10 (xsd:date)</li></ul>
<code>dbpedia-owl:deathDate</code>	<ul style="list-style-type: none"><li><code>dbpedia:Belgium</code></li><li><code>dbpedia:Brussels</code></li></ul>
<code>dbpedia-owl:deathPlace</code>	<ul style="list-style-type: none"><li><code>dbpedia:Brussels</code></li></ul>



Paul Otlet

Author

Paul Marie Ghislain Otlet was a Belgian author, entrepreneur, visionary, lawyer and peace activist; he is one of several people who have been considered the father of information science, a field he called "documentation". [Wikipedia](#)

**Born:** August 23, 1868, [Brussels, Belgium](#)

**Died:** December 10, 1944, [Brussels, Belgium](#)

**Parents:** [Édouard Otlet](#)

**Books:** [Traité de Documentation](#), [more](#)

**Education:** [Université libre de Bruxelles](#), [Catholic University of Leuven](#)

**Influenced by:** [Henri La Fontaine](#), [Melvil Dewey](#), [Edmond Picard](#)

People also search for

[View 10+ more](#)



Henri La Fontaine



Melvil Dewey



Vannevar Bush



Suzanne Briet

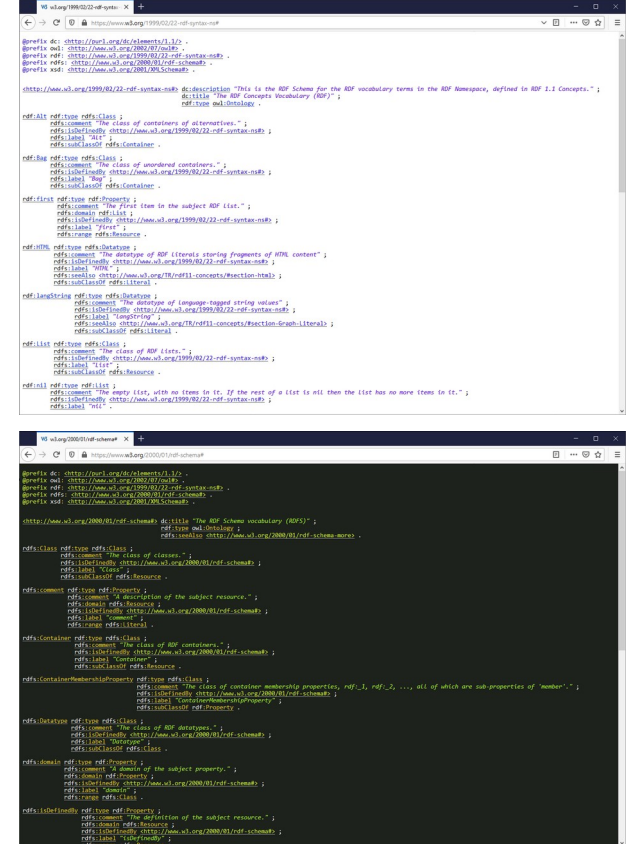


S. R. Rangan...

[Feedback](#)

# RDF Browser

- RDF Browser is a Firefox Add-on that requests RDF files and renders RDF files as Turtle documents with clickable links.
- Supports HTTP CRUD methods in editor mode
- The Add-on is released and maintained by the Chair of Technical Information Systems at Friedrich-Alexander-University Erlangen-Nürnberg.
- Install Addon:  
<https://addons.mozilla.org/en-US/firefox/addon/rdf-browser>
- See source code:  
<https://github.com/kianschmalenbach/rdf-browser>



```

@prefix dc: <http://purl.org/dc/terms/1.1/>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.

<http://www.w3.org/1999/02/22-rdf-syntax-ns#> dc:description "This is the RDF Schema for the RDF vocabulary terms in the RDF namespace, defined in RDF 1.1 Concepts.";
@title "The RDF Concepts Vocabulary (RDF)";
rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#>.

rdfs:Class rdfs:Class {
  rdfs:comment "The class of containers of containers.";
  rdfs:isDefinedBy <http://www.w3.org/1999/02/22-rdf-syntax-ns#>;
  rdfs:label "rdf:List";
  rdfs:subClassOf rdfs:Container .
}

rdfs:Class rdfs:Class {
  rdfs:comment "The class of unordered containers.";
  rdfs:isDefinedBy <http://www.w3.org/1999/02/22-rdf-syntax-ns#>;
  rdfs:label "Set";
  rdfs:subClassOf rdfs:Container .
}

rdfs:Class rdfs:Property {
  rdfs:comment "The first time in the subject RDF list.";
  rdfs:domain rdfs:List;
  rdfs:isDefinedBy <http://www.w3.org/1999/02/22-rdf-syntax-ns#>;
  rdfs:label "First";
  rdfs:range rdfs:Resource .
}

rdfs:Class rdfs:List {
  rdfs:comment "The datatype of RDF (traversal) storage fragments of RDF content";
  rdfs:isDefinedBy <http://www.w3.org/1999/02/22-rdf-syntax-ns#>;
  rdfs:label "List";
  rdfs:range rdfs:Resource .
}

rdfs:Class rdfs:List {
  rdfs:comment "The datatype of language-tagged string values";
  rdfs:isDefinedBy <http://www.w3.org/1999/02/22-rdf-syntax-ns#>;
  rdfs:label "Language-tagged string values";
  rdfs:range <http://www.w3.org/1999/02/22-rdf-syntax-ns#>;
  rdfs:subClassOf rdfs:List .
}

rdfs:Class rdfs:List {
  rdfs:comment "The class of RDF lists.";
  rdfs:isDefinedBy <http://www.w3.org/1999/02/22-rdf-syntax-ns#>;
  rdfs:label "List";
  rdfs:subClassOf rdfs:Resource .
}

rdfs:Class rdfs:List {
  rdfs:comment "The empty list, with no items in it. If the rest of a list is nil then the list has no more items in it.";
  rdfs:isDefinedBy <http://www.w3.org/1999/02/22-rdf-syntax-ns#>;
  rdfs:label "Nil";
  rdfs:subClassOf rdfs:List .
}

```



# Faceted Search and Browsing

- According to [1]:
  - Alternative to keyword-based search, and searching by overall similarity to sample images.
  - Metadata may be faceted, that is, composed of orthogonal sets of categories.
  - The metadata (or an individual facet) may be flat (“by Pablo Picasso”) or hierarchical (“located in Vienna > Austria > Europe”)
  - The metadata (or an individual facet) may be single-valued or multi-valued. That is, the data may allow at most one value to be assigned to an item (“measures 36 cm tall”) or it may allow multiple values to be assigned to an item (“uses oil paint, ink, and watercolor”)



[1] Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst. 2003. Faceted metadata for image search and browsing. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03). ACM, New York, NY, USA, 401-408. DOI=<http://dx.doi.org/10.1145/642611.642681>

# Think-Pair-Share: SPARQL Query



- Express the following keyword search as SPARQL query:



paul otlet books



Paul Otlet > Books

Traité de Documentation  
1934

Mundaneum: Archives of Knowledge

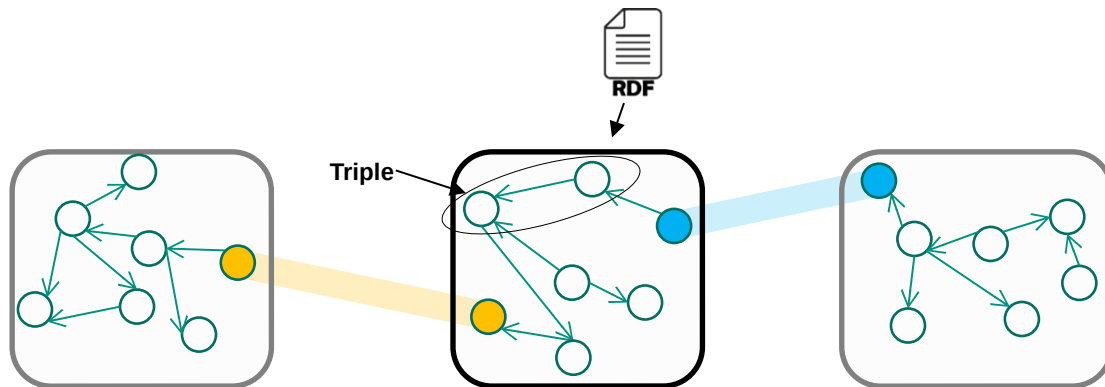
International Organisation and Disse...  
1990

# Outline

- Linked Data Systems Architectures
- Web Architecture Revisited
- Linked Data User Interfaces
- **Linked Data Querying Processors**
- Linked Data Platform (LDP)
- Raising Legacy Systems to Linked Data

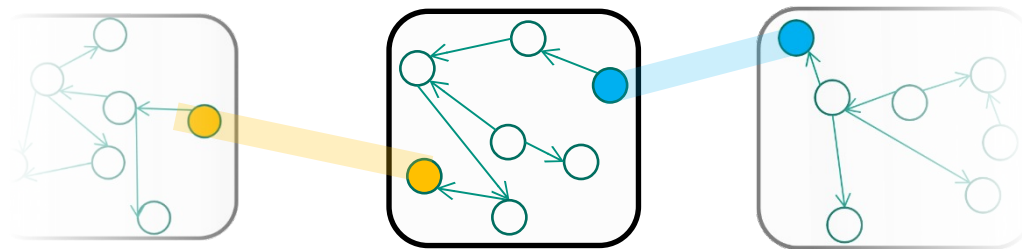
# Operating on a Fixed RDF Dataset

- Until now, both in querying and with entailment, we have assumed that the data over which we operate is fixed at the beginning of the processing.
- That is, we have assumed a fixed RDF Dataset.



# Operating on the Web as RDF Dataset

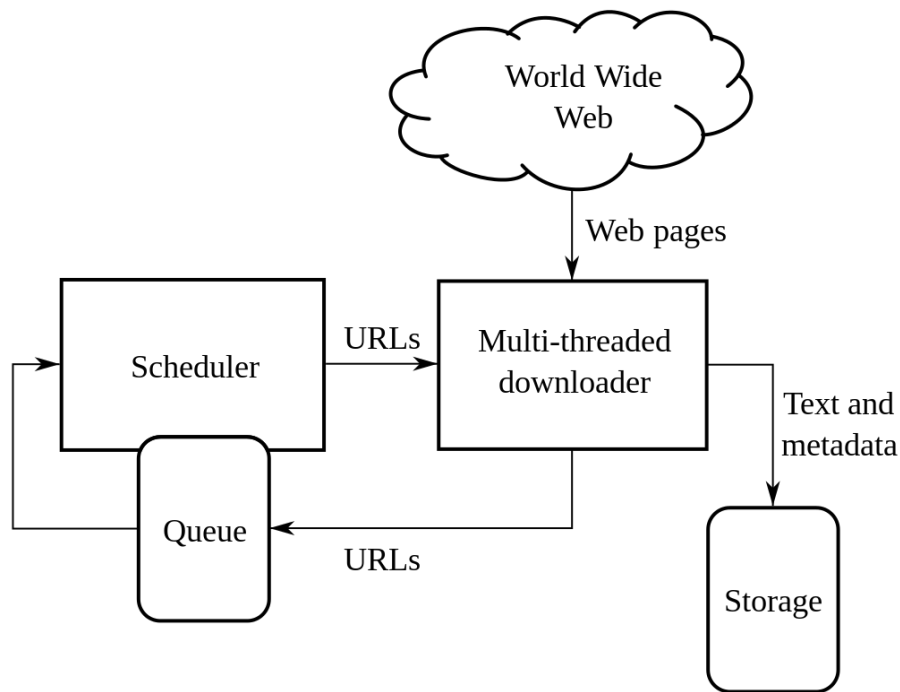
- We would like to use the entire Linked Data web, i.e., a huge RDF dataset *Web*, as basis for querying.
- But the web is too big; downloading the entire web is impractical.
- One of the core features of the web are hyperlinks.
- A user agent starts from an entry point and then follows links.
- Following links can lead to hitherto unknown servers, with unknown data of unknown schema.



- How can we specify a (finite) RDF dataset in a flexible way?

# Crawling Linked Data

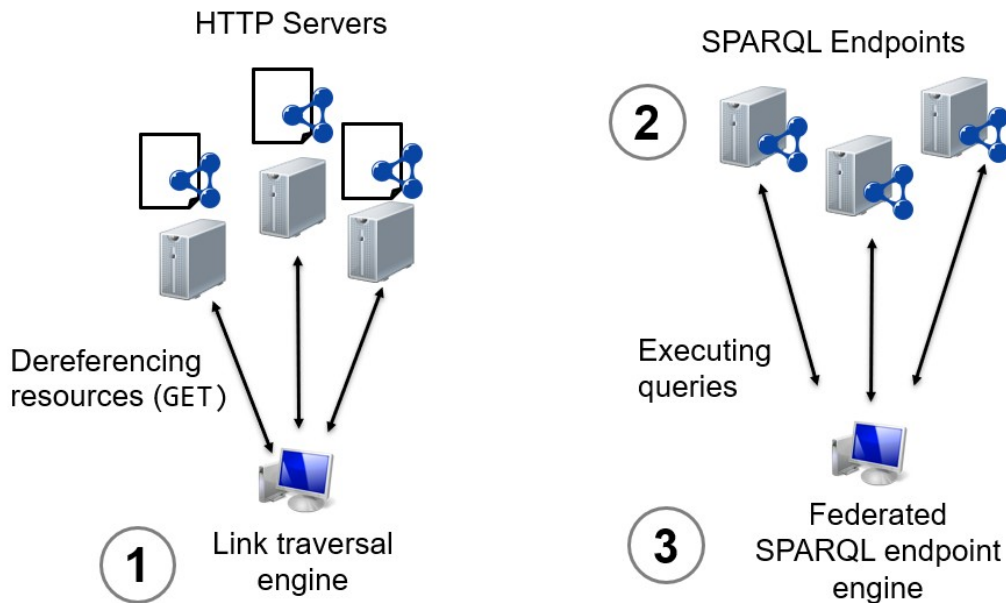
- Some applications need a way to collect Linked Data and access the data as local RDF dataset.
- Start from a set of URIs, and then follow links.
- For crawling Linked Data, see e.g., **LDSpider**<sup>1</sup>
- The result is an **NQuads** file, representing an RDF dataset that can be loaded into a SPARQL system



<sup>1</sup><https://github.com/ldspider/ldspider>

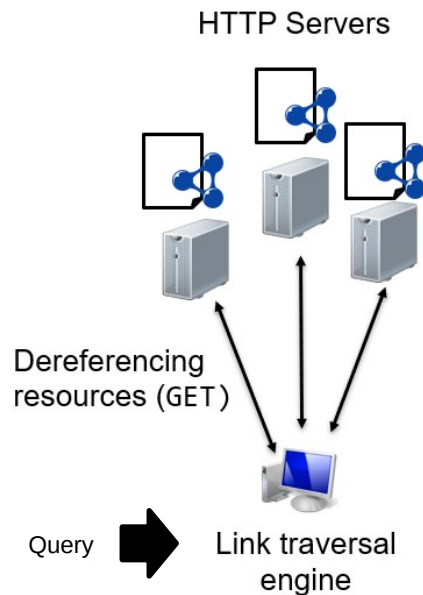
# Querying RDF Data on the Web: An Overview

1. **Link Traversal Engines** parse and combine dereferenced data to evaluate SPARQL queries.
2. **SPARQL Endpoints** can execute SPARQL queries against a local index.
3. **Federated SPARQL Endpoint Engines** execute SPARQL queries against a set of endpoints.



# Link Traversal Engines

- Dereference information resources with RDF descriptions.
- Parse and combine dereferenced data to evaluate SPARQL queries.
- Can follow (traverse) the links specified in the information resources to build more complete answers.





# Link Traversal Engines: Example (1)

- Execute the following query against the DBpedia dataset:

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?x
FROM <http://dbpedia.org/data/India.ttl>
WHERE {
    dbr:India dbo:capital ?x .
}
```

- Execution Process:
  1. Dereference the information resource
  2. Evaluate the triple pattern

# Link Traversal Engines: Example (2)

## 1 . Dereference the information resource

<http://dbpedia.org/data/India.ttl>

(prefixes)

```
dbo:percentageOfAreaWater      "9.6000003814697265625"^^xsd:float ;
dbo:capital      dbr:New_Delhi ;
dbp:capital      dbr:New_Delhi ;
dbo:leader      dbr:Sumitra_Mahajan ,
                dbr:Pranab_Mukherjee ,
                dbr:Mohammad_Hamid_Ansari ,
                dbr:Narendra_Modi ,
                <http://dbpedia.org/resource/T._S._Thakur> ;
dbo:demonym      "Indian"@en ;
dbo:flag      "Flag of India.svg" ;
dbo:longName      "Republic of India"@en ;
dbo:officialLanguage      dbr:Hindi ,
                          dbr:Indian_English ;
dbp:areaSqMi      1269346 ;
```



## Link Traversal Engines: Example (2)

### 2. Evaluate the triple pattern

`dbo:India dbo:capital ?x .`

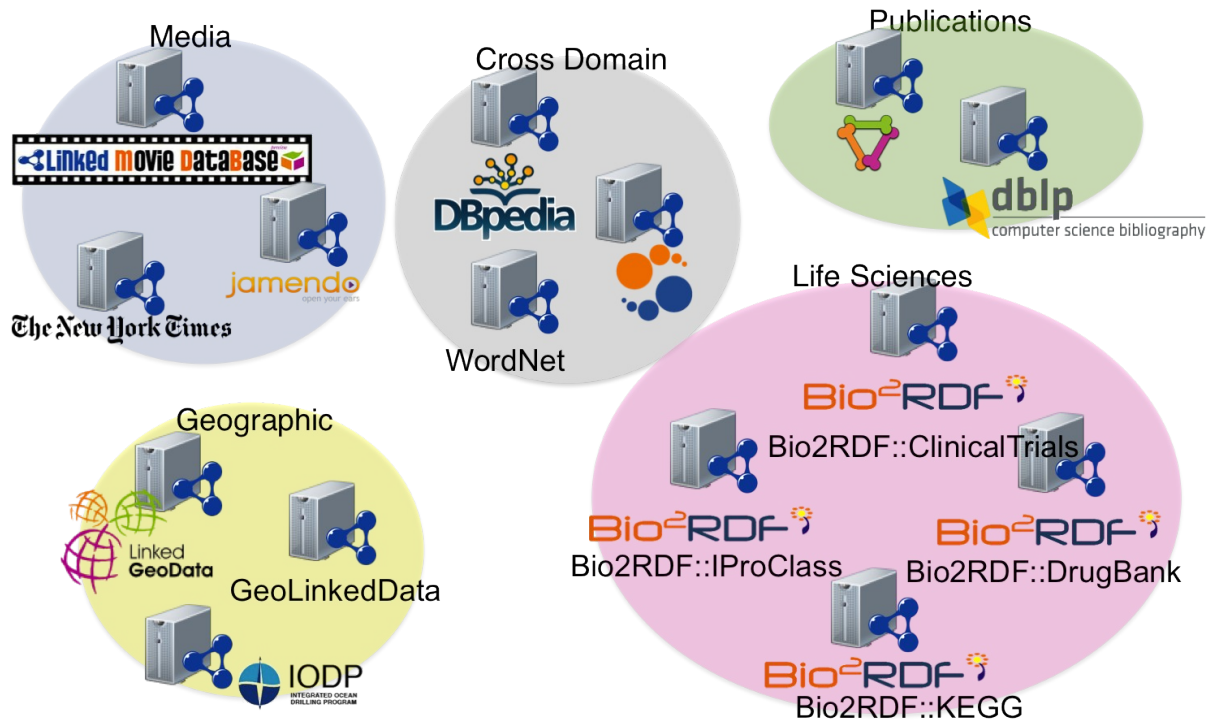
(prefixes)

```
dbo:percentageOfAreaWater    "9.6000003814697265625"^^xsd:float ;
dbo:capital      dbr:New_Delhi ;
dbp:capital      dbr:New_Delhi ;
dbo:leader       dbr:Sumitra_Mahajan ,
                 dbr:Pranab_Mukherjee ,
                 dbr:Mohammad_Hamid_Ansari ,
                 dbr:Narendra_Modi ,
                 <http://dbpedia.org/resource/T._S._Thakur> ;
dbo:demonym      "Indian"@en ;
dbo:flag         "Flag of India.svg" ;
dbo:longName     "Republic of India"@en ;
dbo:officialLanguage  dbr:Hindi ,
                     dbr:Indian_English ;
dbp:areaSqMi     1269346 ;
```



# SPARQL Endpoints

- Servers that can execute SPARQL queries.
- Not all datasets provide a SPARQL endpoint.
- Over **500 SPARQL endpoints** available on the Web.



# Think-Pair-Share: SPARQL Endpoint Example



## Virtuoso SPARQL Query Editor

[About](#) | [Namespace Prefixes](#) | [Inference rules](#) | [SPARQL](#)

Default Data Set Name (Graph IRI)

Query Text

```
SELECT ?z WHERE {  
  dbr:India dbp:capital ?z .  
}
```

(Security restrictions of this server do not allow you to retrieve remote RDF data, see [details](#).)

Results Format:

Execution timeout:  milliseconds (values less than 1000 are ignored)

Options:

☒ Strict checking of void variables ☐

☐ Log debug info at the end of output (has no effect on some queries and output formats)

(The result can only be sent back to browser, not saved on the server, see [details](#))

- Go to <http://dbpedia.org/sparql>

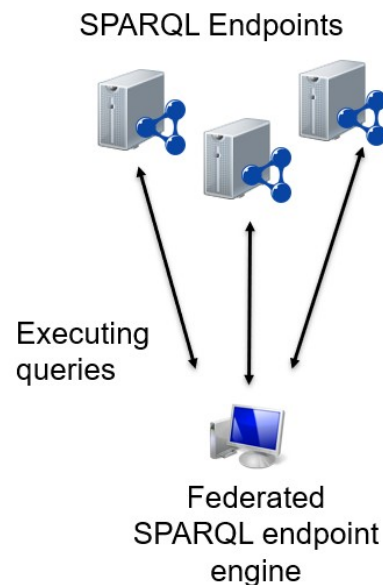
- Tasks:
  - Look for the population of Munich, London and Tokyo



<b>z</b>
<a href="http://dbpedia.org/resource/New_Delhi">http://dbpedia.org/resource/New_Delhi</a>

# Federated SPARQL Endpoint Engines

- Execute SPARQL queries against a set of endpoints.
- Endpoints can be manually specified by the user in the query using the SERVICE clause (from SPARQL 1.1<sup>1</sup>).
- If the user does not specify the endpoint, then the engine selects relevant endpoints:
- Given a user query and a set of endpoints, the engine selects the sources that contain data to answer the query.
- Combine the query answers from each endpoint to produce the final results.



<sup>1</sup><https://www.w3.org/TR/sparql11-overview/>

# Federated SPARQL Endpoint Engine Example A

- Specify the SPARQL endpoint to query with the BGP using SERVICE

```
@PREFIX foaf:<http://xmlns.com/foaf/0.1/>
@PREFIX
geonames:<http://www.geonames.org/ontology#>

SELECT ?name ?location WHERE {
  SERVICE <http://lmdb.org/sparql> {
    ?artist foaf:based_near ?location .
  } .
  SERVICE <http://dbpedia.org/sparql> {
    ?location geonames:parentFeature ?germany .
    ?germany geonames:name 'Federal Republic of
Germany' .
  }
}
```

# Federated SPARQL Endpoint Engine Example B

- Example: If endpoints are not specified by the user, the federated SPARQL engine selects them using source selection.
- Linked Movie DataBase may have solutions for the blue Triple Pattern
- DBPedia might have solutions for the purple Triple Patterns
- The query engine needs to have information about different SPARQL endpoints
- After source selection, see SERVICE

```
@PREFIX foaf:<http://xmlns.com/foaf/0.1/>
@PREFIX geonames:<http://www.geonames.org/ontology#>

SELECT ?name ?location WHERE {
  ?artist foaf:based_near ?location .
  ?location geonames:parentFeature ?germany .
  ?germany geonames:name 'Federal Republic of
Germany' .
}
```



# Outline

- Linked Data Systems Architectures
- Web Architecture Revisited
- Linked Data User Interfaces
- Linked Data Querying Processors
- **Linked Data Platform (LDP)**
- Raising Legacy Systems to Linked Data

# Use Case: The Tax Advisors (1)

- To motivate this lecture's topic, we will observe the fictional tax advisor office Paul & Sons. Any resemblance to actual institutions or people, living or dead, is purely coincidental.
- The office stores its clients' net worth on a local file system.
  - The structure looks like shown on the right.
  - advisors/ contains information about the office's advisors.
  - clients/ contains information about the office's individual clients.
  - nw1/ is one of several folders that represent a net worth that can be owned by one or more clients.
    - advisor\_notes/ is a place where each advisor can store personal notes.
    - assets/ and liabilities/ list all assets and liabilities that contribute to the net worth respectively.
- On the file system, the office can apply the basic operations of persistent storage: Create, Retrieve, Update, Delete (CRUD)



## Use Case: The Tax Advisors (2)

- Each time the assets or liabilities of a client change, the client must inform the office and one employee must create a new file on the local file system. A copy of the new file is sent back to the client to receive verification.
- This workflow takes a long time and many working hours. To improve efficiency, CEO Paul decides to create an HTTP API that allows clients to manage their net worth themselves. Furthermore, some advisors work in home office and could perform their work using the API as well.
- Paul & Sons bought the domain paul-and-sons.com for their Web service.
- In the following sections, we will examine approaches Paul took to finally build a satisfying API via Basic Container.

# Use Case: The Tax Advisors (RESTful Solution)

- With knowledge on HTTP methods and REST Paul can now design a first API for Paul & Sons's storage system:
  1. Every resource is represented in JSON.
  2. Each former folder is now implemented as a collection resource, i. e. a POST request on a collection adds a new resource to the collection.
  3. The server uses HTTPS for communication security.
  4. Basic Authentication is used for user authentication.
  5. Authorization is managed by a database.
  6. The file hierarchy is directly mapped to URIs, e. g. asset 1 of John Z. Smith has now the URI <https://paul-and-sons.com/nw1/assets/a1>.
- Paul asks his friend Tim to evaluate his design. His friend Tim points out some flaws:
  1. The custom behavior of the collection resources and security measures requires an expert to be set up which Paul & Sons has currently not at hand.
  2. The resources provide links to satisfy HATEOAS. However, the link relations are not clear without further knowledge. To solve the problems, Tim advised Paul to checkout the Linked Data Platform

# Linked Data Platform (1)

- Linked Data Platform (LDP) specifies an HTTP interface for servers that allows user agents to perform CRUD operations on Linked Data and other resources.
- LDP is a [W3C Recommendation](#), enabling uniform interactions with servers.
- A server implementing the LDP standard is called LDP server.
- LDP follows the Linked Data principles, i. e. every resource on a LDP server is somewhere described in RDF.
- Given the nature of Linked Data, LDP servers are RESTful.
- The standard specifies interactions and the description of resources.

## Linked Data Platform (2)

- The standard specifies interactions and the description of resources.
- Some freedom is given so implementations may differ to a certain degree.
- Resources on the server are organized like a file system.
  - Directory  $\Leftrightarrow$  LDP Container
  - File  $\Leftrightarrow$  LDP Resource

**Note:** In the following, if not stated otherwise, LDP Resources are called Resources and LDP Container are called Container.

# LDP Terminology (1)

There is a bunch of terms used on LDP following a short list. A more detailed descriptions follow.

- LDP: The Linked Data Platform standard.
- LDP Server: A server implementing the Linked Data Platform standard.
- LDP Resource (*ldp:Resource*): Every resource that is managed by a LDP server. Every LDP Resource is an information resource.
- LDP RDF Source (*ldp:RDFSource*): A LDP Resource with an RDF representation.
- LDP Non-RDF Source (*ldp:NonRDFSource*): A LDP Resource whose representation is not RDF.

## LDP Terminology (2)

- LDP Container (*ldp:Container*): A collection resource whose API follows LDP.
- LDP Basic Container (*ldp:BasicContainer*): A LDP container, only managing containment triples.
- LDP Direct Container (*ldp:DirectContainer*): A LDP container, managing containment triples and membership triples where both point to the same object.
- LDP Indirect Container (*ldp:IndirectContainer*): A LDP container, managing containment triples and membership triples where the later can have an object that is a non information resource.



## LDP Terminology (3)

- Server-managed triple: A triple that is created, updated or deleted by the server. Those actions may be triggered by a user's interaction with Resources.
- Containment triple (P = `ldp:contains`): A server-managed triple expressing that a Resource is contained in a certain Container.
- Membership triple: A server-managed triple created when a new resource is added to an `ldp:DirectContainer` or `ldp:IndirectContainer`.

**Note:** In the following slide the terms will be explained more closely. However, not the whole specification is covered. For more details see the W3C recommendation.

# Resources in LDP

```
/
|-- advisors /
|   |-- ad1
|   +-- ad2
|-- clients /
|   +-- johnzsmith
+-- nw1/
    |-- advisor - notes /
    |   +-- ad1
    |-- assets /
    |   |-- a1
    |   +-- a2
    +-- liabilities /
        +-- l1
```

- LDP Resources are managed by the server, meaning they may be read, created, updated and deleted through the HTTP API of the LDP server.
- Each Resource is identified by an HTTP URI.
- Resources are not limited to RDF but can be any kind of data, e. g. text, pictures, binaries and so on.

**Note:** In the following slide the terms will be explained more closely. However, not the whole specification is covered. For more details see the W3C recommendation.

# HTTP API of LDP Resources

## ■ Methods

- GET: (MUST) Get a representation of the Resource. (4.2.2.1)
- PUT: (optional) Replace the representation of the Resource with the request's body2. (4.2.4.1)
- DELETE: (optional) Remove the Resource. MAY trigger change in server-managed triples.
- PATCH: (optional) Allows implementation-specific update of the Resource's representation.
- POST: Only valid on Containers (more on this topic later).

## ■ Headers

- Resources MUST include a Link header in the response to any request: (4.2.1.4)
  - Link: <http://www.w3.org/ns/ldp#Resource>; rel="type"
  - Note: This is not equivalent to a Resource containing the triple <> a ldp:Resource .
- Each response must include an ETag header. (4.2.1.3)

# Sub-Types of LDP Resources

## ■ Sub-types of Resources

- RDF Sources are information resources that MUST provide an RDF representation. (4.3.1.4)
- Non-RDF Sources may not be able to fully express their state using RDF. (4.4.1.1)
- A POST on this kind of Container creates a new resource and adds a containment triple.

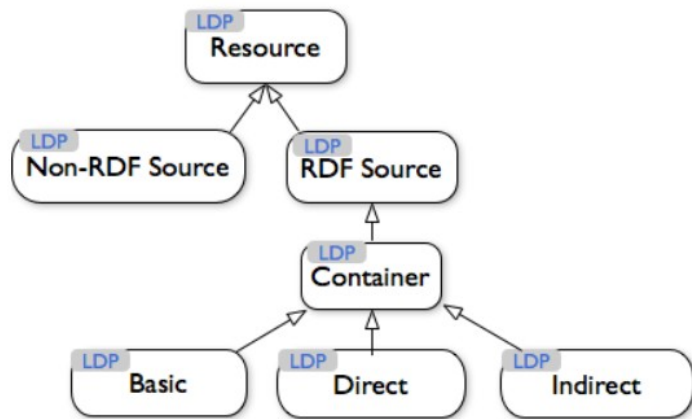
## ■ RDF Sources

- LDP servers must implement content negotiation for RDF Sources and provide Turtle (4.3.2.1) and JSON-LD (4.3.2.3) representations.

## ■ Non-RDF Sources

- When a Non-RDF Source is created the LDP server MAY create an additional RDF Source to describe it. In this case Non-RDF Sources respond with a Link header with rel="describedBy" to the RDF Source. (5.2.3.12)

# Container in LDP



source: sub-classes of LDPR extended to LDPCs (see recommendation)

## ■ LDPCs

- Just like a file system's directories LDP provides Containers to order resources.
- Containers are RDF Sources.
- There are three sub-types of Containers for different use cases.
- A Container's URI always ends with a slash (/).
- Like folders Containers can be nested.

# HTTP API of LDP Containers

- GET: RDF representation of the Container, including the containment triples.
- PUT: MAY only be used to create new Containers. It is illegal to update a Container's representation via PUT. (5.2.4.1)
- DELETE: Containment and membership triples regarding the deleted Resource MUST also be removed. (5.2.5.1)
- PATCH: (optional) Allows implementation-specific update of the Container's representation.
- POST: On success a new Resource is created, and a containment triple is added to the Container's representation. Depending on the sub-type of Container a membership triple may be created as well. (5.2.3.2)
  - An example is shown for each Container sub-type.

# Container Sub-Types

```
/
|-- advisors /
|   |-- ad1
|   +-- ad2
|-- clients /
|   +-- johnzsmith
+-- nw1/
|   |-- advisor - notes /
|   |   +-- ad1
|   |-- assets /
|   |   |-- a1
|   |   +-- a2
|   +-- liabilities /
|       +-- l1
+-- vocab .ttl
```

- For the following description we will assume that Paul & Sons has already implemented their API as a LDP server.
- For each example, the layout of the LDP looks like shown on the left.
- The root (<https://paul-and-sons.com/>) is a Basic Container:  
<> a ldp: BasicContainer ;  
dct: title "The online storage system of Paul and Sons ." ;  
ldp: contains  
    <advisors />, <clients />, <nw1/>, <vocab .ttl > .
- For the examples, a further prefix declaration is used:  
@prefix nw: <https :// solid .ti.rw.fau.de/fld/l4/ vocab .ttl

# Basic Container

- **LDP Basic Containers** are the simplest sub-type and are supported by all LDP implementations. The other sub-types are more complex and may not be supported.
- A POST on this kind of Container creates a new resource and adds a containment triple.



# Basic Container

- Example: advisor Adam (<https://paul-and-sons.com/advisors/ad1>) wants to add his own network. For this we need to add another Container to the root. The representation of the new net worth container should be:

```
<> a ldp: BasicContainer ;  
nw: netWorthOf <https :// paul -and - sons .com/ advisors /ad1 > .
```

- Adam performs:



- The new Container triggered the creation of a new containment triple for the root. The updated representation of <https://paul-and-sons.com/> is:

```
<> a ldp: BasicContainer ;  
dct: title "The online storage system of Paul and Sons ." ;  
ldp: contains <advisors />, <clients />, <nw1/>, <vocab .ttl >, <nw2/> .
```

# Outline

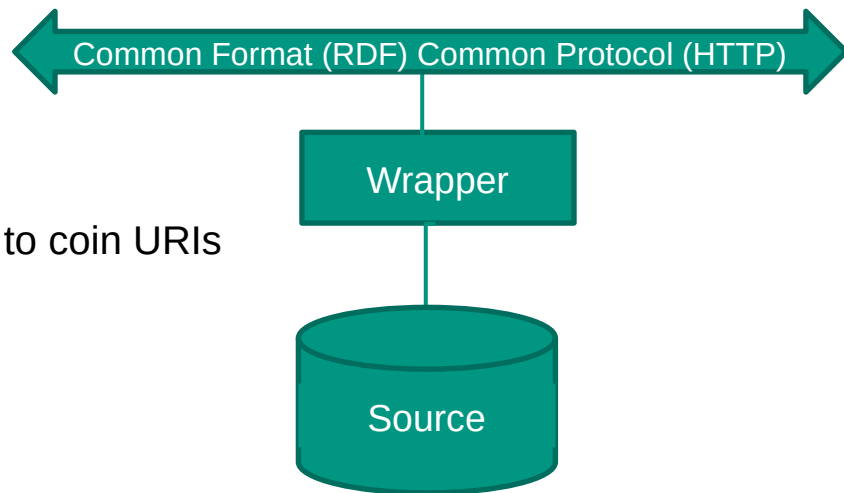
- Linked Data Systems Architectures
- Web Architecture Revisited
- Linked Data User Interfaces
- Linked Data Querying Processors
- Linked Data Platform (LDP)
- **Raising Legacy Systems to Linked Data**

# Raising Legacy Systems to Linked Data

- Given a filesystem on a server as data source:
  - Map the format (to RDF)
  - Provide the data via HTTP
  - Map the schema to a mediated schema and the instance URIs to instance URIs from other sources
  
- When dealing with multiple source systems, bring the source systems to the same level, then integrate data or specify interoperation behavior.
  - Uniform network protocol for access (read and write)
  - Uniform data representation for merging and integration
  
- Steps to consider
  1. Provide Data as RDF (Over HTTP)
  2. Map Identifiers (URIs) Between Sources

# Step 1: Provide Data as RDF (Over HTTP)

- We could just convert files to RDF and publish the RDF files at a HTTP server
  - Alternatively, we could create wrappers (adaptors, shims), which are pieces of software that translate the existing interface to a data source to another interface
- For example, data source is an API serving XML
- Wrapper provides access to data in RDF
- Wrappers for Linked Data have to provide access to RDF data via HTTP and therefore need to coin URIs
- URIs for
  - Instances
  - Schema elements (properties, classes)



## Step 2: Map Identifiers (URIs) Between Sources

- Different sources might use different URIs for
  - Instances
  - Scheme elements (properties, classes)
- Data publishers should either reuse URIs from other people or map URIs from their data to URIs from other people's data

# Think-Pair-Share: Mapping (technical aspect)



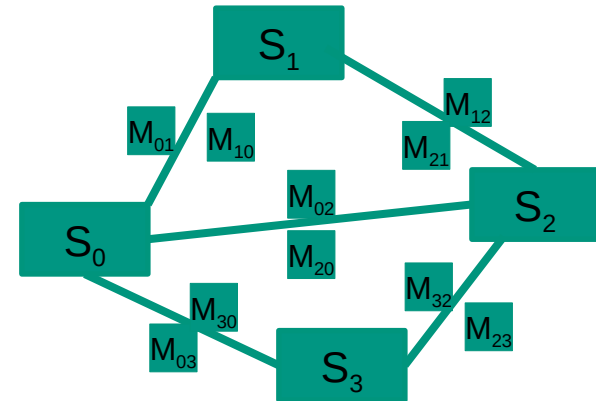
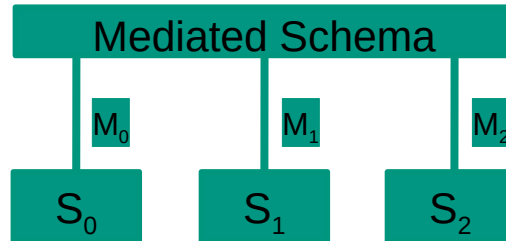
- Given two data sources A and B, how would you map URIs from source A and B?
  - Example URI from source A: `http://a.example.org/foo#bar`
  - Example URI from source B: `http://b.example.org/bar#foo`

```
@prefix a: <http://a.example.org/foo#> .  
@prefix b: <http://b.example.org/bar#> .
```

```
...
```

# Entity Matching/Object Consolidation

- Object consolidation via resource linking („smushing“)
- Object consolidation via entity linking
  - Using similarity metrics



# Think-Pair-Share: Mapping (social aspect)



- In an ideal world, everybody agrees on a common data model, syntax, semantics, interfaces and store data in a central repository
- In the real world, syntactic and semantic transformations are required
- Goal: a group of people works on integration
  - Single-person integration process vs. Multi-persons integration process
- **How to get people to agree on terminology?**



# Entity Matching: OpenRefine

- **OpenRefine** (previously Google Refine) is a spreadsheet application to clean and transform data.
  - **OpenRefine** provides a graphical user interface to edit data and allows for more complex transformation by using *Google Refine Expression Language (GREL)*
  - **OpenRefine** can be used to link and extend your dataset with various webservices
- There will be an exercise with OpenRefine

The top screenshot shows the OpenRefine interface with a dataset of 874 rows. A dialog box titled 'Add columns from reconciled column countryName' is open, showing a list of suggested properties such as 'anthem', 'basic form of government', 'capital', 'coat of arms', 'coat of arms image', 'contains administrative territorial entity', 'continent', 'country', 'currency', 'flag', 'flag image', 'head of government', 'head of state', 'highest judicial authority', and 'legislative body'. The bottom screenshot shows the same dataset with 319 matching rows. A search for 'Germany' is performed, and a dropdown menu is shown with various German entities, including 'Germany', 'country in Central Europe', 'German Empire', 'German Republic', 'German Reich', 'German Empire in the years 1919-1933', 'German Reich', 'official name for the German nation state from 1871 to 1918', 'Germany', 'contributor of the European Parliament', 'Germany', 'Wikimedia disambiguation page', 'Burg Drachenfels', and 'castle'.

# Tarql: SPARQL for Tables



- Tarql is a command-line tool for converting CSV files to RDF using SPARQL 1.1 syntax. It's written in Java and based on Apache ARQ.
- Syntactically, a Tarql mapping is one of the following:
  - A SPARQL 1.1 SELECT query
  - A SPARQL 1.1 ASK query
  - One or more consecutive SPARQL 1.1 CONSTRUCT queries, where PREFIX and BASE apply to any subsequent query
- For more information, refer to the official [Website](#).

□ There will be a short exercise with Tarql.

# Tarql Example with KVV Stations in CSV (GTFS) [1]

## # Some CSV (excerpt)

```
stop_id,stop_name,stop_lat,stop_lon,zone_id,stop_url,location_type,parent_station
"de:07334:1714:1:1","Wörth (Rhein) Alte Bahnmeisterei","49.048742345982","8.26622538039577","","","Pde:07334:1714"
"de:07334:1714:1:2","Wörth (Rhein) Alte Bahnmeisterei","49.0484420719247","8.26673742010779","","","Pde:07334:1714"
"de:07334:1721:1:1","Maximiliansau Eisenbahnstraße","49.0373071007148","8.29789997731824","","","Pde:07334:1721"
"de:07334:1721:2:2","Maximiliansau Eisenbahnstraße","49.0371363175998","8.29896897250649","","","Pde:07334:1721"
"de:07334:1721:3:31","Maximiliansau Eisenbahnstraße","49.0375308845987","8.2966423359203","","","Pde:07334:1721"
"de:07334:1721:3:32","Maximiliansau Eisenbahnstraße","49.0375191065247","8.29674115060156","","","Pde:07334:1721"
```

## # Tarql mapping

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX opnv: <http://example.org/opnv.ttl#>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
```

```
CONSTRUCT { ?stop a opnv:Stop;
                rdfs:label ?stop_name ;
                geo:lat ?lat ;
                geo:long ?lon . }
```

```
WHERE {
  BIND(URI(CONCAT('http://example.org/stops.ttl#',
                  STR(?stop_id))) AS ?stop)
  BIND(xsd:decimal(?stop_lat) AS ?lat)
  BIND(xsd:decimal(?stop_lon) AS ?lon)
}
```

## # Resulting Turtle (excerpt)

```
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix opnv: <http://example.org/opnv.ttl#> .

<http://people.aifb.kit.edu/co1683/2020/hs-dkm/stops.ttl#de:07334:1714:1:1>
  rdf:type          opnv:Stop ;
  rdfs:label        "Wörth (Rhein) Alte Bahnmeisterei" ;
  geo:lat            49.048742345982 ;
  geo:long           8.26622538039577 .
```

[1] <https://www.kvv.de/fahrplan/fahrplaene/open-data.html>

# Learning Goals

- **G 6.1:** Given several data sources in different formats, describe the steps necessary to provide access to the sources as Linked Data.
- **G 6.2:** Explain how user agent and origin server interact.
- **G 6.3:** Compare and contrast different SPARQL query processor architectures.
- **G 6.4:** Describe the API of Linked Data Platform servers and explain what `Idp:BasicContainer` are used for.
- **G 6.5:** Create and delete resources; read and write (update) resource representations with a user agent.