Karlsruhe Institute of Technology

aifb

Distributed Knowledge Graphs

Knowledge Graphs and Linked Data

Dr. Tobias Käfer

AI4INDUSTRY SUMMER SCHOOL, VIRTUAL, 2020

Tobias Käfer

- PostDoc and Project Lead, KIT
  - Projects on
    - DKG: Distributed Knowledge Graphs (EU-COST)
    - KIGLIS: AI in smart cities/networking (BMBF)
    - TraPS: Blockchain and Linked Data (BMBF)
    - DataScore: Research Data Management (ExU)
- PhD, KIT (2018)
- Research stay, UChile (2014)
- Dipl.-Wi.-Ing., KIT (2013)
- Research stay, DERI, Ireland (2011)

- Industry Experience: IBM, Siemens, …

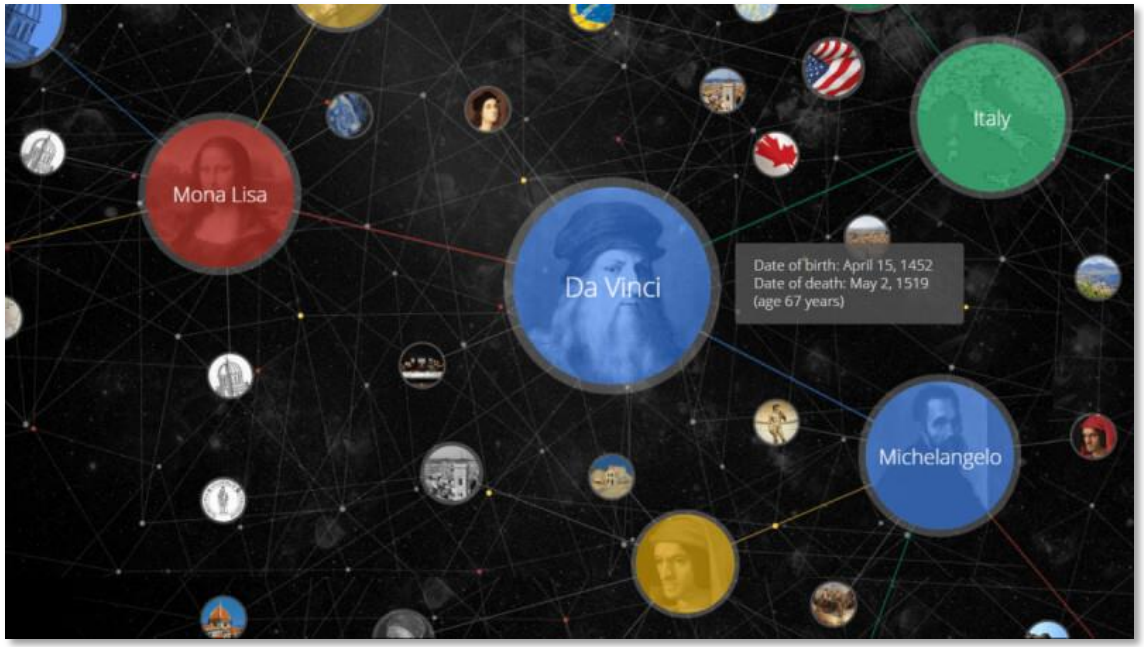
- mailto:tobias.kaefer@kit.edu

Tobias, 10 years ago

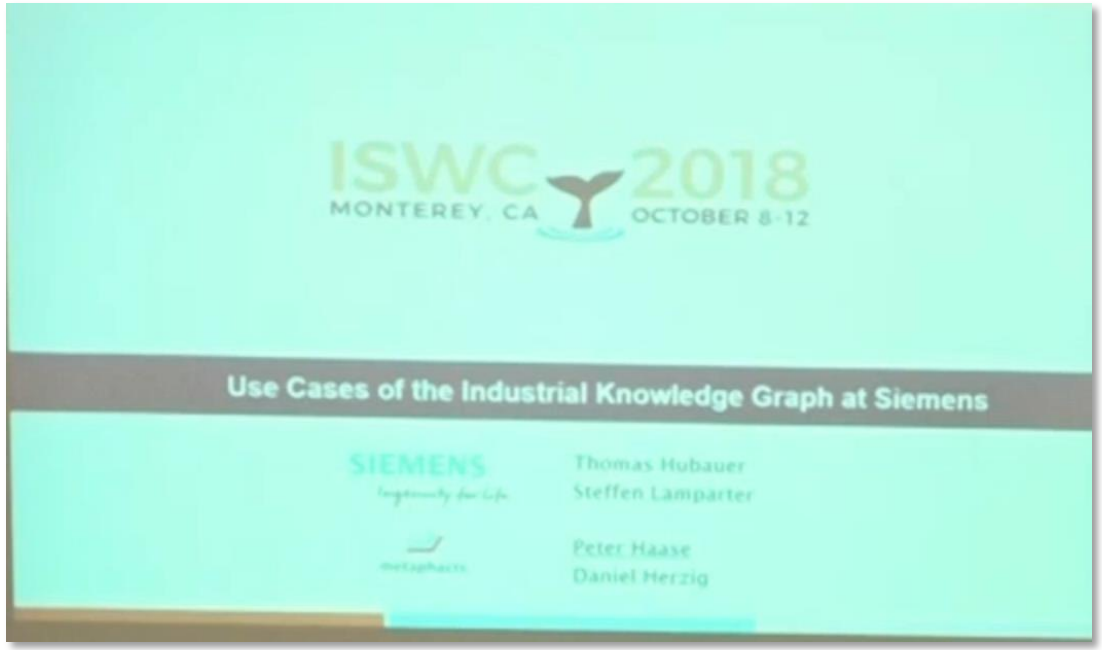
Creative Commons Licensing

- The slides have been prepared by Tobias Käfer, Andreas Harth, and Lars Heling
- This content is licensed under a Creative Commons Attribution 4.0 International license (CC BY 4.0): http://creativecommons.org/licenses/by/4.0/

What are Knowledge Graphs? – By Example



Google coined the term Knowledge Graph (2012) [1]



Siemens uses Knowledge Graphs in industrial settings [2]

[1] https://search.googleblog.com/2012/05/introducing-knowledge-graph-things-not.html http://www.google.com/insidesearch/features/search/knowledge.html (Available in the Web Archive)
[2] Hubauer, Lamparter, Haase, Herzig: Use Cases of the Industrial Knowledge Graph at Siemens. In: Proceedings of the industry track at the 17th ISWC 2018

An Inclusive Definition of a Knowledge Graph

- A Knowledge Graph is…

> *… "a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities." [1]*

[1] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, Antoine Zimmermann: "Knowledge Graphs". https://arxiv.org/abs/2003.02320 (2020)

What are Knowledge Graph Technologies?

Semantic Web Technologies

- Standardised
- Grounded in formal logic
- > 20 years history
- Built for large-scale integration of data from multiple endpoints

- Considerable adoption

Property Graph Technologies

- Typically proprietary
- Only partially formalised
- Younger
- Built to model things as graph and to access data in one endpoint

- Considerable adoption

When are Semantic Web Technologies Applied?

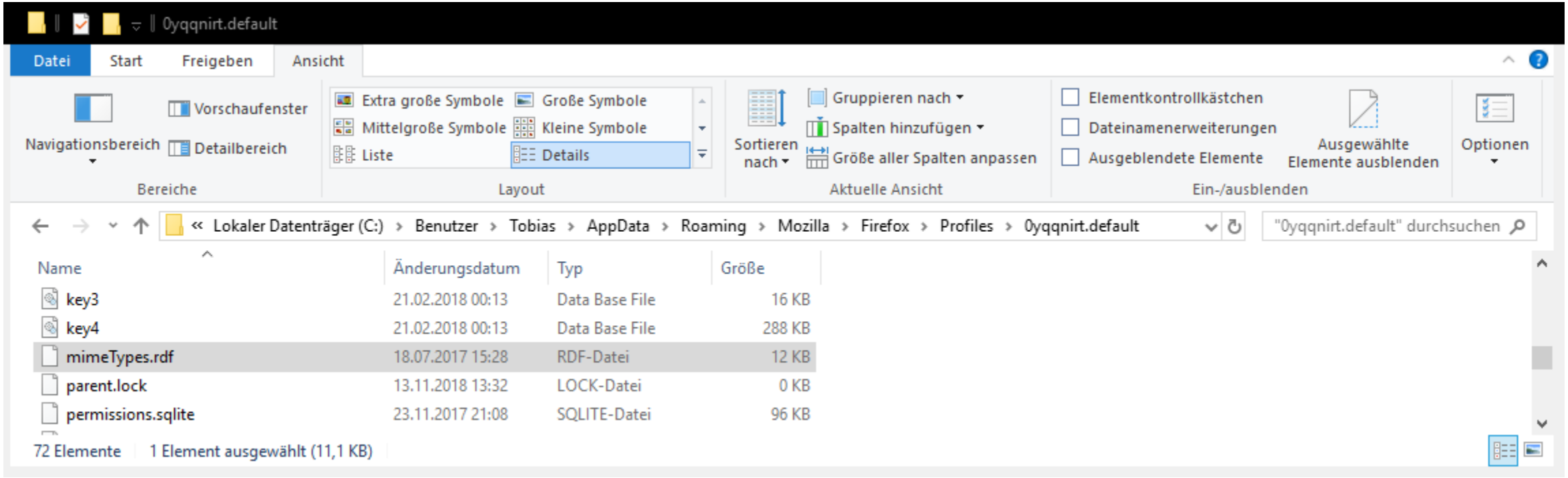
- Graph-based abstraction intuitive in many domains
- Schema heterogeneous and evolving
- Reasoning may be plugged in later
- To integrate different sources
- For data on the web
- …

Where Are All Those Semantic Technologies?

"*Semantic technology vendors […] are beginning to learn that their customers don't want to hear about ontologies, inference rules, and other nuances of the semantic technologies underlying their products. […] As a result of this dynamic,* ***semantic technologies are being absorbed into the platform and hidden from users****. This trend will continue as more and more platforms add semantic capabilities and adopt semantic standards*."

Gartner: "Finding Meaning in the Enterprise: A Semantic Web and Linked Data Primer", 2011

Are You Using Semantic Technologies?

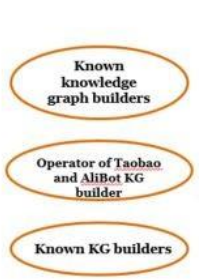


My Firefox profile folder

Who Else is Using Semantic Technologies?

Collected by Prof. Frank van Harmelen

"Who's using knowledge graphs?" Only 9 out of 10 of the most value-creating companies in the world



	Company name	Location	Industry	Change in market cap 2009-2018 ($bn)	Market cap 2018 ($bn)
1	Apple	United States	Technology	757	851
2	Amazon.Com	United States	Consumer Services	670	701
3	Alphabet	United States	Technology	609	719
4	Microsoft Corp	United States	Technology	540	703
5	Tencent Holdings	China	Technology	483	496
6	Facebook	United States	Technology	383(1)	464
7	Berkshire Hathaway	United States	Financial	358	492
8	Alibaba	China	Consumer Services	302(1)	470
9	JPMorgan Chase	United States	Financials	275	375
10	Bank of America	United States	Financials	263	307

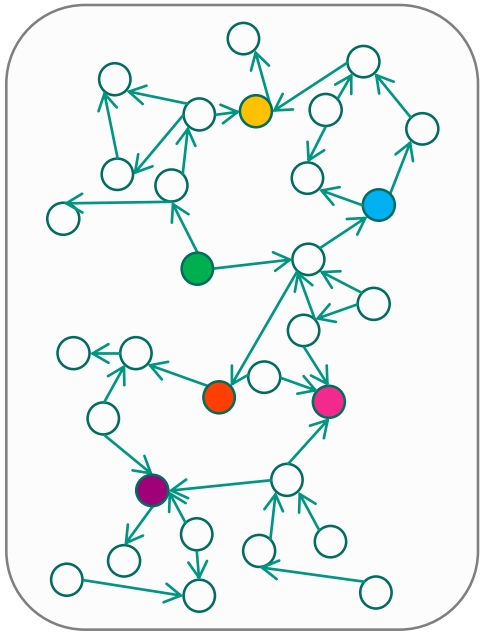
(1) Change in market cap from IPO date
(2) Market cap at IPO date
Source: Bloomberg and PwC analysis

PwC | Data-Centric architecture in business transformation 39

https://www.linkedin.com/pulse/beyond-low-code-hype-knowledge-graph-driven-alan-morrison

Three Buzzwords in Context

Knowledge Graphs



The practice of using graphs for data management

Semantic Web

The vision of intelligent agents that operate on graph-structured data on the web and understand humans

Tim Berners-Lee et al. (2001). "The Semantic Web". Scientific American. 2841 (5): 34.

Linked Data

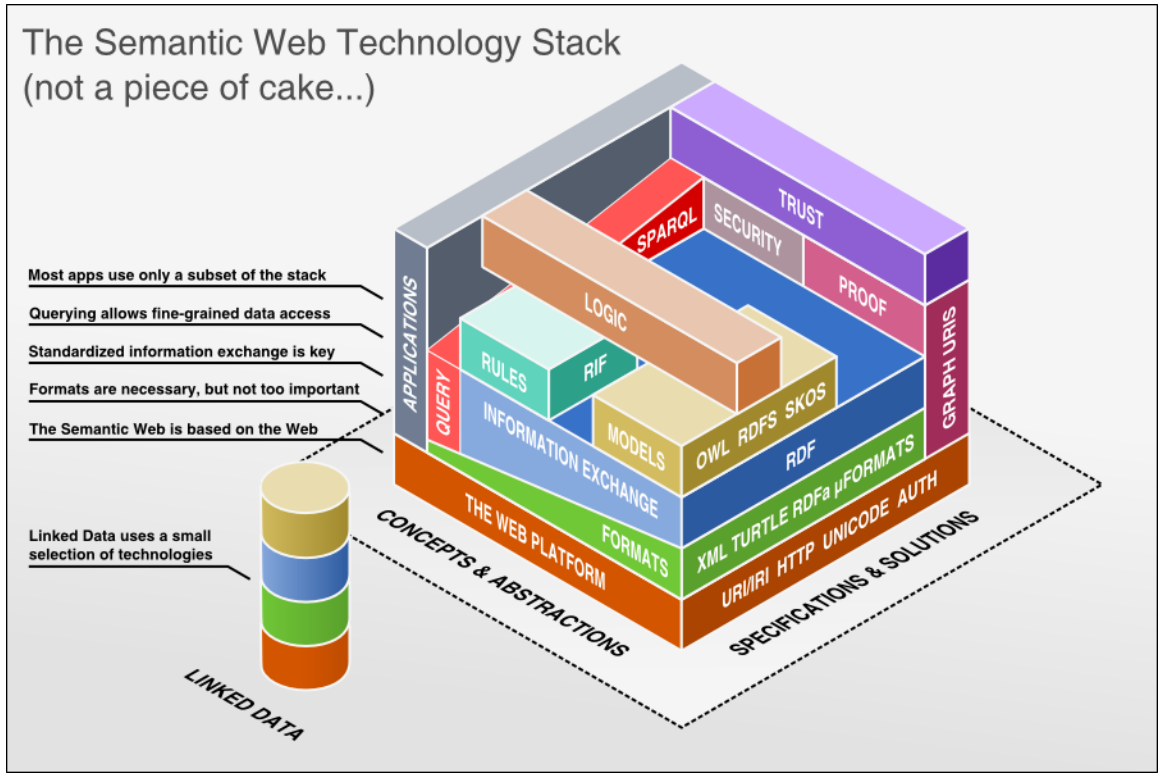
A set of practices to use Semantic Web technologies for publishing data on the web

Tim Berners-Lee presenting Linked Data. TED CC-BY-ND

The Linked Data Principles Determine Our Agenda

- Technologies from the Linked Data Principles:
  - URI
  - HTTP
  - RDF(S)
  - SPARQL
- Extensions for Write Access
- Rules for Reasoning, Link Following, and Programming

→Technologies to build systems with Distributed Knowledge Graphs



Source: http://bnode.org/media/2009/07/08/semantic_web_technology_stack.png

Linked Data Principles

- Postulated by Tim Berners-Lee in 2006.

> *"The Semantic Web isn't just about putting data on the web. It is about **making links**, so that a person or machine can explore the web of data. With linked data, when you have some of it, you can find other, related, data."* [1]

- Collection of best practices governing the publication and consumption of data on the web
- Aim: unified method for describing and accessing resources
- Later we will also see how to manipulate resource state

[1] http://www.w3.org/DesignIssues/LinkedData.html

Linked Data Principles[1]

1. Use URIs to **name *things***.
   - Things are not only documents, but also people, locations, concepts, etc.

2. Use HTTP URIs so that users can **look up** those names.
   - Users refer to humans and machine agents alike.

3. 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).

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

[1] http://www.w3.org/DesignIssues/LinkedData.html

Principle 1: Use URIs as Names for Things

- Point on a distinct resource when you share information
- Linked Data follows a *resource*-centered view of data modelling
- Resources are the basic concept of web architecture

- Example:
  - Assume we would identify a book via its ISBN (9-781497-364783)
  - Using the ISBN scheme from RFC 3187[1] we can use `urn:isbn:9-781497-364783` as resource name for the book

[1] http://ietf.org/rfc/rfc3187.txt

Compact URIs (CURIEs)

- We will work a lot with URIs, but full URIs can be unwieldy
- Thus, there is a syntax for abbreviated URIs[1] called Compact URIs, or CURIEs for short[2]
- CURIEs consist of a prefix (“namespace”) and a local reference (“local part”)
- Assume we declare the prefix `abc` with a value of `http://example.org/doc.ttl#`
- With the prefix `abc` declared, the CURIE `abc:Berlin` expands to `http://example.org/doc.ttl#Berlin`

[1] http://www.w3.org/TR/curie/

[2] CURIEs are an extension to QNames, which are used to abbreviate attribute URIs in XML documents

URIs in Relative Form

- In contrast to absolute HTTP URIs (those starting with http:// and including a hostname), HTTP URIs can also occur in relative form
- They have to be interpreted *relatively* to an absolute URI
- A URI-reference is either a URI or a relative reference[1]
- We can also use the notation known from file systems: "." refers to the current directory, while ".." refers to the parent directory[2]

Relative reference	Base URI	Resolves to the URI
`research/`	`http://example.edu/`	`http://example.edu/research/`
`./academics/`	`http://example.edu/research/`	`http://example.edu/research/academics/`
`../academics/`	`http://example.edu/research/`	`http://example.edu/academics/`
`#people`	`http://example.edu/research/`	`http://example.edu/research/#people`
	`http://example.edu/doc`	`http://example.edu/doc`

[1] http://tools.ietf.org/html/rfc3986#section-4.1
[2] for detailed technical instructions and further examples: http://tools.ietf.org/html/rfc3986#section-5.2

Principle 2: Use HTTP URIs to Allow for Lookup

- Given an identifier for a thing (URI), use HTTP as a mechanism to retrieve more information about that thing

- That is, we require some form of mapping between a
  - **URI as name** (identifying a book, a person, a place or a chemical element) and a
  - **URI as location** (identifying a machine-readable description about the book, the person, the place or the chemical element).

Principle 2: Use HTTP URIs to Allow for Lookup

- Assume we want to know more about a URI-defined resources, say for our book having the URI `urn:isbn:9-781497-364783`
- With the ISBN you can go to your local bookstore, and a clerk there can look up the ISBN in their catalogue
- Or you type the ISBN into a search box of an online bookstore or of a library, to get more information about the book
- Ultimately, there will be a query to a database of things identified via an ISBN, maintained by some organisation

Principle 2: Use HTTP URIs to Allow for Lookup

- HTTP URIs provide an inherent mechanism for lookup and unites logical and physical address
- You can type an identifier into your browser and immediately get some information back → tight connection between identifier and source
- E.g. http://www.w3.org/People/Berners-Lee/card is the URI of Tim Berners-Lee's machine-readable homepage
- No additional information or mediator is needed to access information
- Just type HTTP URI into browser and access HTML, JPEG, PNG, GIF, MP4 files – any content that can be serialised into bytes

Referencing a Resource, Dereferencing a URI

- Referencing a resource is easy: just write the URI
- But what about dereferencing?
- How do you get the referenced resource?
- What do you get?

Referencing a Resource, Dereferencing a URI[1]

- The act of retrieving a representation of a resource identified by a URI is known as **dereferencing** that URI
- Applications, such as browsers, render the retrieved representation for the user
- Most web users do not distinguish between a resource and the rendered representation they receive by accessing it
- **Information resources** associated with a **resource** need to have their own URIs
- They are themselves distinct resources and provide representations

[1] https://www.w3.org/2001/tag/doc/httpRange-14/2007-05-31/HttpRange-14

Referencing a Resource, Dereferencing a URI

- It is important to differentiate between a resource and an informational document **about** that resource[1]
- As you cannot retrieve the resource via your browser, a representation is needed

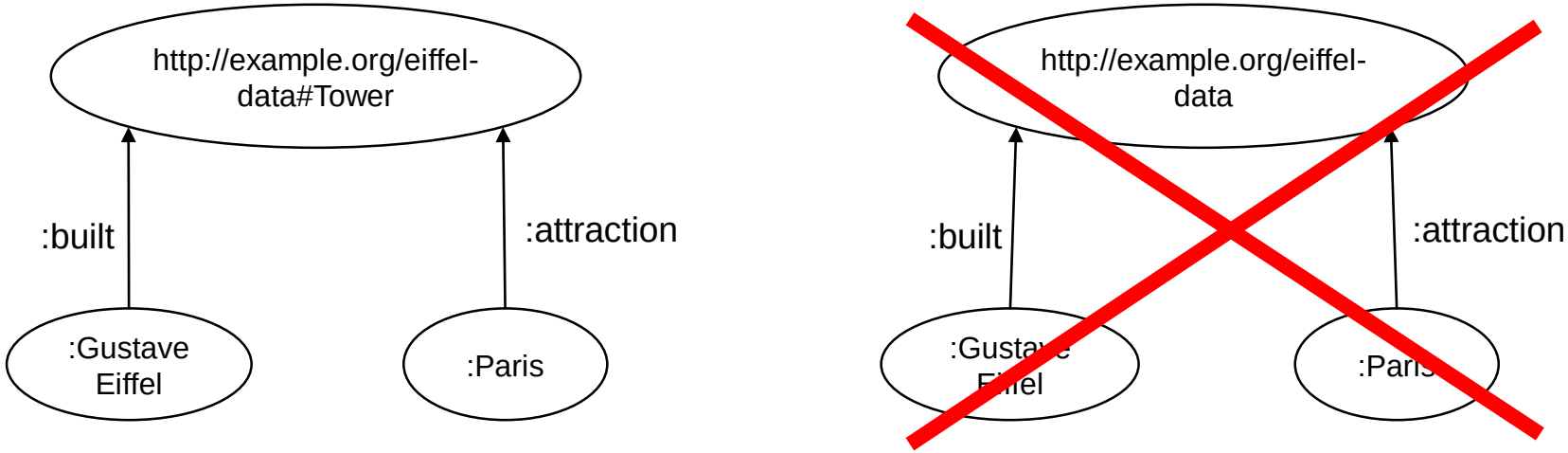
http://example.org/eiffel-data#Tower

http://example.org/eiffel-data

[1] Talking about differentiation: this is also not the Eiffel Tower itself. It is a picture of the Eiffel Tower and the picture's URI is https://upload.wikimedia.org/wikipedia/commons/thumb/8/85/Tour_Eiffel_Wikimedia_Commons_(cropped).jpg/360px-Tour_Eiffel_Wikimedia_Commons_(cropped).jpg

Referencing a Resource, Dereferencing a URI

- As the document about the resource is also a resource itself, it needs its own URI (Information Resource)
- To reference the „Eiffel Tower“, only the URI of the **“resource”** is used:



Referencing a Resource, Dereferencing a URI

- A user that wants information about a given resource might not know the URI of the describing document (the associated information resource)
- In the Semantic Web, two possibilities for providing the information resource of a resource are used: “hash URIs” and “slash URIs”

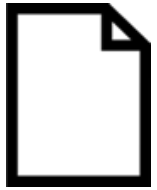
Resource vs. Information Resource

Hash URIs

- Retrieving the document‘s URI by stripping off the hash of a hash URI

`http://example.org/karlsuhe-data#Palace`

Resource



`http://example.org/karlsruhe-data`

Information Resource

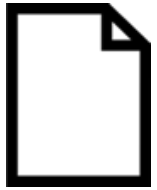
Resource vs. Information Resource

Slash URIs

- Retrieving the document‘s URI by an automated HTTP redirect (303)

`http://dbpedia.org/resource/Karlsruhe_Palace`

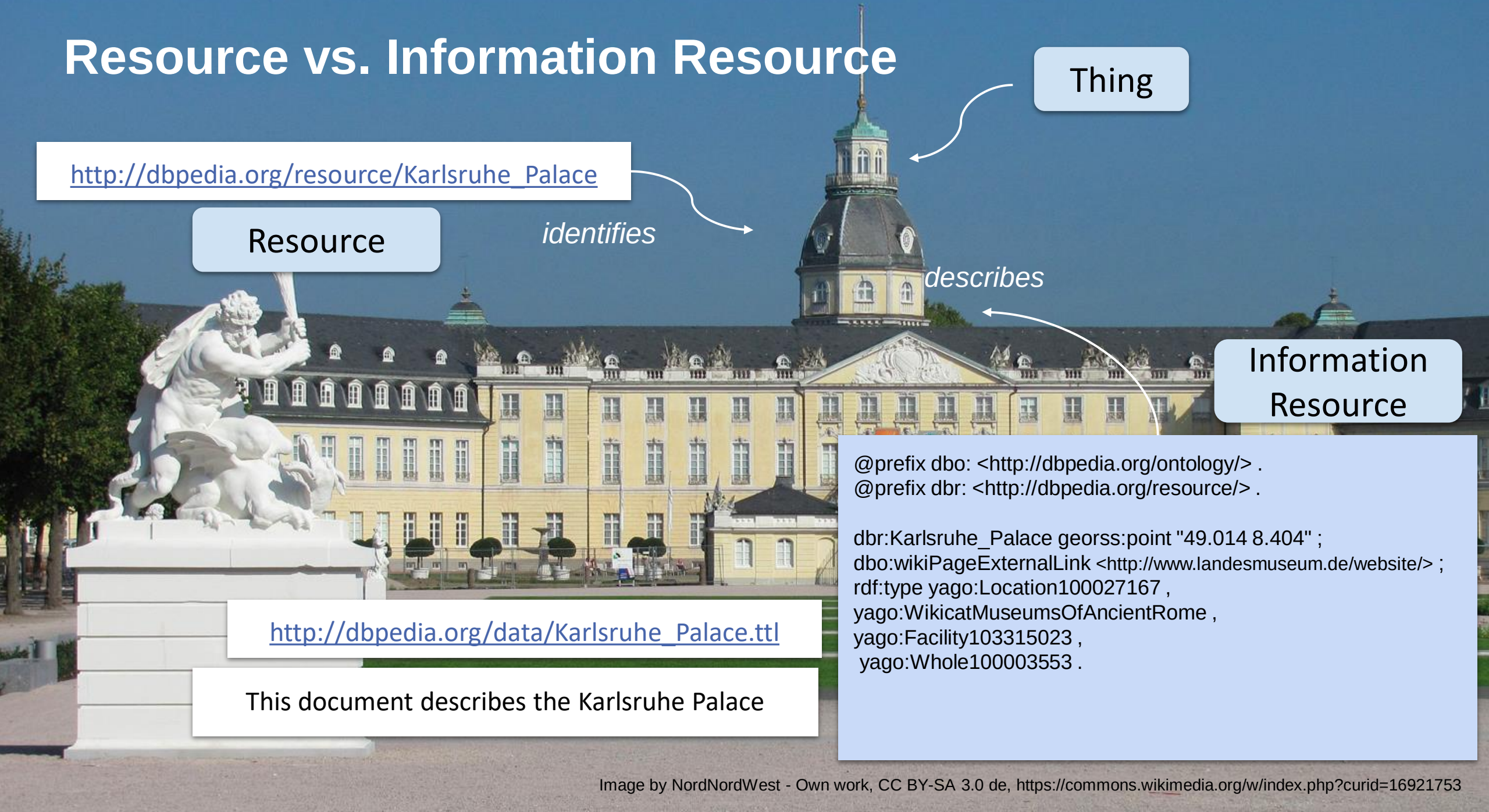
Resource



`http://dbpedia.org/data/Karlsruhe_Palace.ttl`

Information Resource

Resource vs. Information Resource



Thing

http://dbpedia.org/resource/Karlsruhe_Palace

Resource

identifies

describes

Information Resource

http://dbpedia.org/data/Karlsruhe_Palace.ttl

This document describes the Karlsruhe Palace

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

dbr:Karlsruhe_Palace georss:point "49.014 8.404" ;
dbo:wikiPageExternalLink <http://www.landesmuseum.de/website/> ;
rdf:type yago:Location100027167 ,
yago:WikicatMuseumsOfAncientRome ,
yago:Facility103315023 ,
 yago:Whole100003553 .
```

Addressing HTTP-Range 14 using Slash URIs and HTTP Content Negotiation

Let‘s try an example:

- I want to have **information** about the Karlsruhe Palace from DBpedia

1 **HTTP GET request**
Accept Header: text/html

http://dbpedia.org/**resource**/Karlsruhe_Palace

URI represents “***the name of the thing***”

2 **HTTP/2 303 See Other**

http://dbpedia.org/**page**/Karlsruhe_Palace

3 **HTTP GET request**
Accept Header: text/html

URI represents ***“the description of the thing”***

4 **HTML** Document

Image by NordNordWest - Own work, CC BY-SA 3.0 de, https://commons.wikimedia.org/w/index.php?curid=16921753

Addressing HTTP-Range 14 using Slash URIs and HTTP Content Negotiation

Let‘s try an example:

- I want to have **machine-readable information** about the Karlsruhe Palace from DBpedia

1 **HTTP GET request**
Accept Header: text/turtle

`http://dbpedia.org/resource/Karlsruhe_Palace`

URI represents “***the name of the thing***”

2 **HTTP/2 303 See Other**

`http://dbpedia.org/data/Karlsruhe_Palace.ttl`

3 **HTTP GET request**
Accept Header: text/turtle

URI represents ***“the description of the thing”***

4 **RDF** (Turtle) Document

Image by NordNordWest - Own work, CC BY-SA 3.0 de, https://commons.wikimedia.org/w/index.php?curid=16921753

Addressing HTTP-Range 14 using Slash URIs and HTTP Content Negotiation

Karlsruhe Institute of Technology

Let‘s try it ourselves:

- Retrieve **information** about the Karlsuhe Palace from DBpedia

```
curl -L -H "Accept: text/html" http://dbpedia.org/resource/Karlsruhe_Palace
```

- Retrieve **machine readable information** about the Karlsruhe Palace from DBpedia

```
curl -L -H "Accept: text/turtle" http://dbpedia.org/resource/Karlsruhe_Palace
```

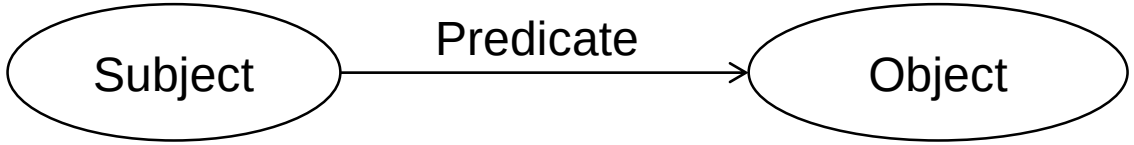
Principle 3: Provide Useful Information

- When somebody looks up a URI, provide useful information using the standards
- **RDF is the data model** for both Semantic Web and Linked Data, providing content meaningful to computational users
- You can eg. write RDF in files, store and query RDF in so-called Triple Stores (databases for RDF), or embed RDF in other formats (eg. HTML)

Resource Description Framework (RDF)

[1]

- RDF is the foundational data model for both Semantic Web and Linked Data
- RDF comes with a formal underpinning → we can mathematically define and proof things
- An RDF *triple* is the basic RDF concept describing information as a subject-property-object structure
- Property (or predicate) specifies relation between subject and object
- Triples can be visualised:



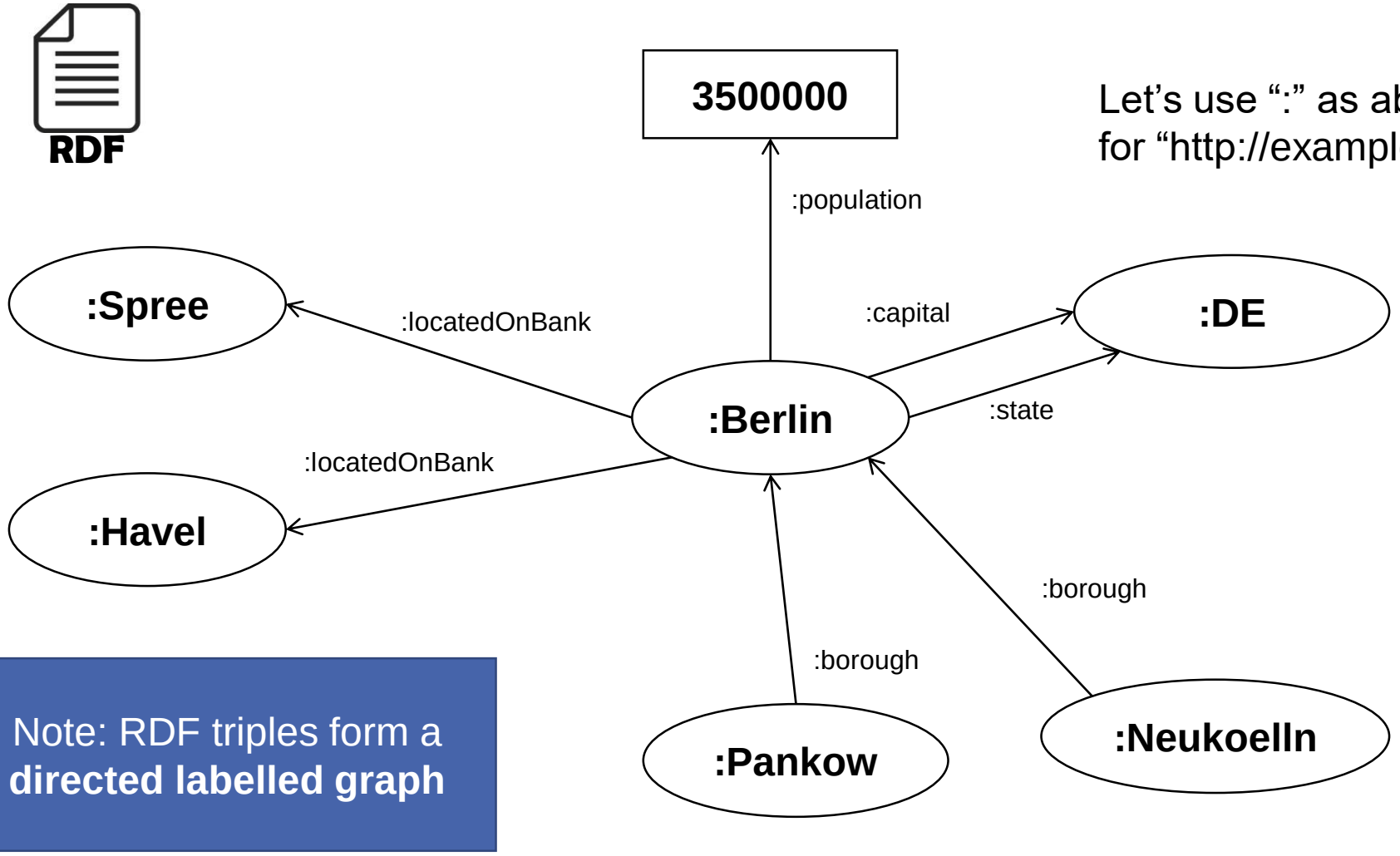
- Multiple triples form an RDF *graph*
- RDF graphs can be visualised as directed labelled graph

[1] http://www.w3.org/RDF/icons/

Facts in „Triples“

- Berlin is the capital of Germany.
- Berlin is a state of Germany.
- Berlin has a population of 3.5 Million.
- Berlin is located on the bank of the Spree.
- Berlin is located on the bank of the Havel.
- Pankow is a borough of Berlin.
- Neukölln is a borough of Berlin.

Example RDF Graph within an RDF Document



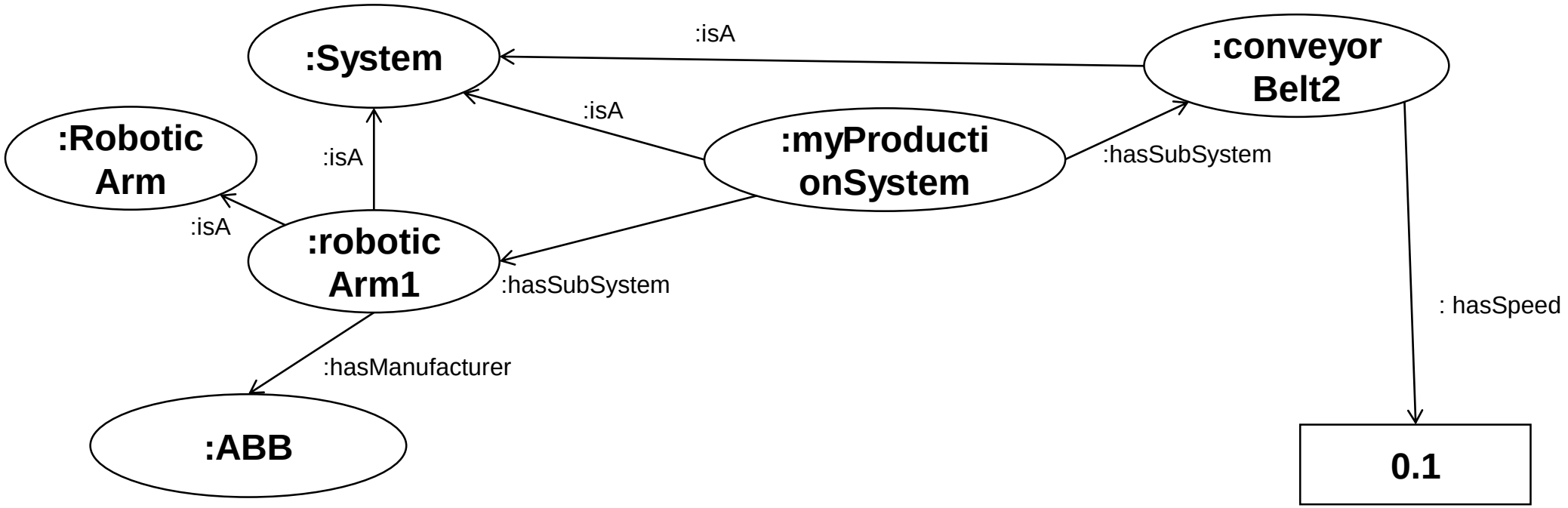
Exercise: Draw an (RDF) Graph

- Use the facts on the right →
- Identify connections, things, and values
- Depict things in circles
- Depict values in rectangles
- Depict connections using arrows
- Draw the graph on a piece of paper

- myProductionSystem is a System
- myProductionSystem has subsystem roboticArm1
- myProductionSystem has subsystem conveyorBelt2
- roboticArm1 is a System
- roboticArm1 is a RoboticArm
- roboticArm1 has manufacturer ABB
- conveyorBelt2 is a System
- conveyorBelt2 has speed 0.1

Sample Solution

Let's use ":" as abbreviation
for "http://example.org/doc.ttl#"



Principle 4: Include Links to Other URIs

- Associating things from one source to things from another source creates the mesh we will later use to perform algorithms on
- Links are required to be able to connect the separate data graphs together
- The graph-structured data model and the re-use of URIs across graphs allows for an easy merging of multiple graphs
- Central points on the web provide URIs for frequently used resources (e.g., DBpedia). Using these allows for a common understanding of descriptions and fast merging of multiple graphs

Distributed Knowledge Graphs

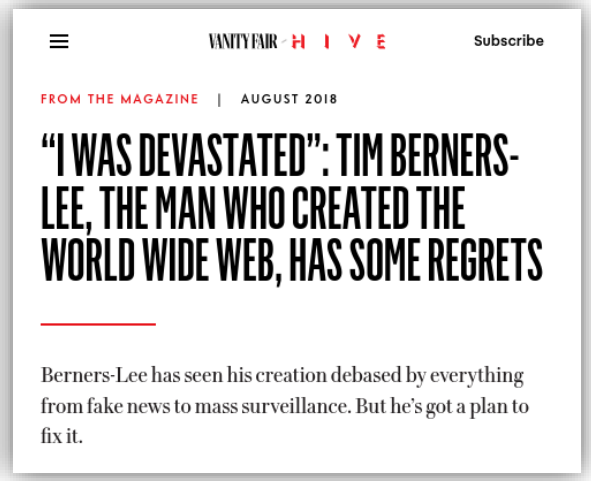
- Linked Data builds on HTTP
- Everybody can run a web server

  `python3 -m http.server`

  - Serves the contents of the working directory (which may contain RDF documents)
- vs. centralised systems of today (Facebook & Co.)

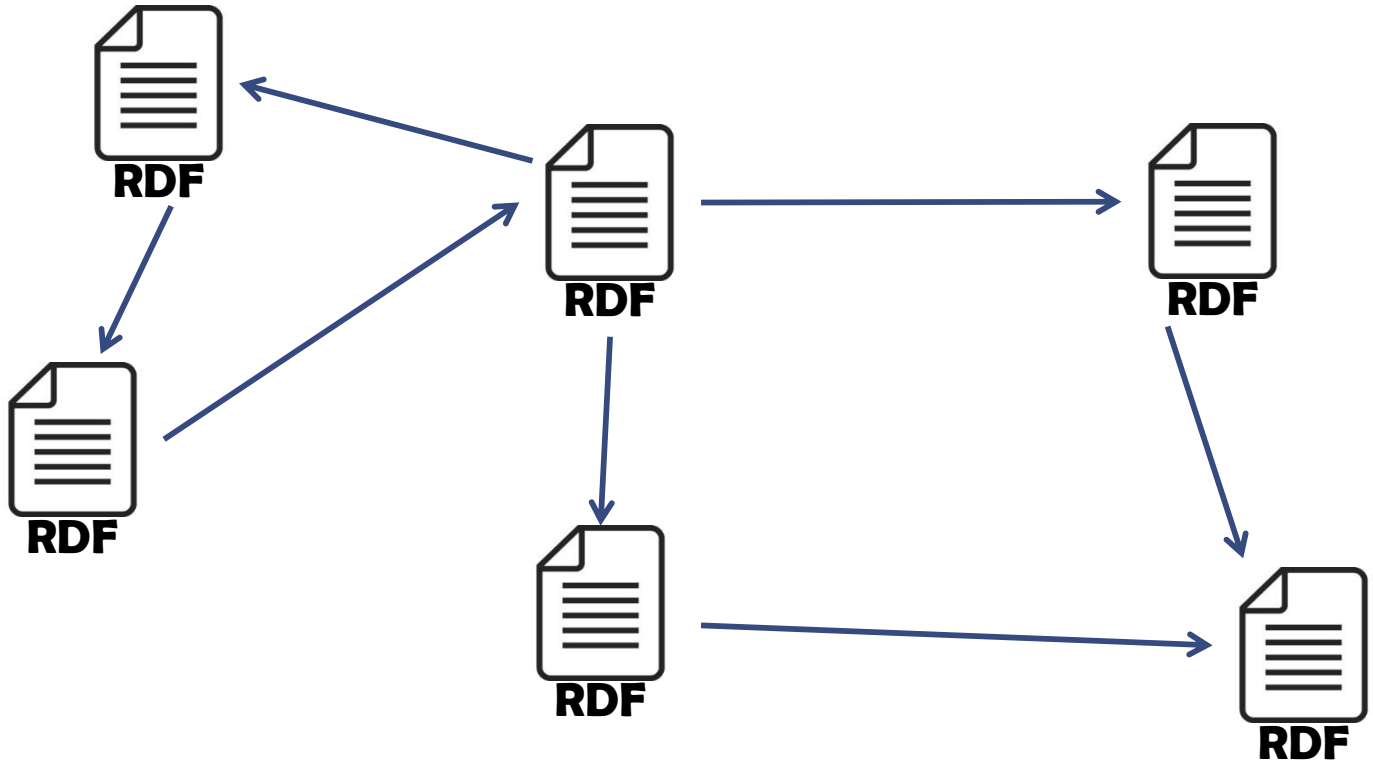
- Decentralised publishing → a distributed system

- Research challenge: Systems/algorithms/… that deal with large amounts of small interlinked RDF documents on the web

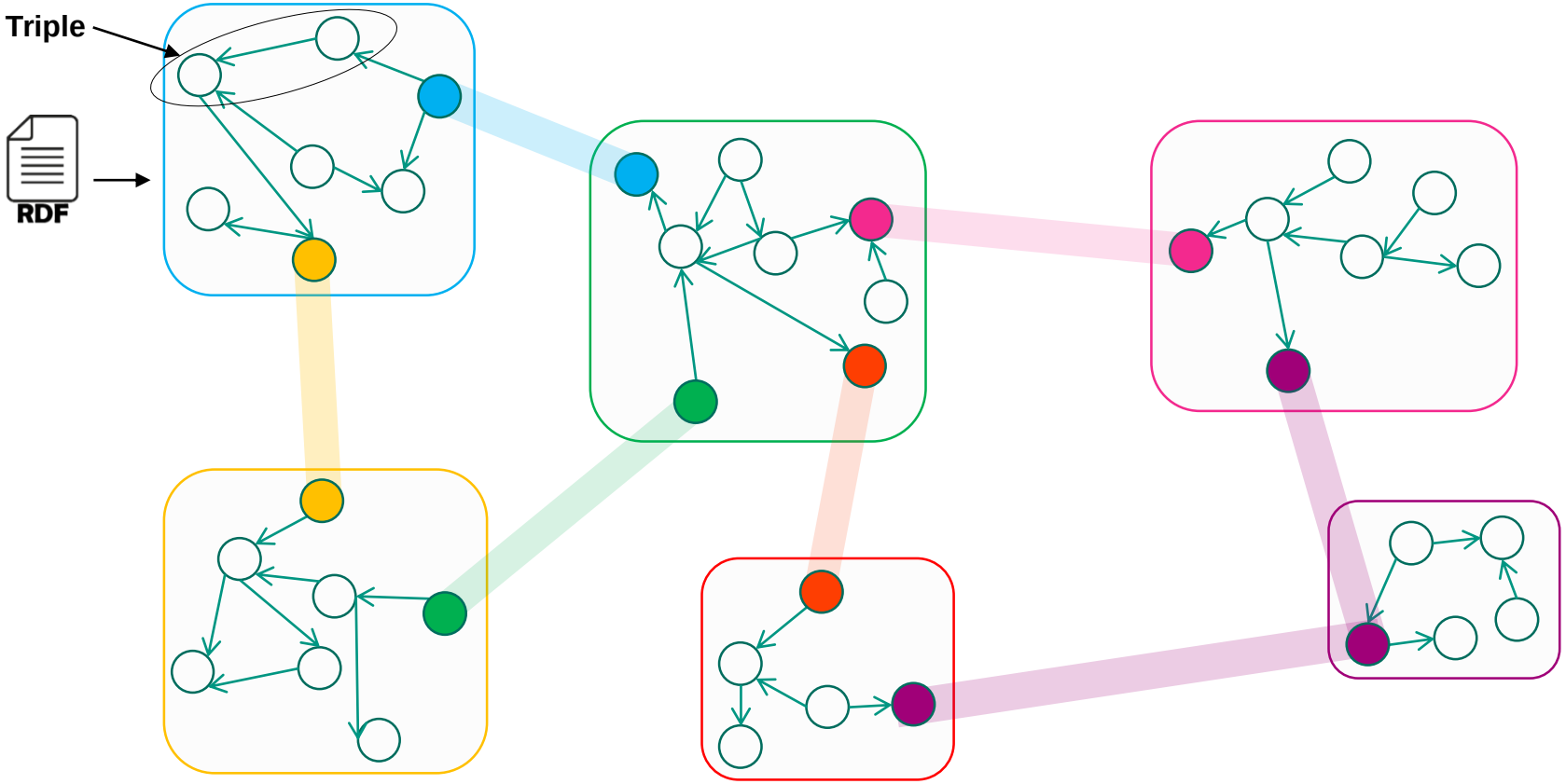


[1] K. Brooker, *Vanit. Fair*, 2018, **60**(696), 62-67.

A Web of RDF Documents…

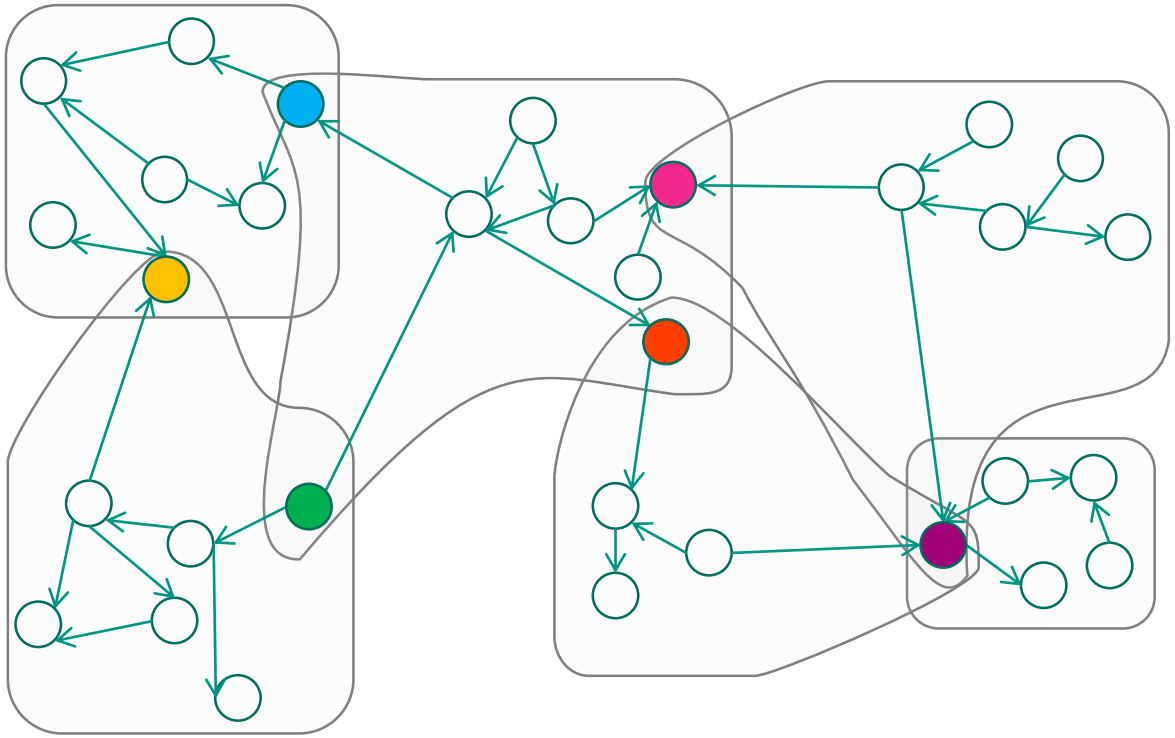


…Using URIs…

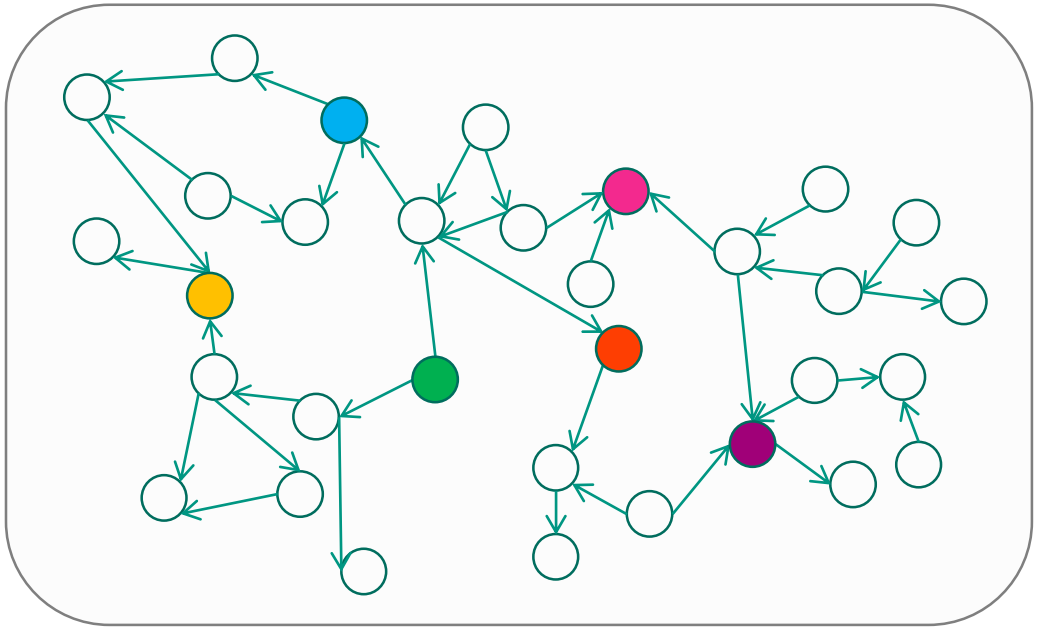


Circles with same color indicates identical resource
Documents and circles in the same color indicate correspondence between resource and information resource

…Can Actually Form…



…a Web of Data



- Each node is one resource, meaningfully linked to other resources, but acquired from different sources → Distributed Knowledge Graph