TU WIEN Informatics

188.399 Introduction to Semantic Systems

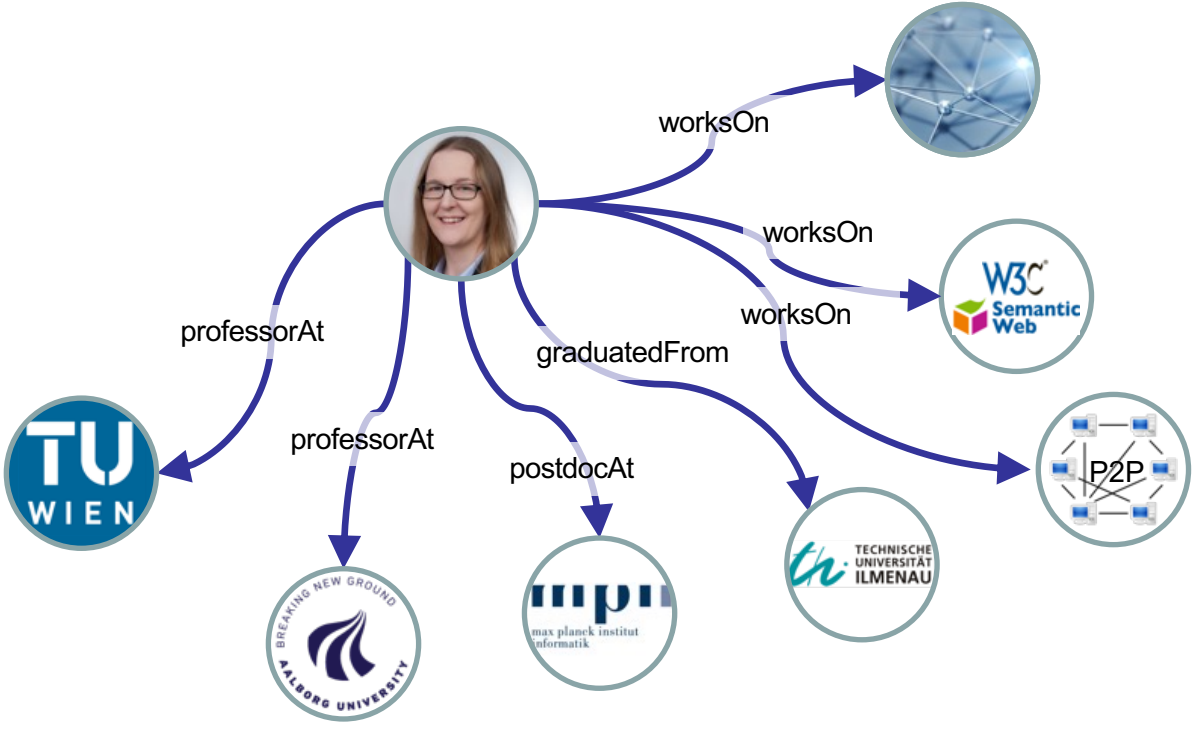
(2.0 VU / 3.0 ECTS)

2023W

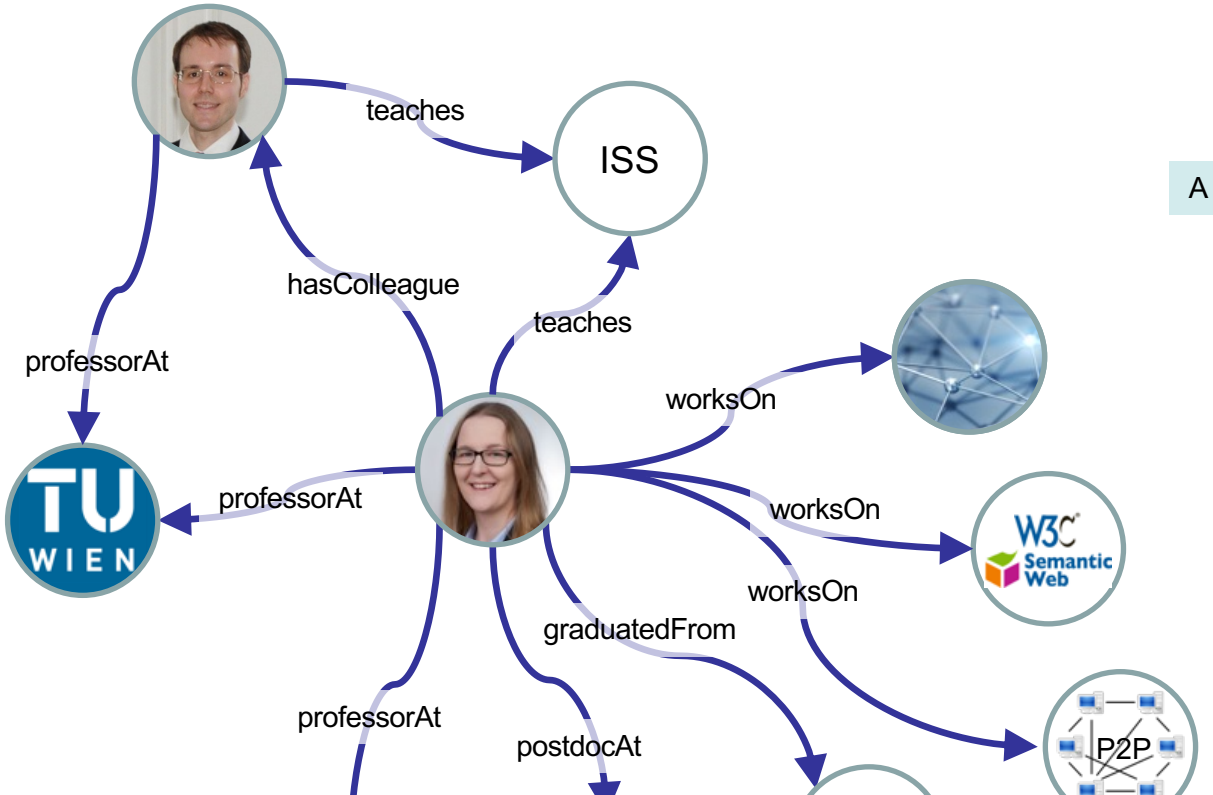
Katja Hose
Emanuel Sallinger

iss@dbai.tuwien.ac.at

Brief self introduction

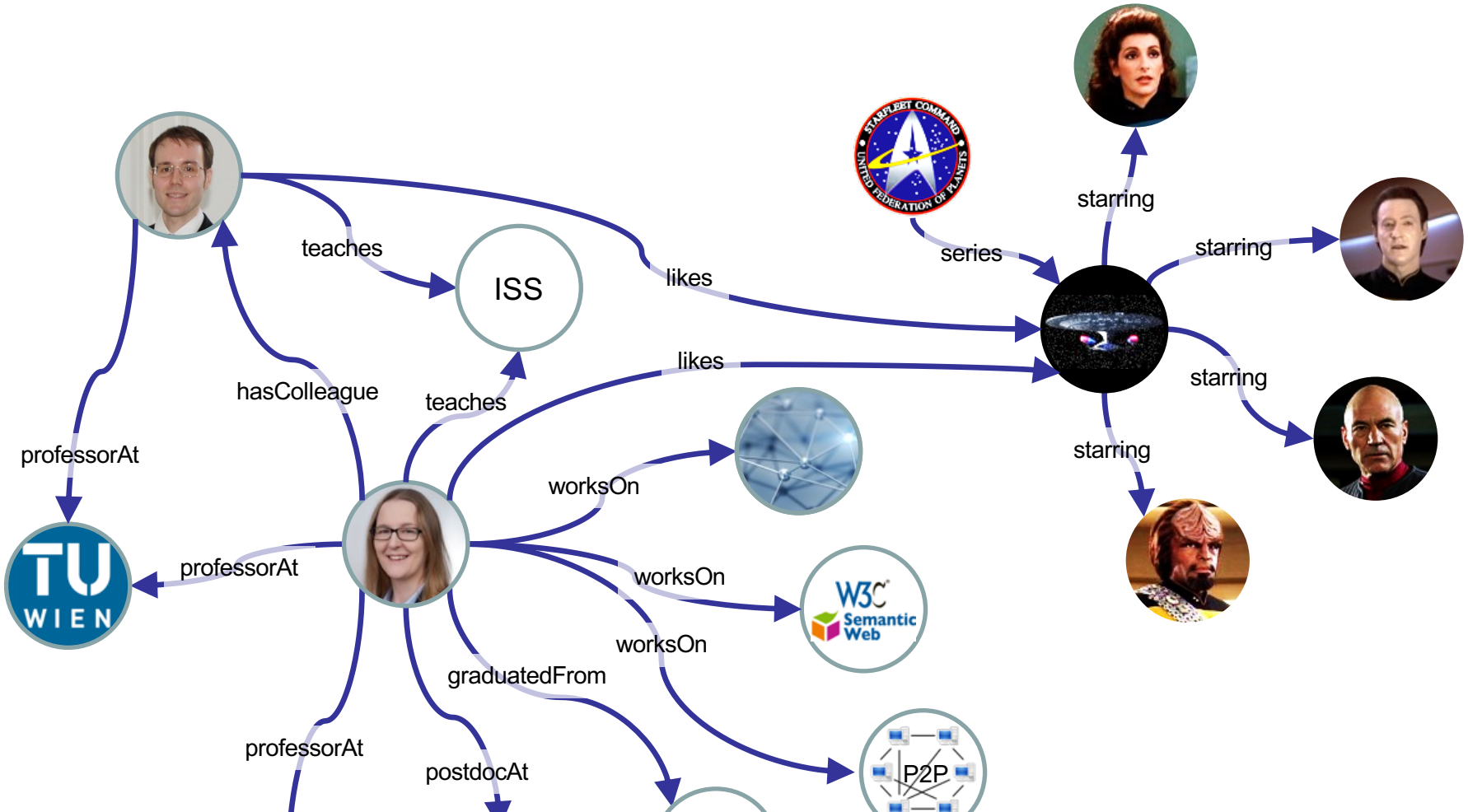


Brief self introduction



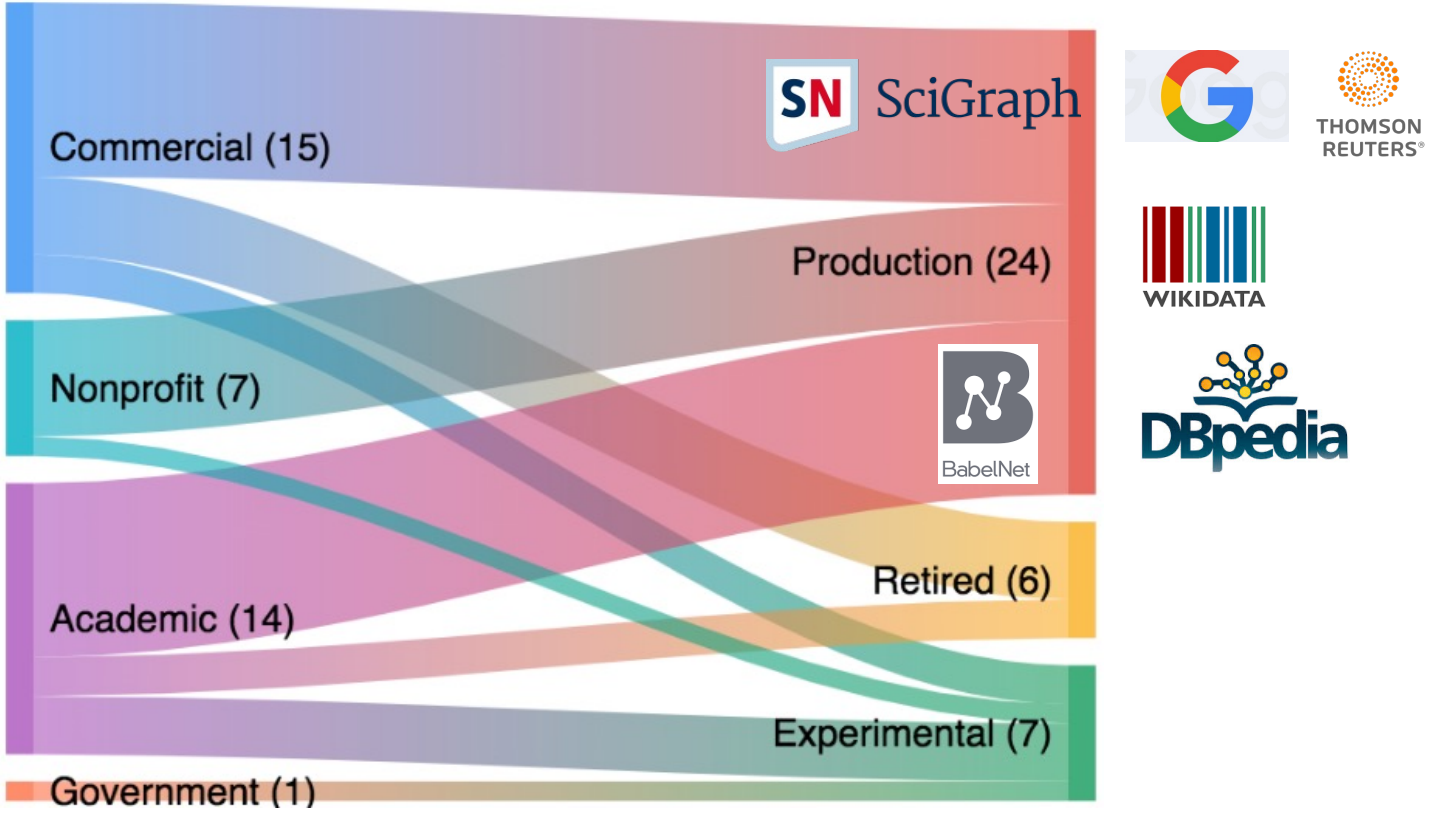
A knowledge graph is a directed labeled graph

Brief self introduction



Lecture Focus

Semantic systems: systems that make use of explicitly represented knowledge, through conceptual structures such as ontologies, taxonomies or **knowledge graphs**.



Knowledge graph proportions by publisher and status

J. P. McCusker et al.: What is a Knowledge Graph?
Sem. Web. Journal, 2018
http://www.semantic-web-journal.net/system/files/swj1954.pdf

Goals of Lecture

What are knowledge graphs (KGs)?

- Definitions
- Examples

How to use semantic technologies and KGs?

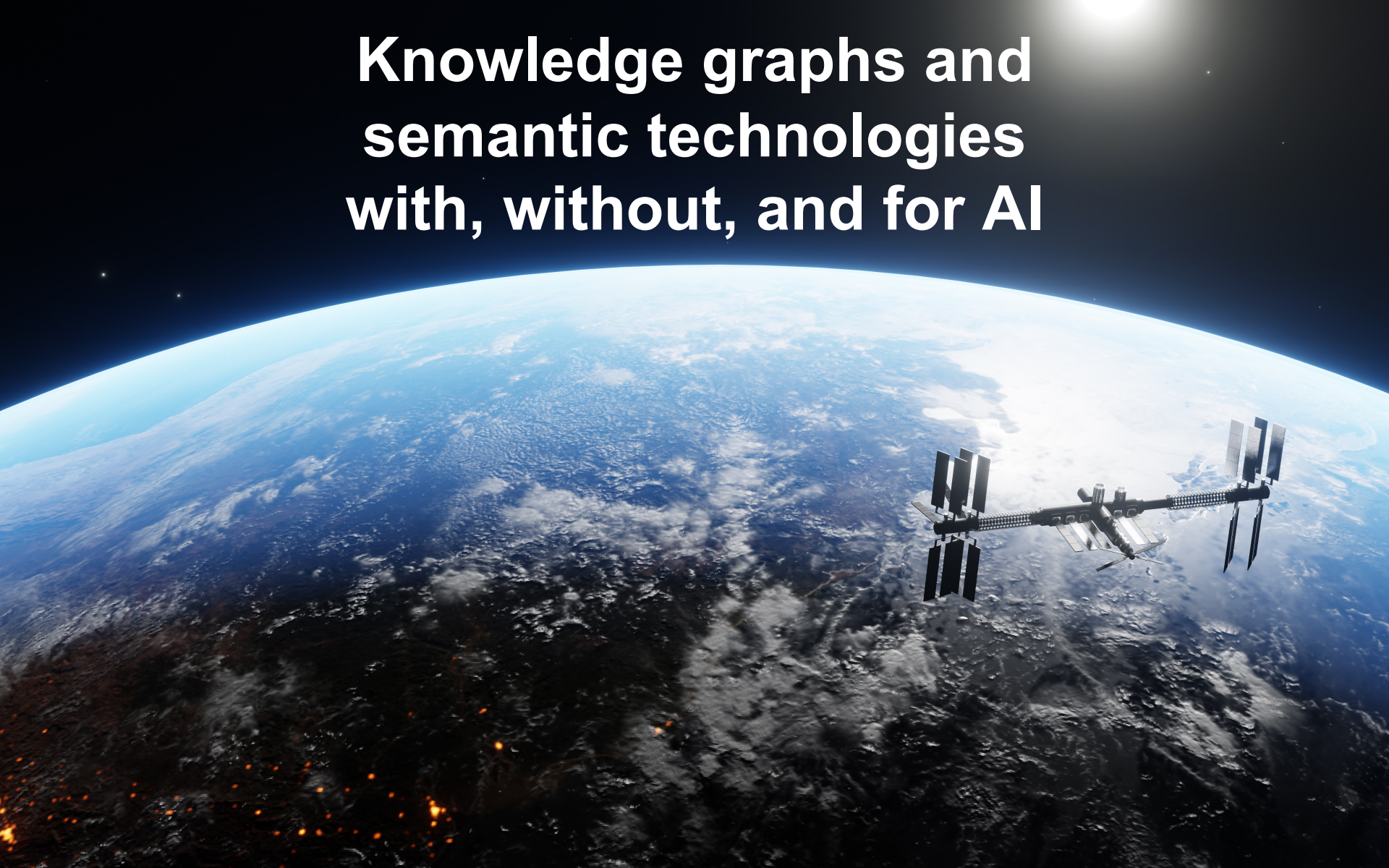
- Ontology modeling
- Reasoning
- Representation languages
- KG management
- KG querying

What can I do with KGs?

- Use cases and applications

Lecture level: For Beginners (no prior knowledge required)

Knowledge graphs and semantic technologies with, without, and for AI



How the journey began…

- Expressing meaning
- Knowledge representation
- Ontologies
- Agents
- Evolution of knowledge

The Semantic Web

A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities

by TIM BERNERS-LEE, JAMES HENDLER and ORA LASSILA

The entertainment system was belting out the Beatles' "We Can Work It Out" when the phone rang. When Pete answered, his phone turned the sound down by sending a message to all the other *local* devices that had a *volume control*. His sister, Lucy, was on the line from the doctor's office: "Mom needs to see a specialist and then has to have a series of physical therapy sessions. Biweekly or something. I'm going to have my agent set up the appointments." Pete immediately agreed to share the chauffeuring. At the doctor's office, Lucy instructed her Semantic Web agent through her handheld Web browser. The agent promptly retrieved information about Mom's *prescribed treatment* from the doctor's agent, looked up several lists of *providers*, and checked for the ones *in-plan* for Mom's insurance within a *20-mile radius* of her *home* and with a *rating* of *excellent* or *very good* on trusted rating services. It then began trying to find a match between available *appointment times* (supplied by the agents of individual providers through their Web sites) and Pete's and Lucy's busy schedules. (The emphasized keywords indicate terms whose semantics, or meaning, were defined for the agent through the Semantic Web.)

In a few minutes the agent presented them with a plan. Pete didn't like it—University Hospital was all the way across town from Mom's place, and he'd be driving back in the middle of rush hour. He set his own agent to redo the search with stricter preferences about *location* and *time*. Lucy's agent, having *complete trust* in Pete's agent in the context of the present task, automatically assisted by supplying access certificates and shortcuts to the data it had already sorted through.

Killer app: Agents

- Retrieve information about prescribed treatment/therapy
- Look for providers/specialists with good ratings in proximity
- Schedule times according to multiple busy schedules
- Share information with other agents and “negotiate”

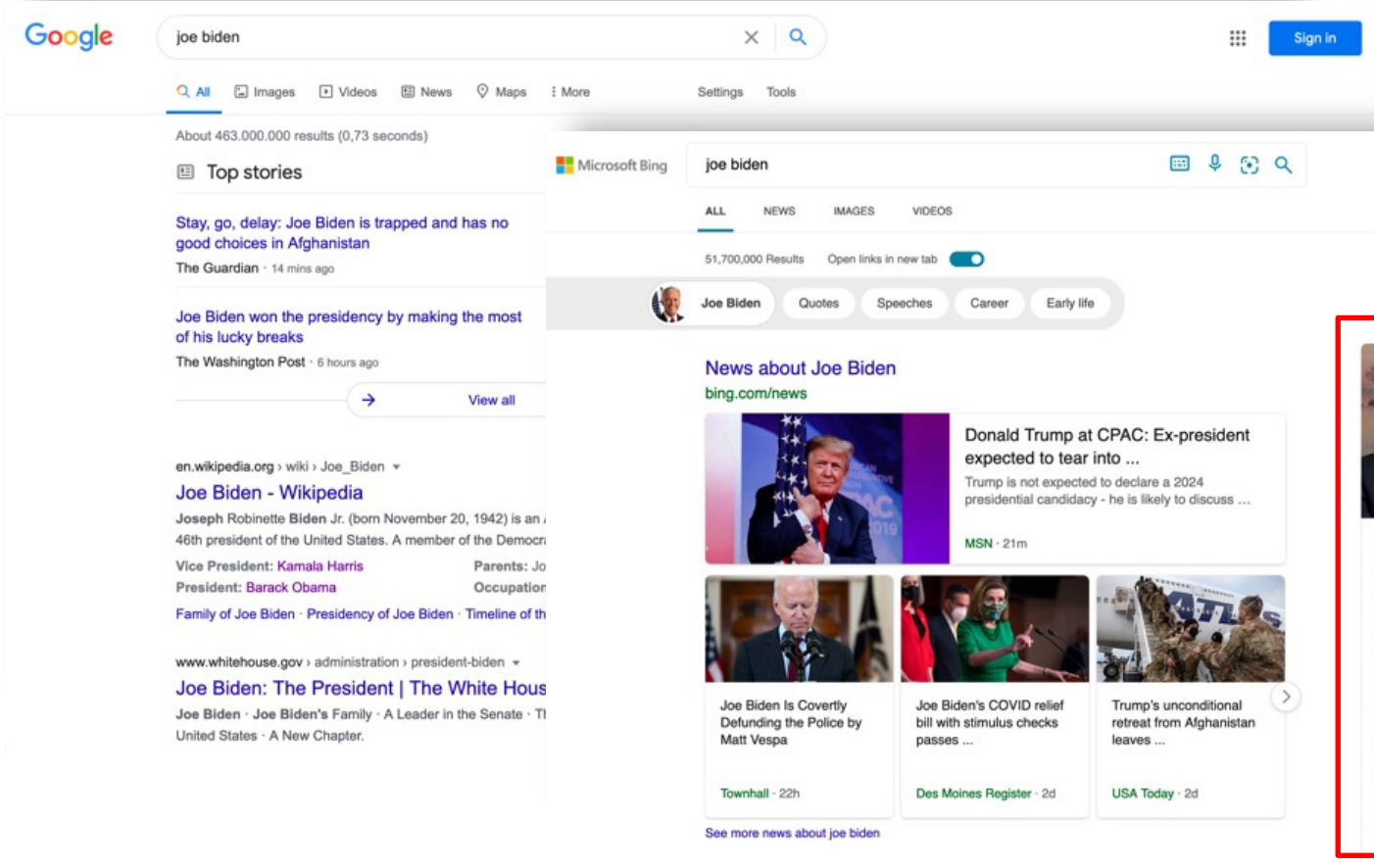
Scientific American, 2001

https://sit.instructure.com/courses/22632/files/2992212?module_item_id=462878

“A really important thing about data is: the more things you have to connect together, the more powerful it is.”

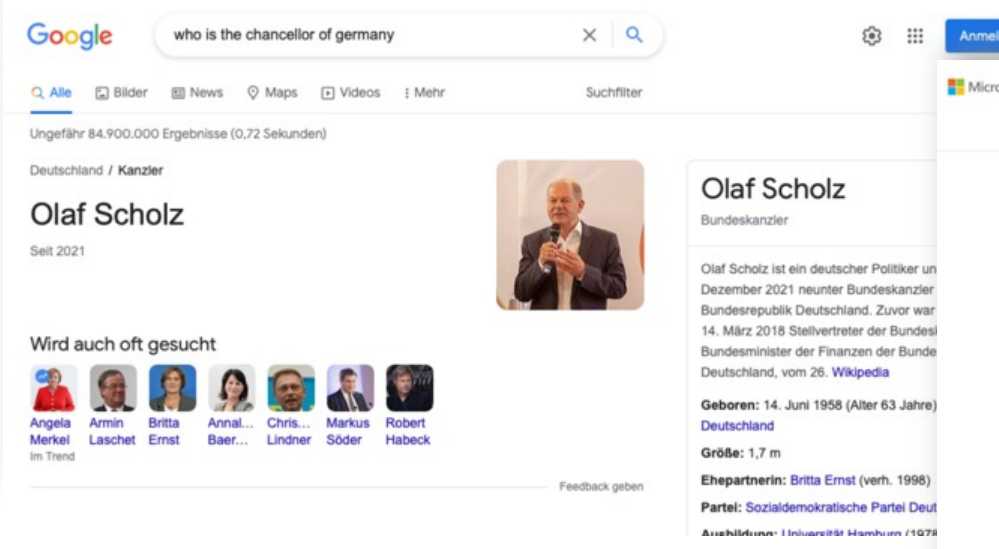
Sir Tim Berners-Lee on Linked Data at TED 2009

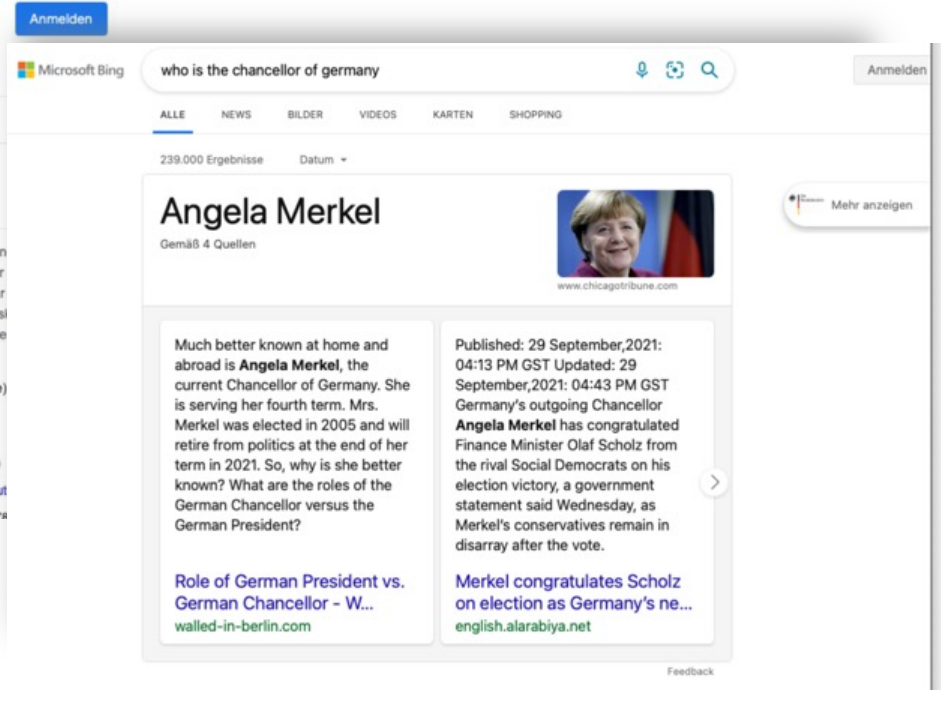
Search engines



Knowledge evolution

December 9, 2021: Who is the chancellor of Germany?





Knowledge graphs in industry

> Knowledge graphs provide the structured data and factual knowledge that drive many products and make them more intelligent and "magical".

N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson, J. Taylor: Industry-scale knowledge graphs: lessons and challenges. Commun. ACM 62(8): 36-43 (2019)

Common characteristics of the knowledge graphs.

	Data model	Size of the graph	Development stage
Microsoft	The types of entities, relations, and attributes in the graph are defined in an ontology.	~2 billion primary entities, ~55 billion facts	Actively used in products
Google	Strongly typed entities, relations with domain and range inference	1 billion entities, 70 billion assertions	Actively used in products
Facebook	All of the attributes and relations are structured and strongly typed, and optionally indexed to enable efficient retrieval, search, and traversal.	~50 million primary entities, ~500 million assertions	Actively used in products
eBay	Entities and relation, well-structured and strongly typed	Expect around 100 million products, >1 billion triples	Early stages of development and deployment
IBM	Entities and relations with evidence information associated with them.	Various sizes. Proven on scales documents >100 million, relationships >5 billion, entities >100 million	Actively used in products and by clients

Knowledge graph adoption

Open knowledge graphs

1.9B Facts

210M Facts

52M Facts

http://linkedlifedata.com/sources.html

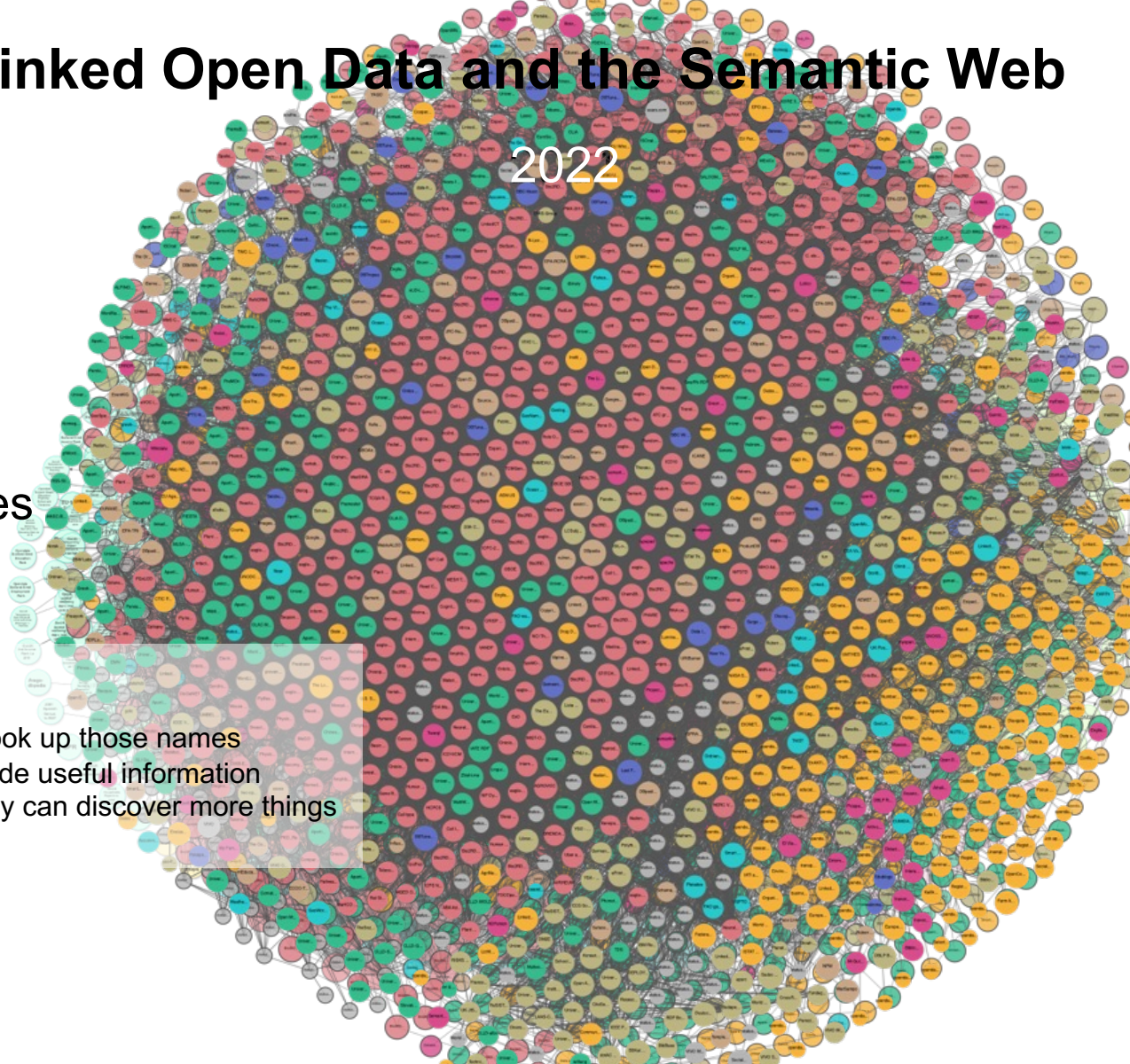
6.7B Facts

https://pubchem.ncbi.nlm.nih.gov/docs/rdf

132B Facts

TU WIEN Informatics

Linked Open Data and the Semantic Web



- Constantly growing
- Ca. 1255 datasets
- Ca. 400 billion triples

Linked Data principles

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names
3. When someone looks up a URI, provide useful information
4. Include links to other URIs so that they can discover more things

https://lod-cloud.net/

TU WIEN Informatics

Take back control of your data

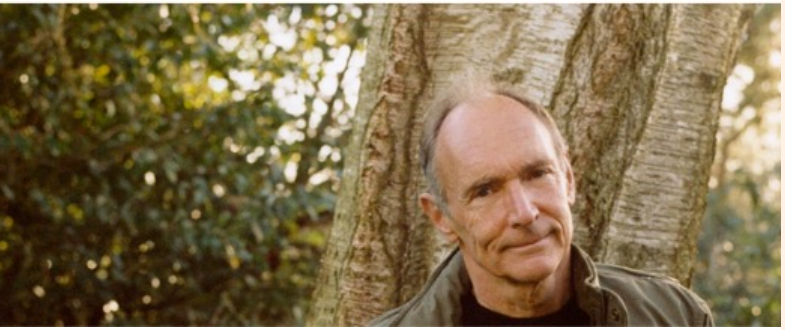
The New York Times

SUBSCRIBE NOW

Digital Privacy | How to Read Privacy Labels | How to Protect Your Digital Life | 10 Simple Tips | How Apps Collect Your Data

He Created the Web. Now He's Out to Remake the Digital World.

Tim Berners-Lee wants to put people in control of their personal data. He has technology and a start-up pursuing that goal. Can he succeed?



Tim Berners Lee's startup Inrupt releases Solid privacy platform for enterprises

Ron Miller @ron_miller / 6:01 AM GMT+1 • November 9, 2020

Comme

FINANCIAL TIMES

HOME WORLD US COMPANIES TECH MARKETS CLIMATE OPINION WORK & CAREERS LIFE & ARTS HOW TO SPEND IT

Technology + Add to myFT

NHS signs up for Tim Berners-Lee pilot to reinvent web

BBC, NatWest and government of Flanders also test new Solid servers from Inrupt

Tim Berners-Lee: "There is a feeling, a zeitgeist, that change is really overdue" © AFP/Getty Images

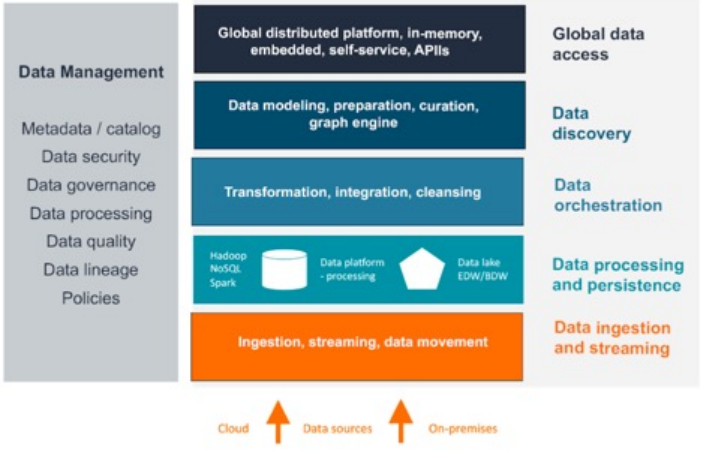
John Thornhill in London NOVEMBER 9 2020

118

Inrupt, the company founded by Tim Berners-Lee to promote a new data architecture for the web, has launched its first enterprise-ready Solid servers

TU WIEN Informatics

Data fabrics & semantic data lakes



A knowledge graph built on semantics and graph data models fits into the data orchestration and discovery layers in a data fabric.

CAMBRIDGE SEMANTICS

Data Fabrics: The Killer Use Case for Knowledge Graphs

Posted by Barbara Petrocelli on Mar 5, 2021 10:54:44 AM

Tweet | Share | Like | Share

This article explores the role of knowledge graphs in data fabrics. If you want to learn more about this topic, please join Forrester Analyst Noel Yuhanna and Cambridge Semantics Co-Founder Ben Szekely at an interactive discussion on this topic. Learn more here.

In a data fabric, knowledge graphs connect siloed data and simplify complicated data.

"A knowledge graph in a data fabric solves the data integration task and bridges the gap between disparate corporate datasets and the actual business requirements.

It allows business users to model, explore and discover how datasets are connected in a human readable model where there is little difference between the conceptual and the physical machine readable models."

Source: Demystifying the Data Fabric | 9.17.20 | Gartner Group

The Data Fabric is an architecture for modern data management that anticipates the need to connect data across the enterprise at speed and scale.

Knowledge graph is central to the success of any Data Fabric as an agile integration overlay that blends complex and diverse data into easy-to-consume data products for end-users, data scientists, and applications.

Neurosymbolic AI

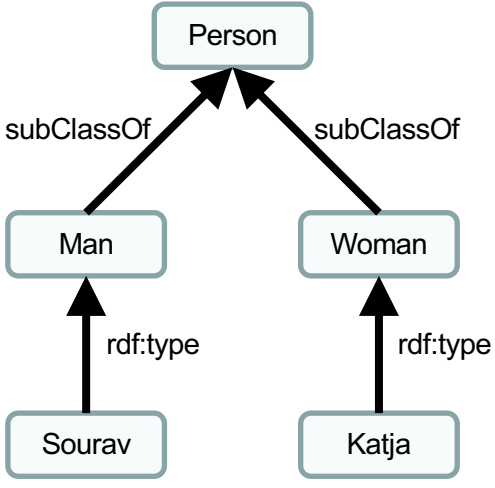
Pure Logic Reasoning

Pure Learning

Neurosymbolic AI

Pure Logic Reasoning

Pure Learning



(Katja, rdf:type, Woman) ∧ (Woman, subClassOf, Person)
→ (Katja, rdf:type, Person)

Neurosymbolic AI

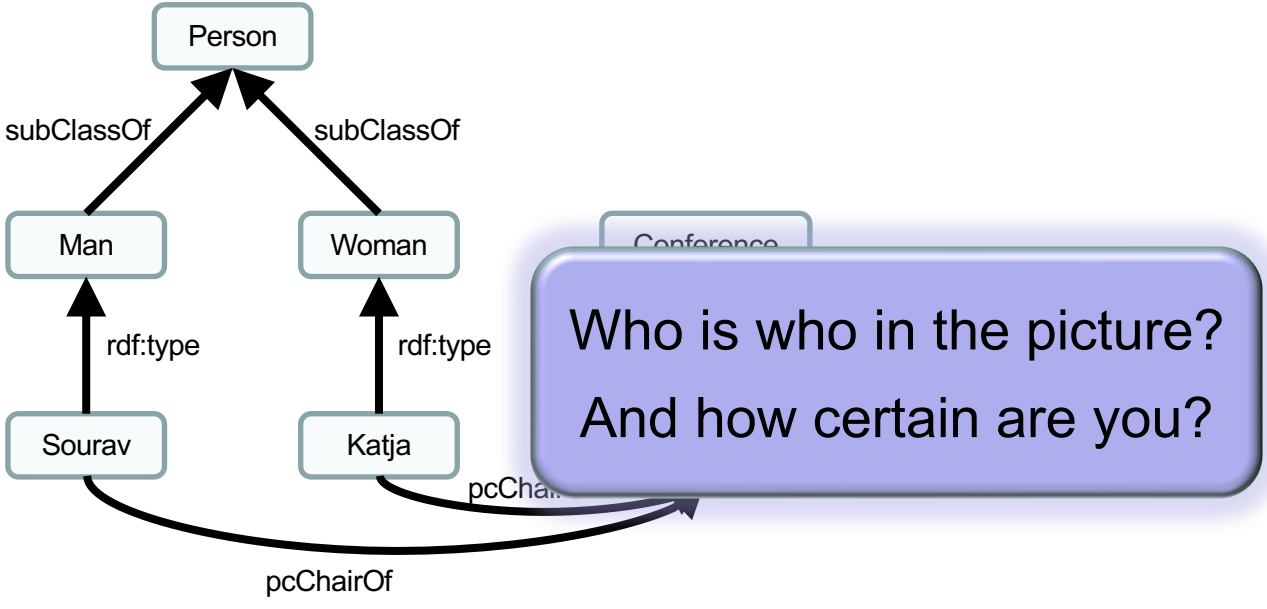
Pure Logic Reasoning

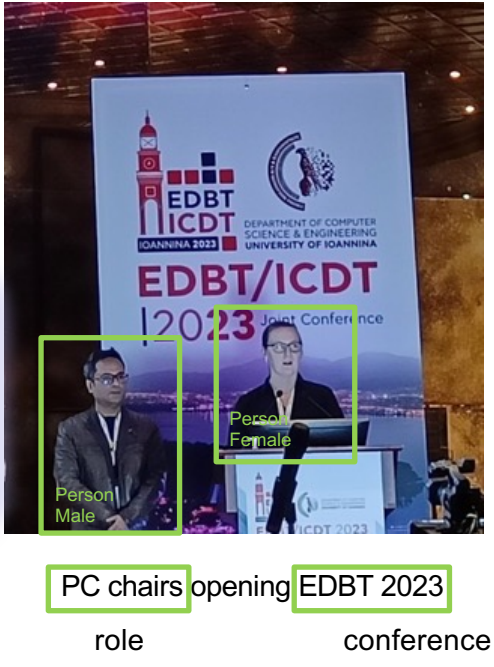
Pure Learning

PC chairs opening EDBT 2023

role conference

Neurosymbolic AI

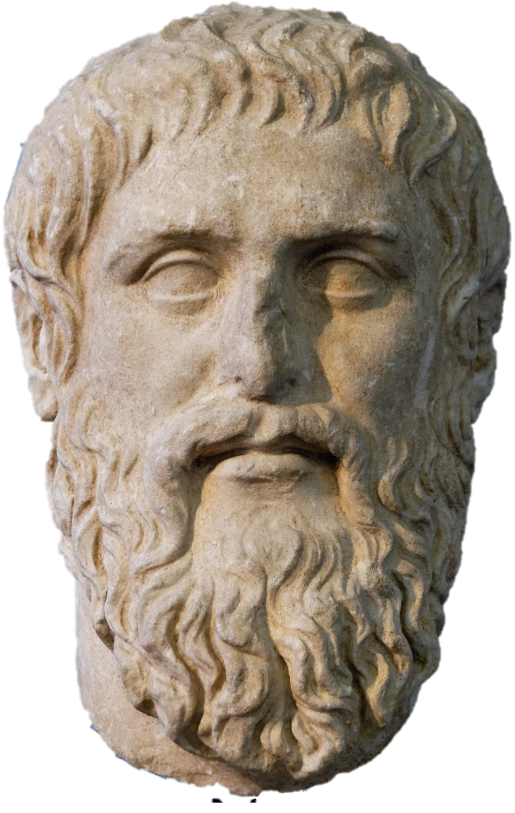




EDBT 2023 has two PC chairs*
These are Katja and Sourav
Katja is a Woman, Sourav is a Man
Men and Women are Persons

The PC chairs of EDBT 2023 are (likely) on the picture
One is (likely) a male person, the other one (likely) female

* Partial Completeness Assumption (PCA)
Galárraga, Teflioudi, Hose, Suchanek. Fast Rule Mining in Ontological Knowledge Bases with AMIE+, VLDBJ 2015



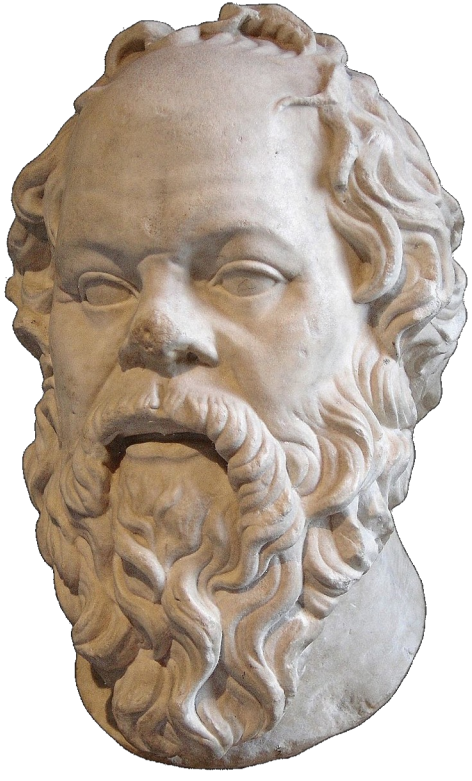
What is knowledge?

Plato:

Knowledge is justified true belief.

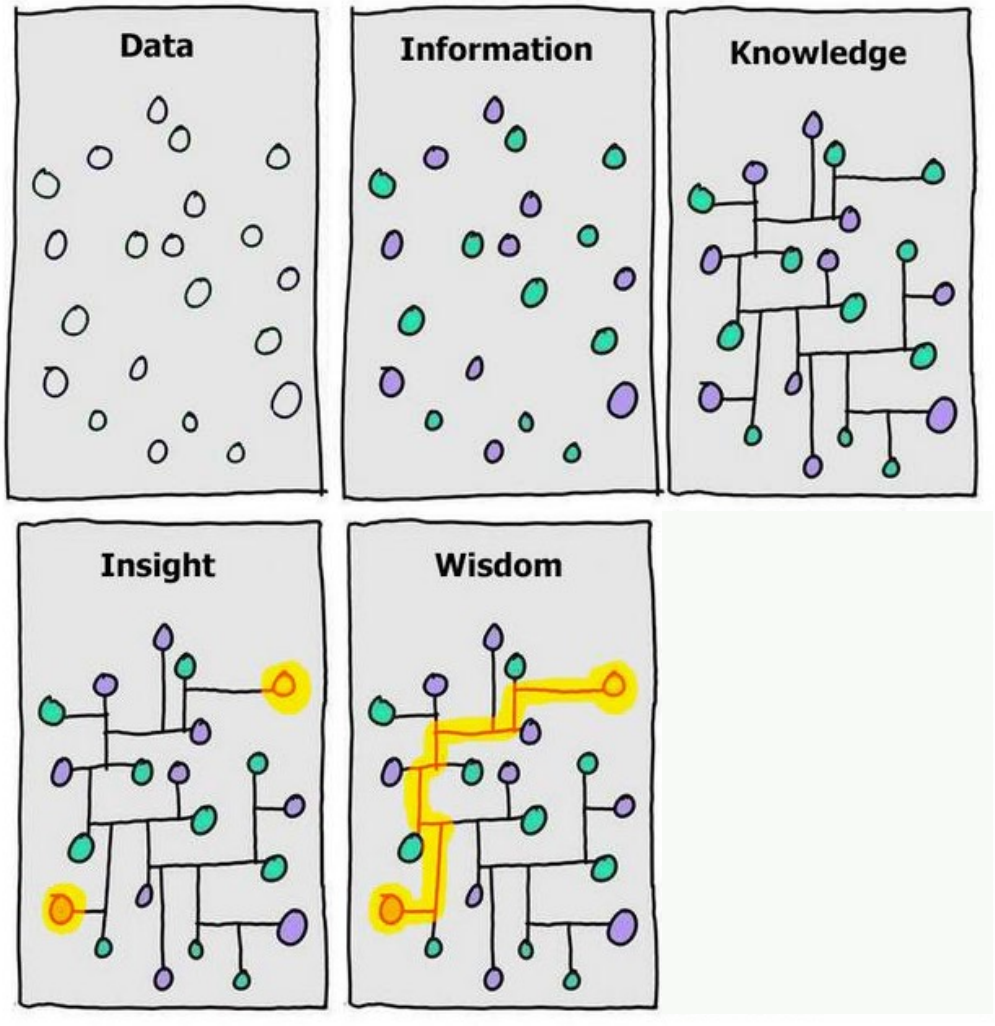
Socrates:

True knowledge exists in knowing
that you know nothing.



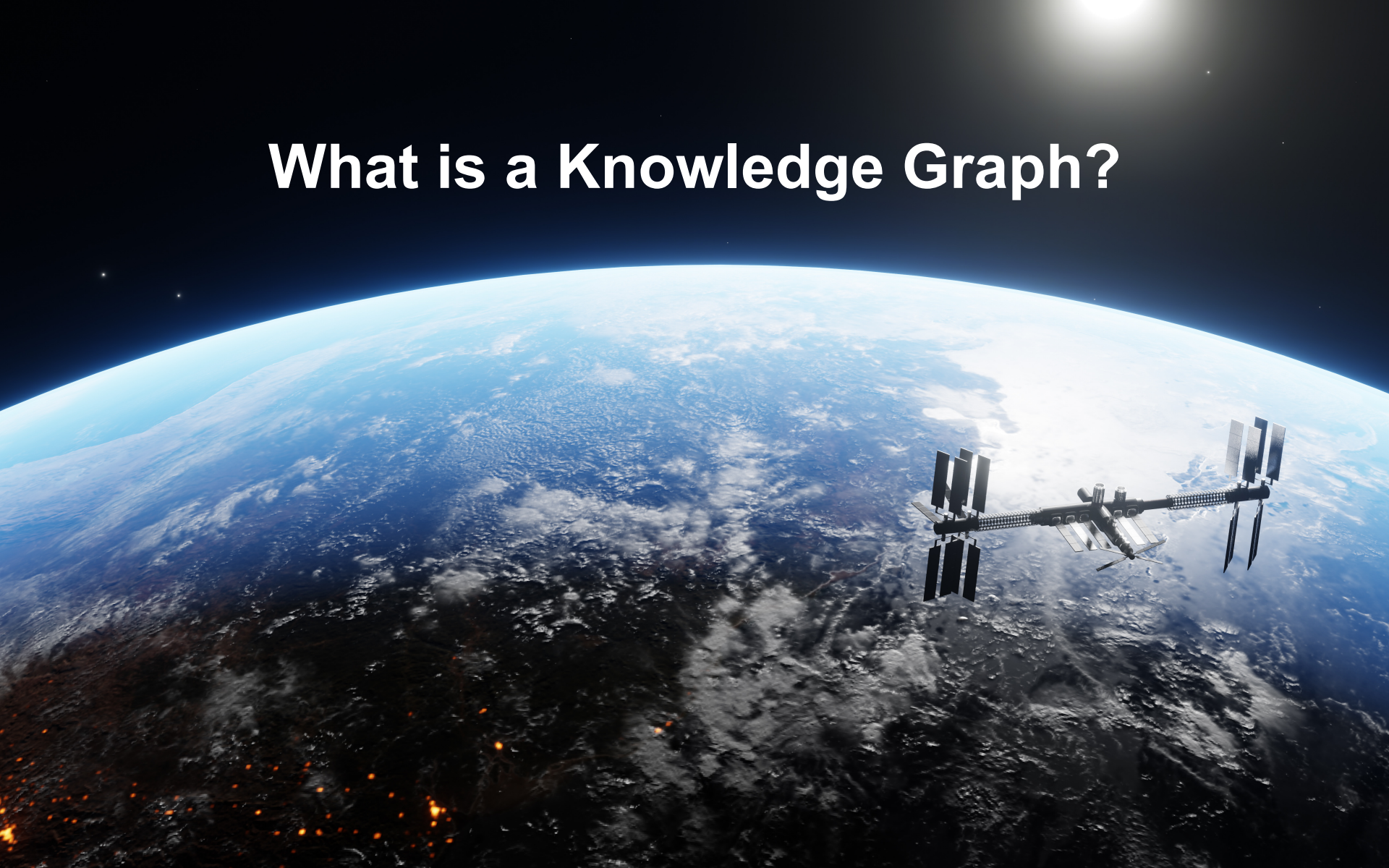
wikipedia.org

Data, Information, Knowledge, Insight, Wisdom … and beyond

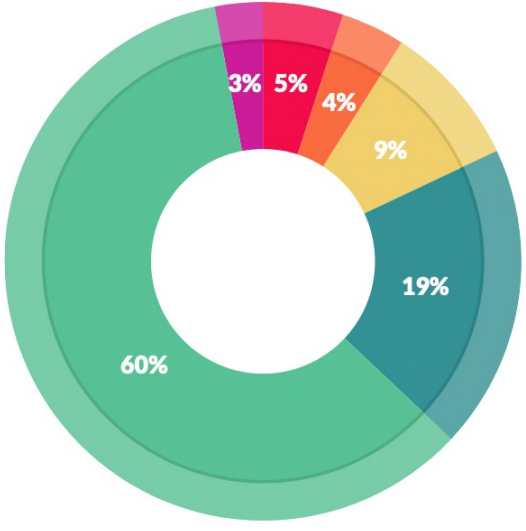


A composition of images shared by David Somerville and Ron Letkeman https://antzfx.wordpress.com/2020/04/08/show-information-

What is a Knowledge Graph?

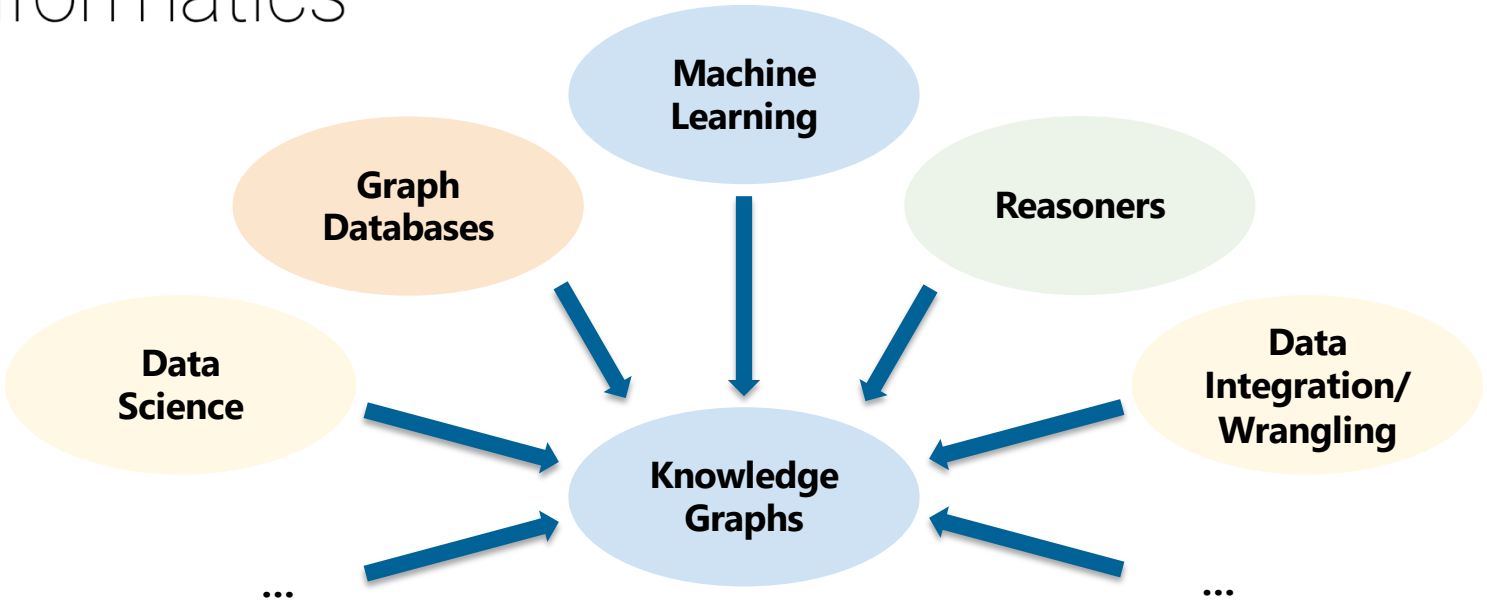


What data scientists spend most of their time on



- *Building training sets: 3%*
- *Cleaning and organizing data: 60%*
- *Collecting data sets; 19%*
- *Mining data for patterns: 9%*
- *Refining algorithms: 4%*
- *Other: 5%*

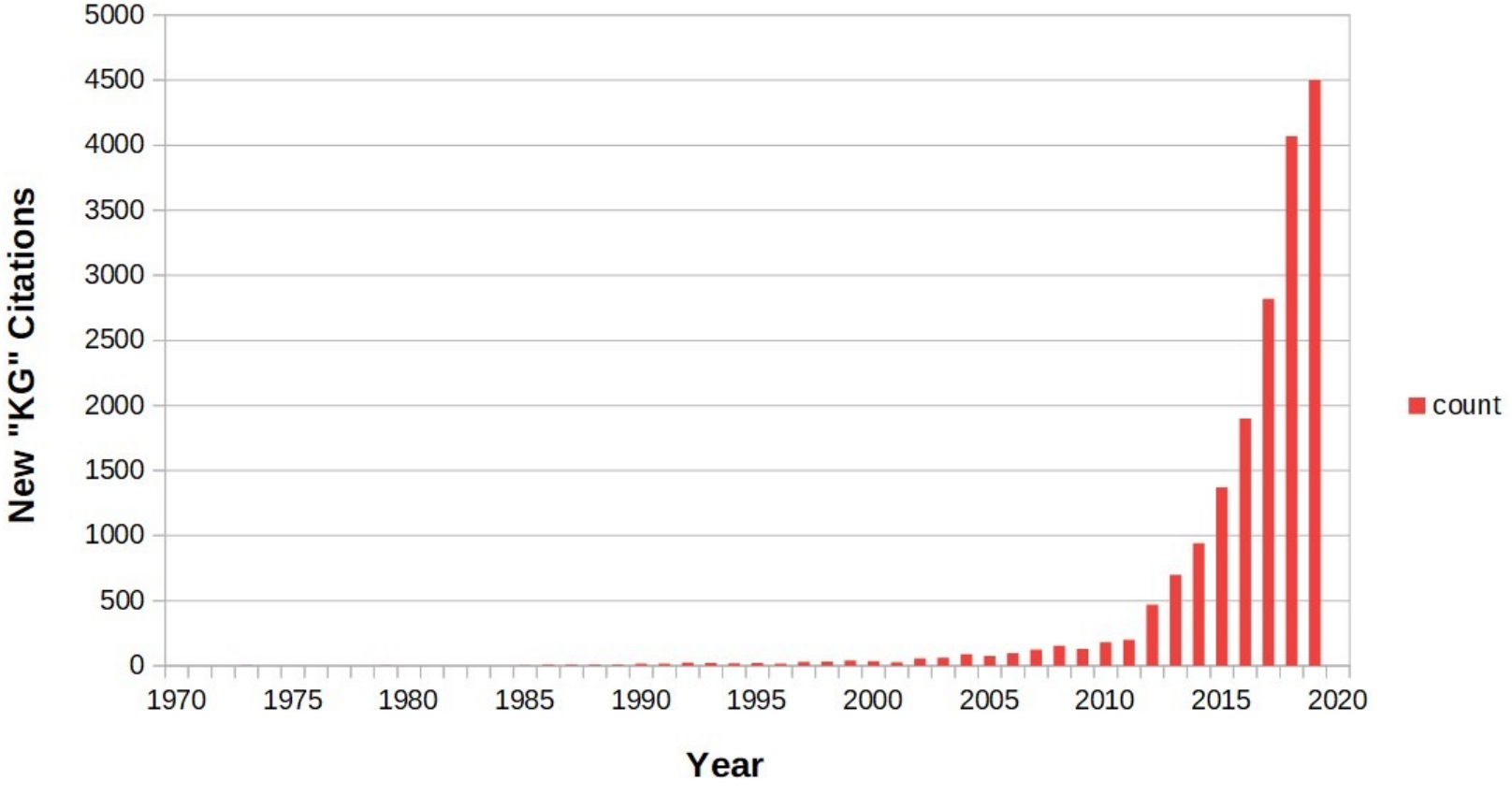
https://visit.figure-eight.com/rs/416-ZBE-142/images/CrowdFlower_DataScienceReport_2016.pdf



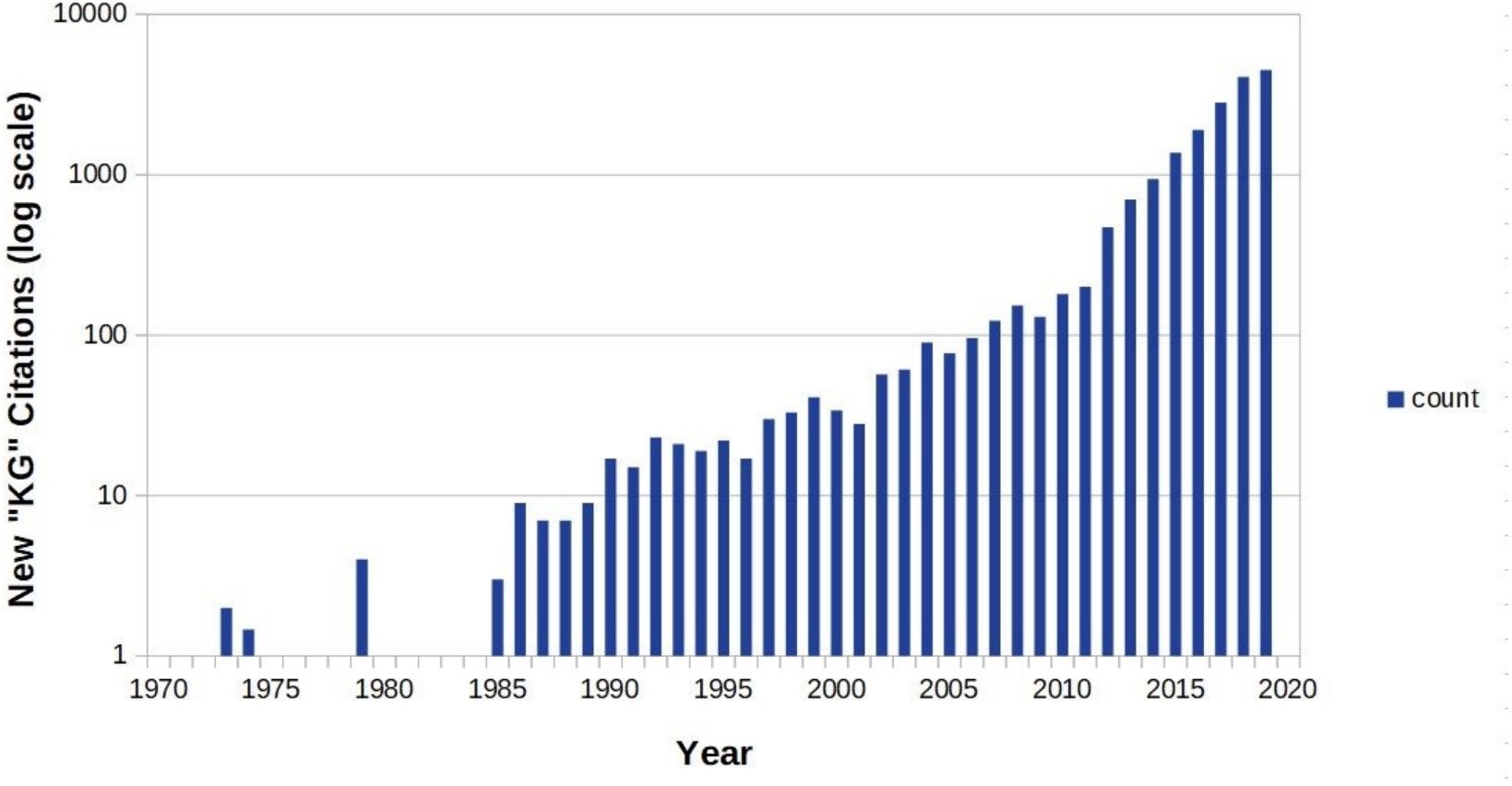
Dgraph ontotext cassandra DeepReason.ai Google Azure Cosmos DB RAINBIRD ...

Cognitum neo4j Gephi SAP HANA ORACLE ontop JanusGraph ...

aws Amazon Neptune AllegroGraph GRAKN.AI MAANA poolparty LogicBlox ...



Source: https://www.mkbergman.com/2244/a-common-sense-view-of-knowledge-graphs/



Source: https://www.mkbergman.com/2244/a-common-sense-view-of-knowledge-graphs/

Google's Viewpoint

"A graph that understands real-world entities and their relationships to one another:
things, not strings"

Google

""Introduction the Knowledge Graph: Things, Not Strings". A. Singhal. Google Blog 2012.

Going Back to 1974

"A mathematical structure with **vertices** as *knowledge units* connected by **edges** that represent the *prerequisite relation*"

"On the Structure of the Teaching-learning Interactive Process". E. Marchi and O. Miguel. I.J. Game Theory 1974.

In this course

"A knowledge graph contains **semantically related** information encoded as **nodes and edges**."

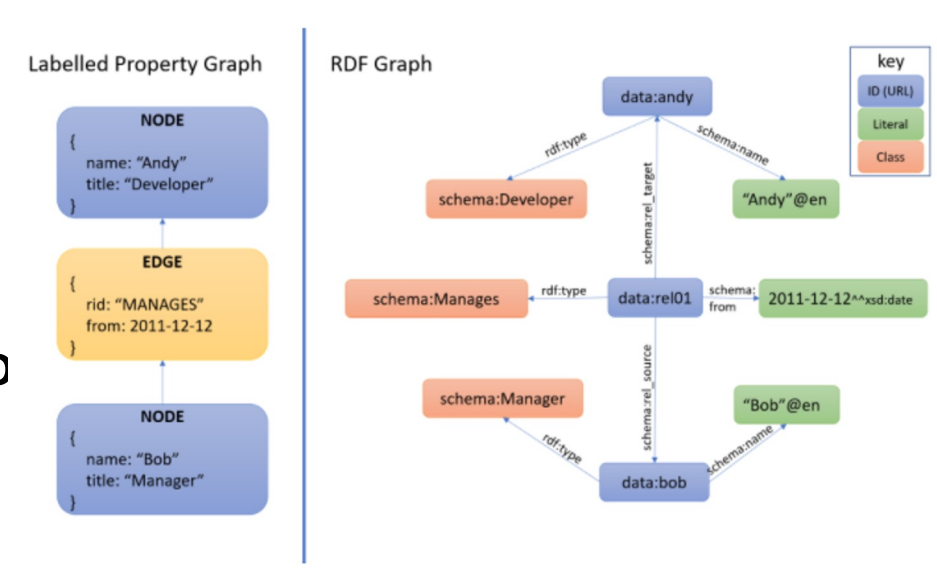
Katja Hose and Emanuel Sallinger, 2023

Enabling Technologies

- Two mainstream technologies (PG vs RDF)

- **Property Graphs (PG)**,
  - e.g., Neo4J, JanusGraph, OrientDB
  - *Programmer-friendly*
  - *Fast*
  - *Edge's attributes*

- **Resource Description Framework (RD**
  - e.g., GraphDB, StarDog, Virtuoso
  - *Globally unique ID*
  - *Interoperability-at-scale*
  - *(shared) Schema-based*



Source: https://terminusdb.com/blog/graph-fundamentals-part-2-labeled-property-graphs/

Graph Databases become mainstream

Source: https://www.bloorresearch.com/technology/graph-databases/#vendorlandscape

188.399 Introduction to Semantic Systems

(2.0 VU / 3.0 ECTS)

2023W

Katja Hose

Emanuel Sallinger

iss@dbai.tuwien.ac.at

Learning Goals

After the lecture you will be able to:

1) explain the vision of semantic technologies
2) semantically model a domain of interest with ontologies
3) apply core semantic representation standards
4) manage and query knowledge graphs
5) build a knowledge graph by transforming (structured) data into a semantic representation

Course Organization and Grading

- 7 Lectures

- Three Individual Assignments, in total worth 60p
  - Submission of technical artifacts and report

- Written Exam (40p)
  - **19.01.2024**
  - You need to score at least 20p on the exam to pass the course
  - You may retake the exam (failed or already passed) on **15.03.2024**
  - **The later result counts!**

Lectures

- Lectures:
  - 06.10.2023
  - 13.10.2023
  - 20.10.2023
  - 17.11.2023
  - 24.11.2023
  - 01.12.2023
  - 15.12.2023

Timeline of assignments

Assignment	Assignment Topics	Available TUWEL	Submission Deadline
A1	Ontology modeling	20.10.2023	03.11.2023 (2 weeks)
A2	Knowledge Graph querying	17.11.2023	01.12.2023 (2 weeks)
A3	Programing Semantic Systems	01.12.2023	22.12.2023 (3 weeks)

Assignments need to be submitted in TUWEL by the specified deadline.

Assignments submitted after the deadline cannot be graded.

Course Grading

Grade = Assignments (max 60p) + Exam (max 40p)

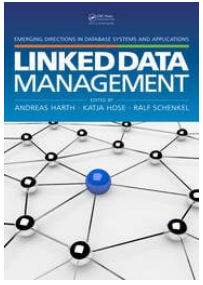
Points	Evaluation	Grade
> 87	Sehr gut	1
86 –75	Gut	2
74 –61	Befriedigend	3
60 –50	Genügend	4
<50	Nicht Genügend	5

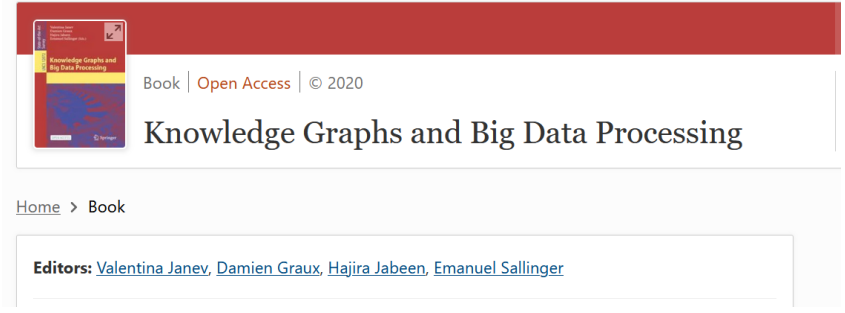
Recommended Literature

Book

Linked Data Management

Edited By Andreas Harth, Katja Hose, Ralf Schenkel



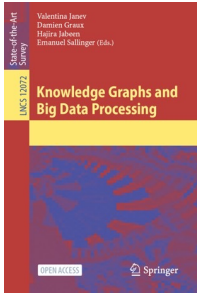


Book | Open Access | © 2020

Knowledge Graphs and Big Data Processing

Home > Book

Editors: Valentina Janev, Damien Graux, Hajira Jabeen, Emanuel Sallinger

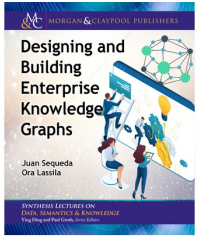


RESEARCH-ARTICLE

Knowledge Graphs

Authors: Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, + 10 Authors Info & Affiliations

ACM Computing Surveys, Volume 54, Issue 4 • July 2021 • Article No.: 71, pp 1–37 • https://doi.org/10.1145/3447772



Next steps

- If you decide not to take the course, make sure to cancel your registration **before Oct 20, 2023 [!]**
- Register for the course in TUWEL to access all course materials. Link available in TISS.

Questions?

General Questions

- TUWEL Forum ... if the answer might be interesting for other students
- Answer questions from colleagues (preferably without posting the solution :-) ...

Specific Questions

- iss@dbai.tuwien.ac.at

TU WIEN Informatics

What Is An Ontology?

Emanuel Sallinger
Katja Hose

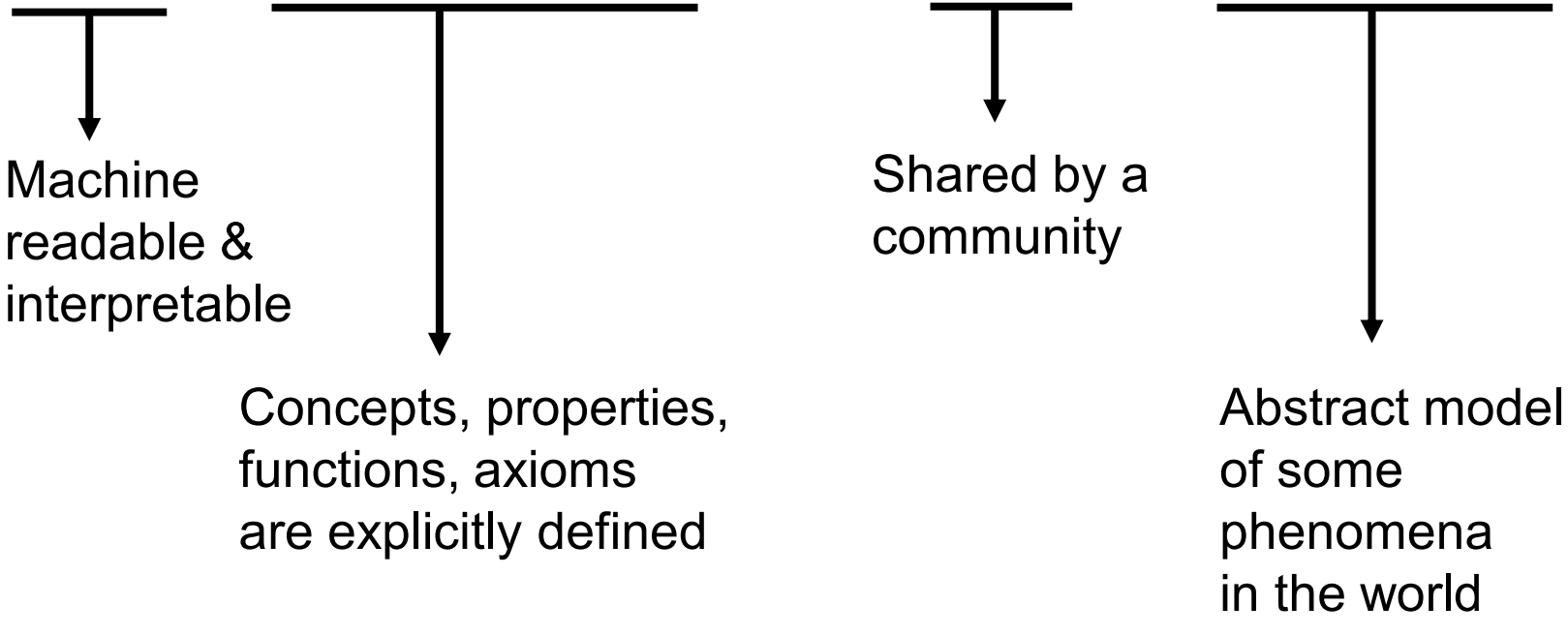
What is an ontology?

- Ontology = ontos (being) + logos (word)
- Philosophy: Systematic explanation of Existence
  - Aristotle (400BC) attempts to establish universal categories for classifying everything that exists
- Mirriam Webster dictionary:
  - A branch of metaphisics concerned with the nature and relations of being;
  - A particular theory about the nature of being and kinds of existents.
- Ontology - is both a science and its result

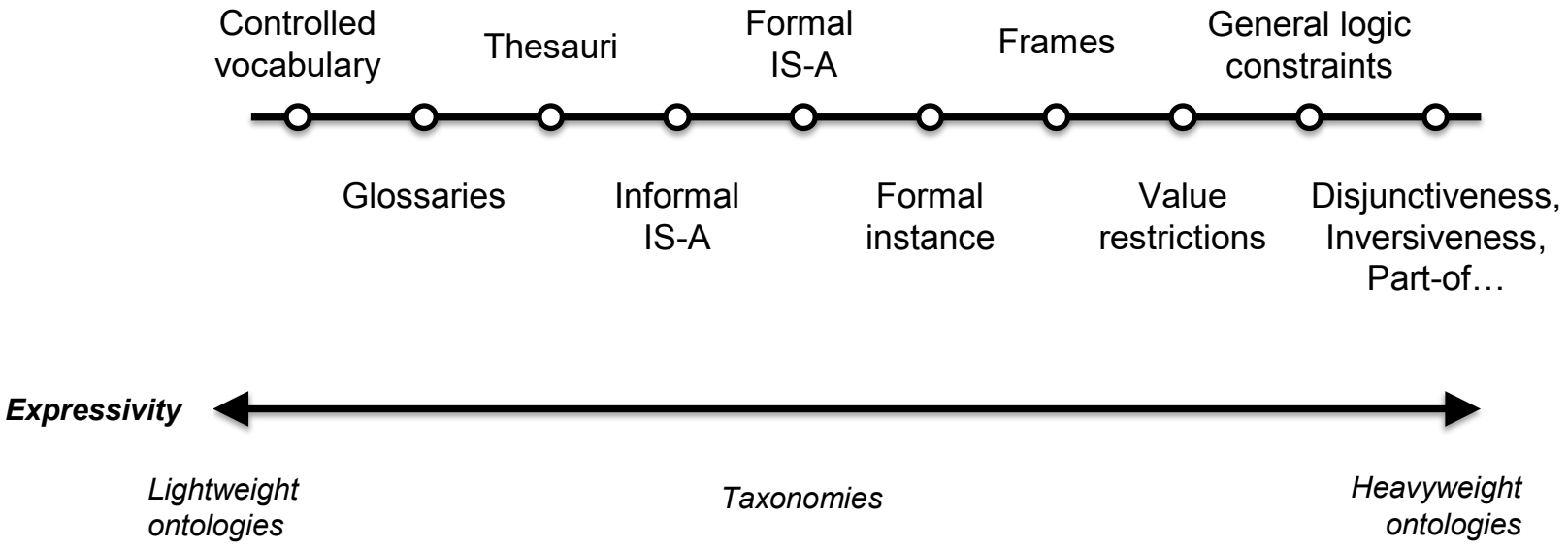
What is an ontology?

Studer(98):

Formal, explicit specification of a shared conceptualization.



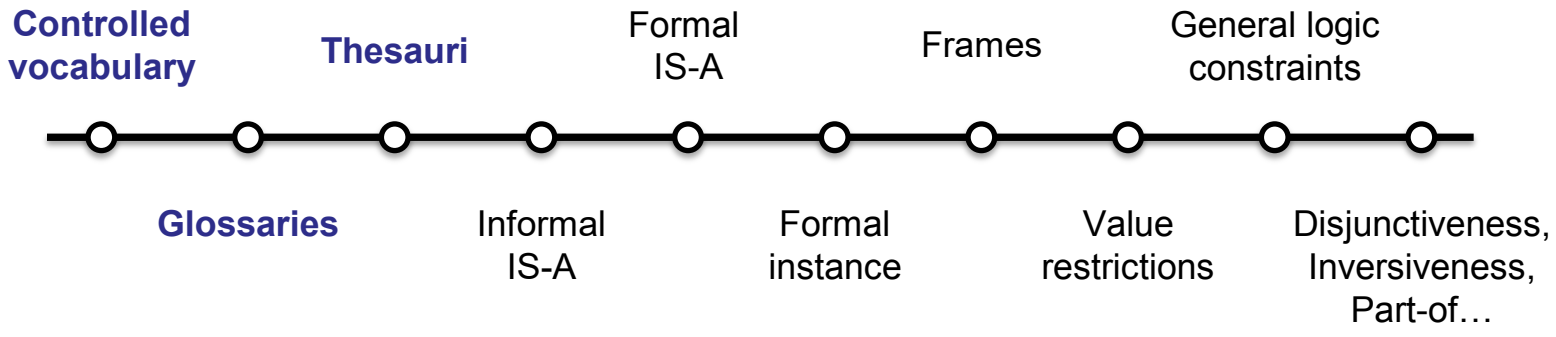
Ontology Types and Categories: Overview



Adapted from lecture slides "Knowledge Engineering with Semantic Web Technologies"
Dr. Harald Sack, Hasso-Plattner-Institut for IT Systems Engineering

Lightweight Ontologies

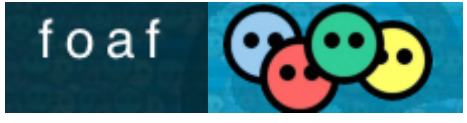
- **Controlled Vocabulary:** finite list of terms (e.g., a catalogue)
- **Glossary:**
  finite list of terms + informal definition in natural language
- **Thesauri:**
  controlled vocabulary + concepts are connected with relations
  - Equivalency (synonyms)
  - Hierarchies (subclasses, superclasses)
  - Homographs (Homonyms)
  - Associations (similar concepts)



Adapted from lecture slides "Knowledge Engineering with Semantic Web Technologies"
Dr. Harald Sack, Hasso-Plattner-Institut for IT Systems Engineering

A popular lightweight ontology

Friend of a Friend



FOAF Basics

- Agent
- Person
- name
- nick
- title
- homepage
- mbox
- mbox_sha1sum
- img
- depiction (depicts)
- surname
- family_name
- givenname
- firstName

Personal Info

- weblog
- knows
- interest
- currentProject
- pastProject
- plan
- based_near
- workplaceHomepage
- workInfoHomepage
- schoolHomepage
- topic_interest
- publications
- geekcode
- myersBriggs
- dnaChecksum

Online Accounts / IM

- OnlineAccount
- OnlineChatAccount
- OnlineEcommerceAccount
- OnlineGamingAccount
- holdsAccount
- accountServiceHomepage
- accountName
- icqChatID
- msnChatID
- aimChatID
- jabberID
- yahooChatID

Projects and Groups

- Project
- Organization
- Group
- member
- membershipClass
- fundedBy
- theme

Documents and Images

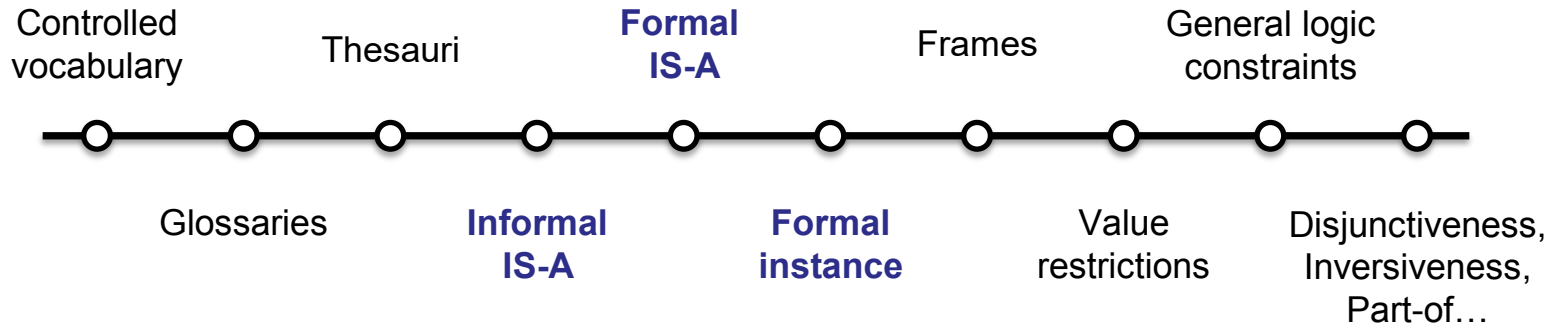
- Document
- Image
- PersonalProfileDocument
- topic (page)
- primaryTopic
- tipjar
- sha1
- made (maker)
- thumbnail

Adapted from Enrico Motta

Taxonomies

Definition of a hierarchical system of groups
[from *τάξις* (*taxis)* = order, arrangement and *νόμος* (*nomos)* = law, science]

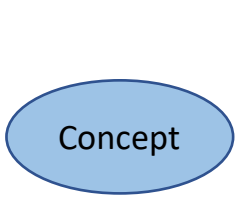
- **Informal IS-A-Hierarchy:** explicit hierarchy of classes
  subclass relations are not strict (e.g. index of a library)
- **Formal IS-A-Hierarchy:**
  explicit hierarchy of classes, subclass relations are strict
- **Formal instance:**
  explicit class hierarchy, strict subclass relations +
  instance-of relations are allowed



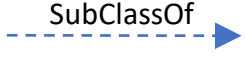
Adapted from lecture slides "Knowledge Engineering with Semantic Web Technologies"
Dr. Harald Sack, Hasso-Plattner-Institut for IT Systems Engineering

Taxonomy

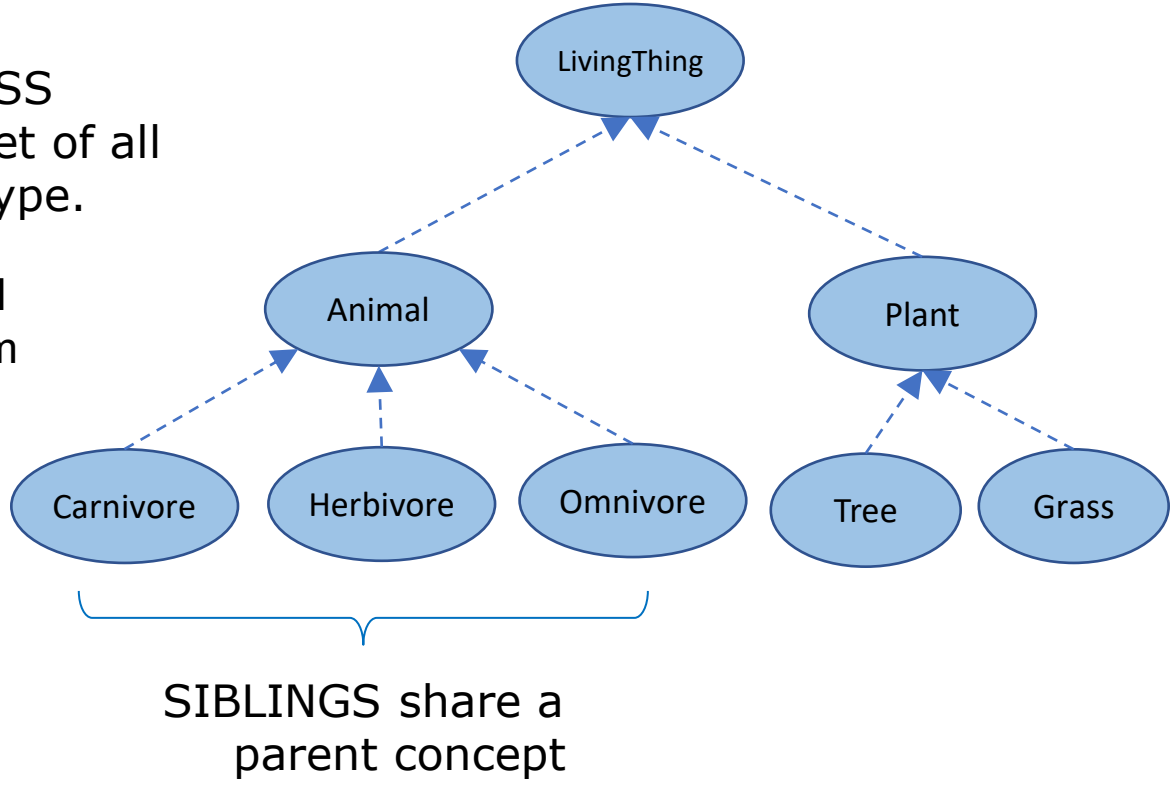
Taxonomy: a controlled vocabulary organized into a hierarchical structure.



A CONCEPT/CLASS represents the set of all entities of that type.

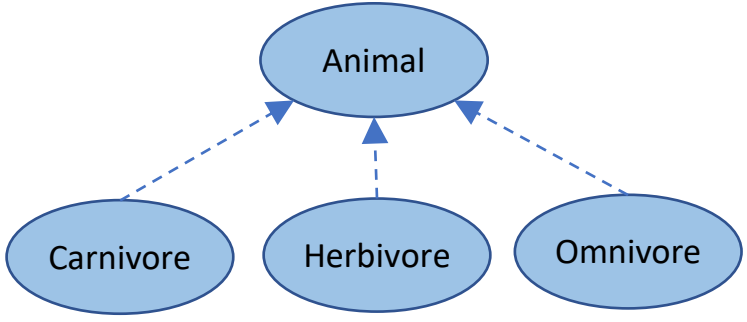


A SUBSUMPTION relation points from more specific to more generic concepts



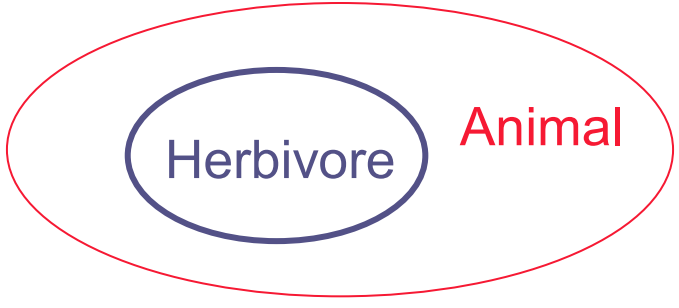
Subsumption/SubClass/Specialization Relation (in Formal IS-A Taxonomies)

Semantics (Meaning): If a *Herbivore* is a subClass of *Animal*, all instances of *Herbivore* are instances of *Animal.*



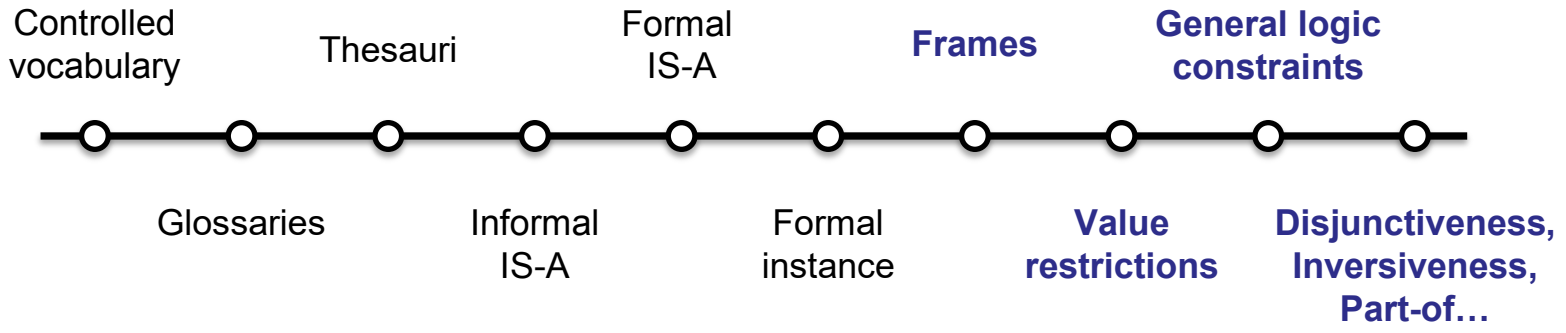
Common Mistake:

Subsumption is often incorrectly used to represent other types of relations such as part of (meronimy).



Heavyweight Ontologies

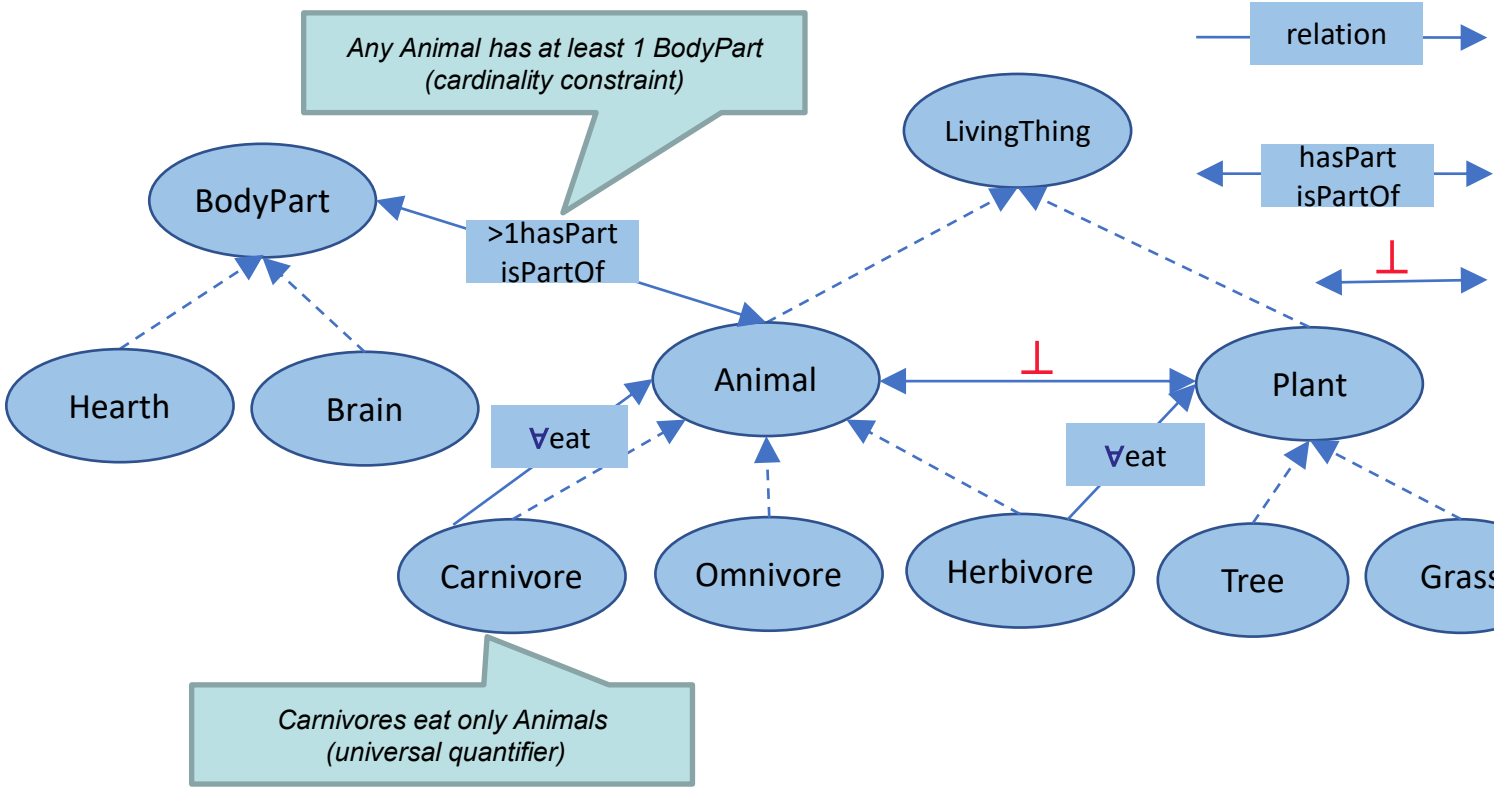
- Rigorous definition and organization of concepts
- Focus on *logic*, i.e., formally correct deductions and inferences
- Careful formal specification, need for consistency



Adapted from lecture slides "Knowledge Engineering with Semantic Web Technologies"
Dr. Harald Sack, Hasso-Plattner-Institut for IT Systems Engineering

Ontology

Ontology: A taxonomy extended with other relations and further constraints.

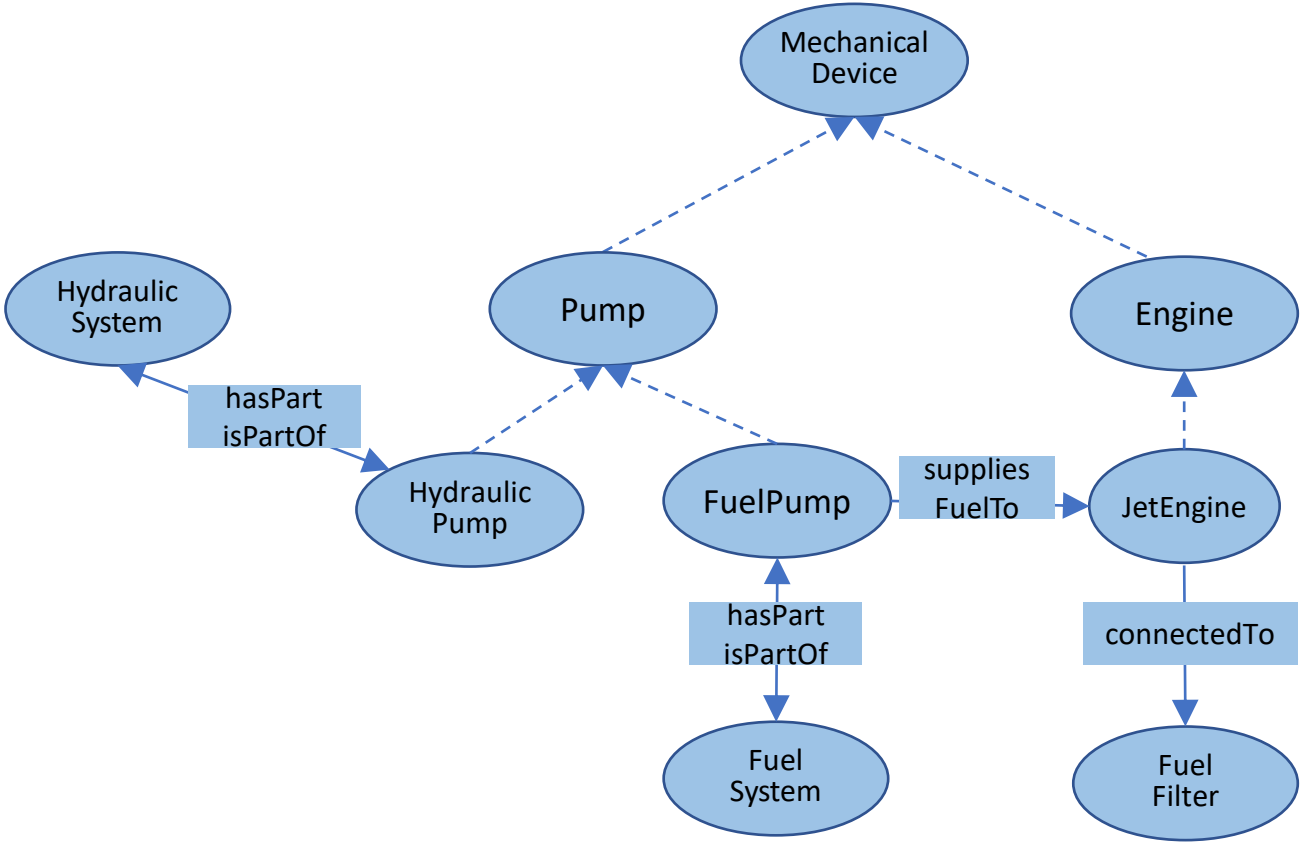


Relations point from a **Domain** to a **Range** concept.

Inverse relations

Disjoint concepts describe non-overlapping instance sets.

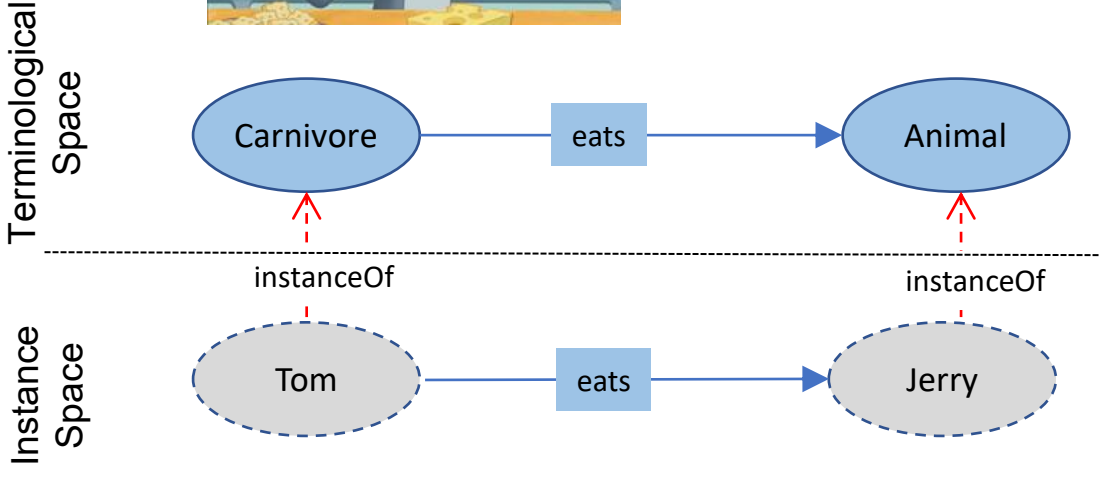
Ontology- Mechanics



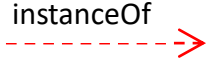
Instances/Entities

Metadata to describe the image:

- *Tom is a Carnivore*
- *Jerry is an Animal*
- *Tom eats Jerry*

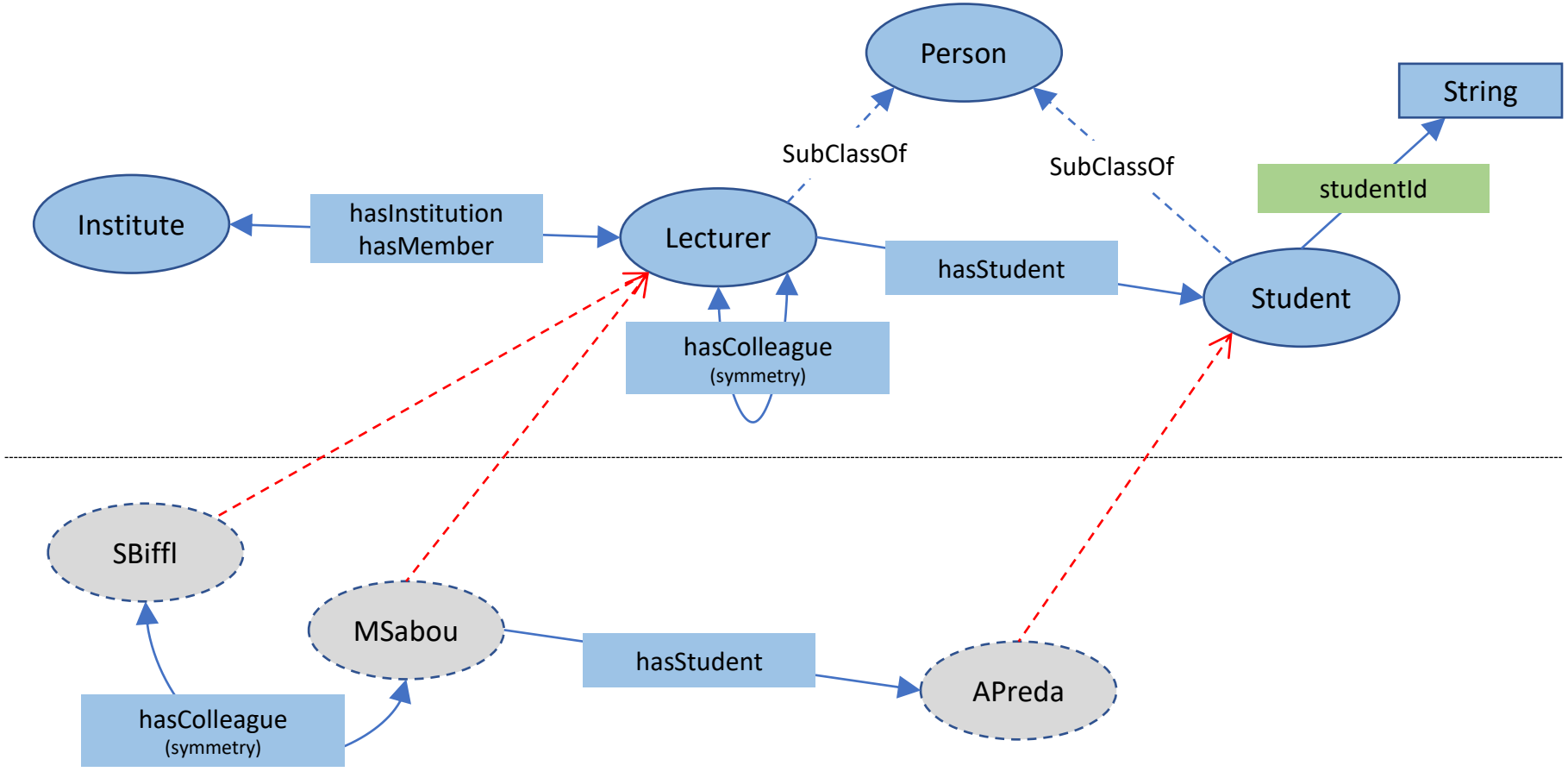


Instance represents an entity in the universe of discourse.



Types the instance with respect to a concept (the instance is of type concept).

More Examples …



Recap: What's inside an ontology?

- Concepts:
  - Denote the main concepts of the domain
  - *E.g.: Carnivore, Animal, Pump, Engine*
- Concept hierarchy
  - Denotes specialization/generalizations
  - *E.g.: Carnivore is a kind of Animal*
- Relations between classes
  - *E.g.: Carnivore eats Animal*
- Restrictions on relations (type, cardinality)
  - *E.g.: any Animal has at least one BodyPart*
- Instances:
  - Denote concrete entities in the domain
  - *E.g.: Tom, Jerry*

TU WIEN Informatics

What Is An Ontology?

Emanuel Sallinger
Katja Hose

TU WIEN Informatics

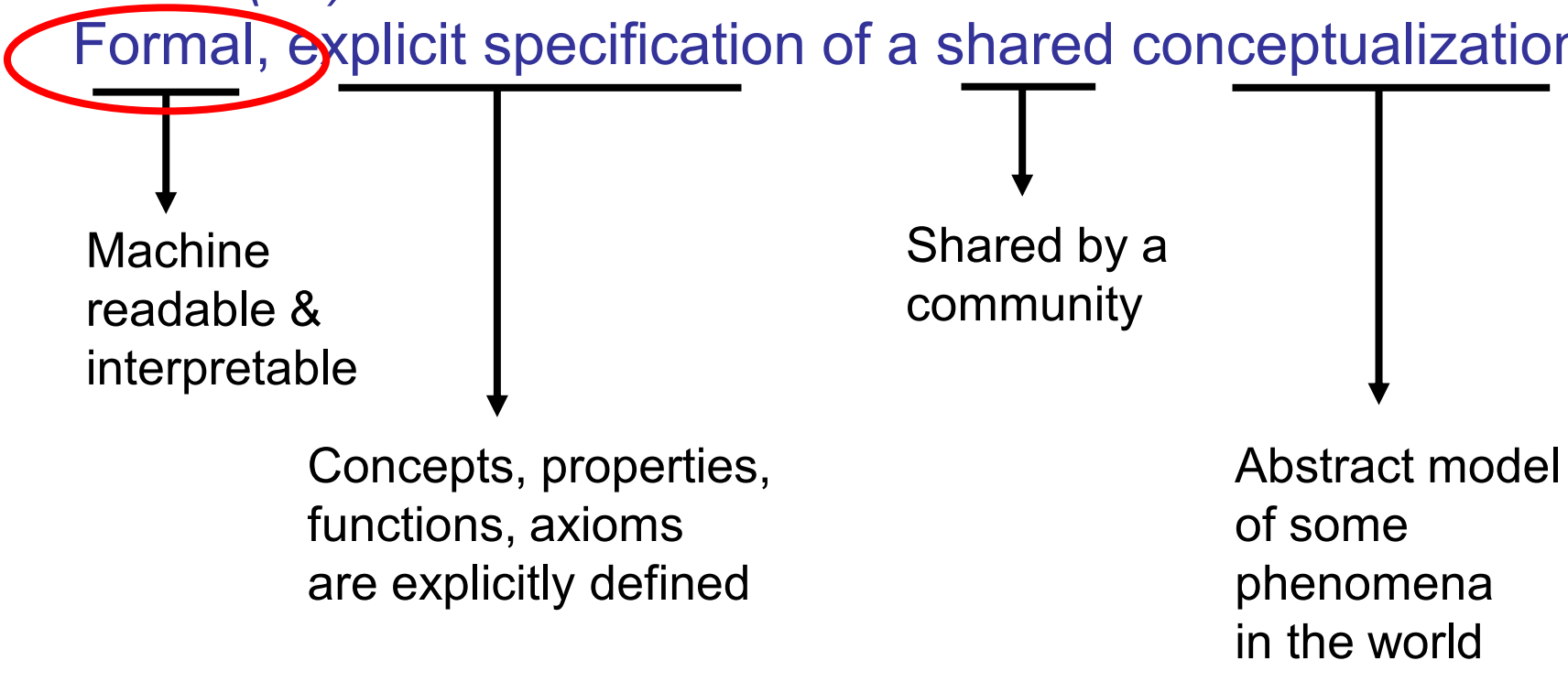
What Is An Ontology?
Description Logics

Emanuel Sallinger
Katja Hose

What is an ontology?

Studer(98):

Formal, explicit specification of a shared conceptualization.

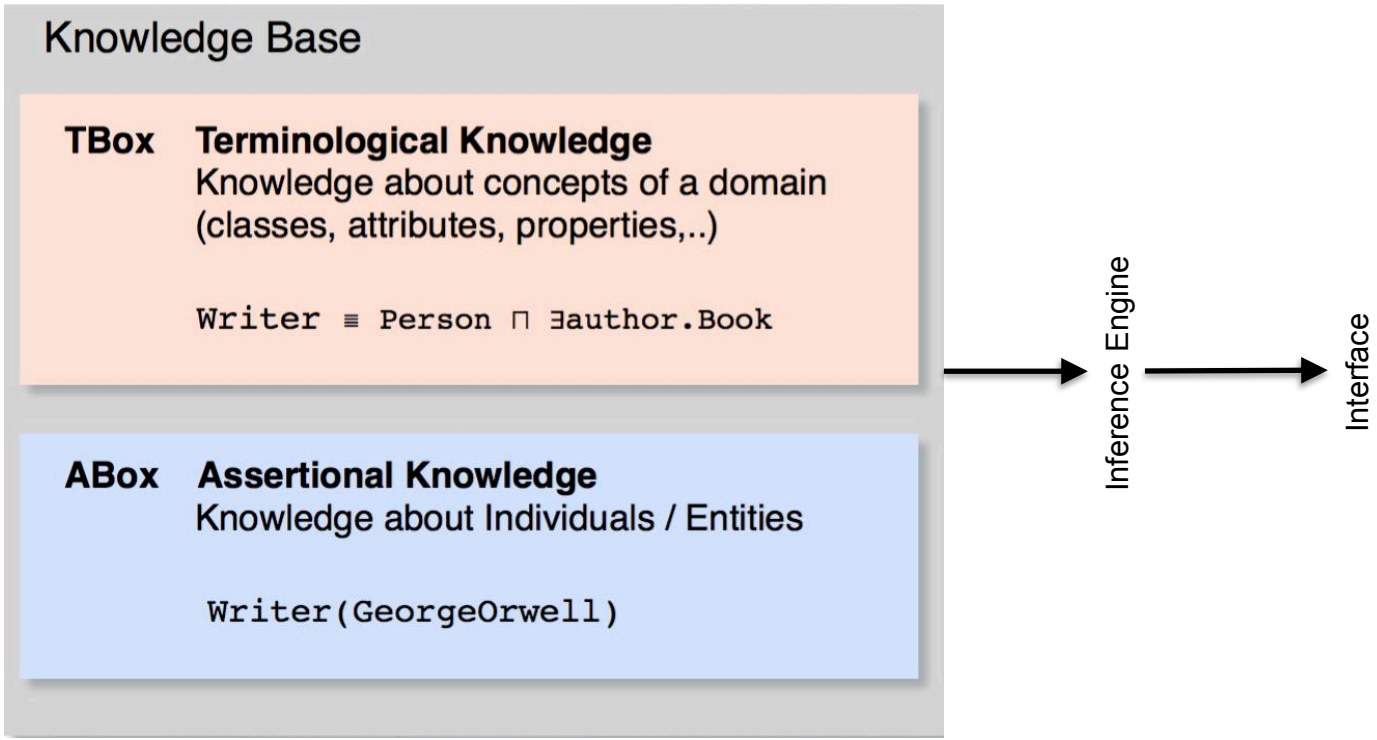


Logics as basis for Ontologies

- Ontologies should be “machine readable”
- … but computers cannot read and make sense of images

- Formal logics are used as basis for representing knowledge
- A formal, logics-based language provides:
  - A syntax (which expressions are correct)
  - A semantics (the meaning of expressions)
  - A calculus (how to determine the meaning of expressions)
  - …. which allow computers to draw “logical” conclusions from the data (i.e., to reason with data)
- Current Semantic Web languages are based on Description Logics (a subset of First Order Logics)

General Description Logic (DL) Architecture



Adapted from lecture slides "Knowledge Engineering with Semantic Web Technologies"
Dr. Harald Sack, Hasso-Plattner-Institut for IT Systems Engineering

DL notation/Syntax

Atomic Types

- Concept names A,B, ...
- Special concepts
  - ⊤ Top (universal concept)
  - ⊥ Bottom concept
- Role names R,S, ...

```
Writer(GeorgeOrwell)
Book(NineteenEightyFour)

Author(NineteenEightyFour,GeorgeOrwell)
```

Constructors

- Negation: ¬C
- Conjunction: C ⊓ D
- Disjunction: C ⊔ D
- Existential quantifier: ∃R.C
- Universal quantifier: ∀R.C

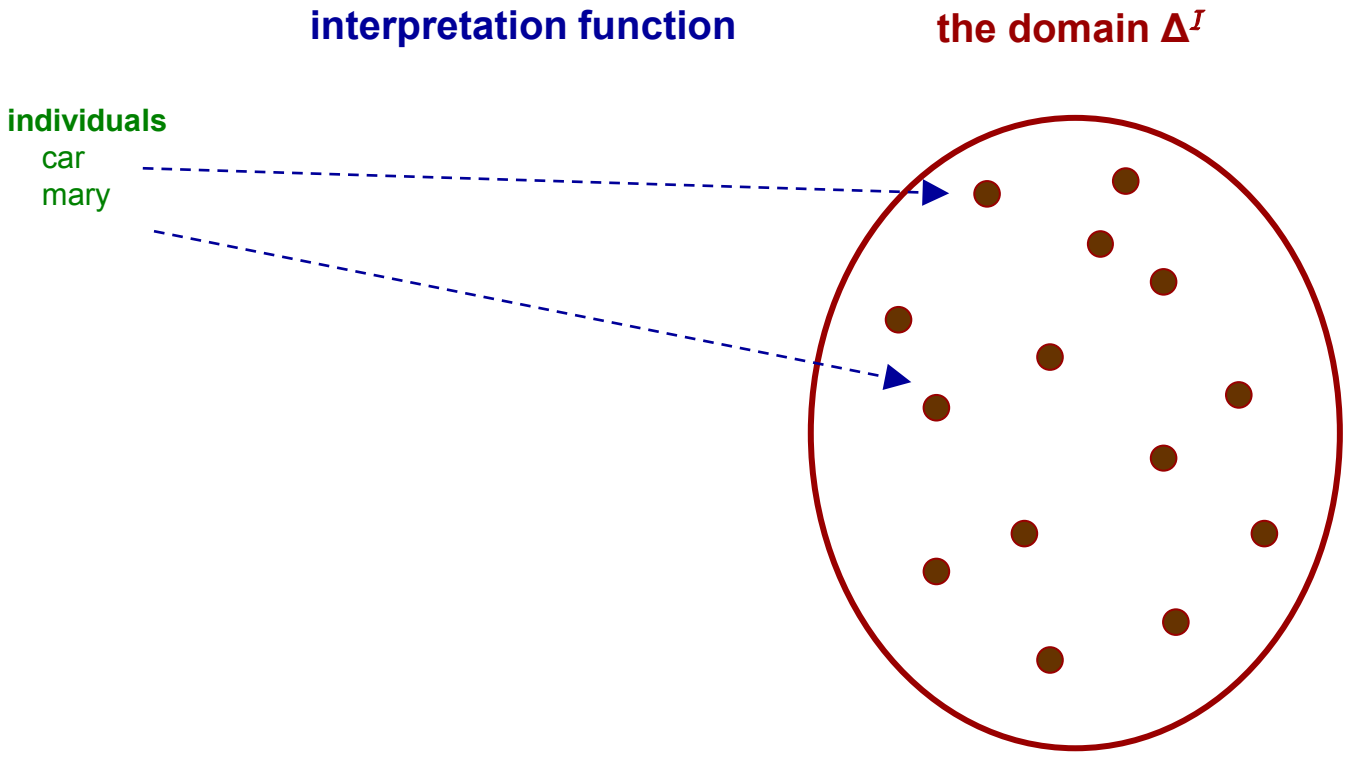
```
Novel ≡ Prose
Novel ⊑ Book
Herbivore ⊑ ¬ Carnivore

PetOwner ⊑ Person ⊓ ∃ hasPet.Animal
Carnivore ⊑ Animal ⊓ ∀eats.Animal
```

Adapted from lecture slides "Knowledge Engineering with Semantic Web Technologies"
Dr. Harald Sack, Hasso-Plattner-Institut for IT Systems Engineering

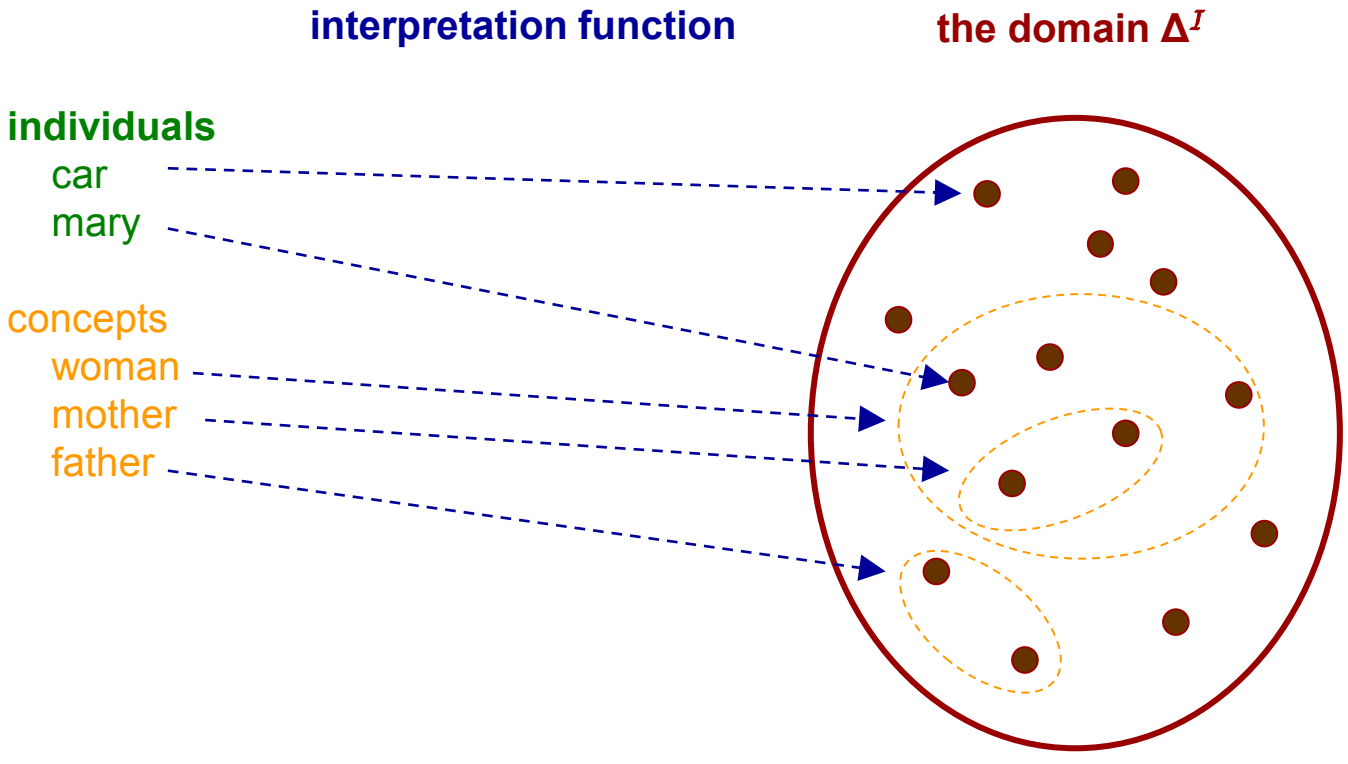
DL Semantics

- Model theoretic semantics: An interpretation consists of
  - A domain of discourse (a collection of objects) **Δ**
  - Interpretative Functions **I** mapping
    - classes to sets of objects
    - properties to sets of pairs of objects
- In a DL, a class description is thus a characterization of the individuals that are members of that class

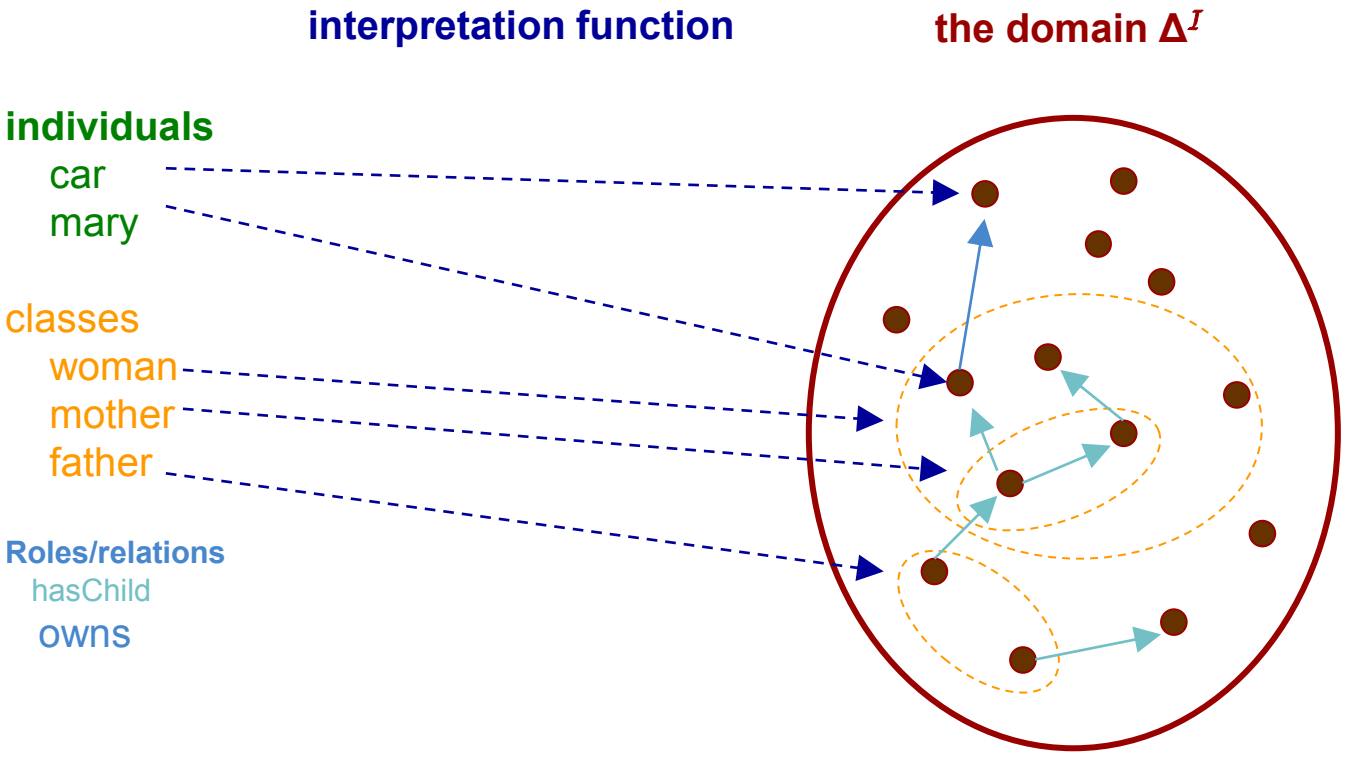
Description Logics Semantics (1/3)



Description Logics Semantics (2/3)



Description Logics Semantics (3/3)



TU WIEN Informatics

What Is An Ontology?
Description Logics

Emanuel Sallinger
Katja Hose

TU WIEN Informatics

How To Build an Ontology?

Ontology Engineering

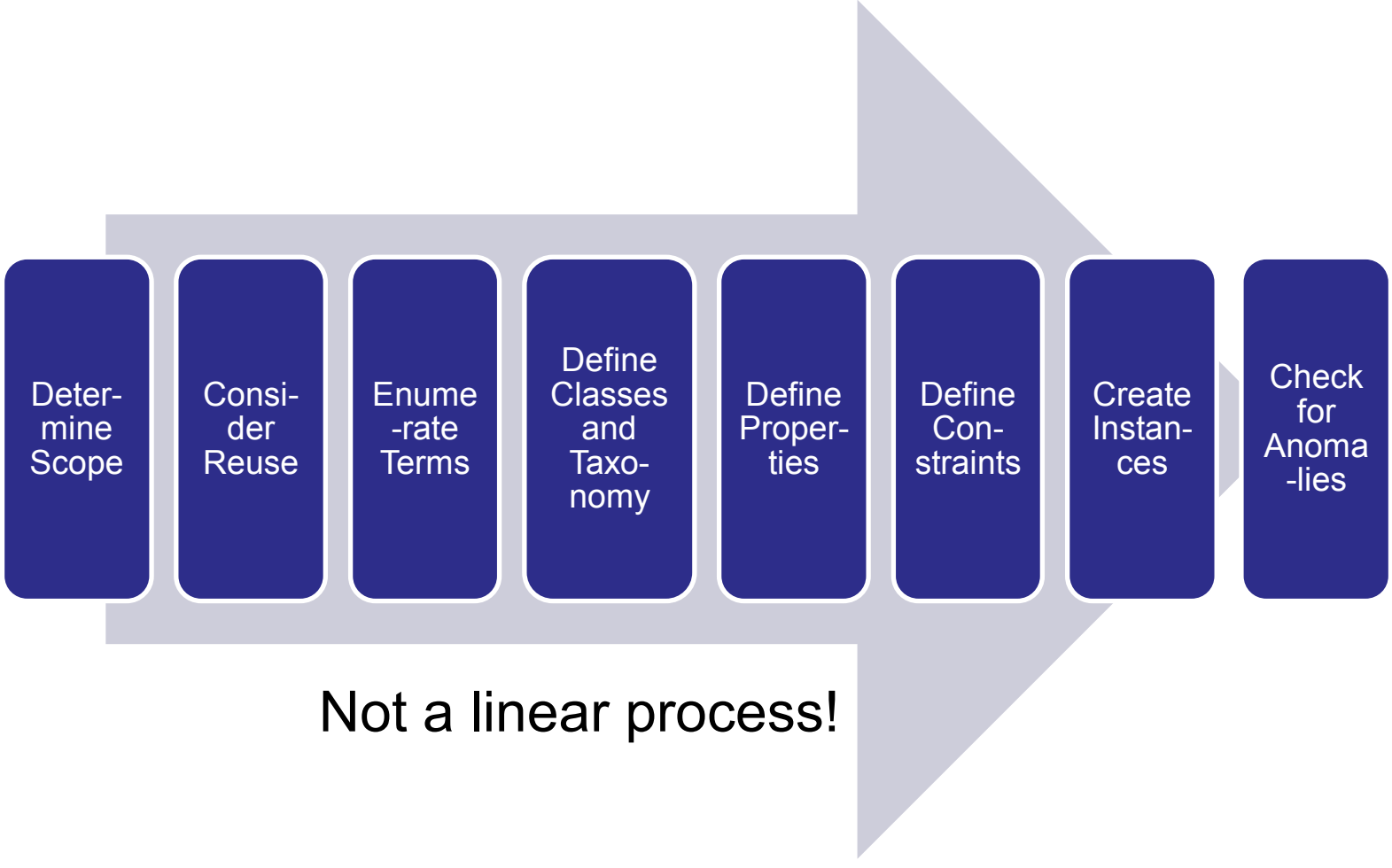
Emanuel Sallinger
Katja Hose

Ontology Engineering

- All activities involved in the development of ontologies
- Methodologies for:
  - specification and design of ontologies (**Ontology Design)**
  - comparison of ontologies **(Ontology Mapping)**
  - combination of ontologies (**Ontology Merging)**
  - retrieving ontologies from a set of information resources (**Ontology Learning)**
  - populating ontologies with individuals from information resources (**Ontology Population)**

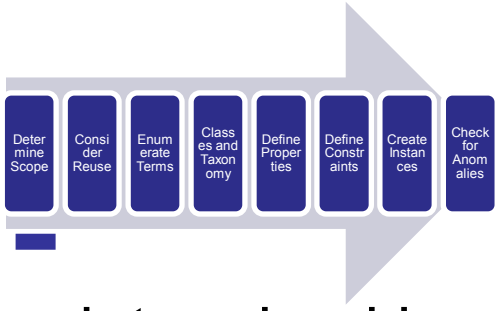
Ontology Development 101

(Noy and McGuinness, 2000)



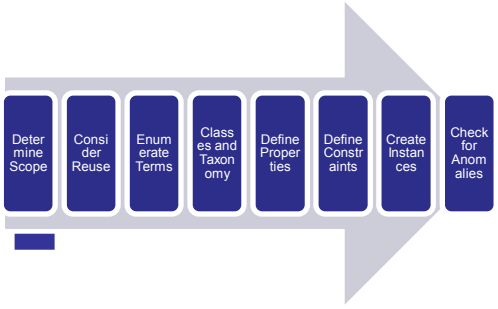
Not a linear process!

Determine Scope



- What is included in the ontology should be determined by:
  - the uses of the ontology
  - future extensions that are already anticipated
- Basic questions to be answered at this stage:
  - What is the **domain** that the ontology will cover?
  - For what are we going to **use** the ontology?
  - For what **types of questions** should the ontology provide answers?
    - Called “Competency Questions”
  - **Who** will use and maintain the ontology?

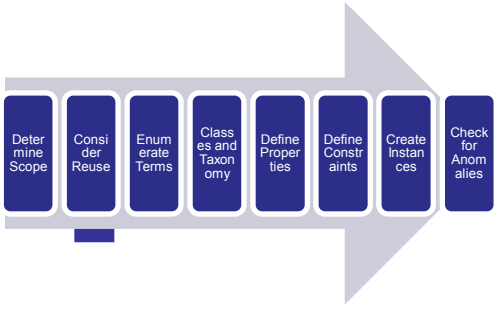
Determine Scope: Competency Questions



Competency questions for a wine recommender system:

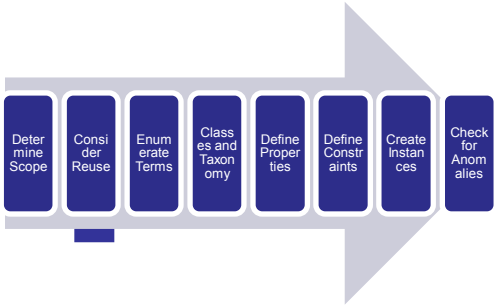
- Which wine characteristics should I consider when choosing a wine?
- Is Bordeaux a red or white wine?
- Does Cabernet Sauvignon go well with seafood?
- What is the best choice of wine for grilled meat?
- Which characteristics of a wine affect its appropriateness for a dish?
- Does a bouquet or body of a specific wine change with vintage year?
- What were good vintages for Napa Zinfandel?

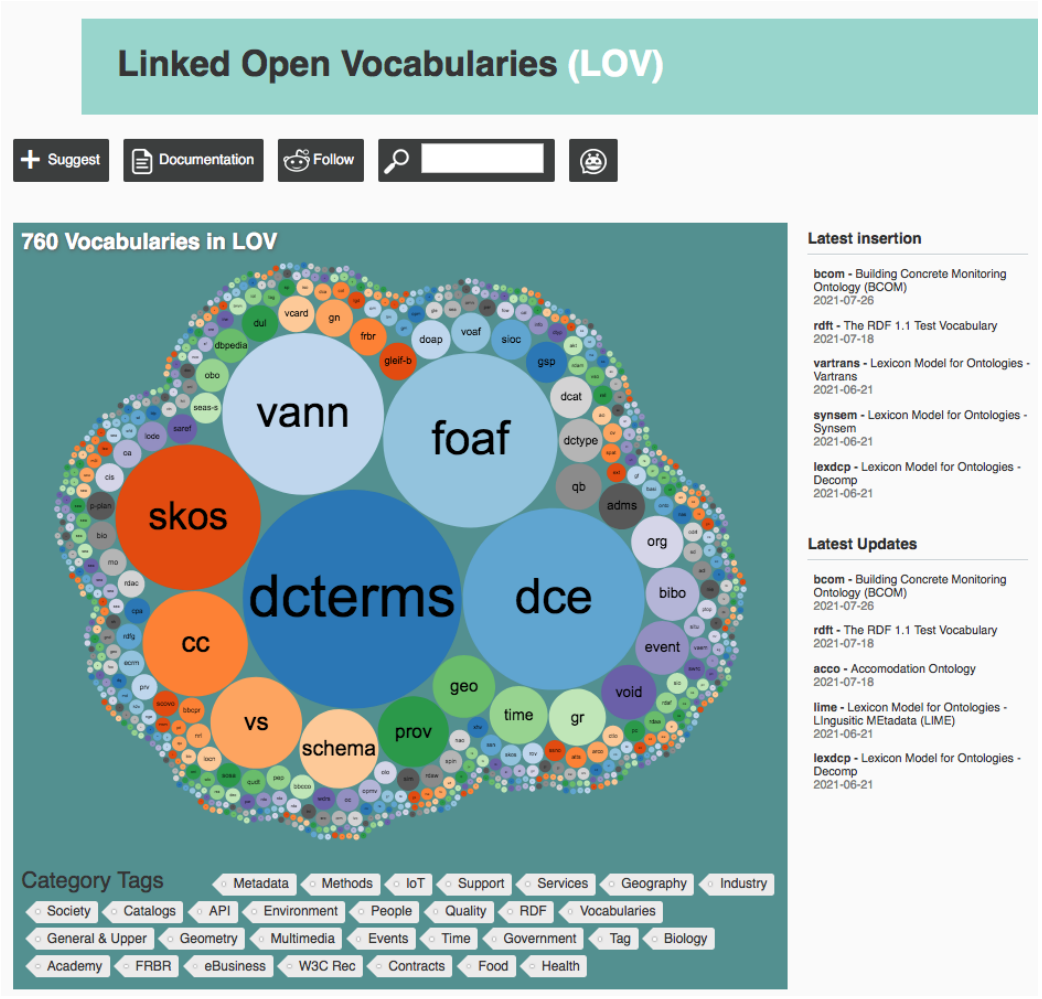
Consider Reuse



- Why reuse other ontologies?
  - to save ontology engineering effort
  - to interact with the tools that use other ontologies
  - to use ontologies that have been validated through use in applications
- Ontology libraries
  - Protégé ontology library (http://protege.stanford.edu)
  - NCBO Bioportal (http://bioontology.org)
  - http://ontologydesignpatterns.org/wiki/Ontology:Main

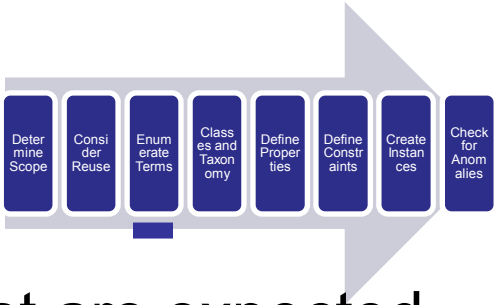
Consider Reuse





https://lov.linkeddata.es, *accessed 13.10.21*

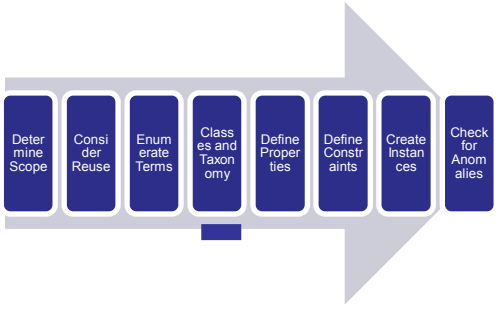
Enumerate Terms



- Write down a list of all the relevant terms that are expected to appear in the ontology
  - What are the terms we need to talk about?
    - Nouns form the basis for class names
  - What are the properties of these terms?
    - Verbs (or verb phrases) form the basis for property names

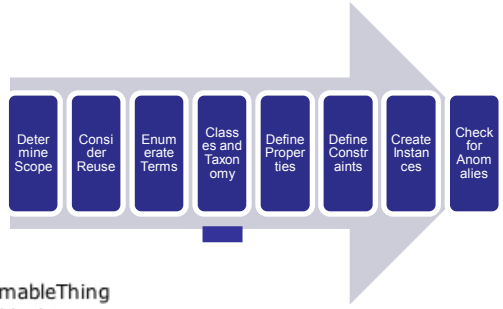
- Examples: The Wine ontology
  - wine, grape, winery, location,
  - wine color, wine body, wine flavor, sugar, content
  - white wine, red wine, Bordeaux wine

Define Classes and a Taxonomy

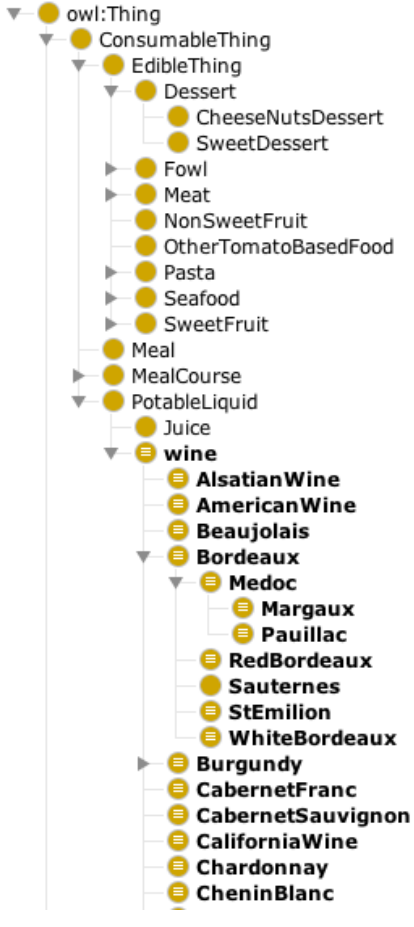


- A class is a concept in the domain
  - a class of wines
  - a class of wineries
  - a class of red wines
- A class is a collection of elements with similar properties
- Instances of classes
  - a glass of California wine you'll have for lunch

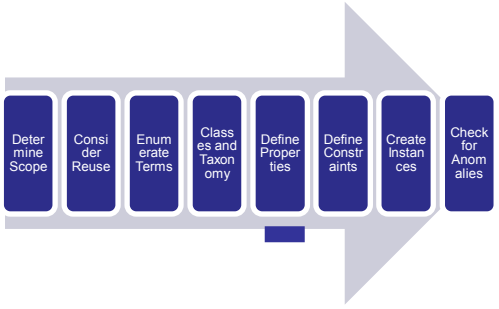
Define Classes and a Taxonomy



- Relevant terms must be organized in a taxonomic hierarchy
  - If A is a subclass of B, then every instance of A must also be an instance of B
- Example
  - *Red wines is a subclass of Wine*
    - *Every red wine is a wine*

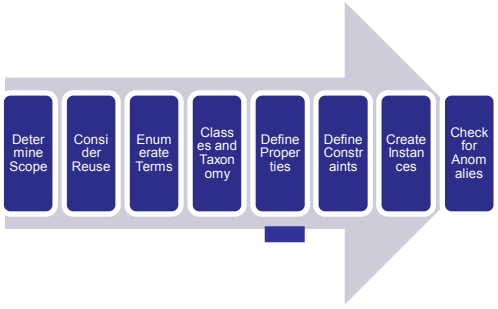


Define Properties



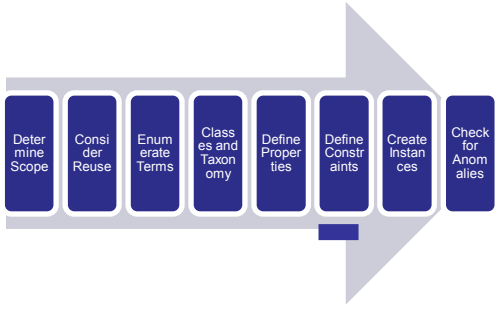
- Often interleaved with the previous step
- Properties in a class definition:
  - describe attributes of instances of the class
    - each wine has color, sugar content, producer, etc.
  - describe relations to other classes:
    - each Wine has maker Winery
    - „parent class“ is the **domain** of property, e.g., Wine
    - „child class“ is the **range** of the property, e.g., Winery
- If A is a subclass of B, every property statement that holds for instances of B must also apply to instances of A
- Properties shared in a hierarchy should be specified in the highest possible class

Define Properties: Property Hierarchies



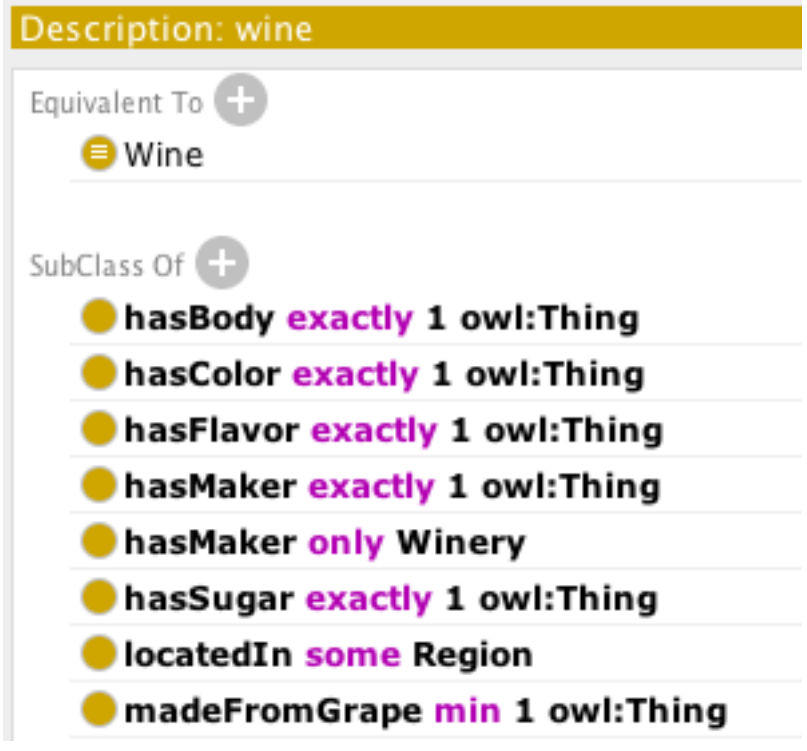
- Properties can be arranged in hierarchies too.

Define Constraints: Cardinality

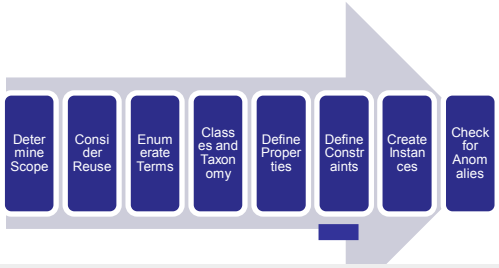


- Cardinality restrictions
  - "each wine is made from at least one grape "
  - "each wine has exactly one maker "

$$\mathit{Wine} \sqsubseteq {>}1\mathit{madeFromGrape}$$
$$\mathit{Wine} \sqsubseteq {=}1\mathit{hasMaker}$$



Define Constraints: Required Values

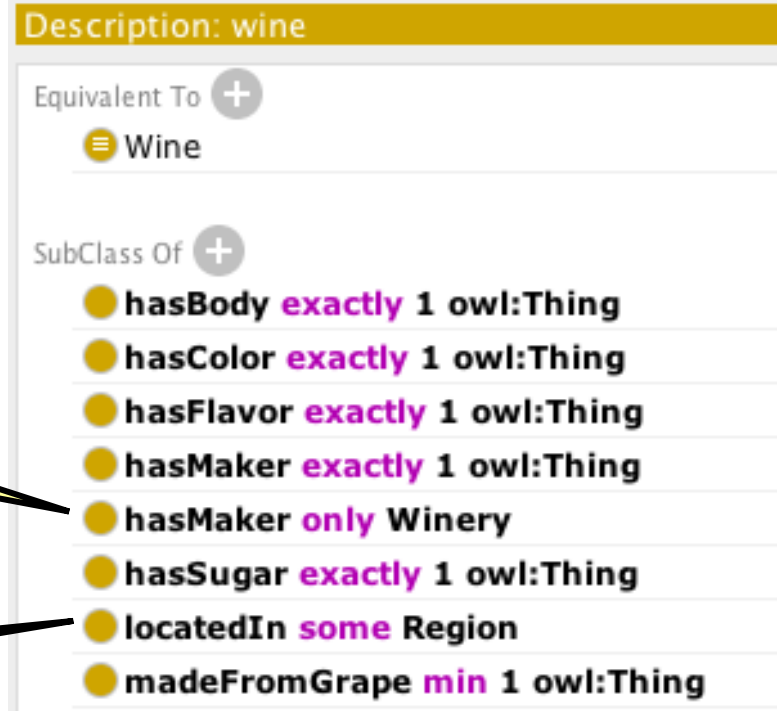


- Universal restriction (all values from):
  - If a property is declared for an instance, all its values must be of a certain type
  - Wines can only be made in Wineries

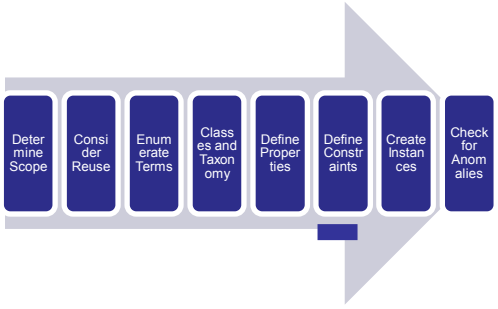
Wherever there is a **hasMaker** property, all its values must be instances of **Winery**.
[There may be zero **hasMaker** properties.]

- Existential restriction (some values from):
  - There **must** be at least one property with that value
  - E.g., Wines are located in regions

There must be at least one **locatedIn** property whose value is **Region**.
[There must be at least one **locatedIn** property.]

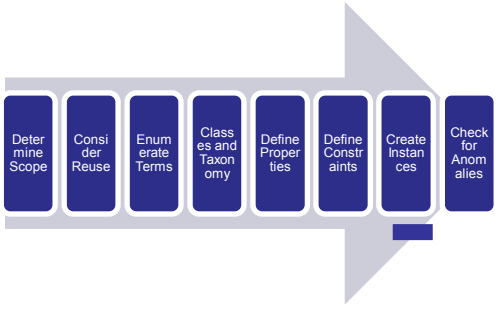


Define Constraints: Property Characteristics



- Symmetry
  - If P(x,y) =>P(y,x)
  - *E.g., marriedTo, friendOf, adjacentRegion*
- Transitivity
  - If P(x,y) and P(y,z) => P(x, z)
  - *E.g., locatedIn, hasAncestor*
- Inverse properties
  - If P(x,y) => InvP(y,x) holds
  - *E.g., hasParent(John, Bob)=> hasChild(Bob, John)*
- Functional values
  - can have only one (unique) value y for each instance x
  - i.e. there cannot be two distinct values y1 and y2 such that the pairs (x,y1) and (x,y2) are both instances of this property
  - E.g., hasHusband

Create Instances



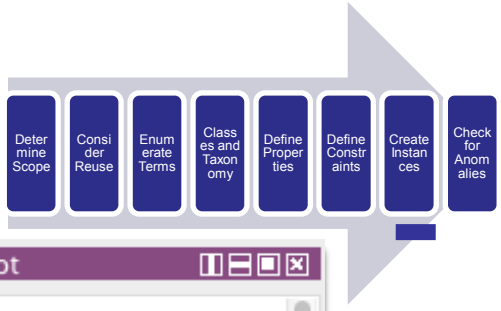
- Create an instance of a class
  - The class becomes a direct type of the instance
  - Any superclass of the direct type is a type of the instance
- Assign property values for the instance frame
  - Property values should conform to the constraints
  - Knowledge-acquisition tools often check it

- Number of instances >> number of classes

- Populating an ontology with instances is usually not done manually
  - retrieved from legacy data sources
  - extracted automatically from a text corpus

See Lecture 6 on KG creation

Create Instances



Description: GaryFarrellMerlot

Types
- Merlot

Same Individual As

Different Individuals

Property assertions: GaryFarrellMerlot

Object property assertions
- hasFlavor Moderate
- hasSugar Dry
- hasMaker GaryFarrell
- locatedIn SonomaRegion
- hasBody Medium

Description: MountadamRiesling

Types
- DryRiesling

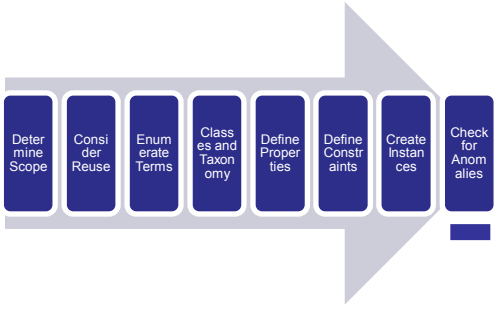
Same Individual As

Different Individuals

Property assertions: MountadamRiesling

Object property assertions
- hasMaker Mountadam
- hasFlavor Delicate
- hasSugar Dry
- locatedIn SouthAustraliaRegion
- hasBody Medium

Check for Anomalies



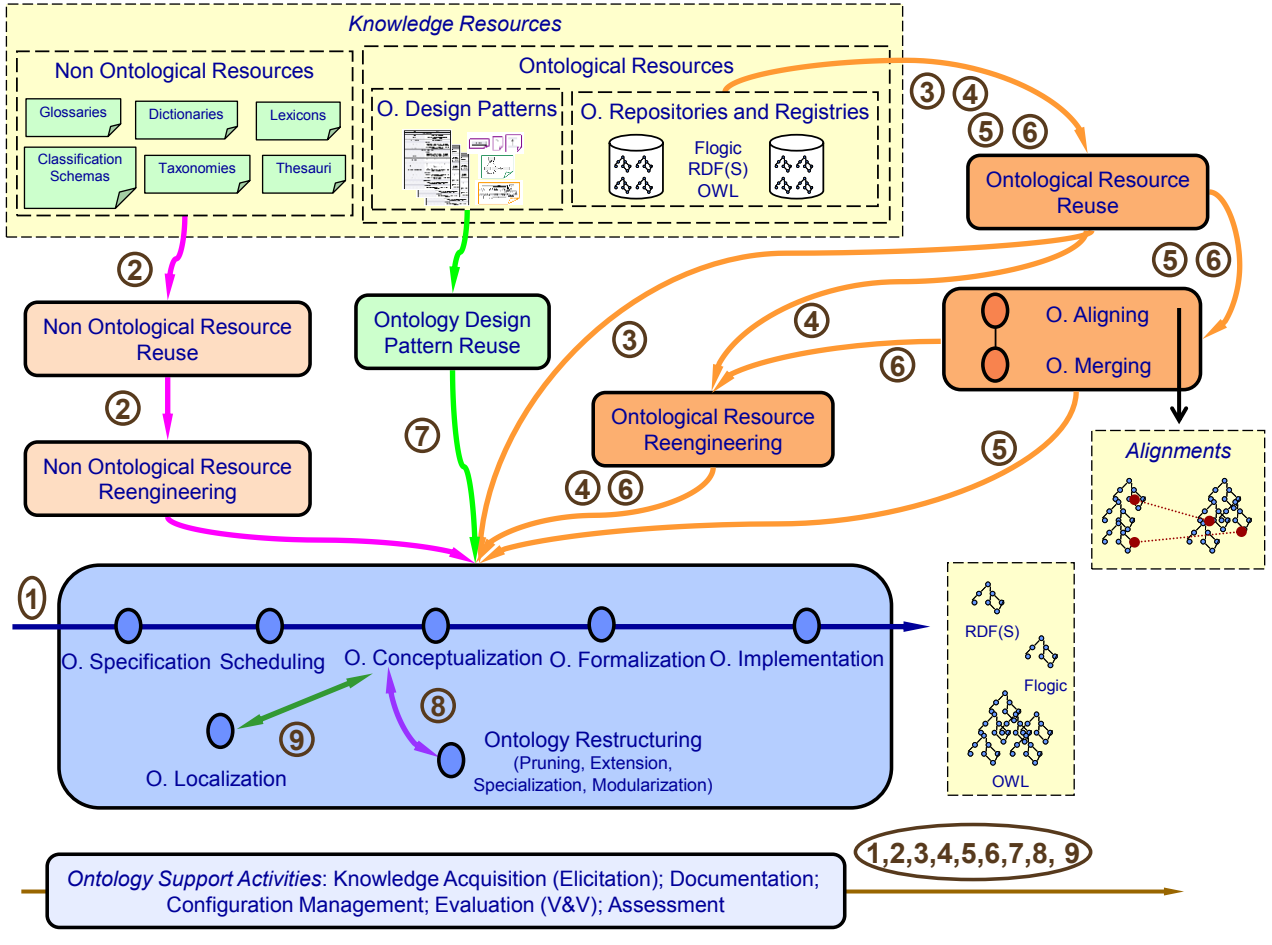
Exploits the formal semantics of ontologies to:

1) Validate ontology models
   - Consistency checking
   - Satisfiability

2) Derive additional knowledge
   - Classification

See “Reasoning” part of Lecture

NeOn Methodology - a much more complex picture



Suárez-Figueroa M.C., Gómez-Pérez A., Fernández-López M. (2012) The NeOn Methodology for Ontology Engineering. Ontology Engineering in a Networked World. Springer

TU WIEN Informatics

How To Build an Ontology?

Ontology Engineering

Emanuel Sallinger
Katja Hose

TU WIEN Informatics

What Can I Do With An Ontology?

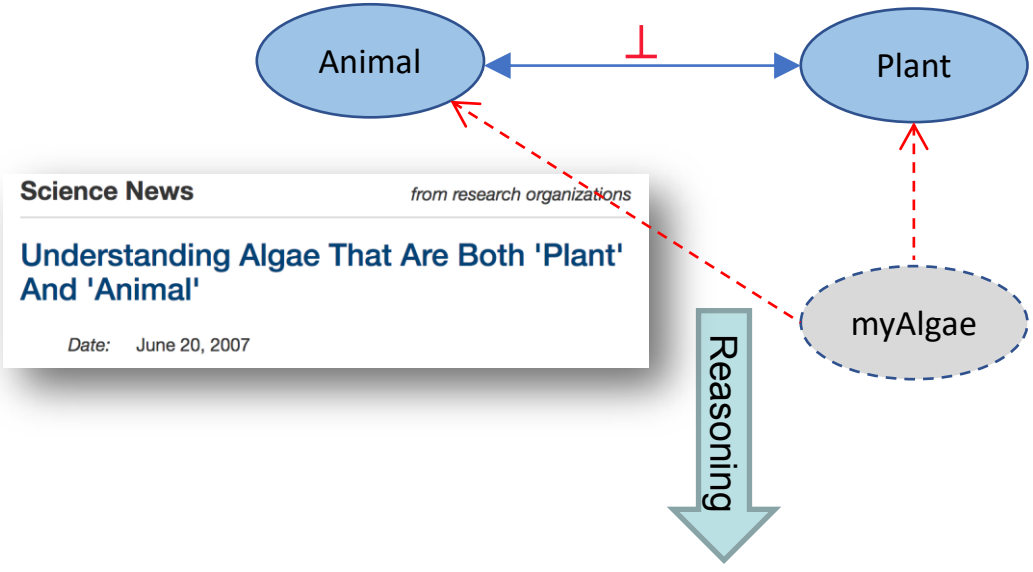
Reasoning

Emanuel Sallinger
Katja Hose

Reasoning with Ontologies

- Thanks to the formal nature of ontologies (their grounding in logic), **reasoning** can be used to **infer** new knowledge based on already declared information.
- Reasoning is performed with **reasoners** (**inference engines**).
- Reasoning allows for, e.g.,:
  - **Consistency checking**: Checking the correctness of the ontology model (e.g., that the model described by the ontology is free of logical contradictions)
  - **Class inferences**: discovering relation between classes that were not explicitly declared.
  - **Instance inferences**: discovering class membership of instances.

Consistency Checking



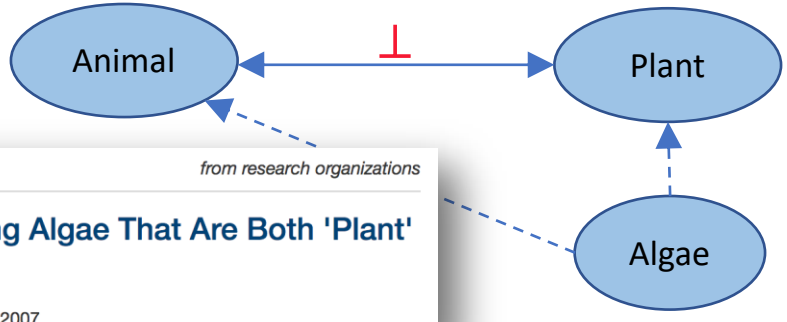
$\mathtt{Animal} \sqsubseteq \neg \mathtt{Plant}$
$\mathtt{Plant(myAlgae)}$

$\mathtt{Animal(myAlgae)}$

Inconsistent ontology: There are logical errors in the ontology

Satisfiability Checking

Are there classes that cannot possibly have any instances?



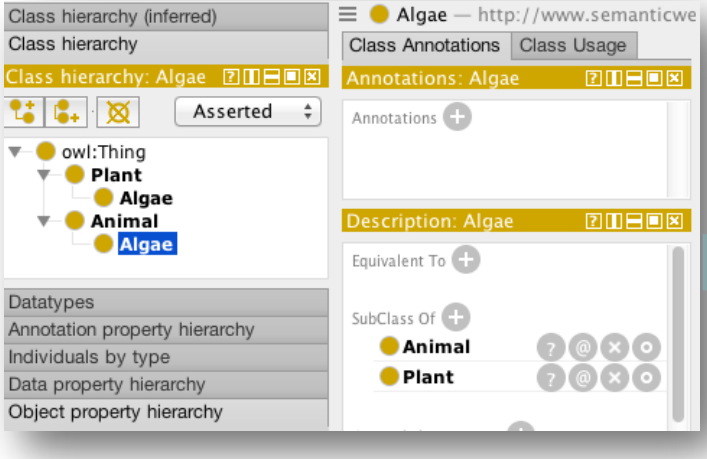
Science News — *from research organizations*

Understanding Algae That Are Both 'Plant' And 'Animal'

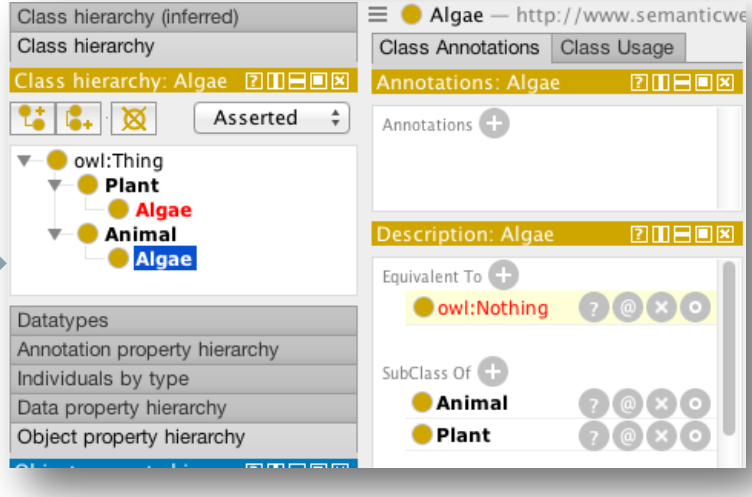
Date: June 20, 2007

$$\mathtt{Algae} \sqsubseteq \mathtt{Plant}$$
$$\mathtt{Animal} \sqsubseteq \neg\,\mathtt{Plant}$$
$$\mathtt{Algae} \sqsubseteq \mathtt{Animal}$$

Unsatisfiable Concept: if there is a contradiction in the ontology that implies that the class cannot have any instances

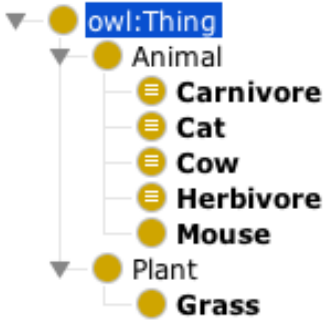


Reasoning

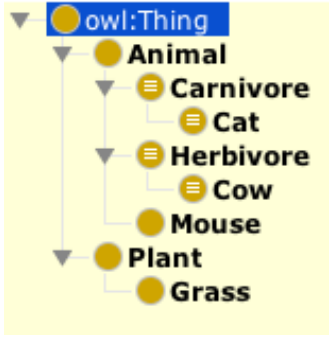


Class Inference

Discover new knowledge based on the given knowledge.



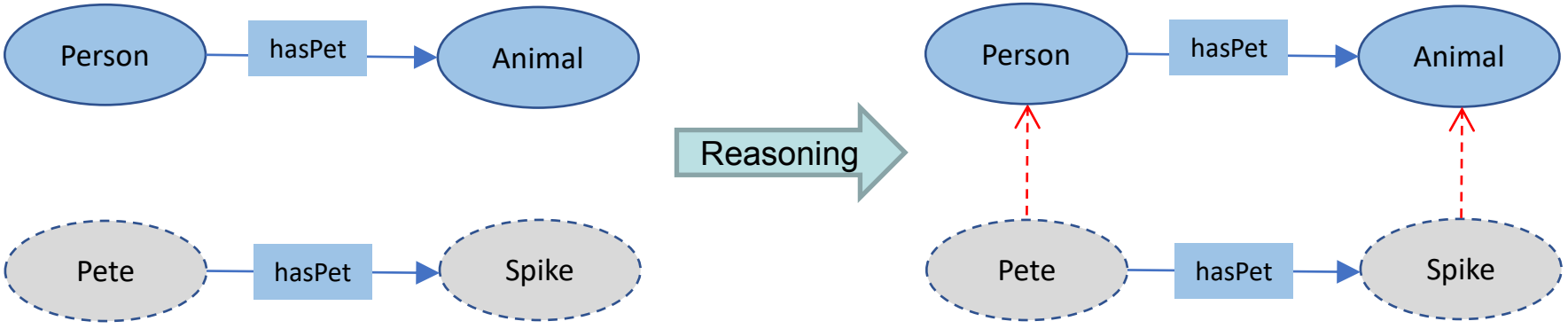
Reasoning



$\mathtt{Carnivore} \sqsubseteq \mathtt{Animal} \sqcap \forall \mathtt{eats.Animal}$
$\mathtt{Herbivore} \sqsubseteq \mathtt{Animal} \sqcap \forall \mathtt{eats.Plant}$
$\mathtt{Cat} \sqsubseteq \mathtt{Animal} \sqcap \forall \mathtt{eats.Mouse}$
$\mathtt{Cow} \sqsubseteq \mathtt{Animal} \sqcap \forall \mathtt{eats.Grass}$
$\mathtt{Mouse} \sqsubseteq \mathtt{Animal}$
$\mathtt{Grass} \sqsubseteq \mathtt{Plant}$

$\mathtt{Cat} \sqsubseteq \mathtt{Carnivore}$
$\mathtt{Cow} \sqsubseteq \mathtt{Herbivore}$

Instance Inference

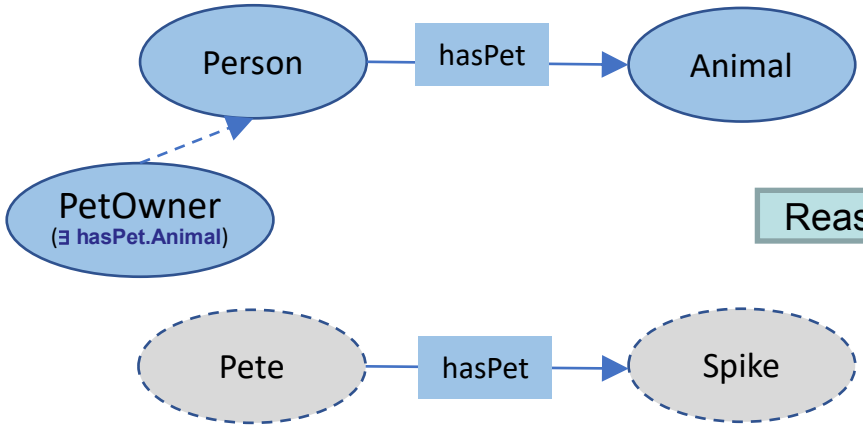



```
hasPet(Person, Animal)

Thing(Pete)
Thing(Spike)
hasPet(Pete,Spike)
```

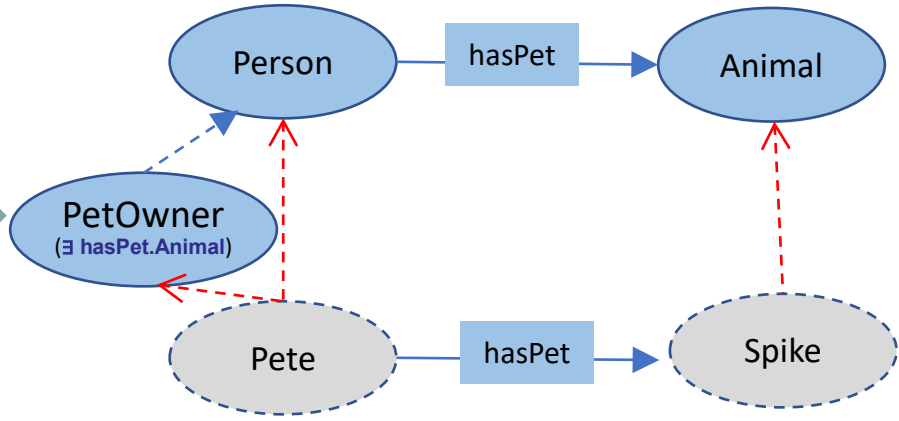
```
Person(Pete)
Animal(Spike)
```

Adapted from Sean Bechhofer, *OWL Reasoning Example*, http://owl.man.ac.uk/2003/why/latest/

Instance Inference



Reasoning

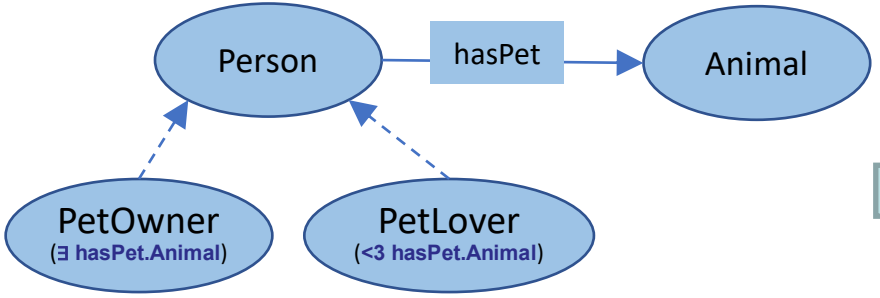



```
hasPet(Person, Animal)
PetOwner ⊑ Person ⊓ ∃ hasPet.Animal

Thing(Pete)
Thing(Spike)
hasPet(Pete,Spike)
```

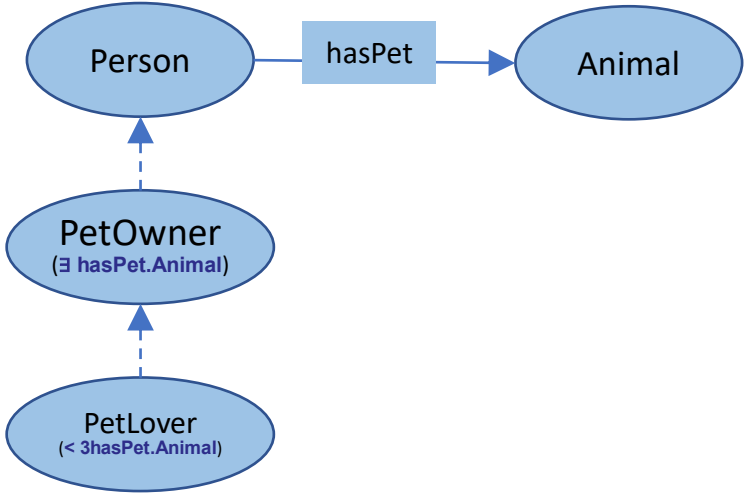
```
Person(Pete)
Animal(Spike)

PetOwner(Pete)
```

Adapted from Sean Bechhofer, *OWL Reasoning Example*, http://owl.man.ac.uk/2003/why/latest/

Class+Instance Inference



Reasoning



$\mathtt{hasPet(Person, Animal)}$
$\mathtt{PetOwner} \sqsubseteq \mathtt{Person} \sqcap \exists\, \mathtt{hasPet.Animal}$
$\mathtt{PetLover} \sqsubseteq \mathtt{Person} \sqcap {>}3\mathtt{hasPet.Animal}$

$\mathtt{Person(Pete)}$
$\mathtt{Animal(Tom),\ Animal(Jerry),\ Animal(Spike)}$
$\mathtt{hasPet(Pete,Spike)}$
$\mathtt{hasPet(Pete,Tom)}$
$\mathtt{hasPet(Pete,Jerry)}$
$\{\mathtt{Spike}\} \sqsubseteq \neg\{\mathtt{Tom, Jerry}\}$

$\mathtt{PetOwner(Pete)}$
$\mathtt{PetLover(Pete)}$
$\mathtt{PetLover} \sqsubseteq \mathtt{PetOwner}$

Adapted from Sean Bechhofer, *OWL Reasoning Example*, http://owl.man.ac.uk/2003/why/latest/

TU WIEN Informatics

What Can I Do With An Ontology?

Reasoning

Emanuel Sallinger
Katja Hose

TU WIEN Informatics

Ontology Editors in Practice

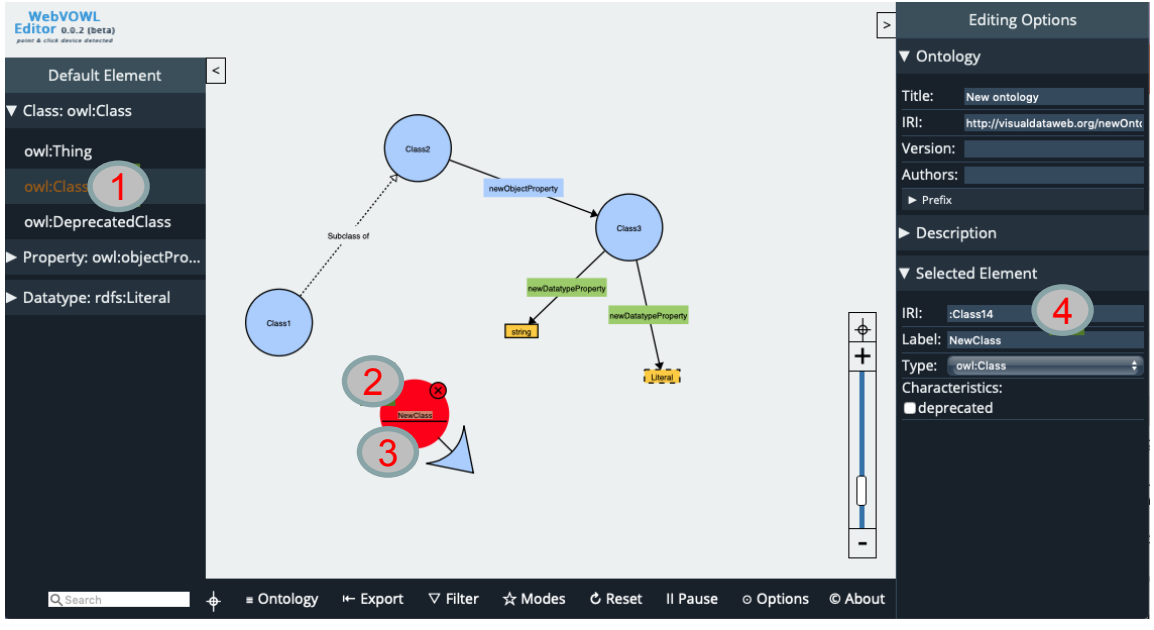
Emanuel Sallinger
Katja Hose

Practical Ontology Engineering

- To create your ontologies you can make use of:
  - WebVOWL (a light-weight, web-based editor)
  - Protégé (a complete ontology editor)
- These slides provide instructions to get started with these tools

Introduction to WebVOWL Editor: Creating a Class

1. Select one of the available class construct in the left panel
2. Double click anywhere on the screen
3. Edit the class name and label by double clicking on the screen OR
4. Use editing options from the right panel;
   Selected Element -> IRI and Label.

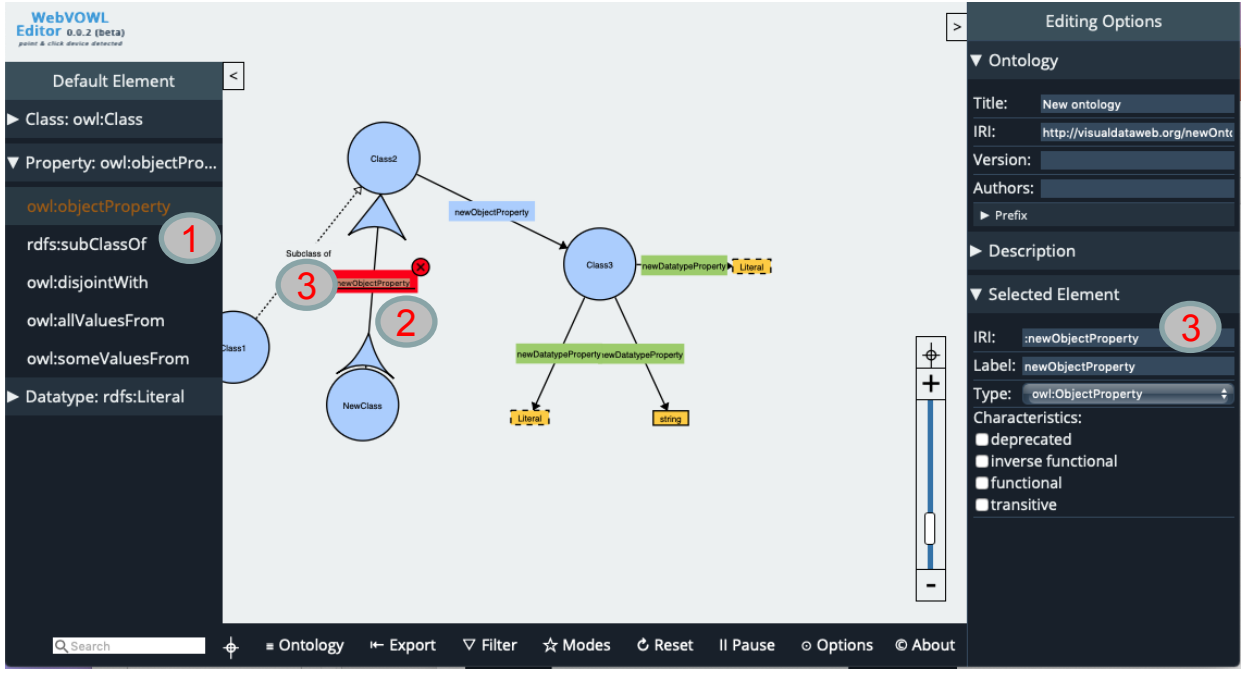


!Attention!

- Can be cumbersome
- Only limited editing functionality)

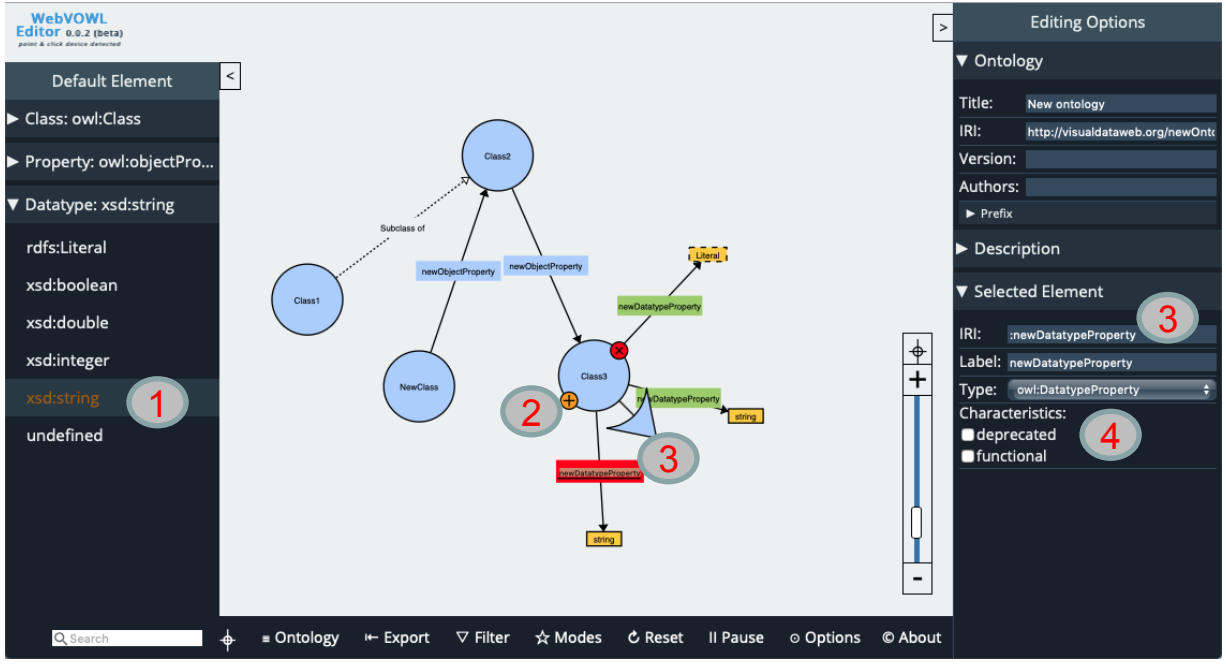
Introduction to WebVOWL Editor: Creating an Object Property

1. Select one of the available construct of Object Property in the left panel
2. Click on the domain (class) of the object property, and drag the arrow to the range (class) of the object property
3. Rename the property by double clicking the property name OR click on the property and edit it in the right panel.



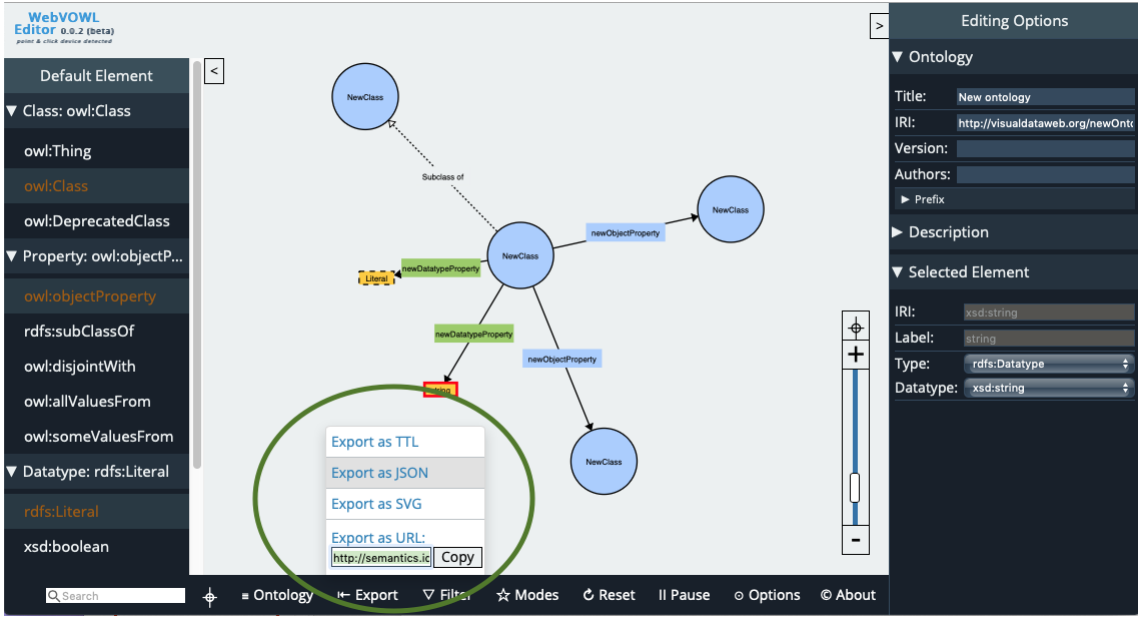
Introduction to WebVOWL Editor: Creating an Datatype Property

1. Select one of the available construct of Datatype Property in the left panel
2. Click on the plus (+) button on the domain (class)
3. Rename the property by double clicking the property name OR click on the property and edit it in the right panel.
4. You can set the datatype by changing the value on the right panel



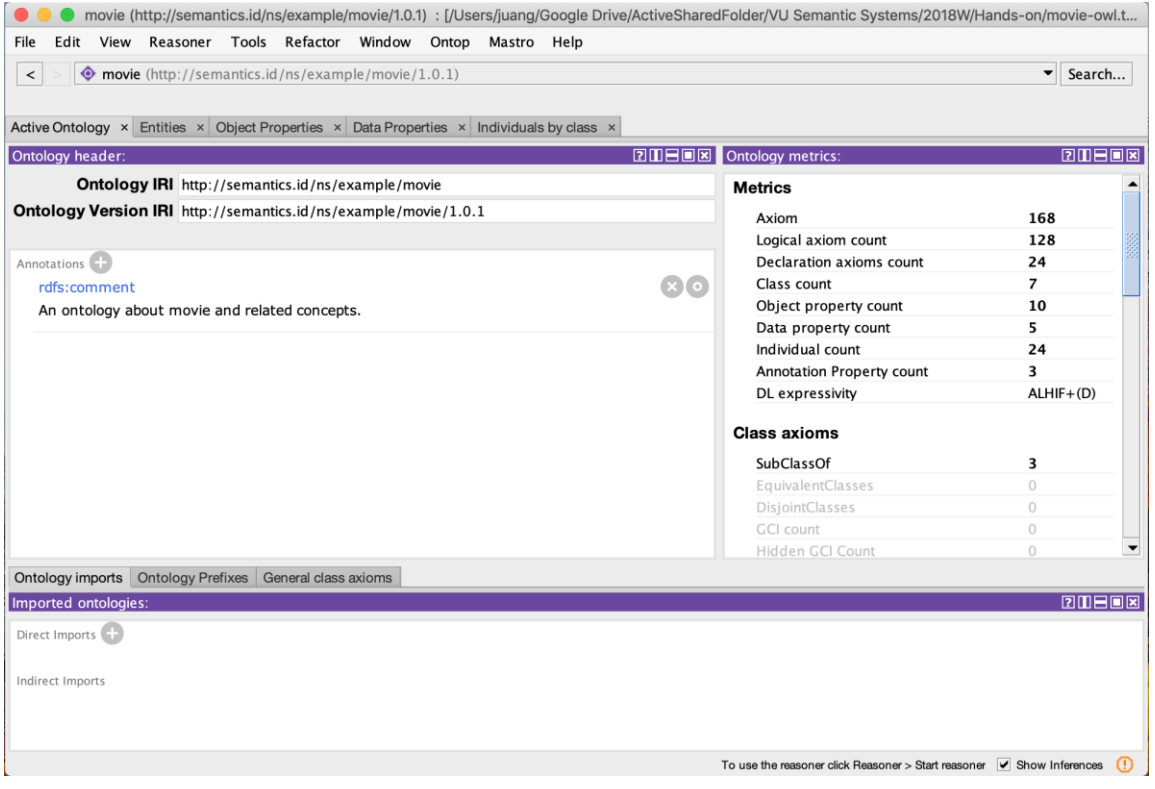
Introduction to WebVOWL Editor: Export Function

- Export as TTL and SVG
  - Bottom menu -> export -> as TTL/SVG
  - JSON and URL exports don’t seem to work at the moment



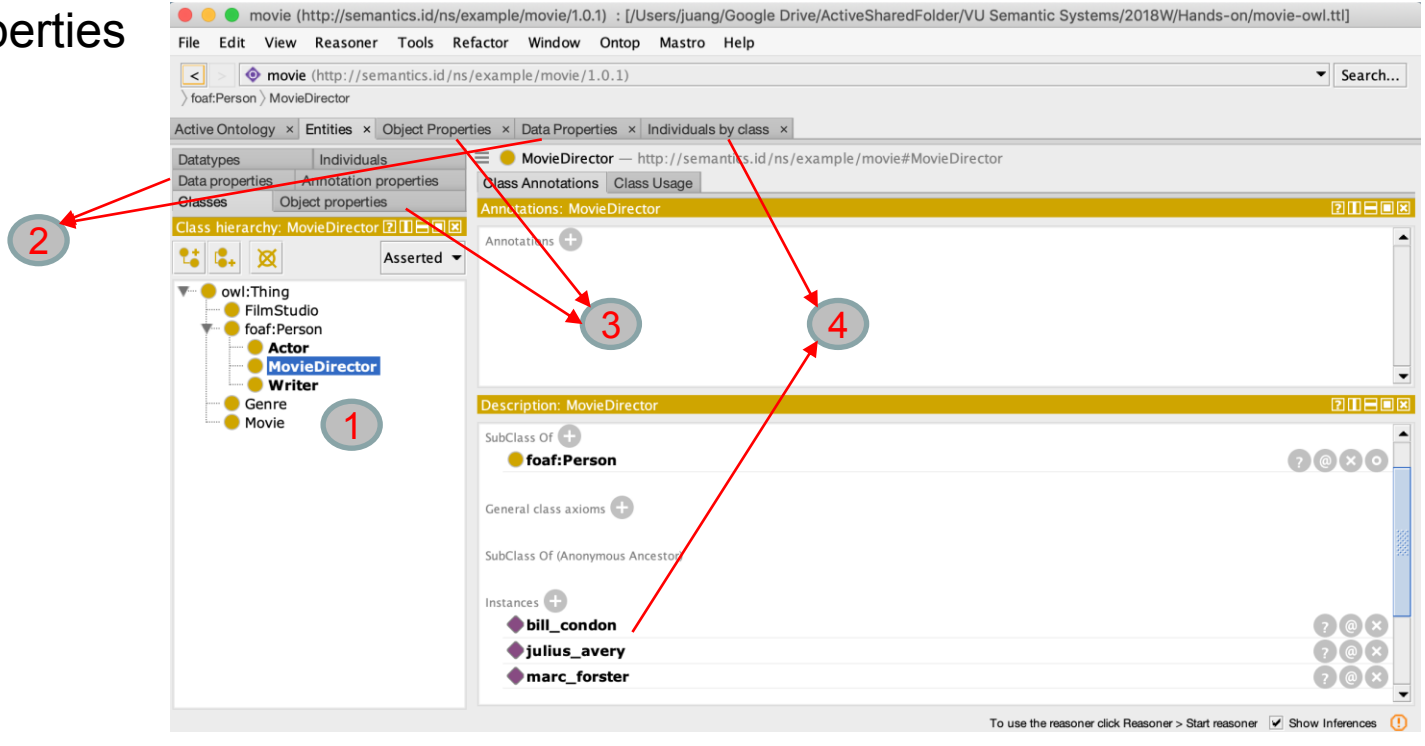
Protégé

- Load your ontology file:
  - File -> Open

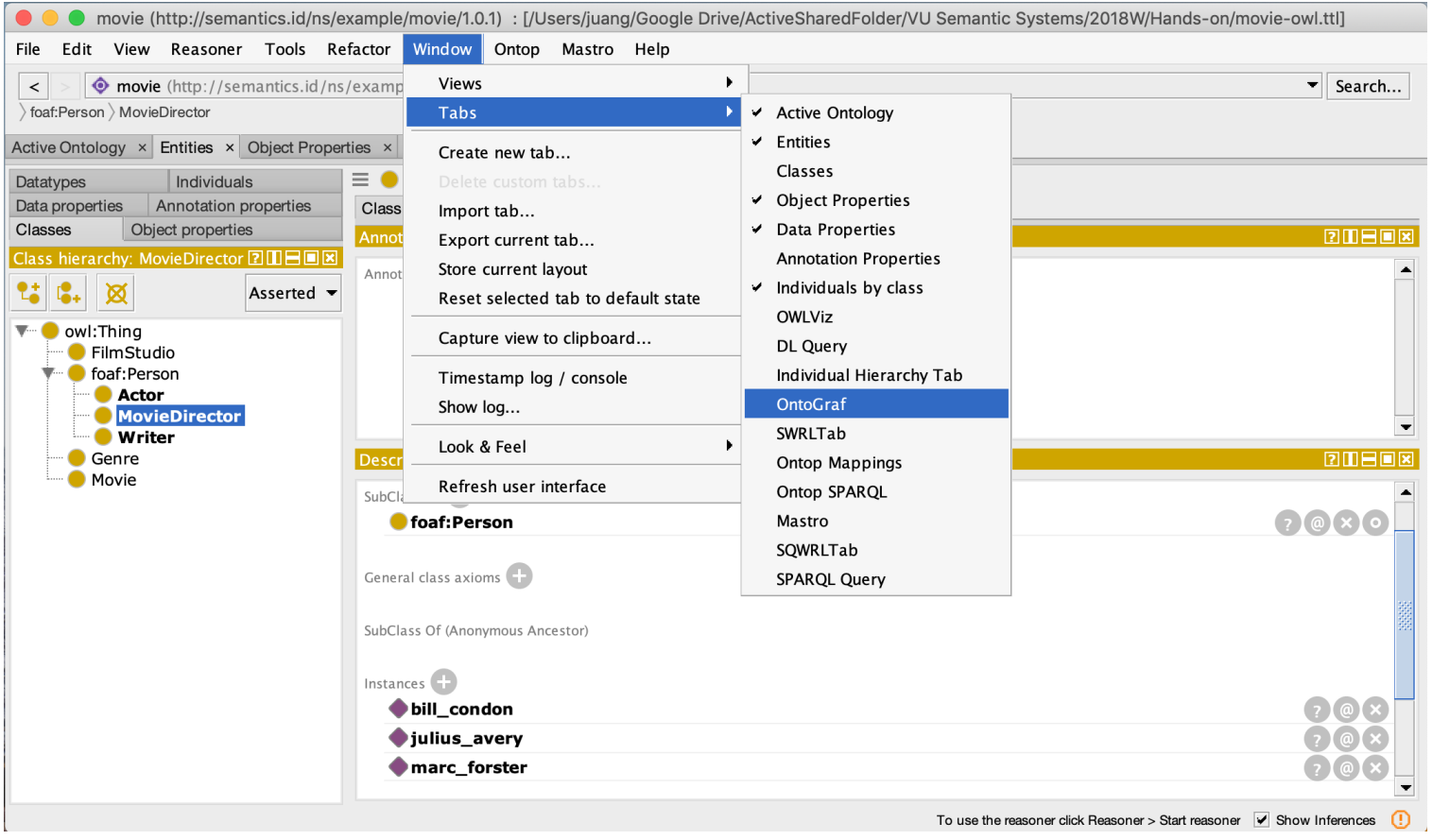


Protégé Editor: Classes, Properties, and Instances

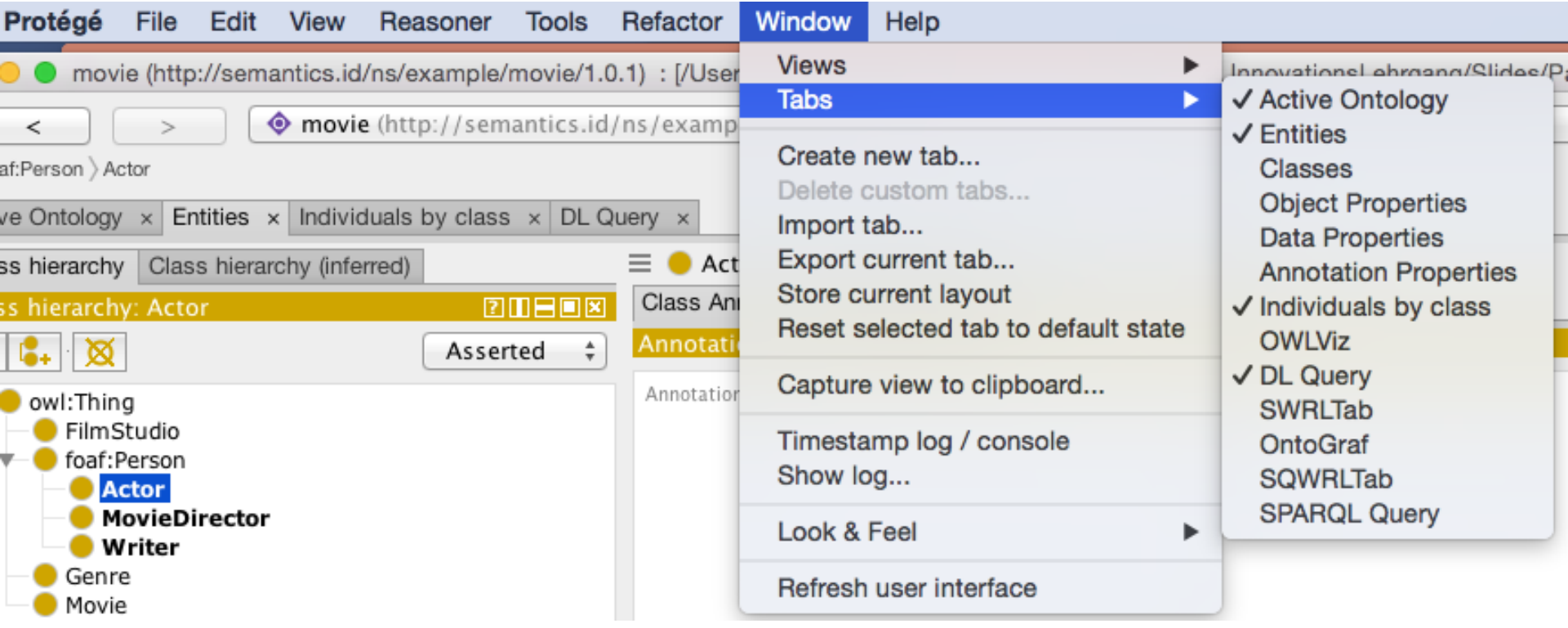
1. Classes
2. Datatype Properties
3. Object Properties
4. Instances



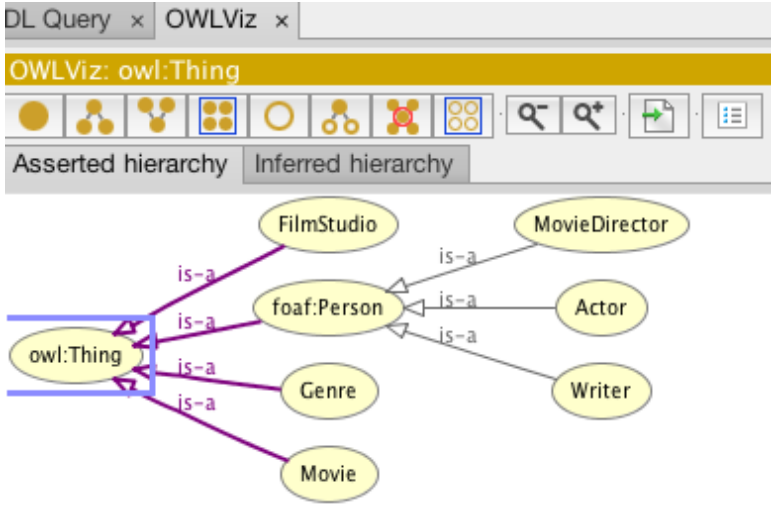
Protégé Editor: Customize your views

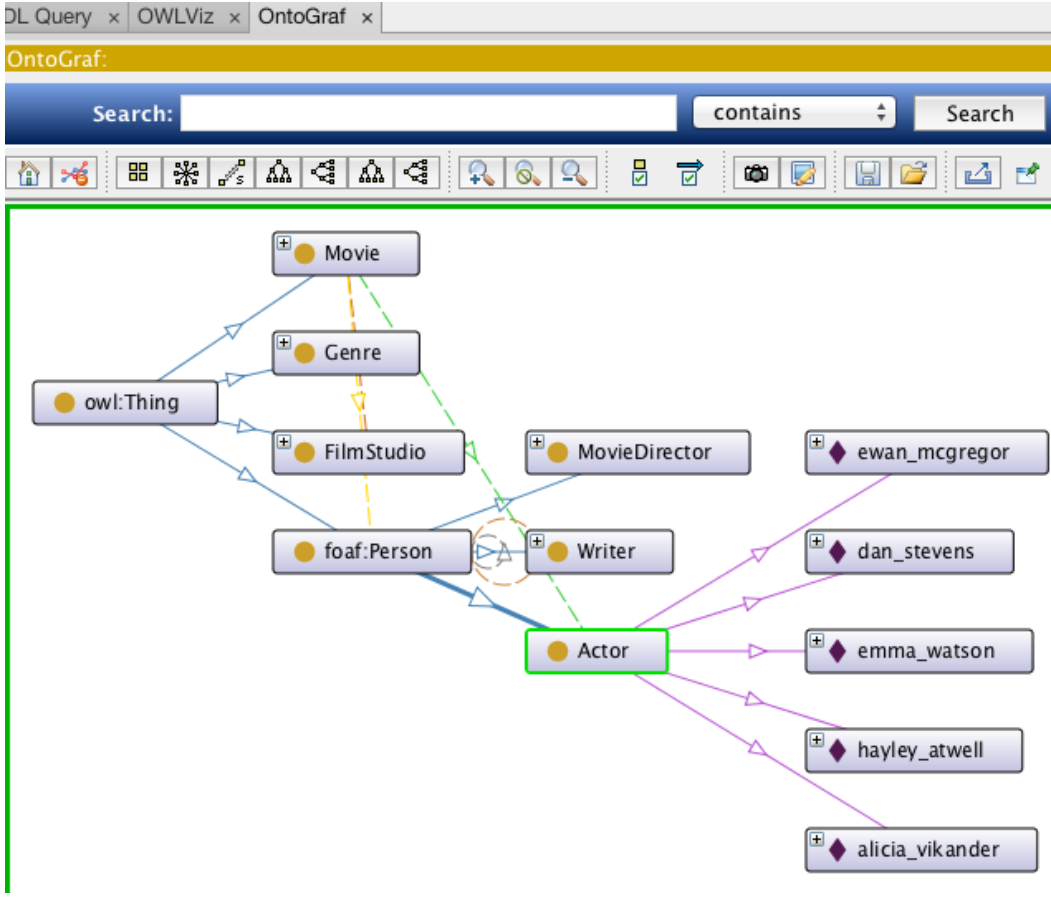


Explore ontology with OWLViz and OntoGraf

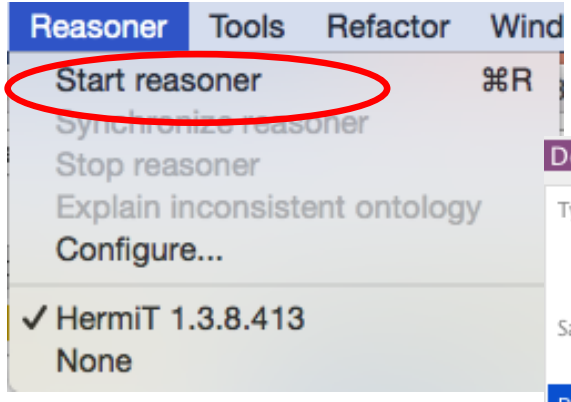


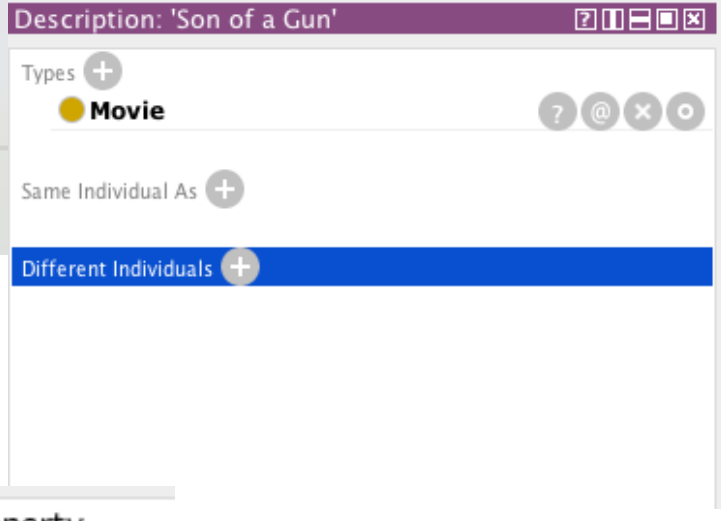
Explore ontology with OWLViz and OntoGraf

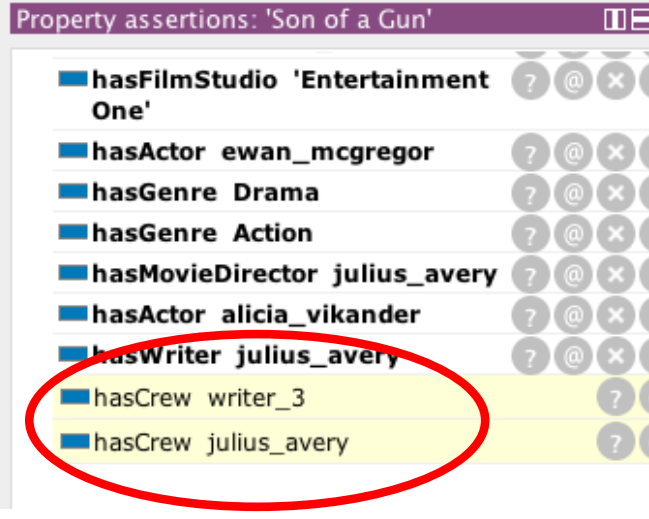




Run the HermiT Reasoner

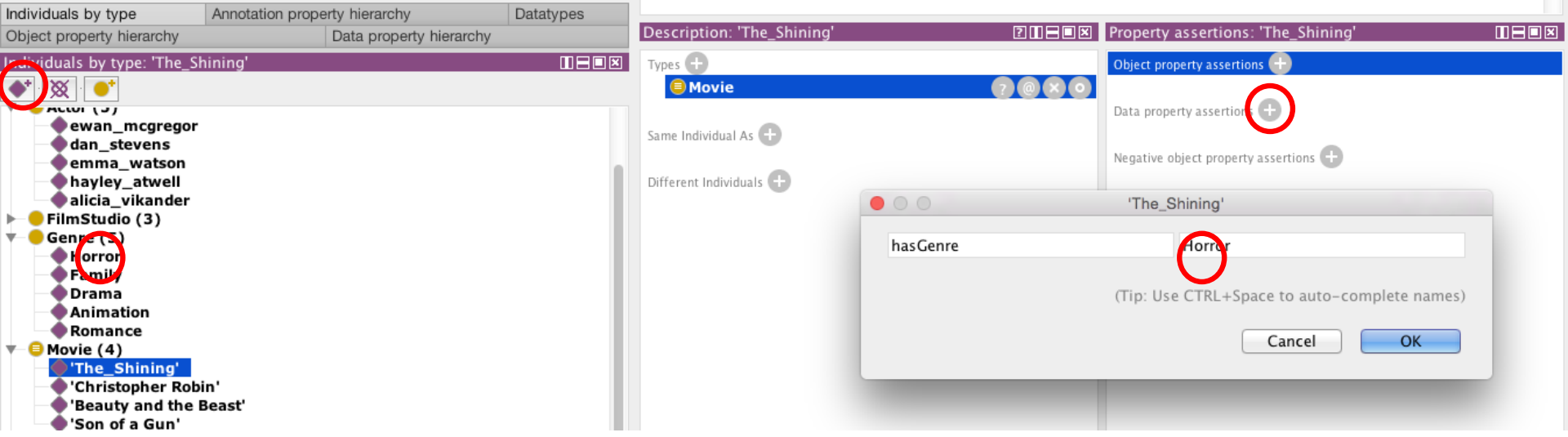




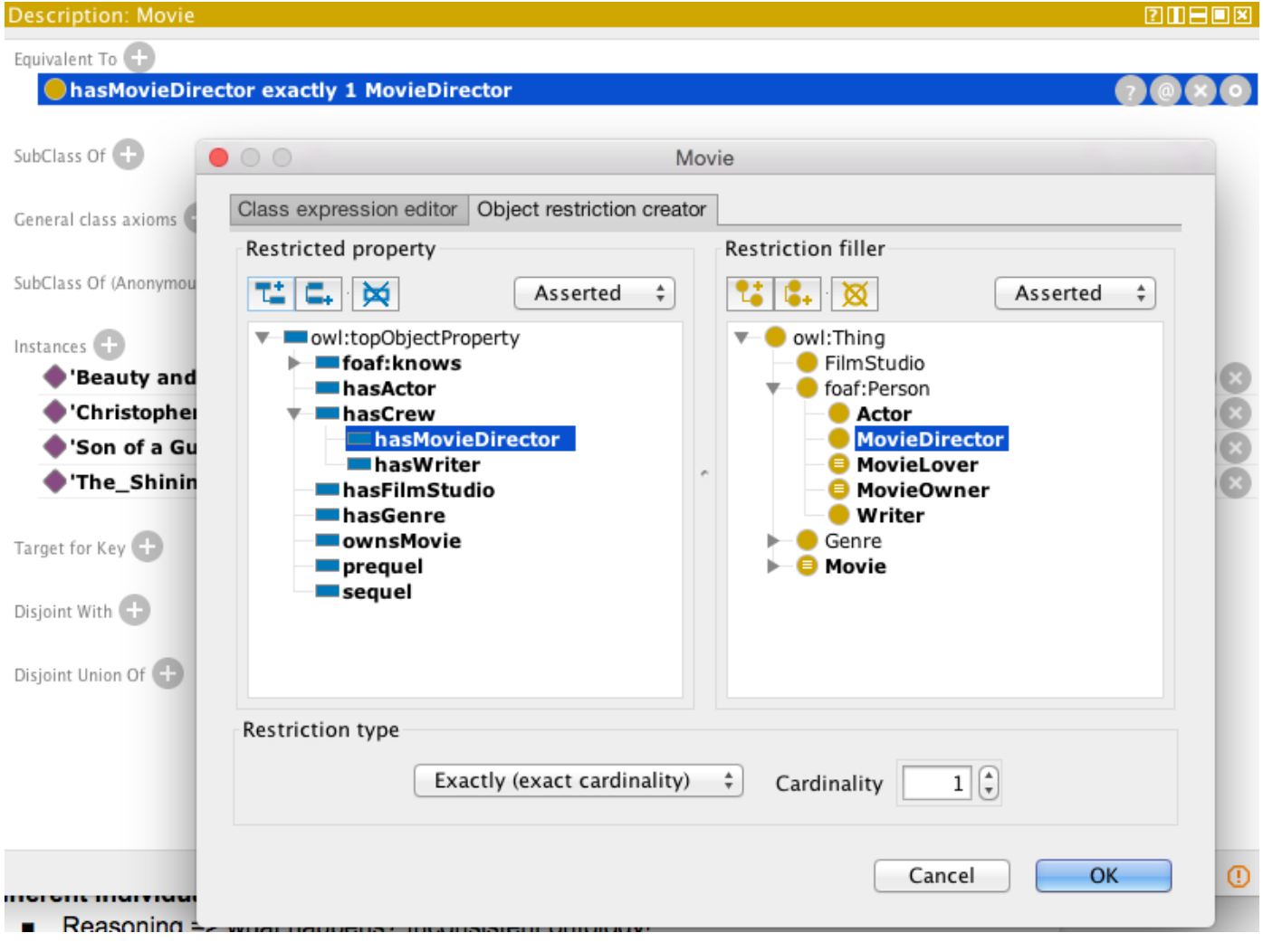


Explanation

Add one movie instance



Cardinality constraint



movie can have exactly one director

TU WIEN Informatics

Ontology Editors in Practice

Emanuel Sallinger
Katja Hose

TU WIEN Informatics

188.399 Introduction to Semantic Systems

(2.0 VU / 3.0 ECTS)

2023W

Katja Hose
Emanuel Sallinger

iss@dbai.tuwien.ac.at

Learning Goals

- RDF and Linked Data
- RDFS
- Ontology Representation and OWL

The Semantic Web

How the journey began…

- Expressing meaning
- Knowledge representation
- Ontologies
- Agents
- Evolution of knowledge

The Semantic Web

A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities

by TIM BERNERS-LEE, JAMES HENDLER and ORA LASSILA

The entertainment system was belting out the Beatles' "We Can Work It Out" when the phone rang. When Pete answered, his phone turned the sound down by sending a message to all the other *local* devices that had a *volume control*. His sister, Lucy, was on the line from the doctor's office: "Mom needs to see a specialist and then has to have a series of physical therapy sessions. Biweekly or something. I'm going to have my agent set up the appointments." Pete immediately agreed to share the chauffeuring. At the doctor's office, Lucy instructed her Semantic Web agent through her handheld Web browser. The agent promptly retrieved information about Mom's *prescribed treatment* from the doctor's agent, looked up several lists of *providers*, and checked for the ones *in-plan* for Mom's insurance within a *20-mile radius* of her *home* and with a *rating* of *excellent* or *very good* on trusted rating services. It then began trying to find a match between available *appointment times* (supplied by the agents of individual providers through their Web sites) and Pete's and Lucy's busy schedules. (The emphasized keywords indicate terms whose semantics, or meaning, were defined for the agent through the Semantic Web.)

In a few minutes the agent presented them with a plan. Pete didn't like it—University Hospital was all the way across town from Mom's place, and he'd be driving back in the middle of rush hour. He set his own agent to redo the search with stricter preferences about *location* and *time*. Lucy's agent, having *complete trust* in Pete's agent in the context of the present task, automatically assisted by supplying access certificates and shortcuts to the data it had already sorted through.

Killer app: Agents

- Retrieve information about prescribed treatment/therapy
- Look for providers/specialists with good ratings in proximity
- Schedule times according to multiple busy schedules
- Share information with other agents and “negotiate”

Scientific American, 2001
https://sit.instructure.com/courses/22632/files/2992212?module_item_id=462878

“A really important thing about data is: the more things you have to connect together, the more powerful it is.”

Sir Tim Berners-Lee on Linked Data at TED 2009

Tim Berners-Lee was honored with the Turing Award for his work inventing the World Wide Web, the first web browser, and "the fundamental protocols and algorithms [that allowed] the web to scale."

Photo: Henry Thomas

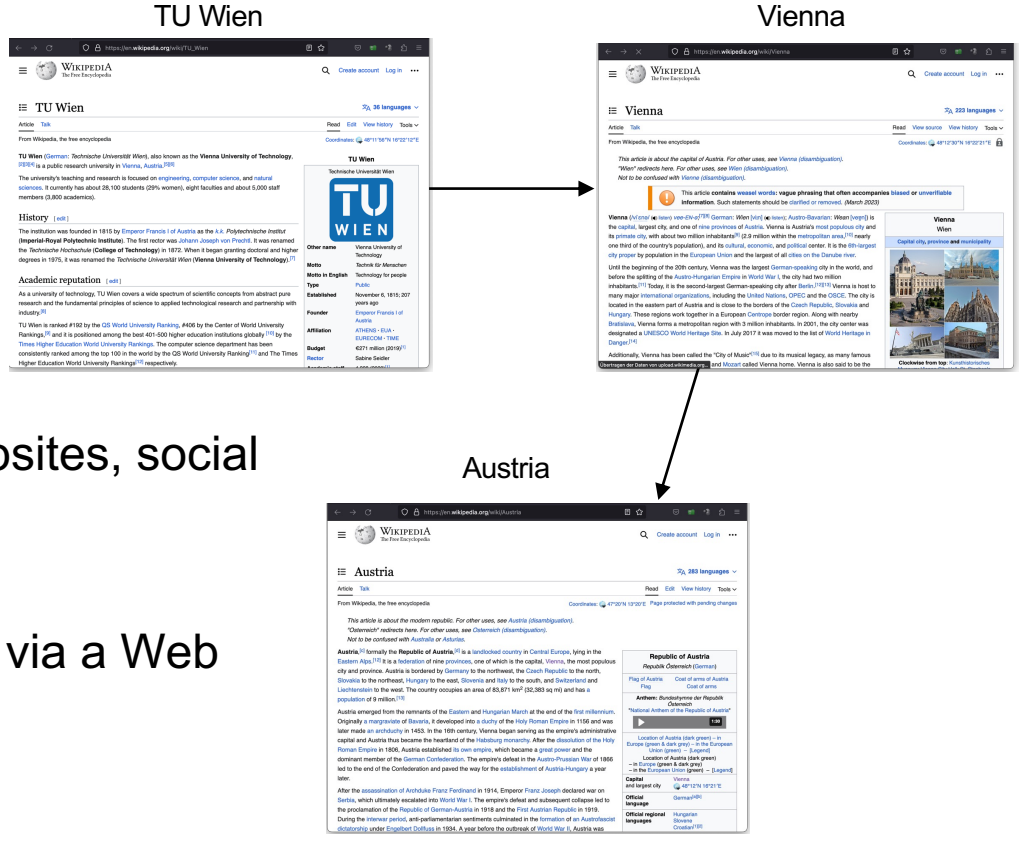
Tim Berners-Lee wins $1 million Turing Award

CSAIL researcher honored for inventing the web and developing the protocols that spurred its global use.

Sir Tim Berners-Lee on Linked Data at TED 2009

Linked Data and the Semantic Web

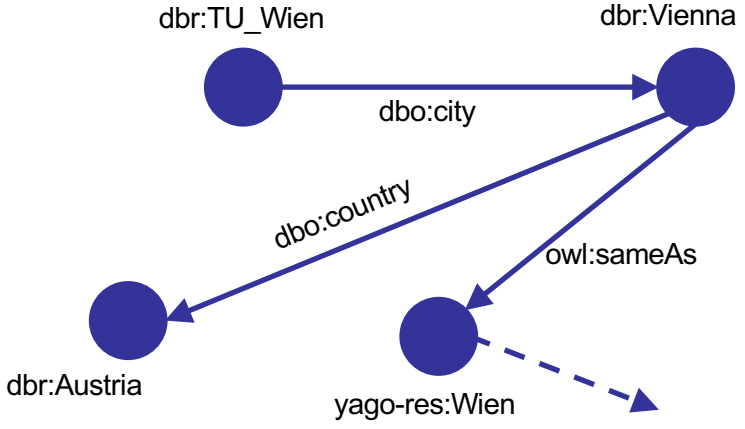
The Web of Documents



- Links between documents (websites, social media, etc.)
- Human-readable knowledge
- Normally accessed by humans via a Web browser
- Links are not qualified

Linked Data and the Semantic Web

The Web of Data



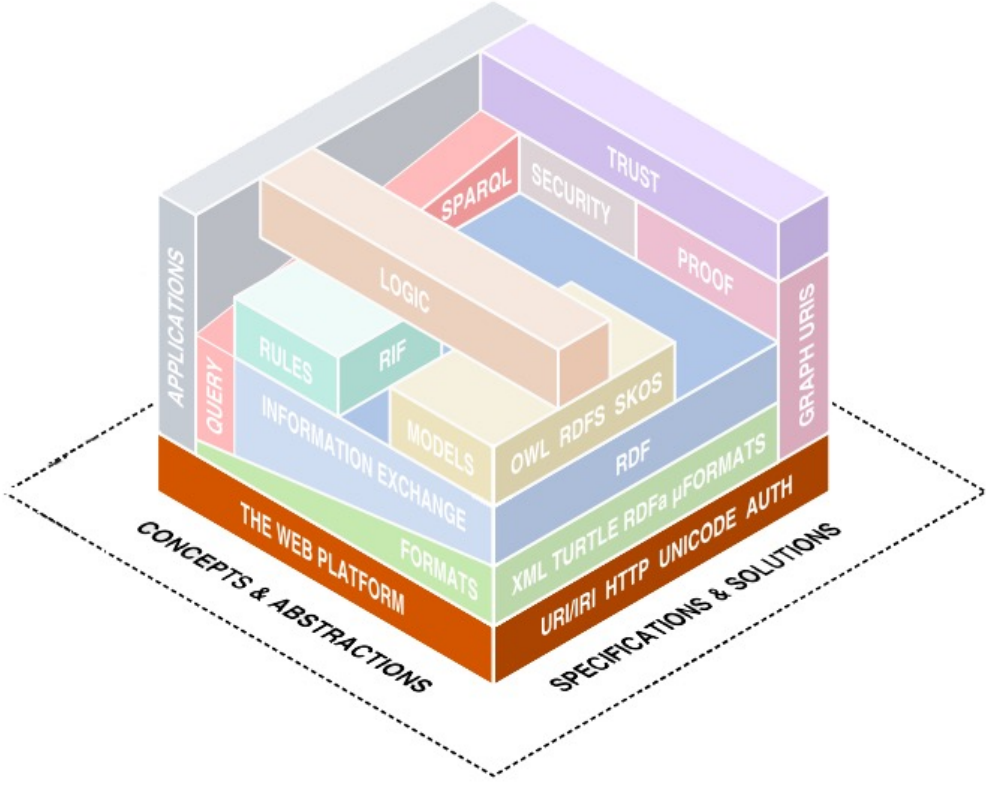
Linked Data

- Links between things/entities
- Machine-readable knowledge
- Normally accessed by machines
- Links are qualified
- A decentralized database

Web Technology Foundations

Web Technology Foundations

The Semantic Web is based on the Web



http://bnode.org/blog/2009/07/08/the-semantic-web-not-a-piece-of-cake

Web Technology Basics

TCP/IP (Transfer Control Protocol/Internet Protocol):
protocols for data transfer via a network

HTTP (Hypertext Transfer Protocol):

- application protocol to transfer information (not just) on the WWW
- request/response protocol between client and server
- content negotiation (header lists acceptable media types etc.)

HTML (Hypertext Markup Language):

- language for the creation of web pages
- describes how it should be displayed by a web browser

XML (Extensible Markup Language):

- general purpose markup language for the creation of special markup languages
- used to facilitate data sharing among different systems

URIs: Names on the Web

Basic principles

- "Everything has an identifier"
- "One name for one resource"

Uniform (different types of identifiers constructed according to a uniform schema)

Resource (whatever may be identified via URI)
(e.g., web pages, books, locations, persons, relations among objects, abstract concepts etc.)

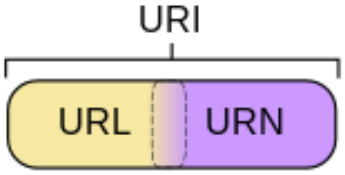
Identifier (to distinguish one resource from another)

Examples:

```
ldap://[2001:db8::7]/c=GB?objectClass?one
mailto:John.Doe@example.com
news:comp.infosystems.www.servers.unix
tel:+1-816-555-1212
telnet://192.0.2.16:80/
urn:oasis:names:specification:docbook:dtd:xml:4.1.2
http://dbpedia.org/resource/Vienna
```


https://tools.ietf.org/html/rfc3986

Names on the Web



URI: Uniform Resource can be

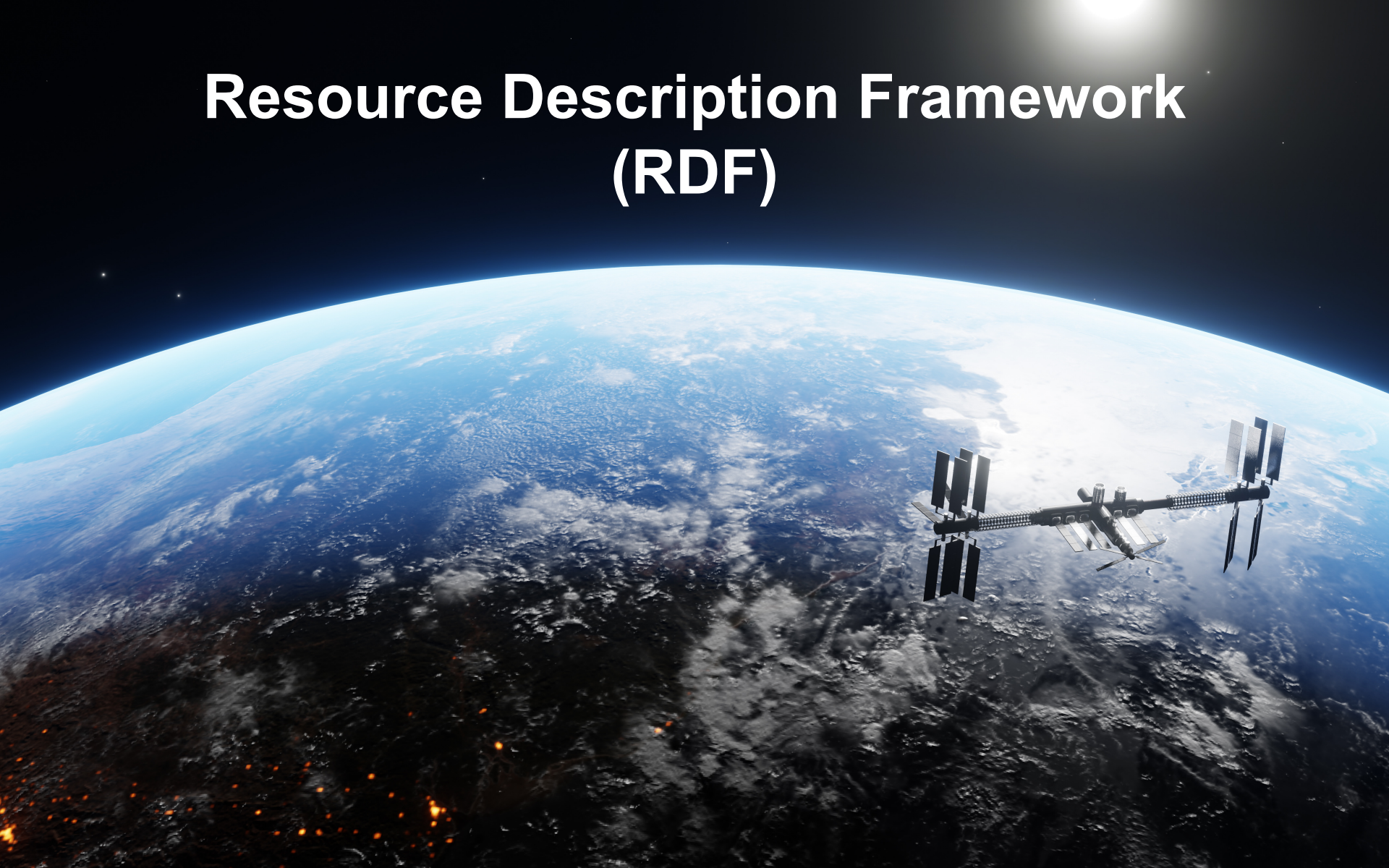
- **URN (RFC 2141):** Uniform Resource Name
  - persistent identifier within a particular namespace
  - Refers to a resource without implying location or how to access it
  - e.g., urn:isbn:0-486-27557-4 refers to a specific edition of Shakespeare's Romeo and Juliet
- **URL (RFC 1738):**
  - Uniform Resource Locator
  - specifies (primary) access mechanism
  - may change during the lifetime of a resource
- **both URN+URL**

IRI (Internationalized Resource Identifier, RFC 3987) are URIs

- Character set of IRIs: Unicode (ISO 10646)
- Character set of URIs: US-ASCII

In the remainder of this presentation occurrences of the term URI can be replaced with IRI.

Resource Description Framework (RDF)



Standard information exchange is key

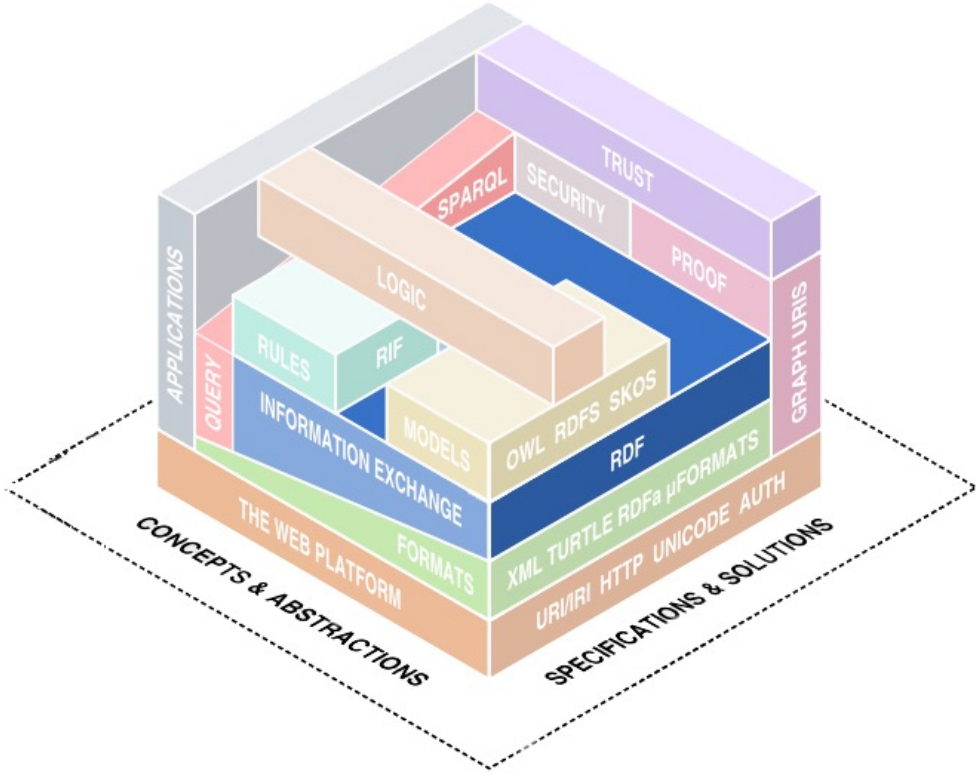


Image: http://bnode.org/blog/2009/07/08/the-semantic-web-not-a-piece-of-cake

RDF

- Resource Description Framework (RDF, W3C standard)
- Basic concepts
  - A resource can basically be anything
    - e.g. persons, places, Web documents, abstract concepts
  - Descriptions of resources
    - Attributes
    - Relationships
  - The framework contains
    - A data model, and
    - Languages and syntaxes

W3C RDF Working Group. 2014. Resource description framework. http://www.w3.org/RDF/

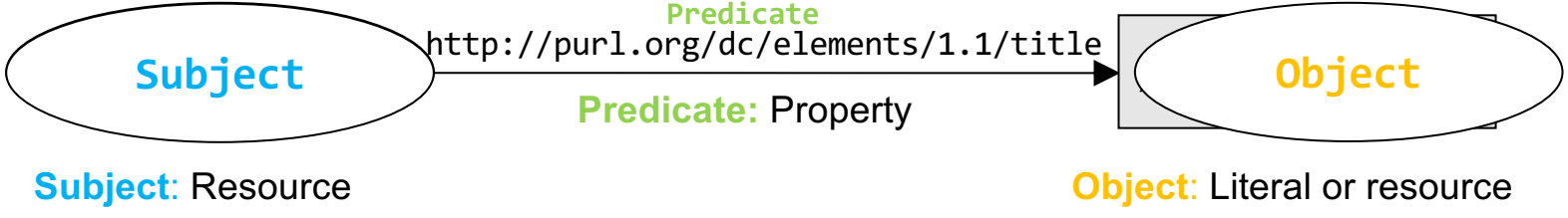
RDF

- Developed in the late 1990s
- Final W3C recommendation in 2004 (RDF 1.1)
- Graph-based model
- Formal semantics → machine-readable
- Knowledge expressed as a list of statements
- All statements follow the same simple schema (**RDF Triple**)
- Abstract model with various syntax notations and serialization formats:
  - Graph notation
  - RDF/XML (historically first serialization format)
  - N3/Turtle
  - N-Triples
  - JSON-LD

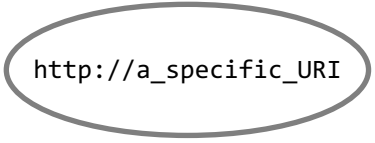
RDF Triples/RDF Statement

Basic RDF building block

- **Subject:** a resource, which may be identified with a URI
- **Predicate:** a URI-identified specification of the relationship between subject and object
- **Object:** a resource or literal to which the subject is related

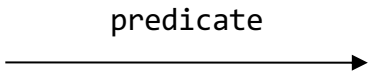


Graph Notation Elements



Resource

Has a unique URI

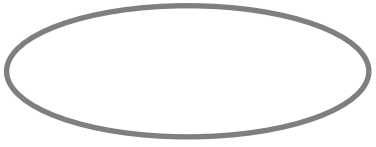


Predicate

Connects two resources

Literal

- Only text, no URI
- Plain (lexical value and optional language tag) or typed



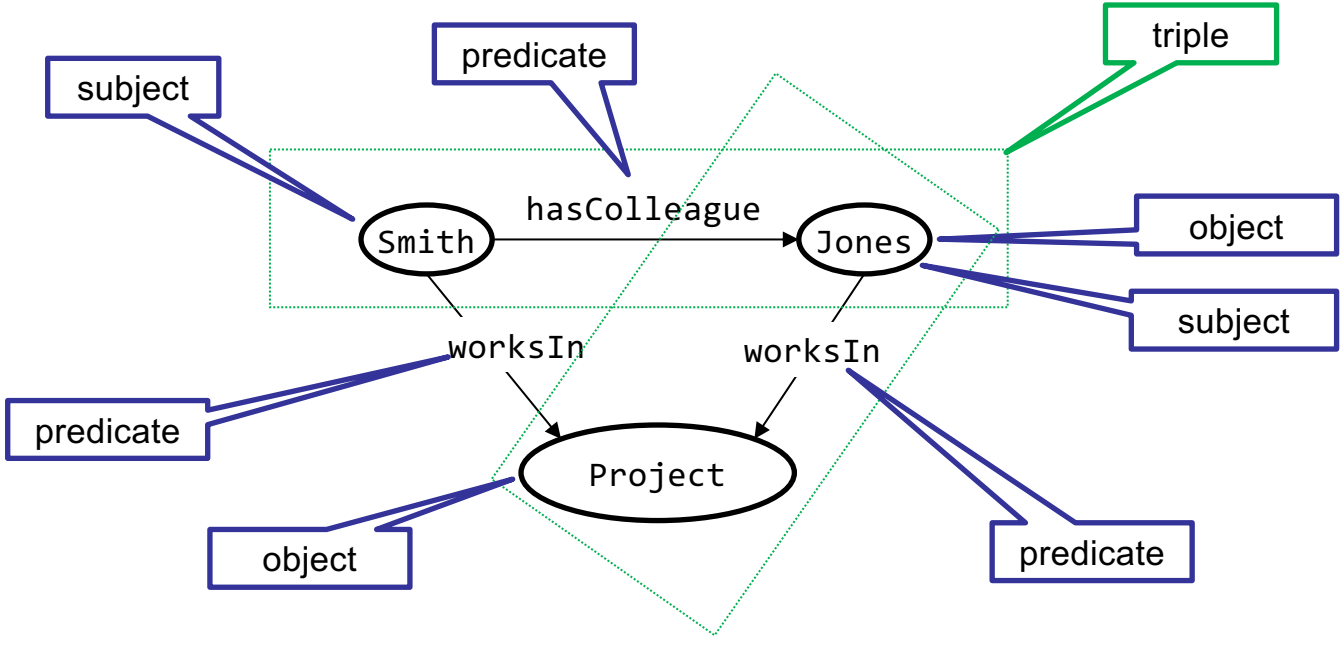
Blank Node

- No URI
- For unnamed objects

RDF Graphs

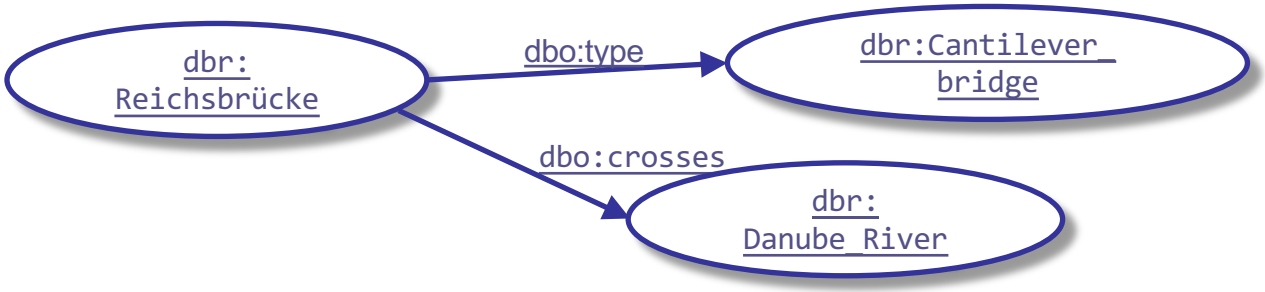
A set of RDF assertions in triples form a labeled directed graph with:

- **Resources:** the subjects and objects are nodes of the graph
- **Predicates:** each predicate becomes a label for an arc connecting the subject to the object.



! The object of one statement can be the subject of another.

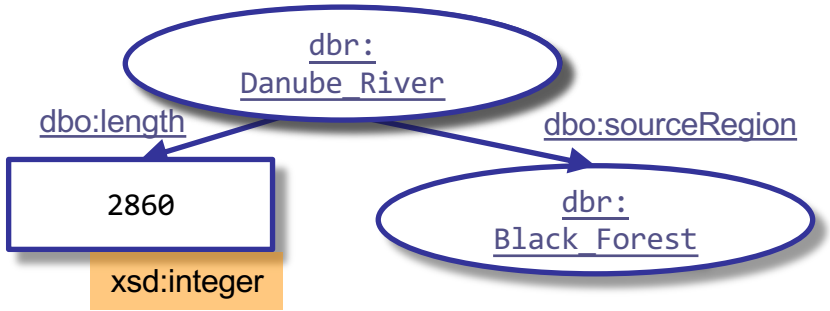
Prefix Notation



http://dbpedia.org/resource/Reichsbrücke

becomes

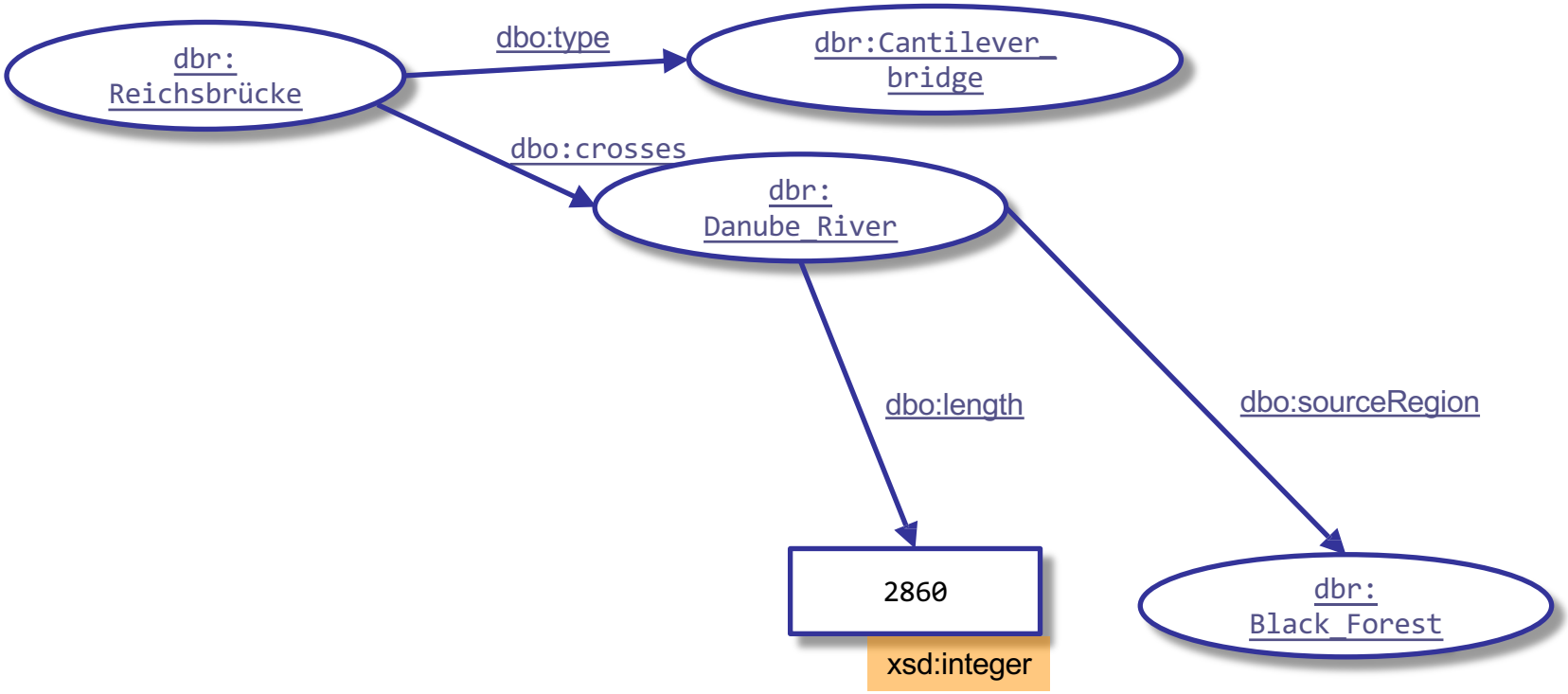
dbr:Reichsbrücke



dbr: http://dbpedia.org/resource/
dbo: http://dbpedia.org/ontology/

RDF DBpedia

Merging/Integrating Multiple Graphs



dbr: http://dbpedia.org/resource/
dbo: http://dbpedia.org/ontology/

RDF Literals: Plain Literals

Plain literals have a **lexical form** (their lexical value) and optionally a **language tag**.

```
“27” ,  “Hello world”@en
```

RDF Literals: Typed Literals

- Formed by pairing a string with a URIref that identifies a particular datatype
- Indicates what datatype should be used to interpret a given literal
- RDF has no built-in set of datatypes of its own (except `rdf:XMLLiteral`)
- RDF datatype concepts are based on a conceptual framework from **XML Schema datatypes** that defines
  - the value space,
  - the lexical space and
  - the lexical-to-value mapping for a datatype (see RDF specifications)

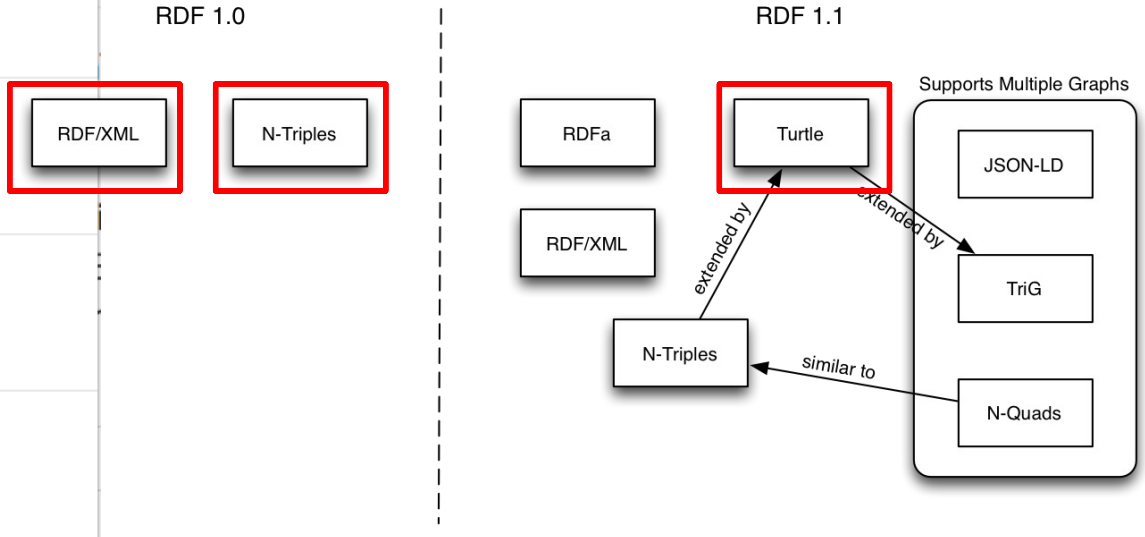
```
"27"^^http://www.w3.org/2001/XMLSchema#integer
```

XML Schema datatypes

	Datatype
Core types	xsd:string
	xsd:boolean
	xsd:decimal
	xsd:integer
IEEE floating-point numbers	xsd:double
	xsd:float
Time and date	xsd:date
	xsd:time
	xsd:dateTime
	xsd:dateTimeStamp
Recurring and partial dates	xsd:gYear
	xsd:gMonth
	xsd:gDay
	xsd:gYearMonth
	xsd:gMonthDay
	xsd:duration
	xsd:yearMonthDuration
	xsd:dayTimeDuration
Limited-range integer numbers	xsd:byte
	xsd:short
	xsd:int
	xsd:long
	xsd:unsignedByte
	xsd:unsignedShort
	xsd:unsignedInt
	xsd:unsignedLong
	xsd:positiveInteger
	xsd:nonNegativeInteger
	xsd:negativeInteger
	xsd:nonPositiveInteger
Encoded binary data	xsd:hexBinary
	xsd:base64Binary
Miscellaneous XSD types	xsd:anyURI
	xsd:language
	xsd:normalizedString
	xsd:token
	xsd:NMTOKEN
	xsd:Name
	xsd:NCName

https://www.w3.org/TR/2013/WD-rdf11-concepts-20130115/#xsd-datatypes

RDF Serialization Formats



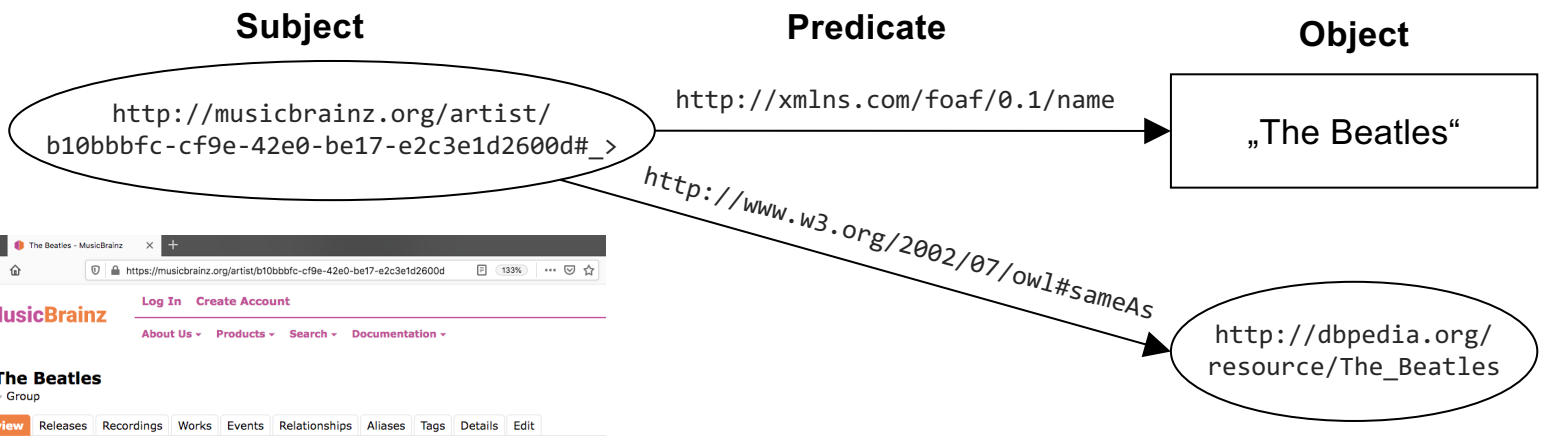
http://www.w3.org/TR/rdf11-new/

RDF Notations: RDF/XML

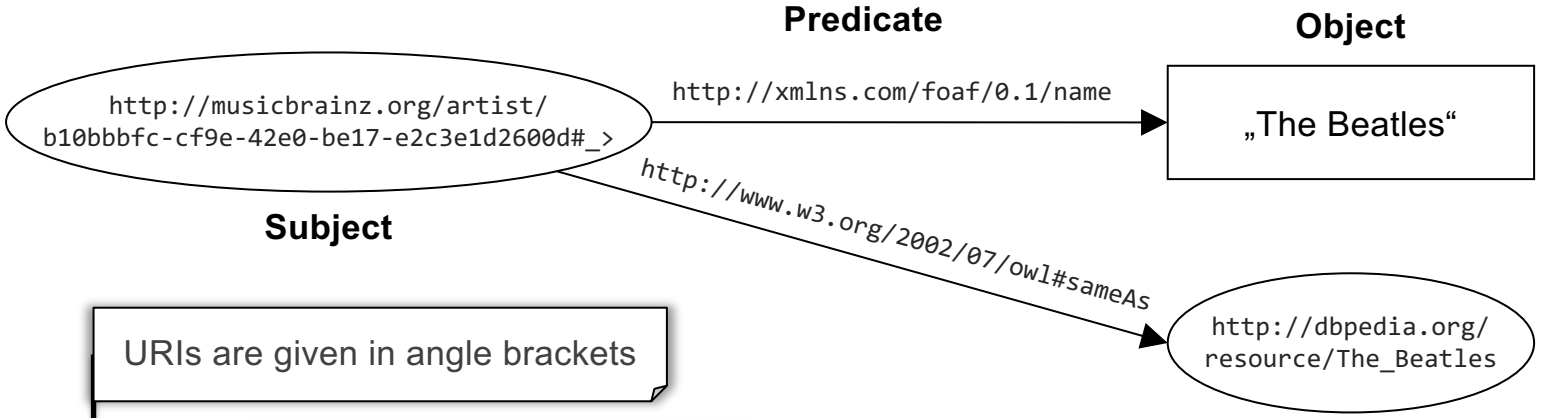
- Useful for inter-machine communication
- Every Description element describes a resource (referred to by an IRI)
- Every attribute or nested element inside a Description is a property of that Resource

```
<rdf:Description rdf:about="http://musicbrainz.org/artist/b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d#_">
  <foaf:name>The Beatles</foaf:name>
  <owl:sameAs rdf:resource="http://dbpedia.org/resource/The_Beatles" />
</rdf:Description>
```

RDF/XML



RDF Notations: N-Triples



Adapted from Euclid learning materials by Barry Norton

RDF Notations: N3

- Easy to read and write
- <subject> <predicate> <object>
- Every subject, predicate, object is identified with a URI
- Two shortcuts for several statements about the same subject
  - “;” introduce another predicate of the same subject
  - “,” introduce another object with the same predicate and subject

```
<#pat> <#child> <#al>, <#chaz>, <#mo>;
       <#age> 24;
       <#eyecolor>  “blue”.
```
N3

Interpretation:
Pat has the children al, chaz, mo;
Pat‘s age is 24;
Pat‘s eyecolor is blue

RDF Notations: Turtle

- "Terse RDF Triple Language"
- Subset of N3
- Many URIs share the same basis → use prefixes for namespace declarations

```
@prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#>.
@prefix owl:<http://www.w3.org/2002/07/owl#>.
@prefix mo:<http://purl.org/ontology/mo/>.
@prefix dbpedia:<http://dbpedia.org/resouce/>.
```

Turtle

Note: You can find commonly used prefixes here: http://prefix.cc

- Unique base:

```
@base <http://musicbrainz.org/>.
```

Turtle

RDF Notations: Turtle

Has a simple shorthand for class membership (**a**):

```
@base <http://musicbrainz.org/>.
@prefix mo:<http://purl.org/ontology/mo/>.
<artist/b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d#_>
                    a mo:MusicGroup.
```

Turtle

is equivalent to

```
<http://musicbrainz.org/artist/b10bbbfc-cf9e-42e0-be17-
e2c3e1d2600d#_>
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  <http://purl.org/ontology/mo/MusicGroup>.
```

N-Triples

RDF Notations: Turtle

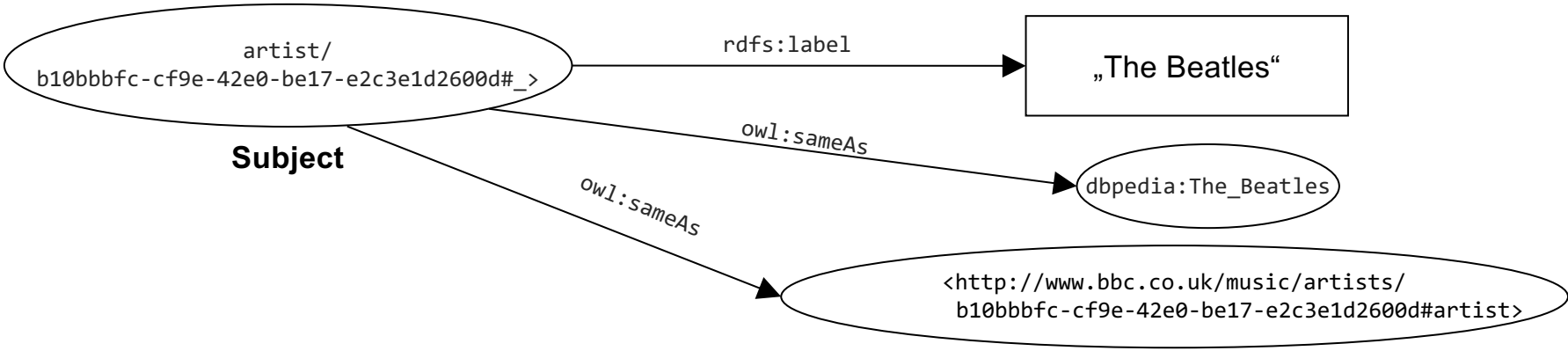
When multiple statements apply to same subject (and predicate) they can be abbreviated:

```
<artist/b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d#_>
        rdfs:label "The Beatles";
        owl:sameAs dbpedia:The_Beatles ,
                   <http://www.bbc.co.uk/music/artists/
                   b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d#artist> .
```

Turtle

Same subject

Same subject and predicate



RDF Notations: Turtle

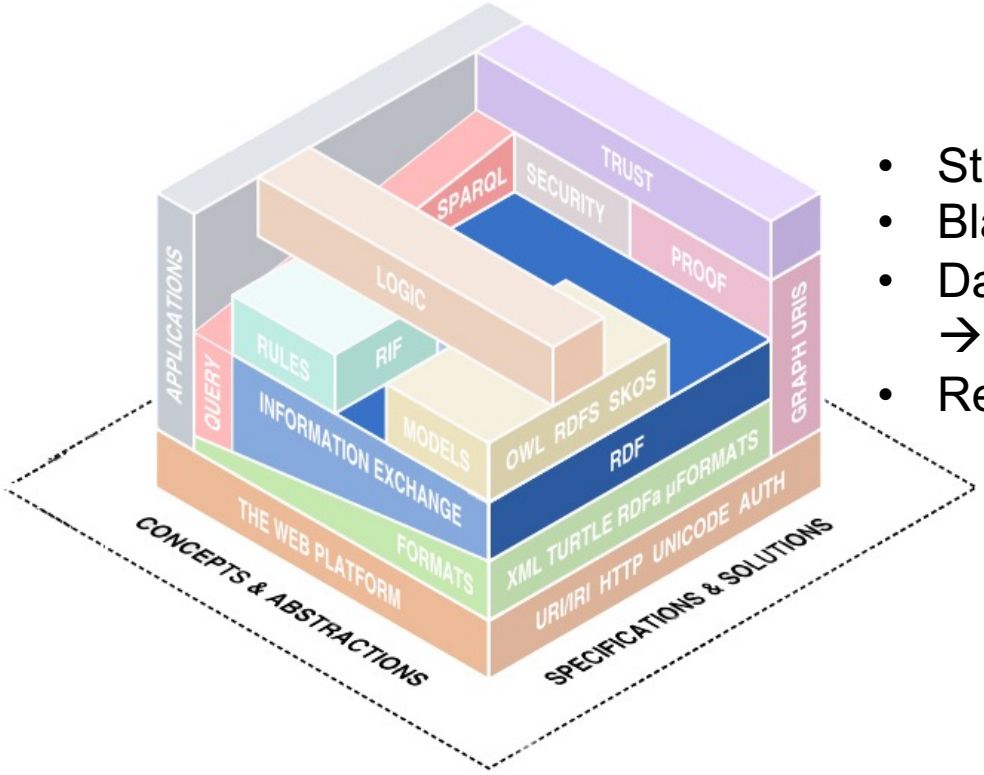
Turtle also provides a simple syntax for data types and language tags for literals:

```
<recording/5098d0a8-d3c3-424e-9367-1f2610724410#_>
        a mo:Signal;
        rdfs:label "All You Need Is Love" ;
        mo:duration "PT3M48S"^^xsd:duration .

dbpedia:The_Beatles dbpedia-owl:abstract
        "The Beatles were an English rock band formed (…) "@en,
        "The Beatles waren eine britische Rockband in den(…) "@de .
```

Turtle

RDF – Advanced Notions

Standard information exchange is key



- Structured values
- Blank nodes
- Data structures (container, collection)
  → see appendix
- Reification

Image: http://bnode.org/blog/2009/07/08/the-semantic-web-not-a-piece-of-cake

Structured Values in RDF

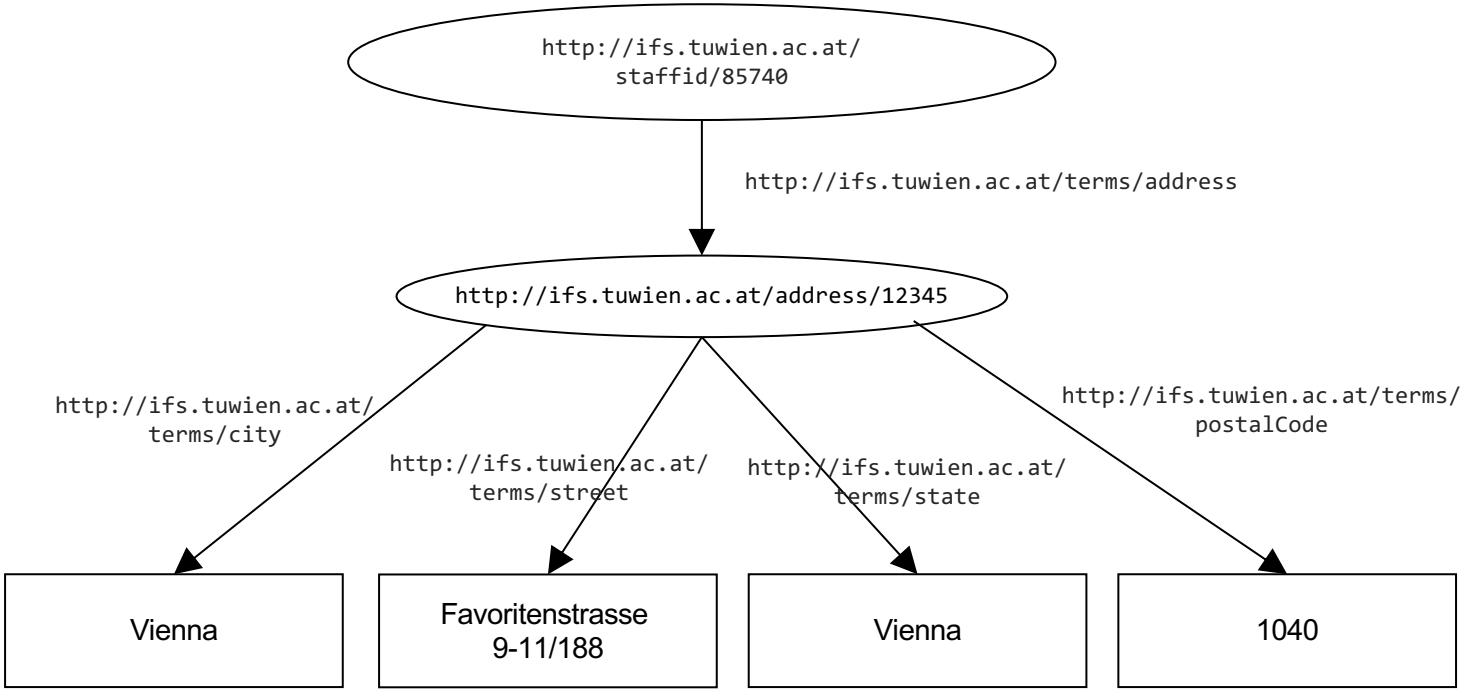
Consider the triple:

```
exstaff:85740
        exterms:address
        "Favoritenstrasse 9-11/188, 1040 Vienna" .
```

How do we represent address as a **structure** consisting of separate *street, city, state, and postal code* values? (N-ary relation)

→ Consider **the "aggregated thing" to be described** (like an address) **as a resource**, and then make statements about that new resource

Structured Values in RDF



Adapted from Manolis Koubarakis: "Knowledge Technologies"
http://cgi.di.uoa.gr/~artint2/STORAGE/lectures/2013.../introduction-to-rdf.ppt

Structured Values in RDF

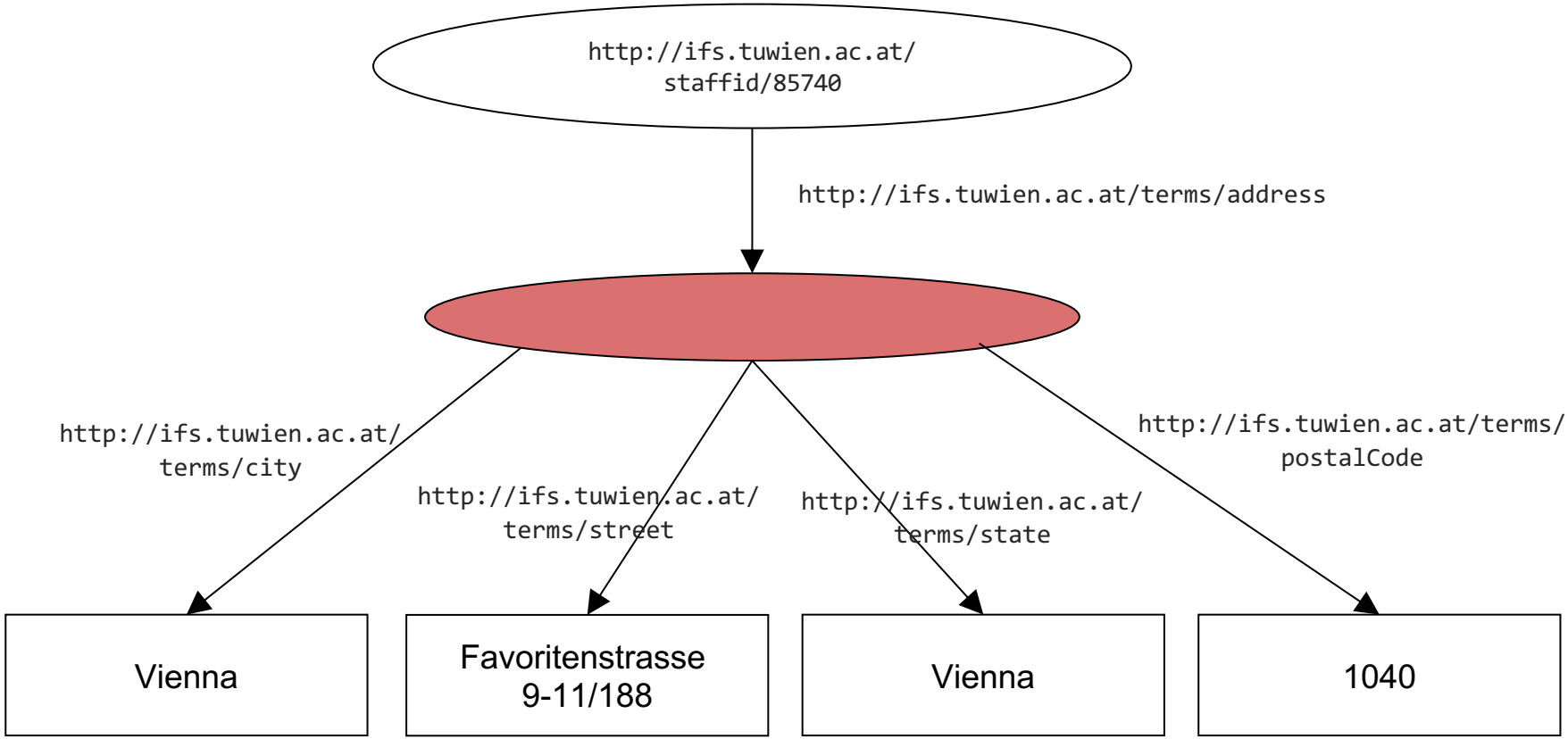
Or in triples notation:

```
exstaff:85740 exterms:address exaddressid:85740 .
exaddressid:85740 exterms:street "Favoritenstrasse 9-11/188" .
exaddressid:85740 exterms:city "Vienna" .
exaddressid:85740 exterms:state "Vienna" .
exaddressid:85740 exterms:postalCode "1040" .
```

```
exstaff:85740 exterms:address exaddressid:85740 .
exaddressid:85740 exterms:street "Favoritenstrasse 9-11/188" ;
                  exterms:city "Vienna" ;
                  exterms:state "Vienna" ;
                  exterms:postalCode "1040" .
```

Turtle

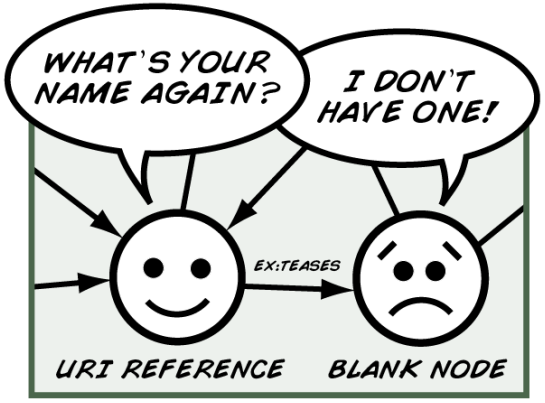
RDF Blank Nodes



Adapted from Manolis Koubarakis: "Knowledge Technologies"
http://cgi.di.uoa.gr/~artint2/STORAGE/lectures/2013.../introduction-to-rdf.ppt

RDF Blank Nodes

- Represent unnamed resources
- Used widely, with varying intentions:
  - Make statements about resources that do not have URIs (but that are described in terms of relationships with other resources that do)
  - group related information
  - represent n-ary relationships



http://milicicvuk.com/blog/?p=4

Issues:

- "Breaks" reasoning, because simple entailment in the presence of existentials is NP-complete [but: often not a big deal in practice*]
- Use in Linked Data discouraged [but: common in practice*]

* A. Hogan, M. Arenas, A. Mallea, and A. Polleres, "Everything you always wanted to know about blank nodes," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 27, pp. 42–69, 2014.

Blank Nodes Using Triples

```
exstaff:85740 exterms:address ? .
? exterms:street "1501 Grant Avenue" .
? exterms:city "Bedford" .
? exterms:state "Massachusetts" .
? exterms:postalCode "01730" .
```

Turtle

Is this a valid notation for a blank node?
→ Menti

Blank Node Identifiers

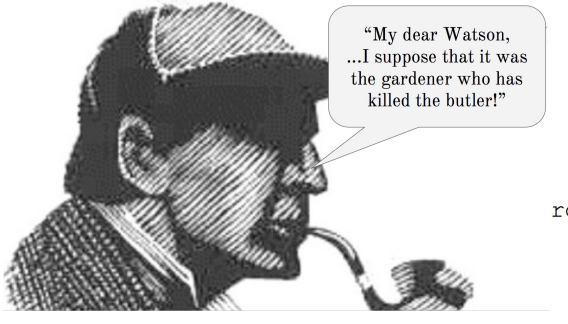
```
exstaff:85740 exterms:address _:johnaddress .
_:johnaddress exterms:street "1501 Grant Avenue" .
_:johnaddress exterms:city "Bedford" .
_:johnaddress exterms:state "Massachusetts" .
_:johnaddress exterms:postalCode "01730" .
```
Turtle

- Each distinct blank node must be given a different blank node identifier
- Blank node identifiers have significance only within a single graph
- Blank node identifiers may only appear as subjects or objects in triples (not predicates)

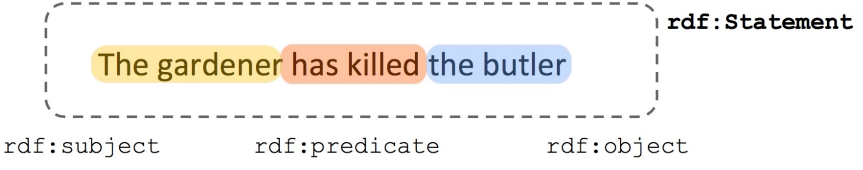
Reification

Used for making statements about statements to
- model data provenance
- formalize statements about reliability and trust
- define metadata about statements

Caveat:
- definition of infinite recursions and cycles

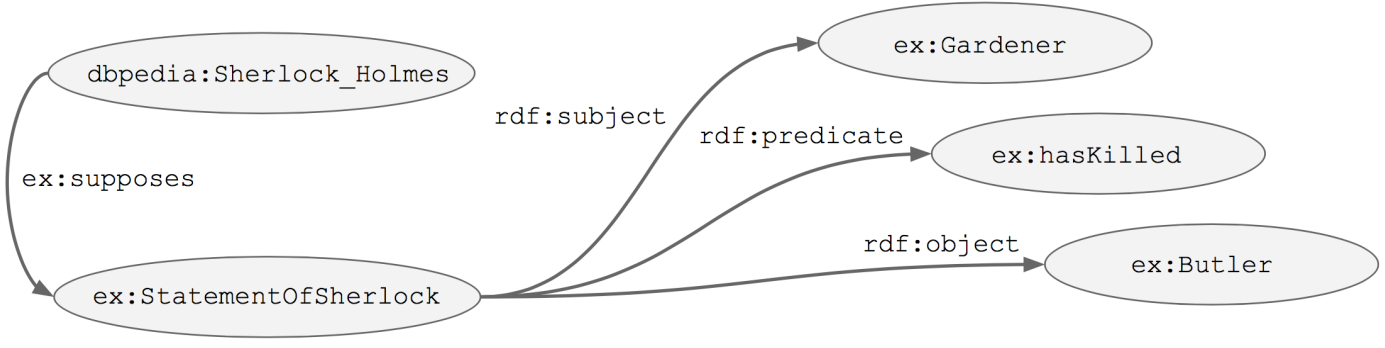


- `rdf:subject` - the described resource
- `rdf:predicate` - the original property
- `rdf:object` - the value of the property



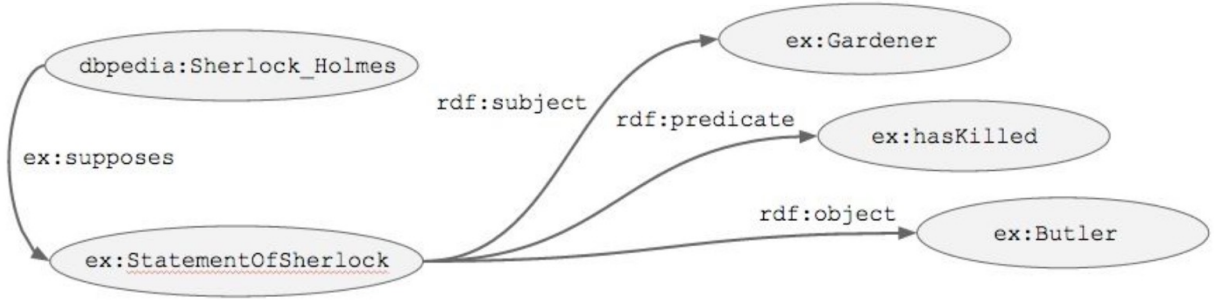
Reification Example

Sherlock Holmes supposes that the gardener has killed the butler




```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dbpedia <http://dbpedia.org/resource/> .
@prefix ex: <http://example.org/Crimestories#> .
```

Reification Example

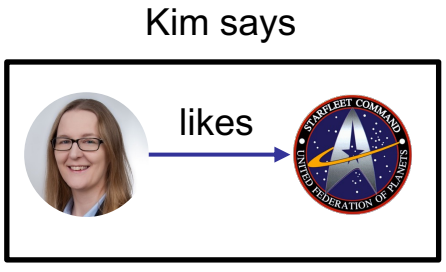



```
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex:      <http://example.org/Crimestories#> .

dbpedia:SherlockHolmes ex:supposes ex:StatementOfSherlock .
ex:StatementOfSherlock a rdf:Statement ;
    rdf:subject   ex:Gardener ;
    rdf:predicate ex:hasKilled ;
    rdf:object    ex:Butler .
```

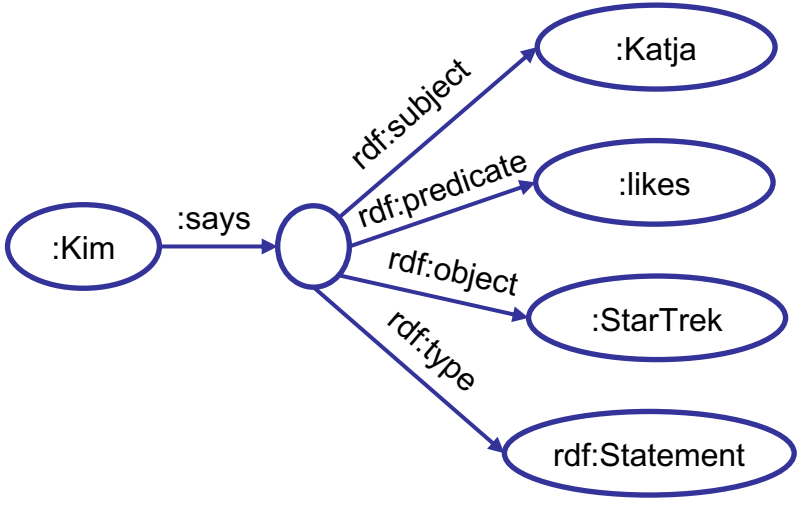
Turtle

Reification Alternatives

Standard reification



:Katja :likes :StarTrek



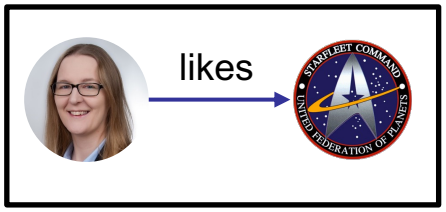
:Kim	:says	_:s1
_:s1	rdf:type	rdf:Statement
_:s1	rdf:subject	:Katja
_:s1	rdf:predicate	:likes
_:s1	rdf:object	:StarTrek

Pros: it works on any basic RDF/SPARQL engine
Cons: number of triples and query triple patterns is at least quadrupled

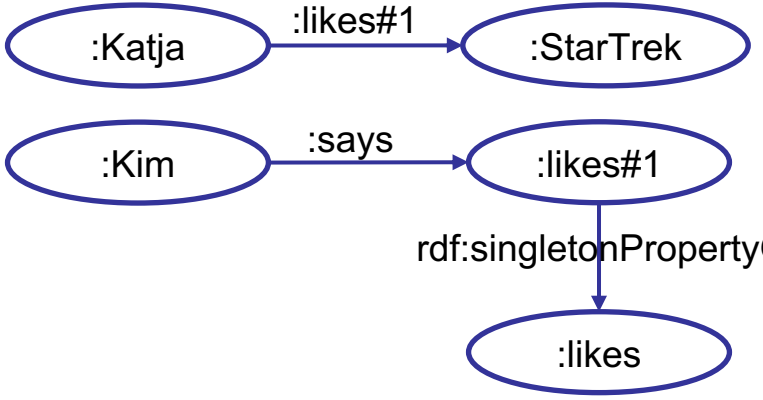
Reification Alternatives

Singleton property

Kim says



:Katja :likes :StarTrek



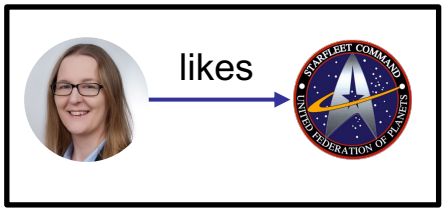
:Katja :likes#1 :StarTrek
:Kim :says :likes#1
:likes#1 rdf:singletonPropertyOf :likes

Pros: space efficient
Cons: it requires additional text indexes for predicates

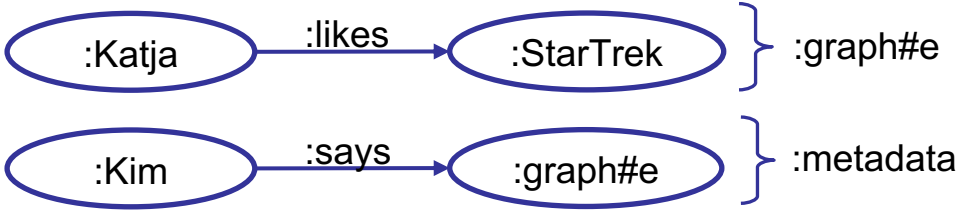
Reification Alternatives

Named graphs

Kim says



:Katja :likes :StarTrek

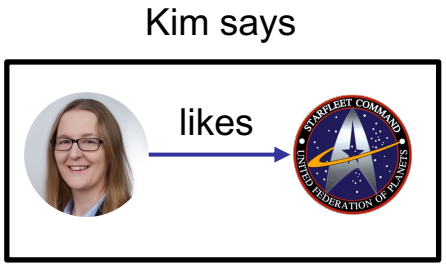


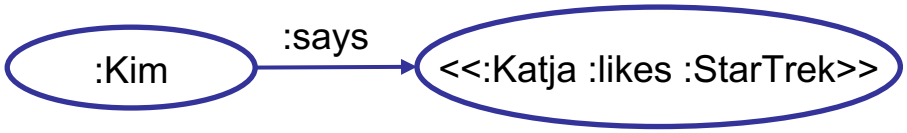
:Katja :likes :StarTrek :graph#e
:Kim :says :graph#e :metadata

Pros: relatively space efficient
Cons: it requires efficient graph indexing – not always supported

Reification Alternatives

RDF-star (RDF*)





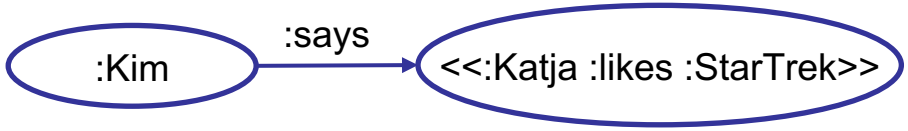
:Kim :says <<:Katja :likes :StarTrek>>

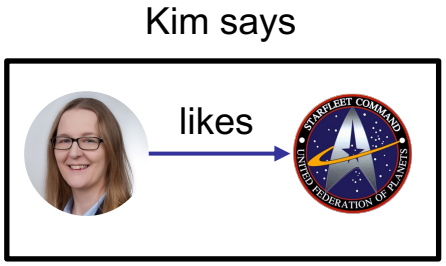
Pros: nested triples treated as first-class citizens
Cons: not supported by all systems

https://w3c.github.io/rdf-star

Reification Alternatives

RDF-star (RDF*)





:Kim :says <<:Katja :likes :StarTrek>>

W3C recommendation – not yet a standard

Currently not supported by all triple stores.
Supported (to a certain degree) by GraphDB, Stardog, Jena, Oxigraph

Details:
G. Abuoda, C. Aebeloe, D. Dell'Aglio, A. Keen, K. Hose:
StarBench: Benchmarking RDF-star Triplestores
In: QuWeDa icw. ISWC 2023

https://w3c.github.io/rdf-star

RDF Vocabulary

Classes

- **`rdf:XMLLiteral`** - the class of XML literal values
- **`rdf:Property`** - the class of properties
- **`rdf:Statement`** - the class of RDF statements
- **`rdf:Alt, rdf:Bag, rdf:Seq -`**
  containers of alternatives, unordered containers, and ordered containers*
- **`rdf:List`** - the class of RDF Lists
- **`rdf:nil`** - an instance of `rdf:List` representing the empty list

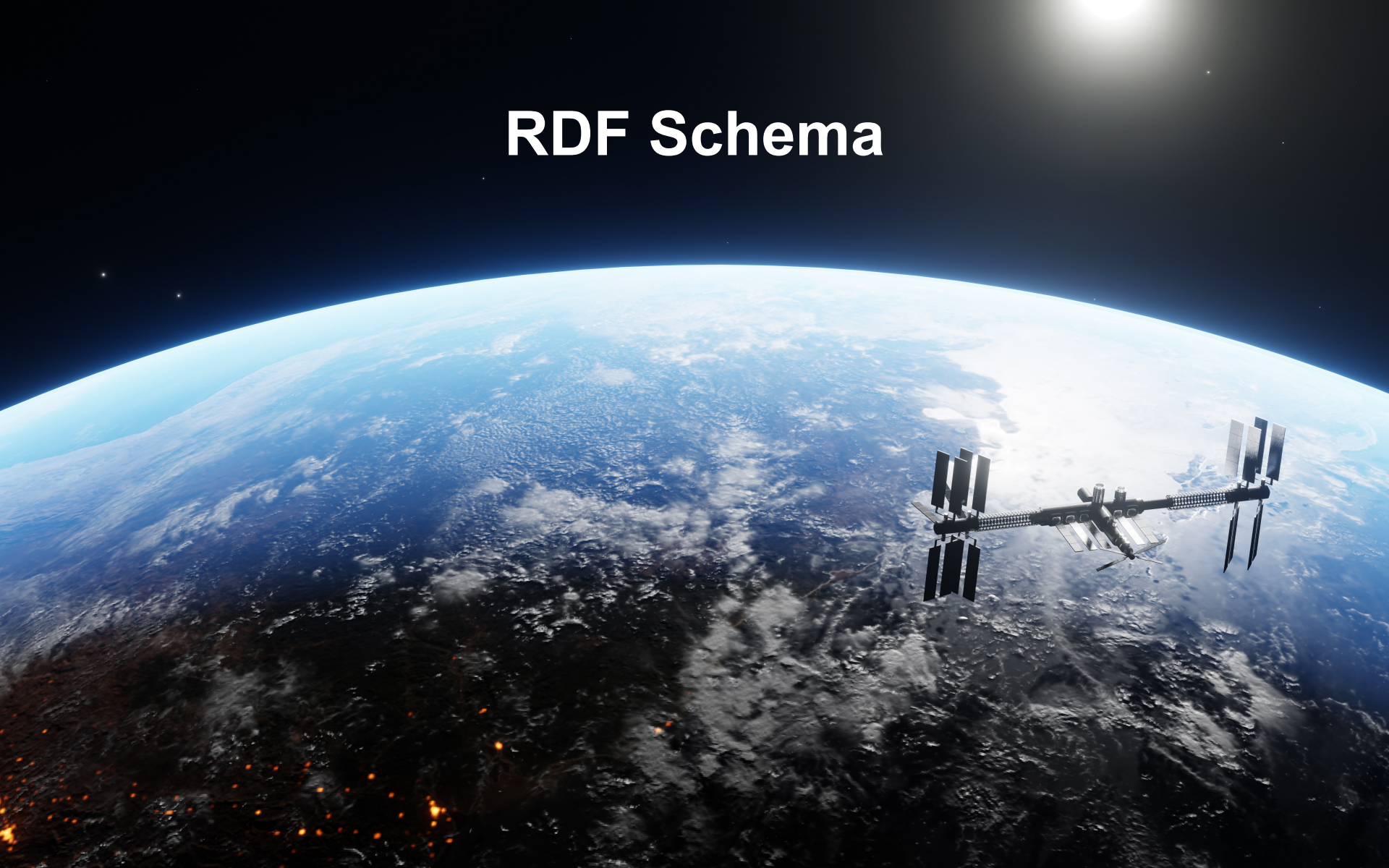
Properties

- **`rdf:type`** - an instance of `rdf:Property` used to state that a resource is an instance of a class
- **`rdf:first`** - the first item in the subject RDF list
- **`rdf:rest`** - the rest of the subject RDF list after rdf:first
- **`rdf:value`** - idiomatic property used for structured values
- **`rdf:subject`** - the subject of the subject RDF statement
- **`rdf:predicate`** - the predicate of the subject RDF statement
- **`rdf:object`** - the object of the subject RDF statement

* `rdfs:Container` is a super-class of the three

https://en.wikipedia.org/wiki/Resource_Description_Framework#RDF_vocabulary

RDF Schema



RDFS

Specifying conceptual models:
specifying **lightweight** ontologies with RDFS.

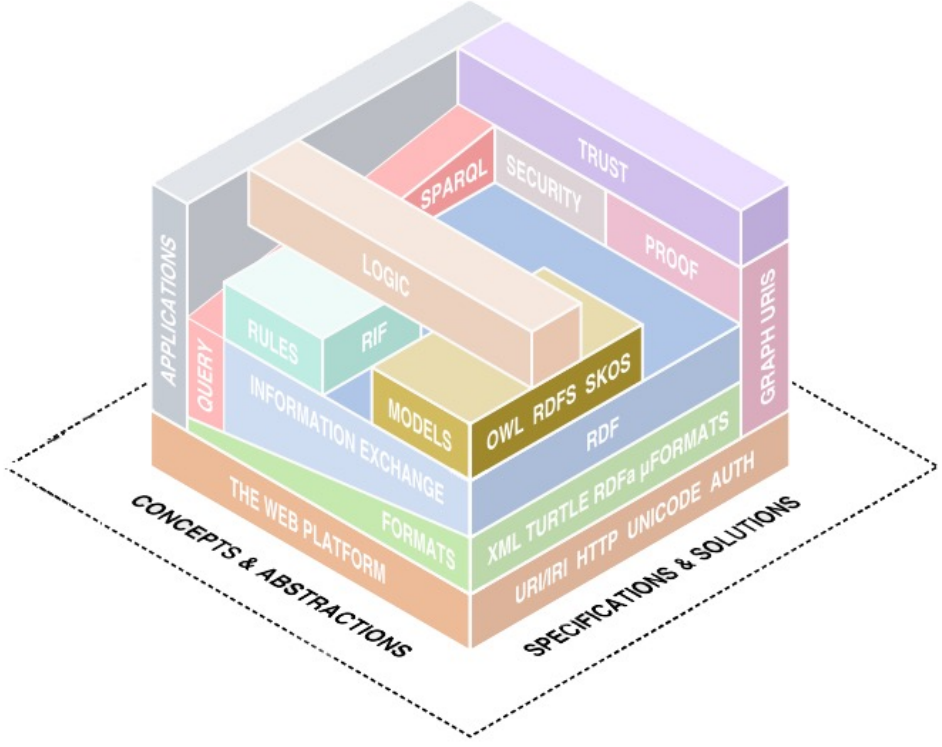


Image: http://bnode.org/blog/2009/07/08/the-semantic-web-not-a-piece-of-cake

Semantic Web Language Design Principles

- On the Web “Anyone can say Anything about Any topic” – **AAA Slogan**

- **Non-unique Naming Assumption**
  - The same entity could be known by more than one name
  - E.g., PersonA can be the same instance as PersonB
  - Disjointness needs to be specified explitly

- **Open World Assumption**
  - “Missing information is not evaluated as negative information!”
  - Assumes incomplete information by default, in this way, it allows for reasoning on incomplete models.
  - If fact f in a knowledge base is missing, not f cannot be inferred
  - E.g., likes(PersonA, DrinkB)
    - CWA: likes(PersonA, DrinkC) – No
    - OWA: likes(PersonA, DrinkC) – don’t know; PersonA may also like other drinks…

RDFS (RDF Schema)

- First version in April 1998, W3C recommendation in Feb. 2004
- Every RDF Schema document is an RDF document

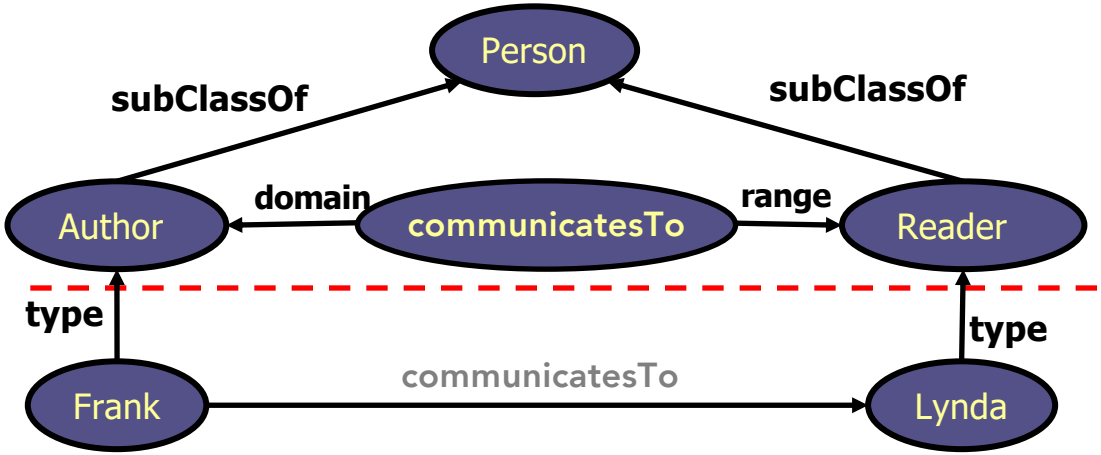
Purpose:

1. **Nominate**
   - the ‘types’ (i.e., classes) of things we might make assertions about, and
   - the properties we might apply, as predicates in these assertions, to capture their relationships

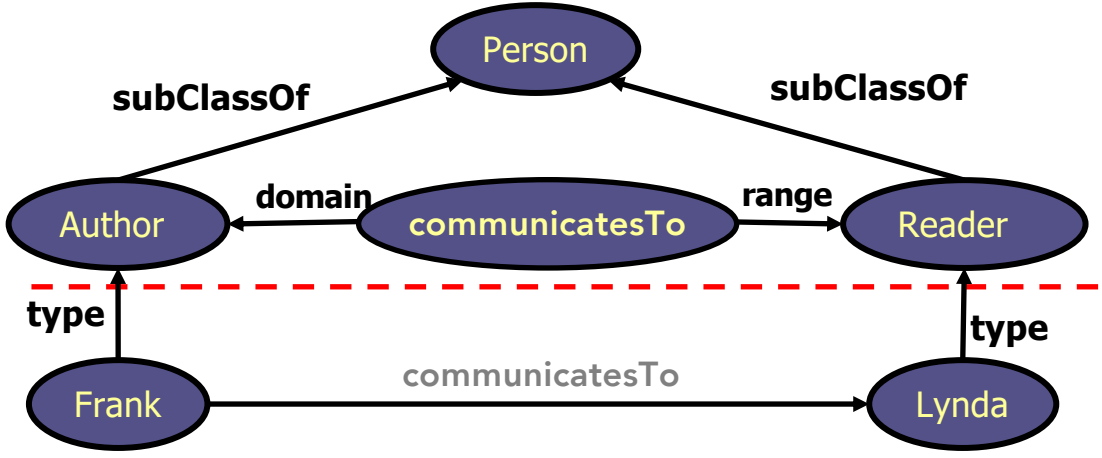
2. **Inference**
   Given a set of assertions, using these classes and properties, specify what should be inferred about assertions that are implicitly made

What does RDF Schema add?

- Defines the **vocabulary** for RDF
- Organizes this vocabulary in a **typed hierarchy**
  - class, subClassOf, type
  - property, subPropertyOf
  - domain, range



Inferences with RDF Schema




```
<Person,rdf:type,rdfs:Class>
<Reader,rdfs:subClassOf,Person>
<Author,rdfs:subClassOf,Person>
<communicatesTo,rdf:type,rdfs:Property>
<communicatesTo,rdfs:domain,Author>
<communicatesTo,rdfs:range,Reader>
<Frank,communicatesTo,Lynda>
```

```
<Frank,rdf:type,Author>
<Lynda,rdf:type,Reader>
```

Signatures: Domain and Range

rdfs:domain

- Restricts the scope of a property

```
mo:member rdfs:domain mo:MusicGroup .
```
Turtle

rdfs:range

- Used to restrict the type of values of a property

```
mo:member rdfs:range foaf:Agent .
```
Turtle

 Adapted from Euclid learning materials by Barry Norton

RDF and RDFS Vocabularies

RDF Classes:
- rdf:Property, rdf:Statement,
- rdf:XMLLiteral
- rdf:Seq, rdf:Bag, rdf:Alt,
- rdf:List

RDF Properties:
- rdf:type, rdf:subject,
- rdf:predicate, rdf:object,
- rdf:first, rdf:rest, rdf:_n
- rdf:value

RDF Resources:
- rdf:nil

RDFS Classes
- rdfs:Resource
- rdfs:Class
- rdfs:Literal
- rdfs:Datatype
- rdfs:Container
- rdfs:ContainerMembershipProperty

RDFS Properties
- rdfs:domain
- rdfs:range
- rdfs:subPropertyOf
- rdfs:subClassOf
- rdfs:member
- rdfs:seeAlso
- rdfs:isDefinedBy
- rdfs:comment
- rdfs:label

W3C http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/
http://www.w3.org/TR/rdf-schema/

Main RDFS Constructs

Classes:

- **`rdfs:Resource`** all resources within RDF are implicitly members of this class
- **`rdfs:Class`** declares a resource as a class (type or category) for other resources; instantiation of classes with `rdf:Type`
- **`rdfs:Literal`** literal values such as strings and integers (plain or typed)
- **`rdfs:Datatype`** class of datatypes (both an instance of and a subclass of `rdfs:Class`; each instance of `rdfs:Datatype` is a subclass of `rdfs:Literal`)

Properties:

- **`rdfs:subClassOf`** allows to declare hierarchies of classes; properties of super class are inherited
- **`rdfs:subPropertyOf`** all resources related by one property are also related by another
- **`rdfs:domain`** defines the scope of a property (see slide)
- **`rdfs:range`** defines the range of values of a property (see slide)

(no inference)

- **`rdfs:comment`** used to provide a human-readable description of a resource
- **`rdfs:seeAlso`** used to indicate a resource that might provide additional information
- **`rdfs:isDefinedBy`** indicates a resource defining the subject source

RDFS Limitations

- Allows basic inference
- Poor expressivity
- Does not allow complex expressions besides membership

Specifying conceptual models:
specifying **heavyweight** ontologies with OWL.

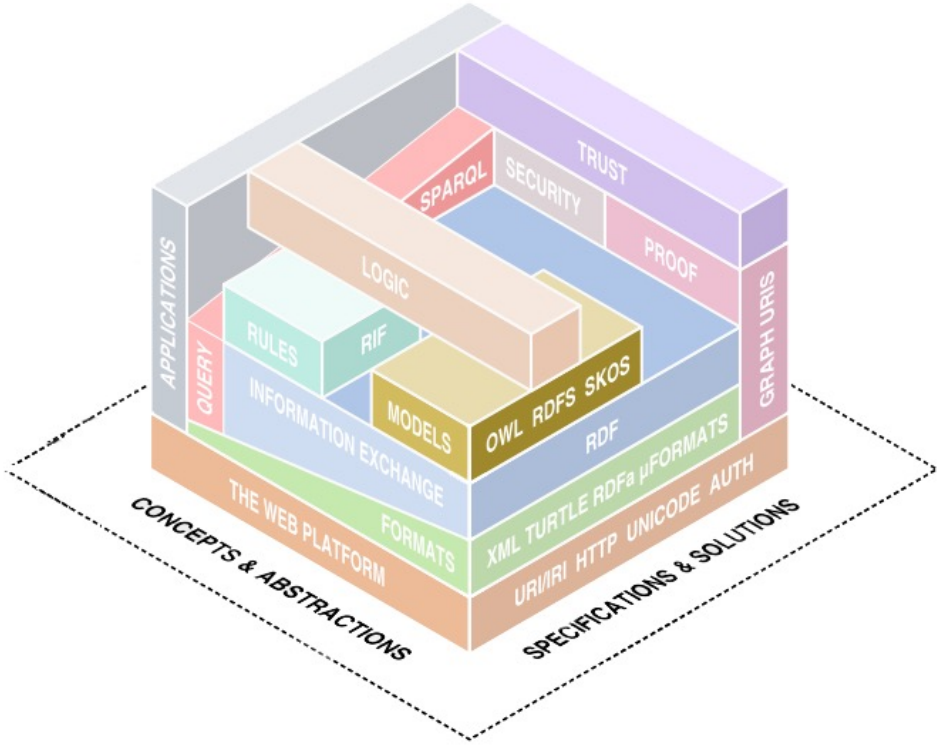


Image: http://bnode.org/blog/2009/07/08/the-semantic-web-not-a-piece-of-cake

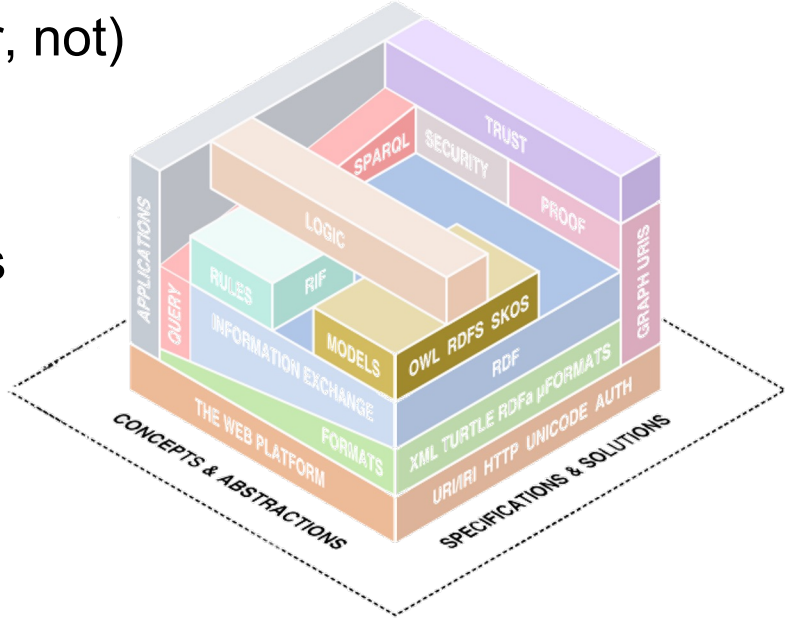
Web Ontology Language (OWL)

- W3C Recommendation for the representation of ontologies
- Fully-fledged knowledge representation language for the Web
- Provides a wider range of ontological constructs and avoids some of the potential confusion in RDFS
- Formal foundations: Description Logics (DL)
  - well-defined semantics
  - Well-understood properties (e.g., computational complexity)
  - allows to verify consistency of defined knowledge
  - inferencing allows to make implicit knowledge explicit
  - many tools available
- OWL ontologies can be exchanged as RDF documents

OWL: Extending RDFS

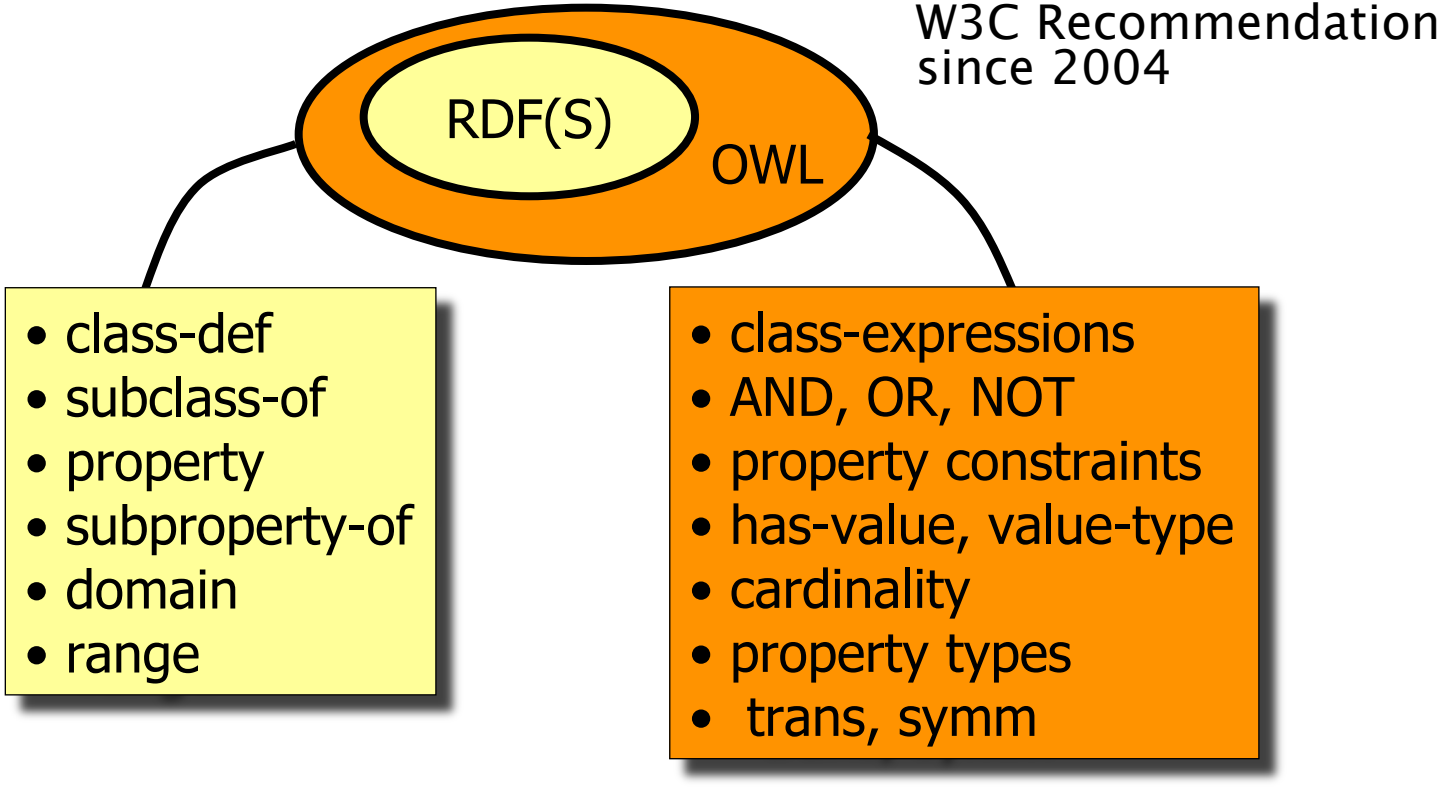
OWL extends RDF Schema to a fully-fledged knowledge representation language for the Web

- Logical expressions (and, or, not)
- (In)equality
- Local properties
- Required/optional properties
- Required values
- Enumerated classes
- Symmetry, inverse



http://teaching-wiki.sti2.at/uploads/a/a7/SW-07-OWL.pdf

OWL – Web Ontology Language



Classes

- Aka. “type”, “concept”, “category”
- Sets of individuals belong to a class
- Class membership is dependent on logical description, not name
- There exist two predefined classes
  - **`owl:Thing`** (class that contains all individuals) ⊤
  - **`owl:Nothing`** (empty class) ⊥

Definition of a class:

```
:Book a owl:Class .
```

Individuals

- Definition of individuals via class membership

```
:NineteenEightyfour a :Book .
```

```
Book(NineteenEightyfour)
```

- Individuals may be (and are likely to be) a member of multiple classes
- Individuals can also be defined without direct class membership as named entity

```
:FrankSmith a owl:NamedIndividual .
```

Properties

- Used to relate individuals, binary in OWL
- Two property variants:

1. ObjectProperties

```
:author a owl:ObjectProperty .
```

2. DatatypeProperties

```
:publicationYear a owl:DatatypeProperty .
```

ObjectProperties

Domain and Range of object properties

```
:author a owl:ObjectProperty ;
          rdfs:domain          :Book ;
          rdfs:range :Writer .
```

Domain: $\exists$author.$\top \sqsubseteq$ Book
Range: $\top \sqsubseteq \forall$author.Writer

DatatypeProperties

Domain and range of datatype properties

```
:publicationYear a owl:DatatypeProperty ;
                 rdfs:domain           :Book ;
                 rdfs:range xsd:integer .
```

$$\exists \text{publicationYear}.\top \sqsubseteq \text{Book}$$
$$\top \sqsubseteq \forall \text{publicationYear}.\text{Integer}$$

Many XML datatypes can be used

Properties and Individuals

```
:Book a owl:Class .
:Writer a owl:Class .
:GeorgeOrwell a :Writer .

:author a owl:ObjectProperty ;
            rdfs:domain :Book;
            rdfs:range  :Writer .

:publicationYear a owl:DatatypeProperty ;
                        rdfs:domain :Book;
                        rdfs:range  xsd:integer .

:NineteenEightyFour a :Book ;
         :author   :GeorgeOrwell ;
         :publicationYear 1948 .
```

Tbox and Abox do not have to be separated in OWL!

Class Hierarchies

```
:Novel a owl:Class ;
         rdfs:subClassOf :Book .

:Book a owl:Class ;
         rdfs:subClassOf :Work .

:Work a owl:Class .
```

$$\texttt{Novel} \sqsubseteq \texttt{Book}$$
$$\texttt{Book} \sqsubseteq \texttt{Work}$$

(Via inference, it can be entailed that Novel is also a subclass of Work)

Class Hierarchies and Disjunctiveness

```
:Book a owl:Class .
:Writer a owl:Class .
:Novel a owl:Class ;
         rdfs:subClassOf :Book .
:Poet a owl:Class ;
         rdfs:subClassOf :Writer .
:Book owl:disjointWith :Writer.
```

$$\texttt{Novel} \sqsubseteq \texttt{Book}$$
$$\texttt{Poet} \sqsubseteq \texttt{Writer}$$
$$\texttt{Book} \sqcap \texttt{Writer} \sqsubseteq \bot$$

(Via inference, it can be entailed that `Novel` and `Poet` are also disjoint classes)

Class Hierarchies and Equivalence

```
:Writer a owl:Class .
:Author a owl:Class .
:Poet a owl:Class ;
         rdfs:subClassOf :Writer .

:Writer owl:equivalentClass :Author .
```

$Poet \sqsubseteq Writer$

$Writer \equiv Author$

(Via inference it can be entailed that `Poet` is also an `Author`)

Individuals – equivalence

```
:NineteenEightyfour a :Novel ;
         :author :GeorgeOrwell ;
         :publicationYear 1948;
         owl:sameAs :ARX012345 .
:Novel a owl:Class ;
         rdfs:subClassOf :Book .
:Book a owl:Class.
```

Via inference it can be entailed that ARX012345 is a Book

Differences of Individuals via owl:differentFrom:

```
:ARX012345 a :Novel ;
         owl:differentFrom :ARX0123456 .
```

OWL Recap

Overview of the basic constructs:

1. Classes:

```
:Book a owl:Class .
:Novel a owl:Class ;
      rdfs:subClassOf :Book
:Book owl:disjointWith :Writer .
:Writer owl:equivalentClass :Author .
```

2. Properties:

```
:author a owl:ObjectProperty;
                rdfs:domain :Book;
                rdfs:range :Writer .

:publicationYear a owl:DatatypeProperty;
      rdfs:domain : Book;
      rdfs:range xsd:integer .
```

3. Individuals:

```
:NineteenEightyfor a :Book .
:NineteenEightyfour owl:sameAs :ARX012345 .
```

OWL Recap

Overview of the basic constructs:

1. Classes:

```
:Book a owl:Class .
:Novel a owl:Class ;
      rdfs:subClassOf :Book
:Book owl:disjointWith :Writer .
```

Further Reading:
OWL Complex Classes in Appendix 2

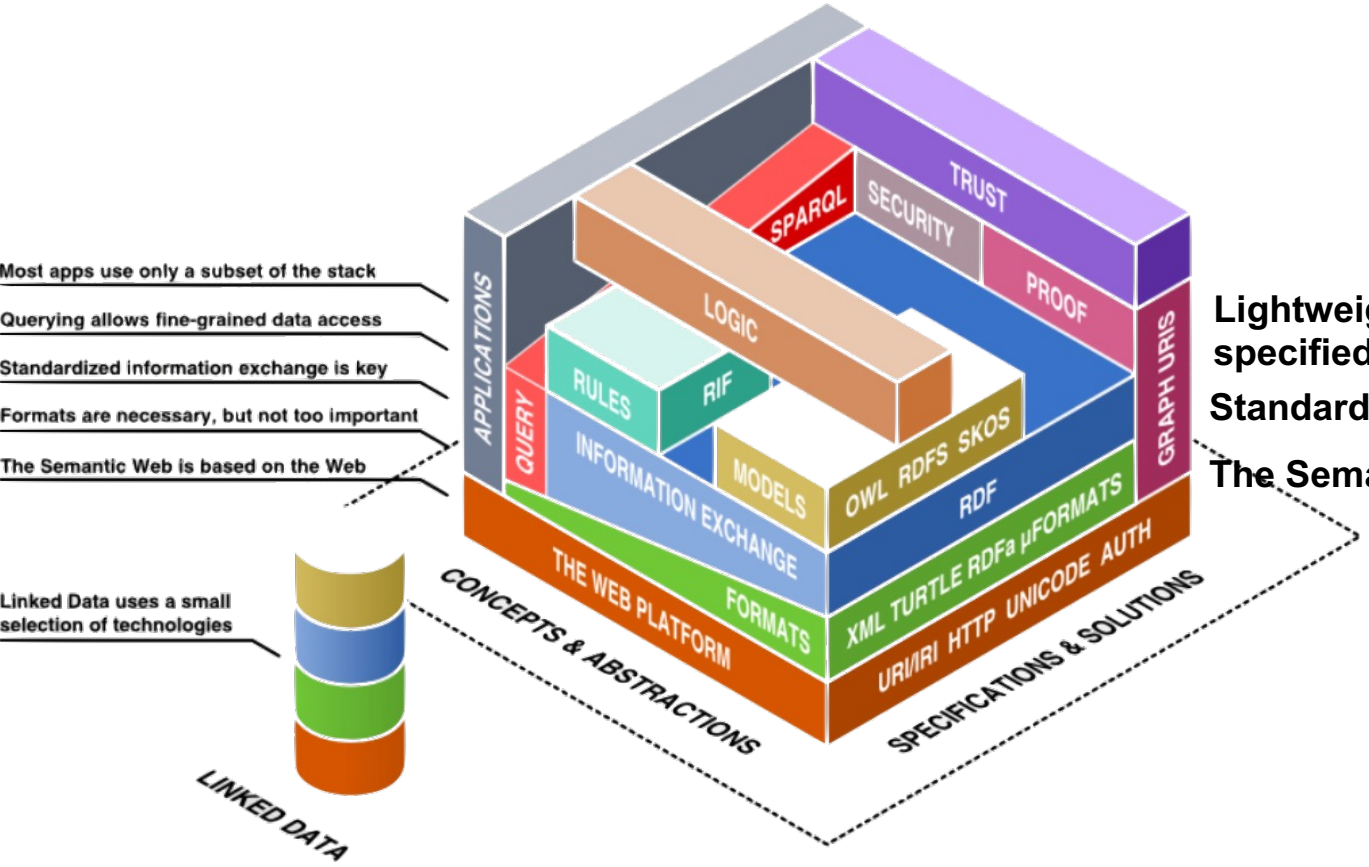
2. Pro

```
:publicationYear a owl:DatatypeProperty;
      rdfs:domain : Book;
      rdfs:range xsd:integer .
```

3. Individuals:

```
:NineteenEightyfor a :Book .
:NineteenEightyfour owl:sameAs :ARX012345 .
```

Summary: The Semantic Web Technology Stack



Lightweight and heavyweight ontologies specified with RDF(S) and OWL

Standard information exchange is key

The Semantic Web is based on the Web

Every layer can access the functionality of the layers below

http://bnode.org/blog/2009/07/08/the-semantic-web-not-a-piece-of-cake

Summary

- RDF and Linked Data
- RDFS
- Ontology Representation and OWL

Read Appendix 2 in preparation for the assignment!

Appendix 1
RDF

Blank Node Identifiers

```
exstaff:85740 exterms:address _:johnaddress .
_:johnaddress exterms:street "1501 Grant Avenue" .
_:johnaddress exterms:city "Bedford" .
_:johnaddress exterms:state "Massachusetts" .
_:johnaddress exterms:postalCode "01730" .
```
Turtle

- Each distinct blank node must be given a different blank node identifier
- Blank node identifiers have significance only within a single graph
- Blank node identifiers may only appear as subjects or objects in triples (not predicates)

RDF Data Structures: Lists

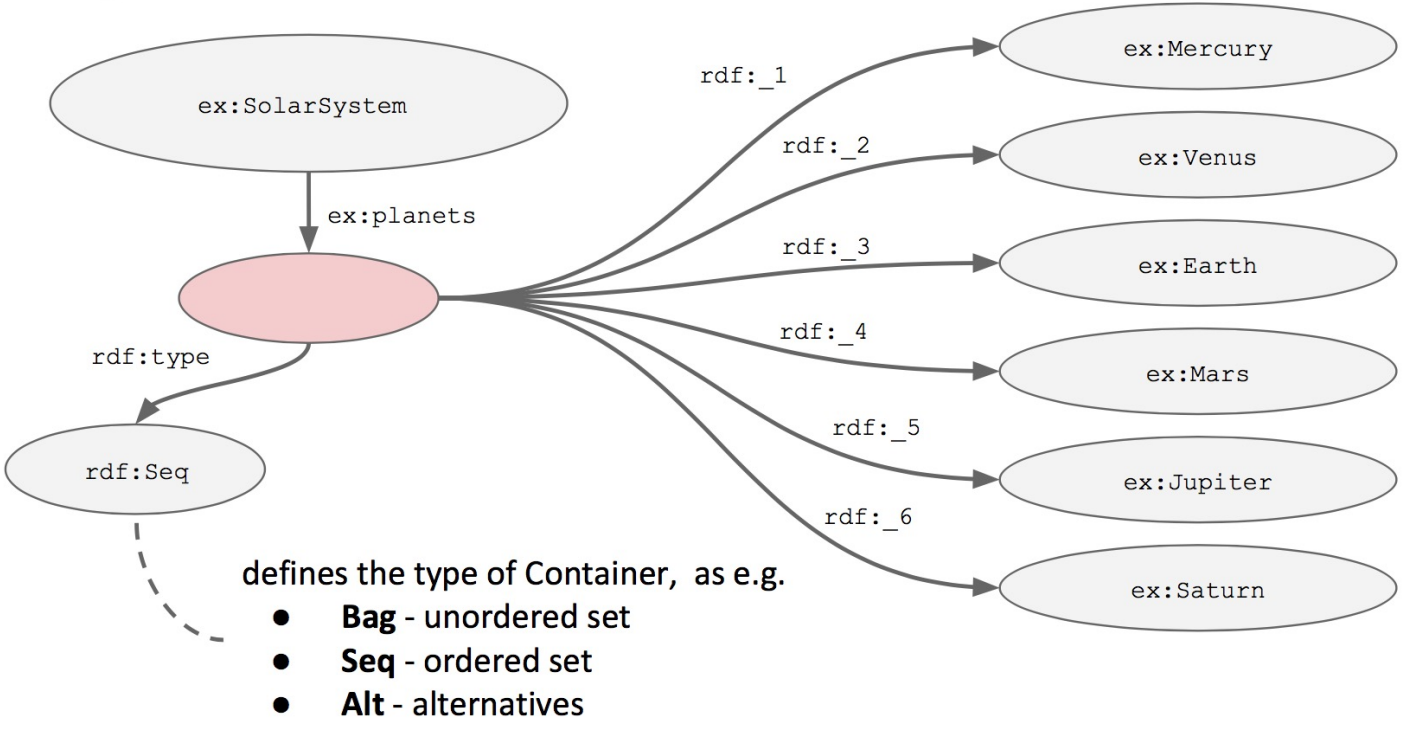
- General Data structure to enumerate resources or literals
- Only shortcuts without additional semantic expressivity (syntactic sugar)

Distinguish:

- **Container:**
  open list, i.e. extension (new entries) possible

- **Collections:**
  closed list, i.e. no extension possible

RDF Container (Open Lists)

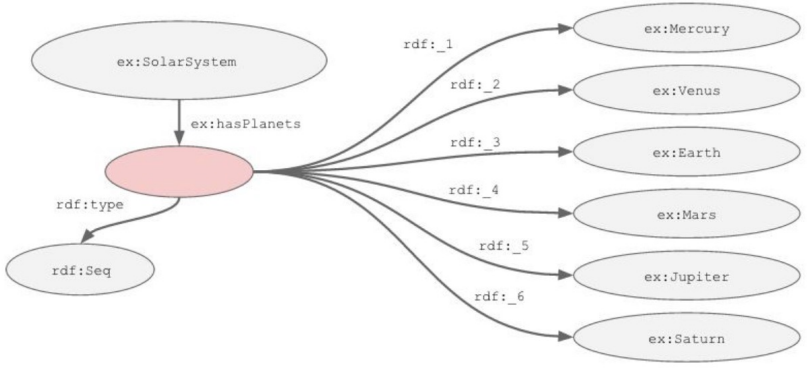


RDF Container (Open Lists)

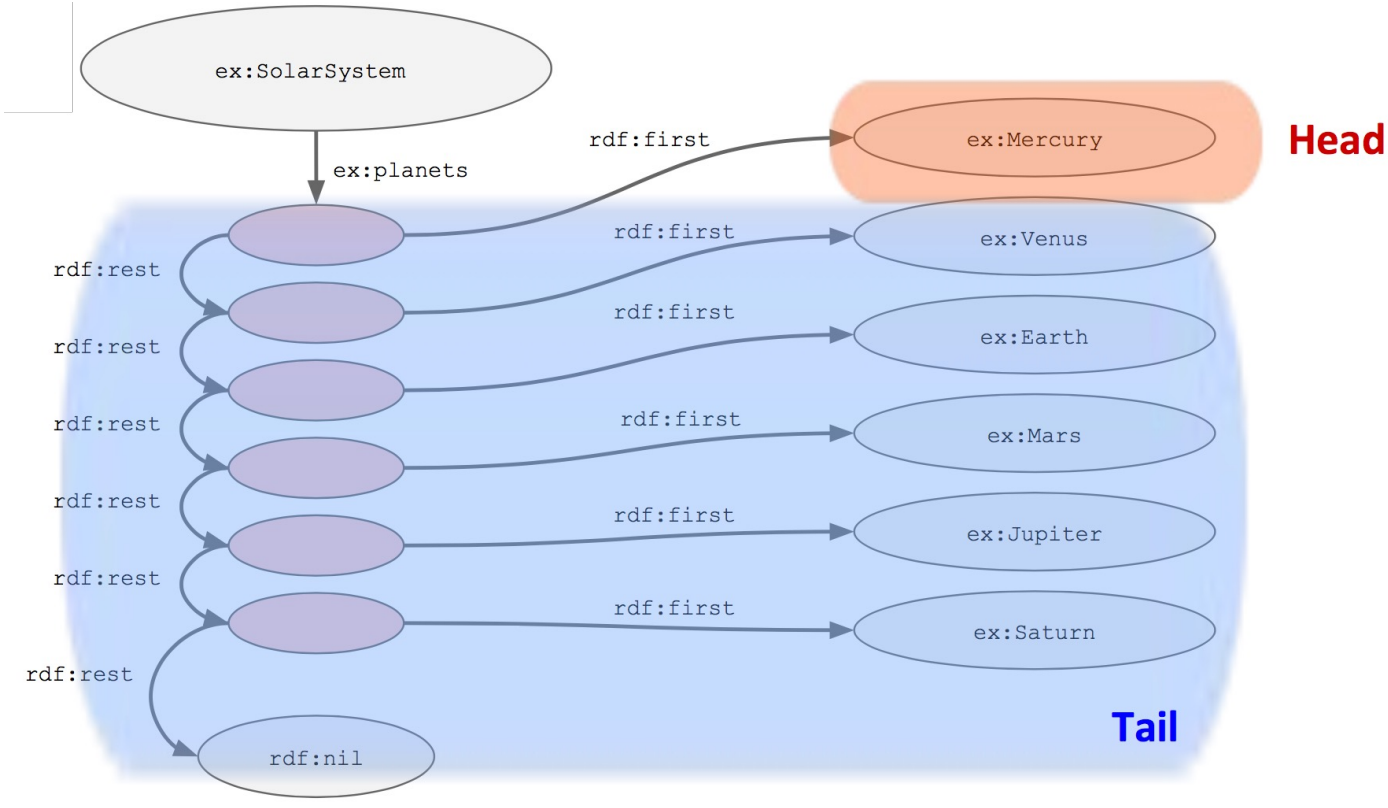
```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex: <http://example.org/test#> .

ex:SolarSystem ex:planets  [
  a rdf:Seq ;
  rdf:_1 ex:Mercury ;
  rdf:_2 ex:Venus ;
  rdf:_3 ex:Earth ;
  rdf:_4 ex:Mars ;
  rdf:_5 ex:Jupiter ;
  rdf:_6 ex:Saturn
] .
```

Turtle

= **rdf:type** rdf:Seq

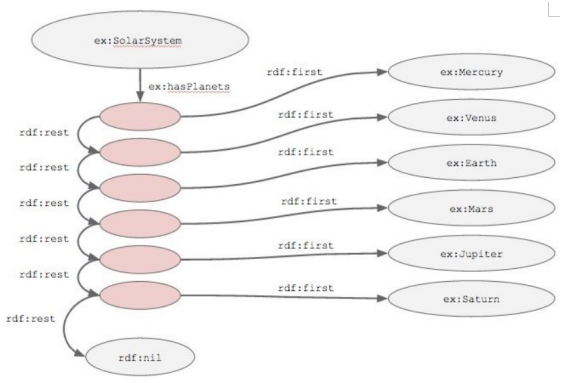


RDF Collection (Closed Lists)



RDF Collection (Closed Lists)

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex: <http://example.org/test#> .

ex:SolarSystem ex:planets [
  rdf:first ex:Mercury ; rdf:rest [
   rdf:first ex:Venus ; rdf:rest [
    rdf:first ex:Earth ; rdf:rest [
     rdf:first ex:Mars ; rdf:rest [
      rdf:first ex:Jupiter ; rdf:rest [
       rdf:first ex:Saturn ;
       rdf:rest rdf:nil
     ]
    ]
   ]
  ]
 ]
] .
```




```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex: <http://example.org/test#> .

ex:SolarSystem ex:planets (
  ex:Mercury ex:Venus ex:Earth ex:Mars ex:Jupiter ex:Saturn
) .
```

Turtle

Turtle provides a nice syntax for RDF collections: simply put elements into parentheses

Appendix 2
OWL Complex Classes and Property Restrictions

Closed Classes (Nominals)

```
:Novel a owl:Class ;
:AnimalFarm a :Novel .
:NineteenEightyfour a :Novel .
:NovelsInStore a owl:Class ;
               owl:oneOf
              (:AnimalFarm
               :NineteenEightyfour) .
```

$NovelsInStore \sqsubseteq \{AnimalFarm, NineteenEightyfour\}$

This says that there are only two novels available in the store.

Property Restrictions

- Are used to describe complex classes via properties

- Restrictions on values:
  - owl:hasValue
  - owl:allValuesFrom
  - owl:someValuesFrom

- Restrictions on cardinality
  - owl:cardinality
  - owl:minCardinality
  - owl:maxCardinality

Property Restrictions with Constants

```
:Writer a owl:Class .
:Book a owl:Class .
:GeorgeOrwell a :Person .
:author a owl:ObjectProperty ;
        rdfs:domain :Book ;
```

```
:OrwellsBooks a owl:Class ;
              rdfs:subClassOf
            [ a owl:Restriction ;
              owl:onProperty :author ;
              owl:hasValue :GeorgeOrwell
 ] .
```

$OrwellsBooks \sqsubseteq author.(GeorgeOrwell)$

- Class OrwellsBooks is described via fixed value assignment (=constant) of the individual GeorgeOrwell to the property author

Property Restrictions: owl:allValuesFrom

Fixes all instances of a specific class C as allowed range for a property P → (Universal Quantification) ∀P.C

```
:Poetry a owl:Class ;
        rdfs:subClassOf
      [ a owl:Restriction ;
        owl:onProperty :author ;
        owl:allValuesFrom :Poet
] .
```

Poetry ⊑ ∀author.Poet

```
:Poet a owl:Class .
:Poetry a owl:Class rdfs:subClassOf :Writer .
:author a owl:ObjectProperty ;
        rdfs:domain :Book ;
        rdfs:range :Writer .
```

Value restriction: Universal restriction

Universal restriction over an object property expression: contains those individuals that are connected through an object property expression **only** to instances of a given class expression.

- for a given property, universal restrictions *do not specify the existence* of a relationship. They merely state that if a relationship exists for the property then it must be to individuals that are members of a specific class.
- may be denoted by the universal quantifier (∀). In OWL `owl:allValuesFrom.`
- It can be read as **only** or **no other than**
  - **VegetarianPizza hasTopping only VegetarianTopings**
  - **VegetarianPizza hasTopping no other than VegetarianTopings**

```
<owl:Class rdf:about=”VegetarianPizza">
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="hasTopping"/>
                        <owl:allValuesFrom rdf:resource=”VegetarianTopping"/>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

How about an individual that has no “hasTopping” relation? Will it be also considered a “Vegetarian” Pizza?

“trivial satisfaction of universal restriction”

Property Restrictions: owl:someValuesFrom

Describes that there **must exist** an individual for property `P` and fixes its range to class `C` → (existential quantification) $\exists P.C$

```
:Reader a owl:Class ;
        rdfs:subClassOf
           [ a owl:Restriction ;
             owl:onProperty :reads ;
             owl:someValuesFrom :Book
           ] .
```

```
:Reader a owl:Class .
:Book a owl:Class .
:reads a owl:ObjectProperty ;
         rdfs:domain :Person ;
         rdfs:range :ReadingMaterial .
```

$Reader \sqsubseteq \exists reads.Book$

Value restriction: Existential restriction

An existential restriction describes a class of individuals that have *at least one (some)* relationship along a specified property to an individual that is a member of a specified class.

- To be a member of the class, the individual *must have at least one relation* along the given property to individuals that are members of a specific class (necessary condition).
- may be denoted by the existential quantifier (∃). In OWL `owl:someValuesFrom.`
- It can be read as **some** or **at least one**
  - **Pizza hasBase some PizzaBase**
  - **Pizza hasBase at least one PizzaBase**

```
<owl:Class rdf:about=”Pizza">
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="hasBase"/>
                        <owl:someValuesFrom rdf:resource=”PizzaBase"/>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

Combining Existential And Universal Restrictions

A common pattern is combining universal and existential restrictions to define classes that have *at least one* relation of a property and all values of the property range over individuals of a certain class.

```
<owl:Class rdf:about="VegetarianPizza">
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="hasTopping"/>
                        <owl:someValuesFrom rdf:resource="VegetarianTopping"/>
                </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="hasTopping"/>
                        <owl:allValuesFrom rdf:resource="VegetarianTopping"/>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

Cardinality Restrictions

- `owl:cardinality` restricts to an exact number
- `owl:minCardinality`, `owl:maxCardinality` restricts to upper / lower bounds

```
:Tetralogy a owl:Class ;
       rdfs:subClassOf
       [ a owl:Restriction ;
         owl:onProperty  :hasVolumes ;
         owl:cardinality 4
       ] .
```

$\texttt{Tetralogy} \sqsubseteq (=4)\texttt{hasVolumes}$

```
:hasVolumes a owl:DatatypeProperty ;
            rdfs:domain :Thing ;
            rdfs:range xsd:integer .
```

Property Specifications

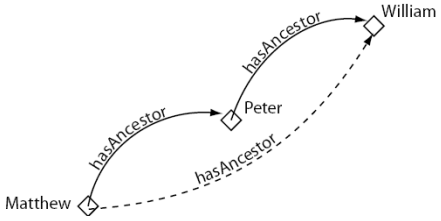
- Property hierarchies can be created via specializations: rdfs:subPropertyOf
- Inverse properties are defined via owl:inverseOf
- Identical properties are defined via owl:equivalentProperty

```
:isMadeOf a owl:ObjectProperty ;
          rdfs:subPropertyOf  :consistsOf .

:reads a owl:ObjectProperty ;
       owl:inverseOf  :isReadBy .
```

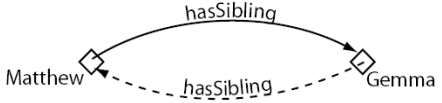
$$isMadeOf \sqsubseteq consistsOf$$

$$reads^- \equiv isReadBy$$

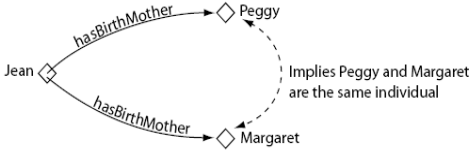
Property Characteristics



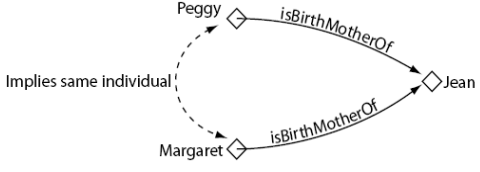
- **owl:TransitiveProperty**
  e.g.: if `isPartOf(a,b)` and `isPartOf(b,c)`
  then it holds that `isPartOf(a,c)`



- **owl:SymmetricProperty**
  e.g.: if `isNeighborOf(a,b)`,
  then it holds that `isNeighborOf(b,a)`

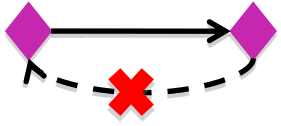


- **owl:FunctionalProperty**
  e.g.: if `hasMother(a,b)` and `hasMother(a,c)`,
  then it holds that `b=c`

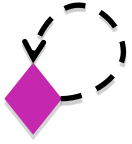


- **owl:InverseFunctionalProperty**
  e.g.: if `isBirthMotherOf(b,a)` and `isBirthMotherOf(c,a)`
  then it holds that `b=c`

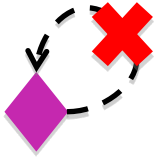
Property Relationships



owl:AsymmetricProperty
e.g.: if it holds that `isLeftOf(a,b)`
then it is not possible that also `isLeftOf(b,a)`



owl:ReflexiveProperty
e.g.: `isRelatedTo(x,x)`



owl:IrreflexiveProperty
e.g.: if `isParentOf(x,y)`
then $x \neq y$

Example: Transitive Properties

```
:isPublishedBefore a owl:ObjectProperty ;
        a owl:TransitiveProperty ;
        rdfs:domain owl:Book ;
        rdfs:range owl:Book .
:AnimalFarm a owl:Book ;
        :isPublishedBefore :NineteenEightyfour .
:BraveNewWorld a :Book ;
         : isPublishedBefore  :AnimalFarm .
```

Via inference it can be entailed that BraveNewWorld has also been published before NineteenEightyfour .

Disjunctive Properties

Two properties R and S are **disjunctive**, if two individuals x,y are never related via both properties

```
:hasParent a owl:ObjectProperty ;
   owl:propertyDisjointWith :hasChild .
```

Shortcut for several disjunctive properties:

```
[] rdf:type owl:AllDisjointProperties ;
   owl:members ( :hasParent :hasChild :hasGrandchild ) .
```

OWL Recap

Basic constructs:

1. **Classes:**

```
:Book a owl:Class .
:Novel a owl:Class ;
     rdfs:subClassOf :Book
:Book owl:disjointWith :Writer .
:Writer owl:equivalentClass :Author .
```

2. **Properties:**

```
:author a owl:ObjectProperty;
               rdfs:domain :Book;
               rdfs:range :Writer .

:publicationYear a owl:DatatypeProperty;
     rdfs:domain : Book;
     rdfs:range xsd:integer .
```

3. **Individuals:**

```
:NineteenEightyfor a :Book .
:NineteenEightyfour owl:sameAs :ARX012345 .
```

- **Complex classes** (logical class constructors, property restrictions)
- **Property relationships**

TU WIEN Informatics

188.399 Introduction to Semantic Systems

(2.0 VU / 3.0 ECTS)

2023W

Katja Hose
Emanuel Sallinger

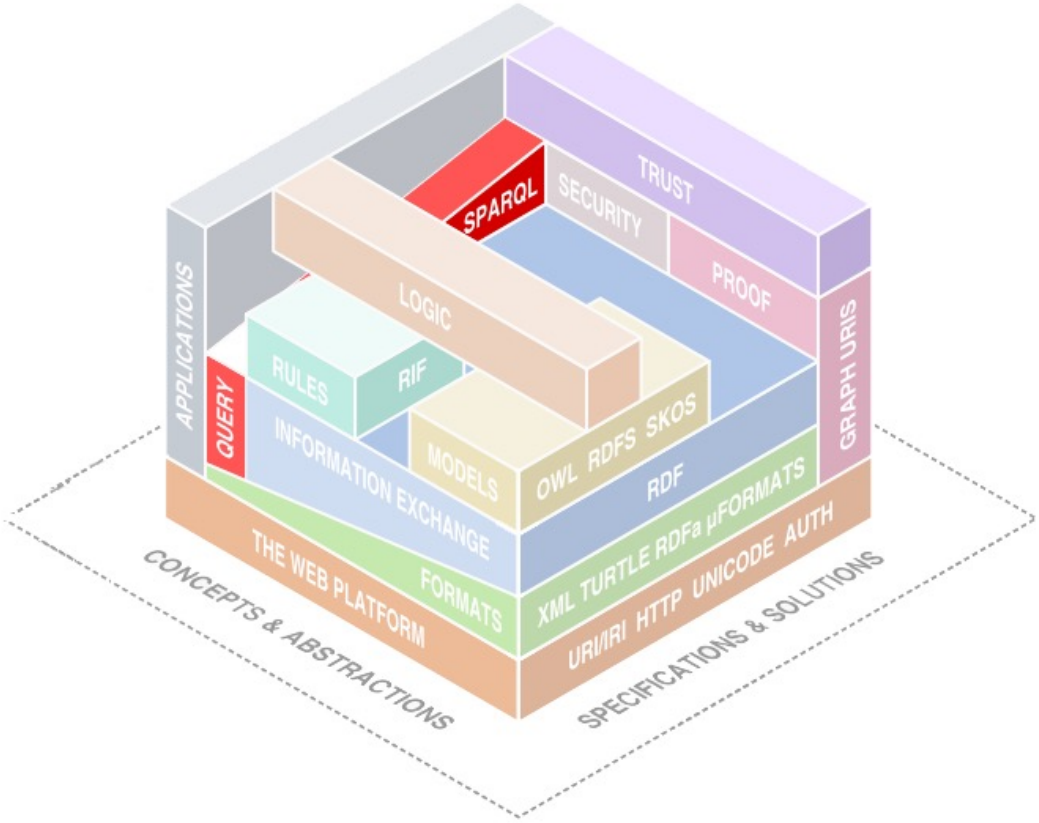
iss@dbai.tuwien.ac.at

Learning Goals

- Querying Knowledge Graphs with SPARQL
  - Basic SPARQL
  - Advanced SPARQL
- Updating Knowledge Graphs with SPARQL

SPARQL

Semantic Web Foundations



http://bnode.org/blog/2009/07/08/the-semantic-web-not-a-piece-of-cake

SPARQL - Agenda

- Basics
  - **Introductory notions**
  - The SELECT query
  - Other query types
- Advanced notions
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - Federated queries
  - Entailment regimes

Introduction

- SPARQL = **S**PARQL **P**rotocol and **R**DF **Q**uery **L**anguage
  - Semantic query language
  - Developed by W3C RDF Data Access Working group (DAWG)
  - January 2008: SPARQL 1.0
  - March 2013: SPARQL 1.1
    - https://www.w3.org/TR/sparql11-query/

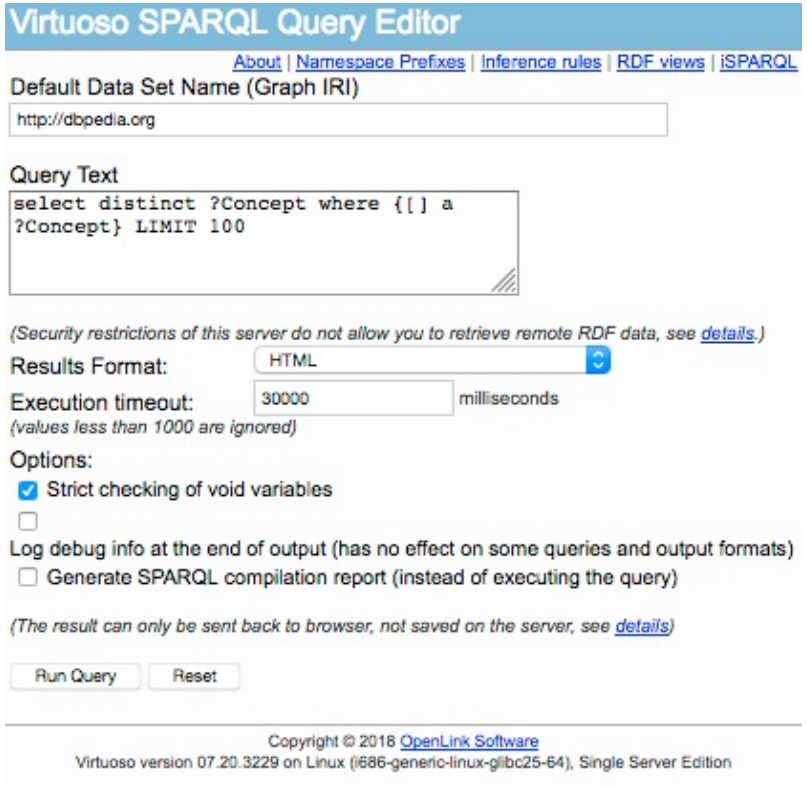
- SPARQL lets us
  - Retrieve and manipulate data stored in RDF
  - Explore data by querying unknown relationships
  - Perform complex joins of disparate databases in a single, simple query
  - Transform RDF data from one vocabulary to another

- Triple stores
  - OpenLink Virtuoso (https://virtuoso.openlinksw.com)
  - Apache Jena ARQ (http://jena.apache.org)
  - Many more: https://www.w3.org/wiki/SparqlImplementations

SPARQL Fundamentals

- **RDF graph:** Set of RDF assertions, manipulated as a labeled directed graph.
- **RDF dataset**: set of RDF triples; comprised of
  - one default graph
  - zero or more named graphs
- **SPARQL protocol client:** HTTP client that sends requests for SPARQL Protocol operations (queries or updates)
- **SPARQL protocol service:** HTTP server that services requests for SPARQL Protocol operations
- **SPARQL endpoint:** The URI at which a SPARQL Protocol service listens for requests from SPARQL clients

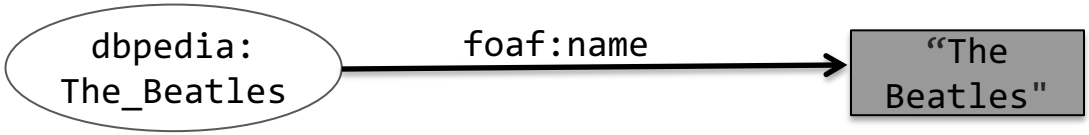
https://dbpedia.org/sparql



SPARQL Syntax Basics

RDF triple

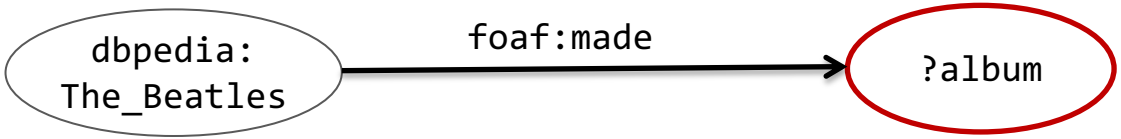
```
dbpedia:The_Beatles foaf:name "The Beatles" .
```



RDF triple pattern

```
dbpedia:The_Beatles foaf:made ?album .
```

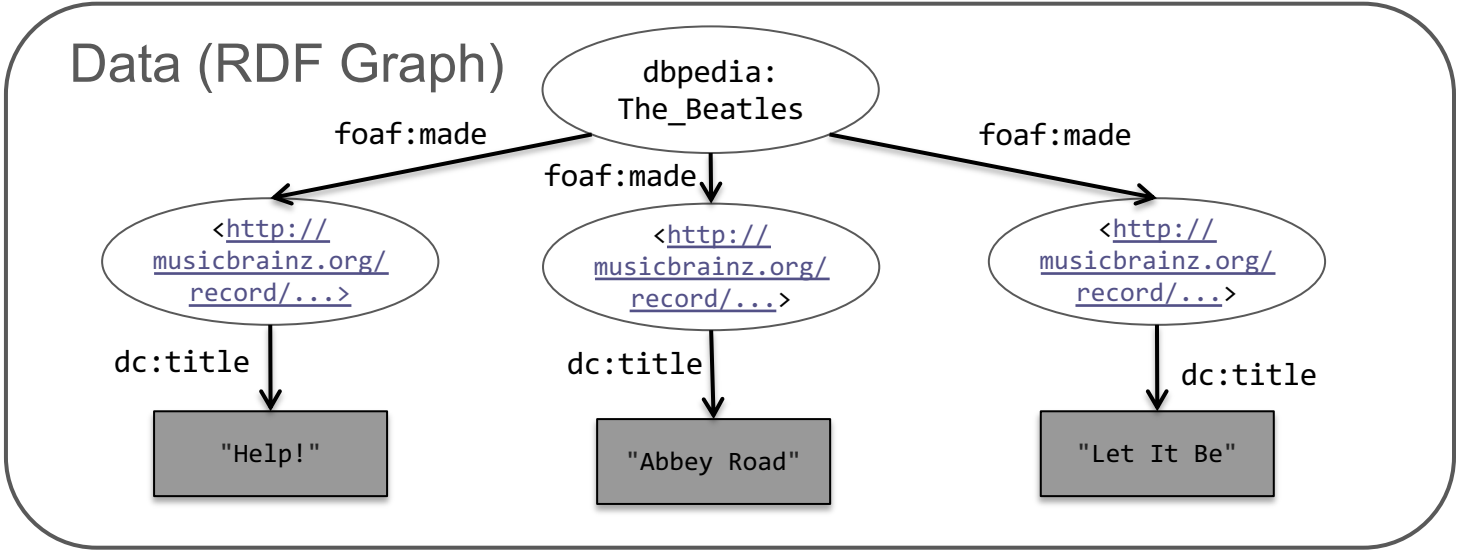
Just like triples, but any of their parts can be replaced by a variable.

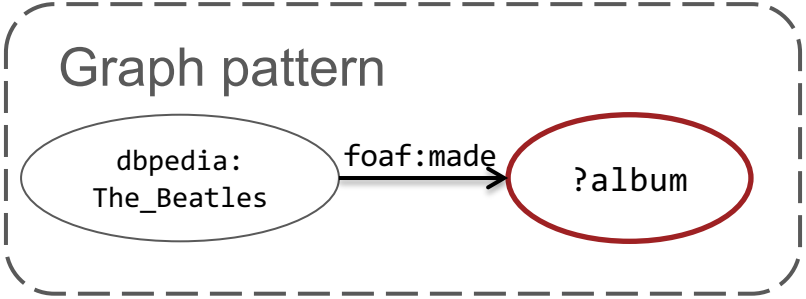


Other triple pattern examples:

```
?album mo:track ?track .
?s ?p ?o .
```

SPARQL Query Example



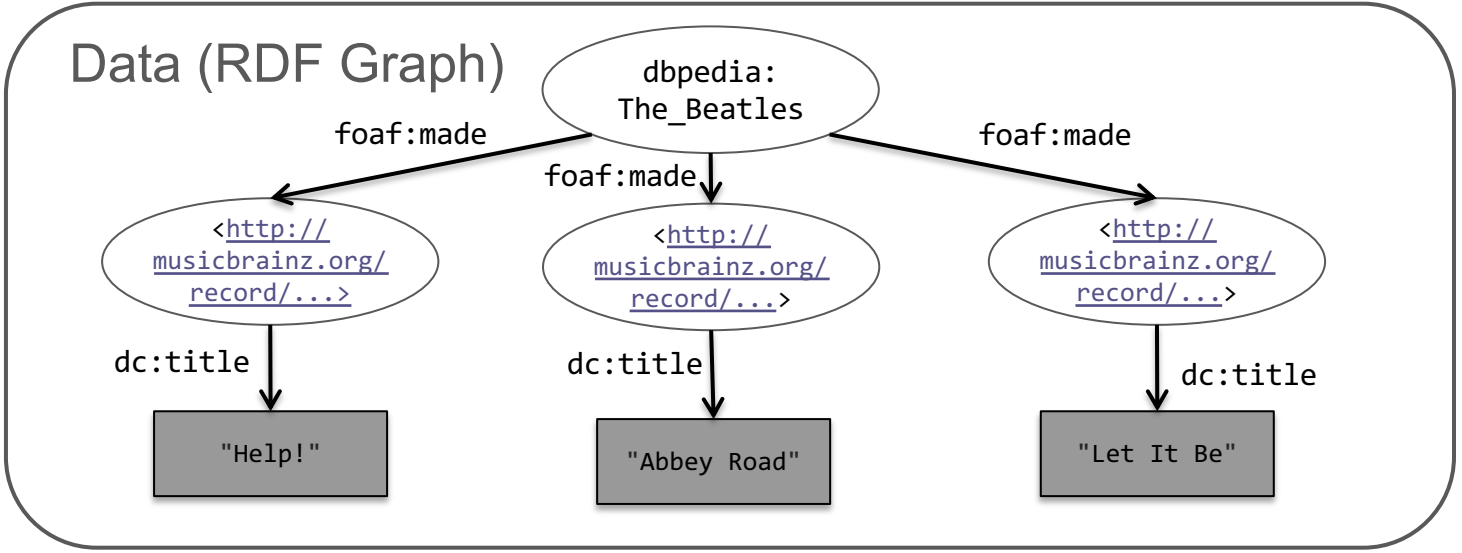


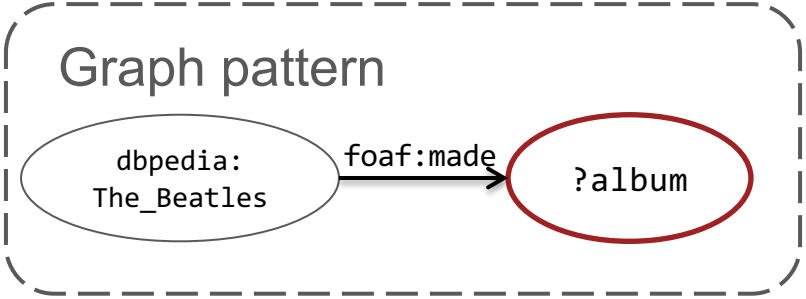
SPARQL

Results

What are the results?
→ Menti

SPARQL Query Example



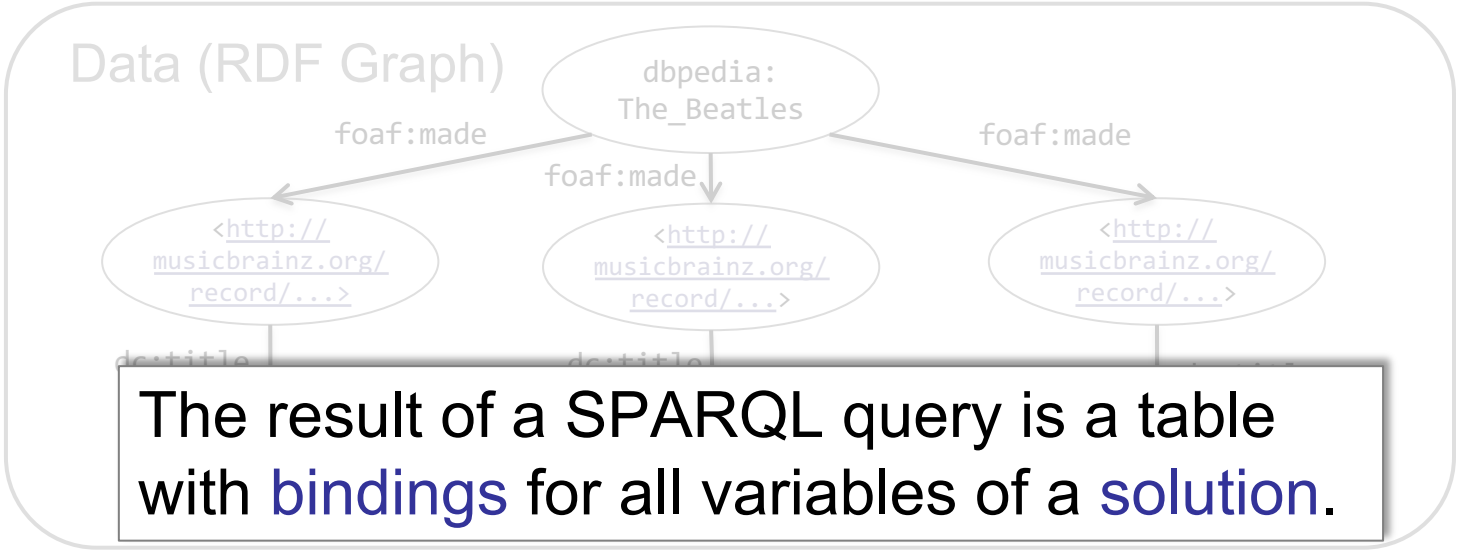


SPARQL

Results

?album
<http://musicbrainz.org...>
<http://musicbrainz.org...>
<http://musicbrainz.org...>

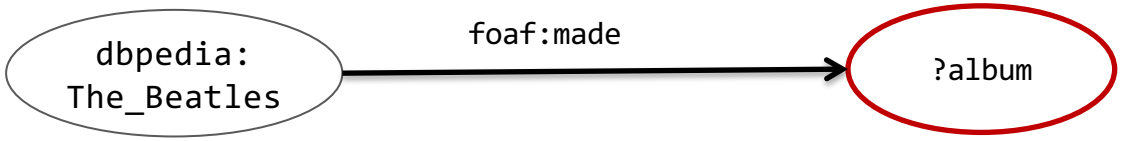
SPARQL Query Example



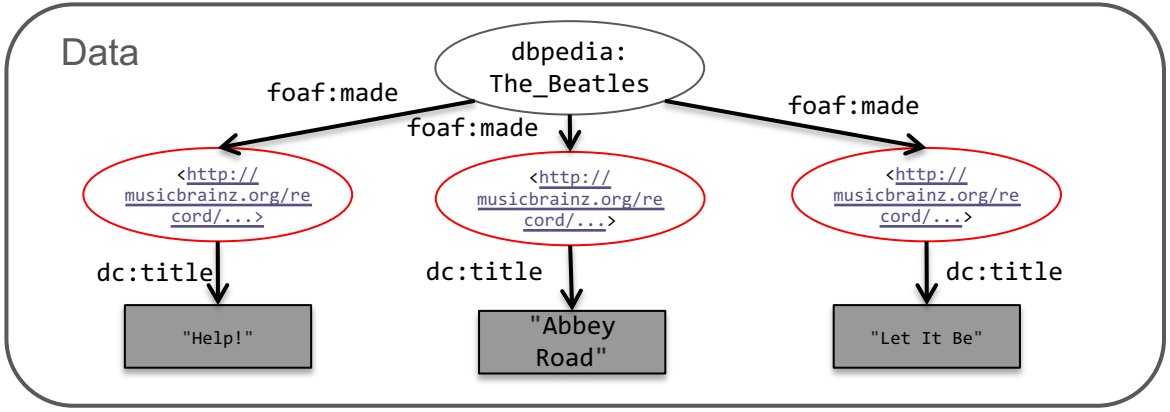
The result of a SPARQL query is a table with bindings for all variables of a solution.

SPARQL Queries

- Key idea: Pattern matching
- **Queries** (graph patterns):
  - describe subgraphs of an RDF graph
  - Roughly correspond to RDF graphs specified in Turtle syntax but containing variables (prefixed by **?**)



- **Results**: Subgraphs matching the graph pattern
  → or rather a table with the variables of those subgraphs



Adapted from EUCLID learning materials – "Querying Linked Data"

Structure of a SPARQL query

```
# Prefix declarations, for abbreviating URIs
PREFIX dbpedia: http://dbpedia.org/resource/
...

# Result clause, identifying what information to return from the query
SELECT/ASK/CONSTRUCT ...

# Dataset definitions, stating what RDF graph(s) are being queried
FROM  <http://musicbrainz.org/20130302>

# Query pattern, specifying what to query for in the underlying dataset
WHERE {
    ...
}

# Query modifiers, slicing, ordering, rearranging query results
ORDER BY ...
```

SPARQL - Agenda

- Basics
  - Introductory notions
  - **The SELECT query**
  - Other query types
- Advanced notions
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - Federated queries
  - Entailment regimes

Anatomy of a SELECT query

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?album
FROM  <http://musicbrainz.org/20130302>
WHERE {
    dbpedia:The_Beatles foaf:made ?album .
}
```

Query form:

- SELECT retrieves variables and their bindings as a table
- Other query forms: ASK, DESCRIBE or CONSTRUCT

?album
<http://musicbrainz.org...>
<http://musicbrainz.org...>
<http://musicbrainz.org...>

Anatomy of a SELECT query

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?album
FROM  <http://musicbrainz.org/20130302>
WHERE {
    dbpedia:The_Beatles foaf:made ?album .
}
```

Data set specification

- This clause is optional
- `FROM` or `FROM NAMED`
- Indicates the sources for the data against which to find matches

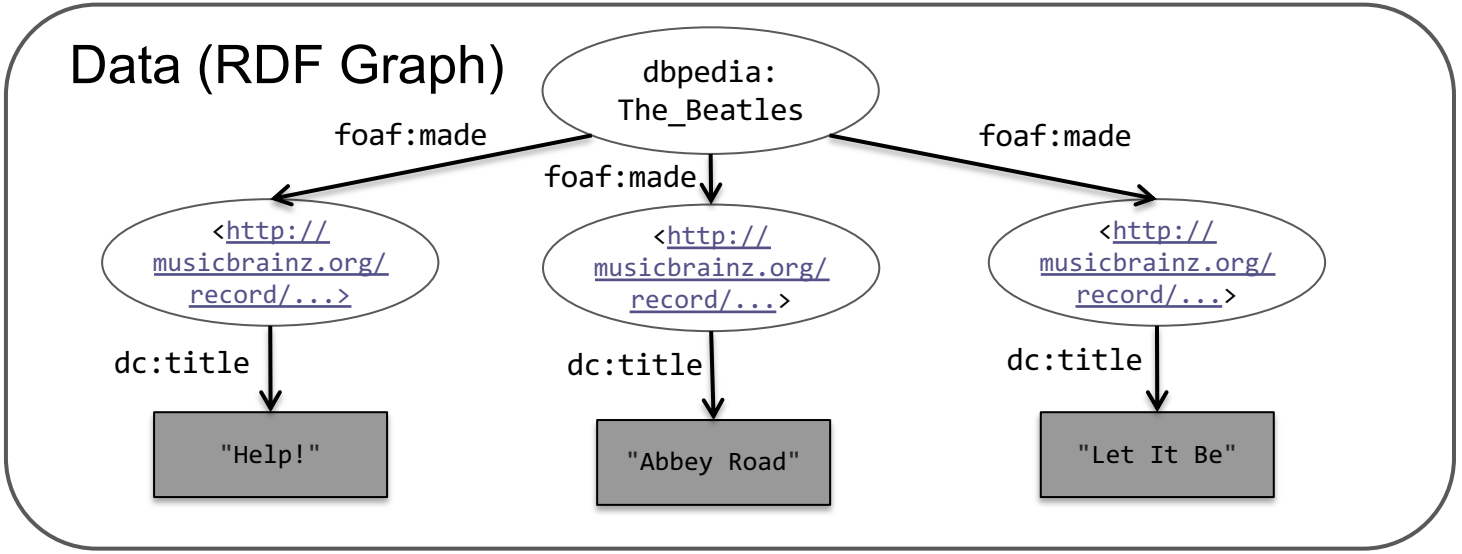
Anatomy of a SELECT query

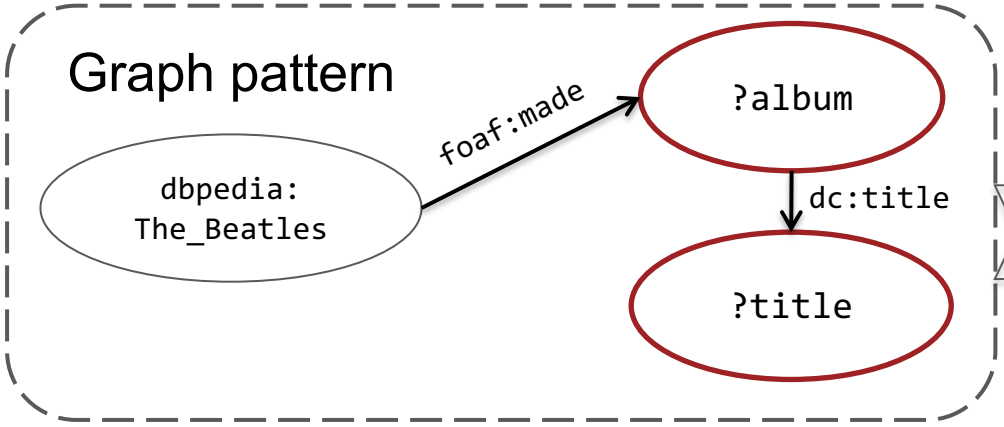
```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?album
FROM  <http://musicbrainz.org/20130302>
WHERE {
    dbpedia:The_Beatles foaf:made ?album .
}
```

Query pattern

- Generalizes Turtle with variables and keywords
  - The final "." after last triple pattern is optional
- One or multiple triple patterns between "{}"
  - together describing a graph pattern to be matched against RDF triples specified in the FROM clause

SPARQL Query Example

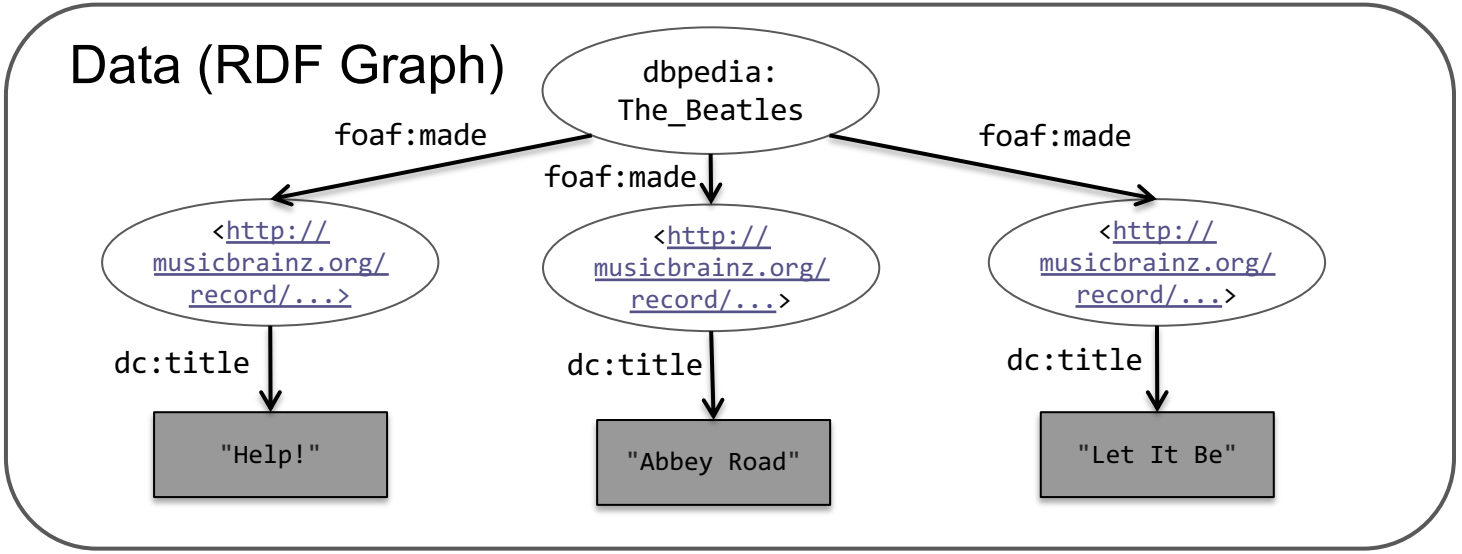


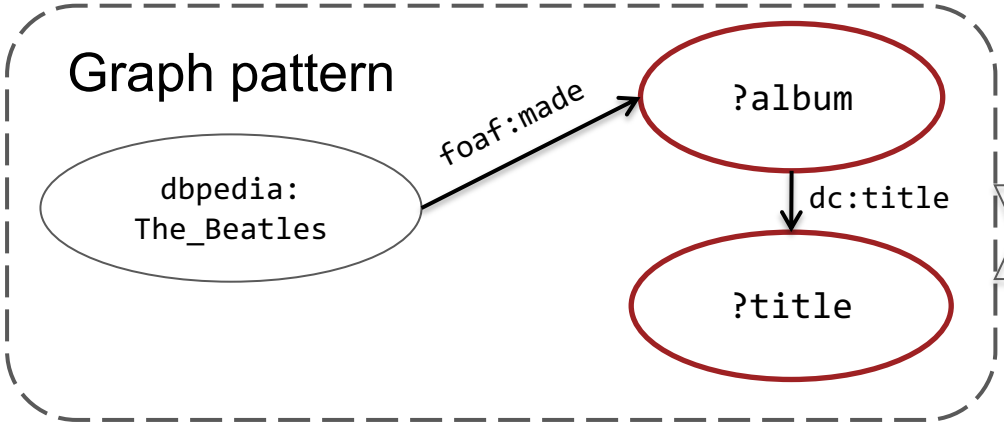


Results

What are the results?
→ Menti

SPARQL Query Example





Results

?album	?title
<http://...>	"Help!"
<http://...>	"Abbey Road"
<http://...>	"Let It Be"

Anatomy of a SELECT query

```
PREFIX dbpedia, foaf, dc, mo

SELECT *
FROM  <http://musicbrainz.org/20130302>
WHERE {
    dbpedia:The_Beatles foaf:made ?album .
    ?album dc:title ?title .
}
ORDER BY ?title
```

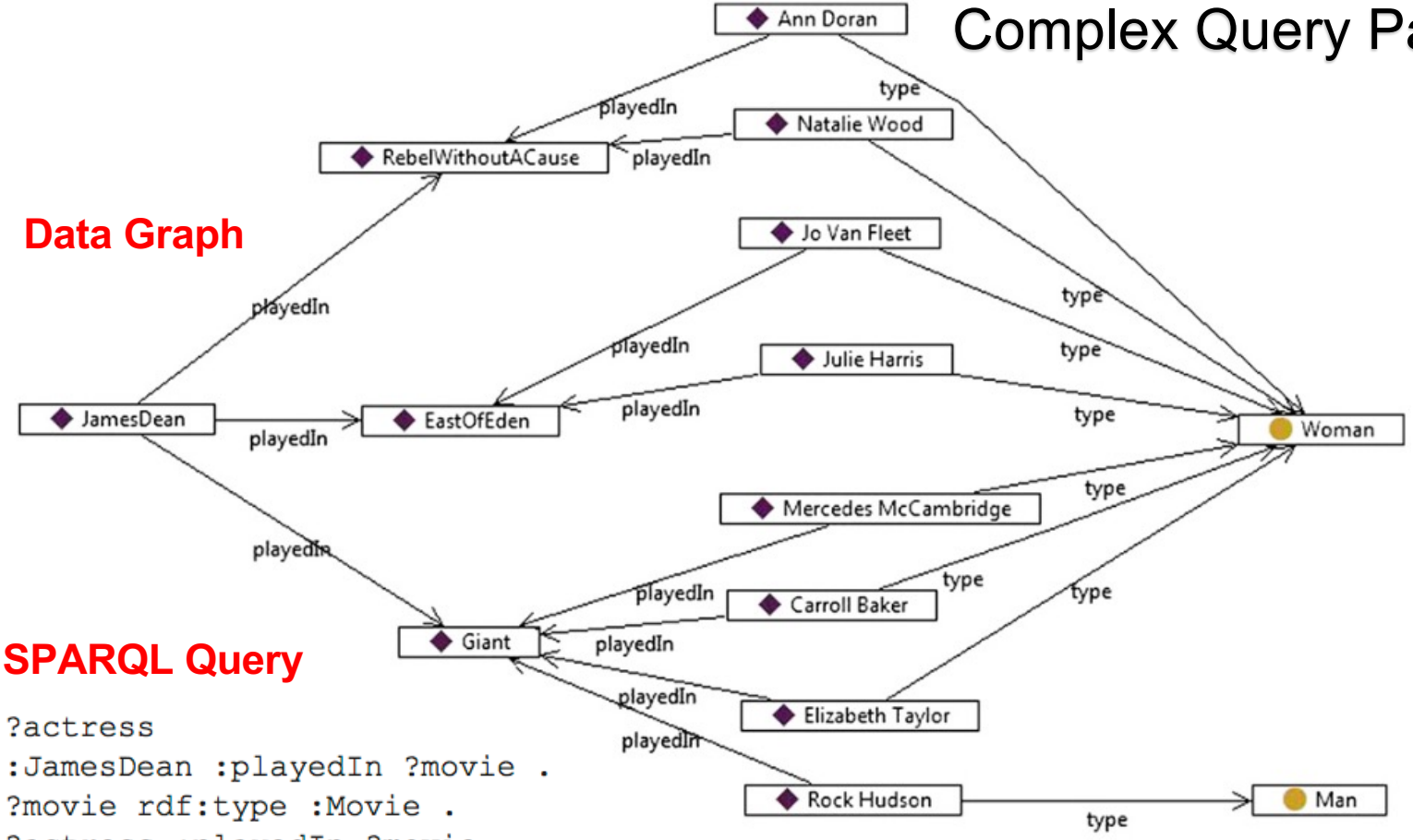
Solution modifier:

- Modify the result set
- `ORDER BY`, `LIMIT` or `OFFSET`
- `GROUP BY`
- `SELECT *`

?album	?title
<http://...>	"Abbey Road"
<http://...>	"Help!"
<http://...>	"Let It Be"

Complex Query Patterns

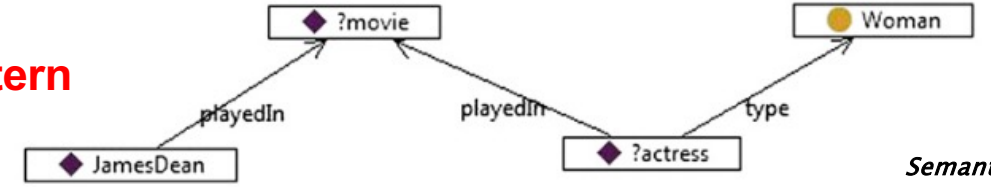
Data Graph



SPARQL Query

```
SELECT ?actress
WHERE {:JamesDean :playedIn ?movie .
       ?movie rdf:type :Movie .
       ?actress :playedIn ?movie .
       ?actress rdf:type :Woman }
```

Query Pattern



SPARQL - Agenda

- Basics
  - Introductory notions
  - The SELECT query
  - **Other query types**
- Advanced notions
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - Federated queries
  - Entailment regimes

Query Types

SPARQL supports the following query types

- **SELECT**
  returns variables and their bindings directly
- **ASK**
  tests whether or not a query pattern has a solution
  Returns true or false
- **DESCRIBE**
  returns a single RDF graph containing RDF data about a resource
- **CONSTRUCT**
  returns a single RDF graph specified by a graph template

SPARQL ASK

ASK - tests whether or not a query pattern has a solution. Returns yes/no

Query: *Is Paul McCartney a member of 'The Beatles'?*

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX mo: <http://purl.org/ontology/mo/>

ASK WHERE { dbpedia:The_Beatles mo:member dbpedia:Paul_McCartney. }
```

In your own words:
What does this query mean?

Results

true

SPARQL ASK

ASK - tests whether or not a query pattern has a solution. Returns yes/no

Query: *Is Elvis Presley a member of 'The Beatles'?*

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX mo: <http://purl.org/ontology/mo/>

ASK WHERE { dbpedia:The_Beatles mo:member dbpedia:Elvis_Presley. }
```

In your own words:
What does this query mean?

Results

false

SPARQL ASK

ASK - tests whether or not a query pattern has a solution. Returns yes/no

Query: *Is Elizabeth Taylor dead?*

```
ASK WHERE { :ElizabethTaylor :diedOn ?any . }
```

Results

false

(no triple matches this pattern)

Query: *Were any of the actors of "Giant" born after 1950?*

```
ASK WHERE { ?any :playedIn :Giant .
            ?any :bornOn ?birthday .
            FILTER (?birthday > "1950-01-01"^^xsd:date)}
```

Results

false

(everyone playing in Giant was born before 1950)

SPARQL DESCRIBE

Takes the resources within the solution and provides information about them as RDF statements. They can be identified by:

- Specifying **explicit IRIs**

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
DESCRIBE dbpedia:Paul_McCartney
```

- **Bindings of variables** in the WHERE clause

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX mo: <http://purl.org/ontology/mo/>

DESCRIBE ?member
WHERE { dbpedia:The_Beatles mo:member ?member .}
```

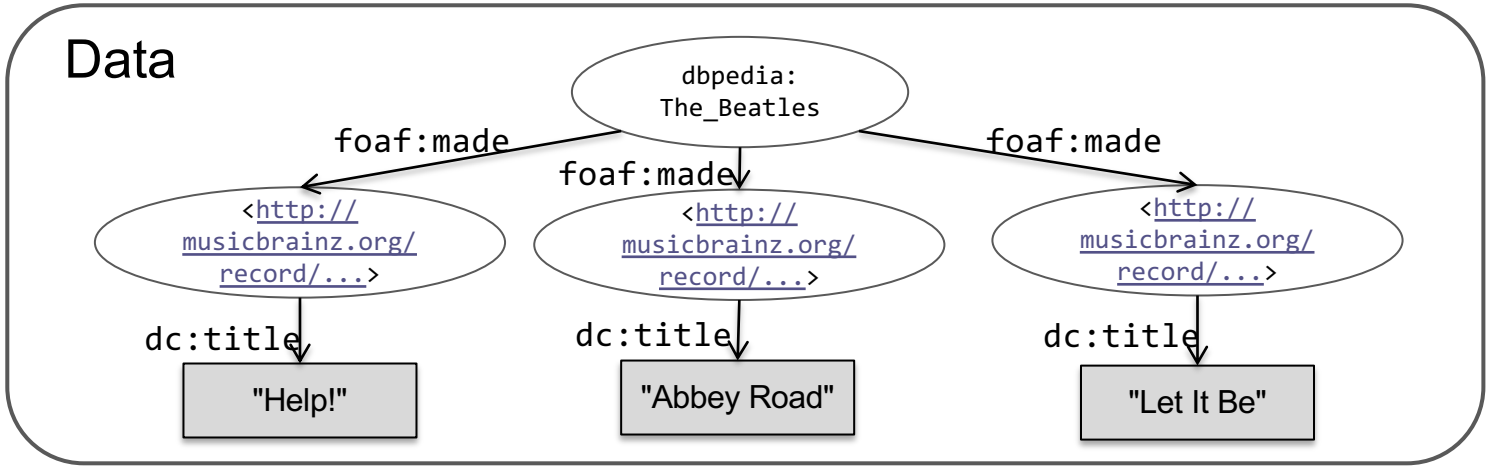
Very useful for exploration when you are confronted with a new dataset that you don't know the structure of.

SPARQL - Agenda

- Basics
  - Introductory notions
  - The SELECT query form
  - Other query forms
- Advanced notions
  - **CONSTRUCT queries**
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - Federated queries
  - Entailment regimes

SPARQL CONSTRUCT

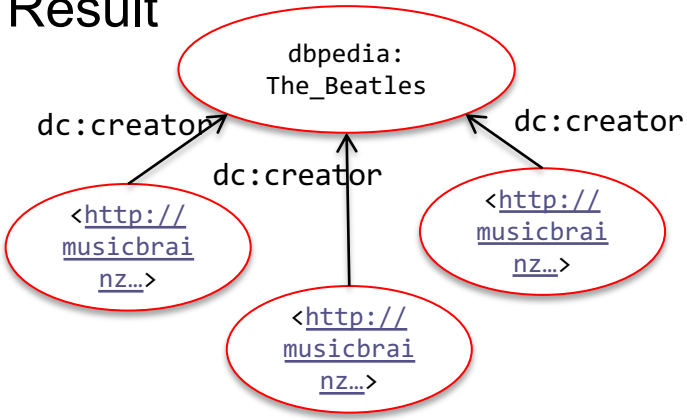
Returns RDF statements created from variable bindings



Query

```
CONSTRUCT {
  ?album dc:creator dbpedia:The_Beatles .}
WHERE {
  dbpedia:The_Beatles foaf:made ?album .}
```

Result



Adapted from EUCLID learning materials – "Querying Linked Data"

SPARQL CONSTRUCT

Query: *Create the `dc:creator` descriptions for albums and their tracks recorded by 'The Beatles'.*

```
PREFIX dbpedia, foaf, mo, dc ...
CONSTRUCT {
  ?album dc:creator dbpedia:The_Beatles .
  ?track dc:creator dbpedia:The_Beatles .}
WHERE {
  dbpedia:The_Beatles foaf:made ?album .
  ?album mo:track ?track .}
```

SPARQL CONSTRUCT

CONSTRUCT queries allow phrasing rules of the type
“IF this is found THEN generate that”

```
CONSTRUCT {?q1 :hasSibling ?q2} WHERE {?q1 :hasBrother ?q2}
CONSTRUCT {?q1 :hasSibling ?q2} WHERE {?q1 :hasSister ?q2}
CONSTRUCT {?q1 :hasParent ?q2} WHERE {?q1 :hasFather ?q2}
CONSTRUCT {?q1 :hasParent ?q2} WHERE {?q1 :hasMother ?q2}
```

```
CONSTRUCT {?q1 :hasUncle ?q2}
WHERE {?q2 :hasSibling ?parent .
       ?q2 a :Man .
       ?q1 :hasParent ?parent }
```

Semantic Web for the Working Ontologist, p90

SPARQL - Agenda

- Basics
  - Introductory notions
  - The SELECT query form
  - Other query forms
- Advanced notions
  - CONSTRUCT queries
  - **Filter & Order**
  - Aggregate & Group
  - Union
  - Subqueries
  - Federated queries
  - Entailment regimes

FILTER

- Different types of **filters and functions** may be used
- Filters use comparison and logical operators
- Positioned **within** the WHERE clause

Query: *Retrieve the albums and tracks recorded by 'The Beatles', where the duration of the song is more than 300 secs. and no longer than 400 secs.*

```
PREFIX dbpedia, foaf, dc, mo

SELECT ?album_name ?track_title ?duration
WHERE {
    dbpedia:The_Beatles foaf:made ?album .
    ?album dc:title ?album_name ;
           mo:track ?track .
    ?track dc:title ?track_title ;
          mo:duration ?duration;
    FILTER (?duration > 300000 && ?duration < 400000)
}
```

Filter expressions as regular expressions over strings

Query: *Create the `dc:creator` descriptions of the albums recorded by 'The Beatles' whose title contains the word "love".*

```
PREFIX dbpedia, foaf, dc

CONSTRUCT {?album dc:creator dbpedia:The_Beatles .}

WHERE {
    dbpedia:The_Beatles foaf:made ?album .
    ?album dc:title ?album_name ;
    FILTER (REGEX(?album_name, ".*love.*", i))
}
```

Solution Modifier: DISTINCT

- Elimination of duplicates with `DISTINCT` and `REDUCED`
- `REDUCED` does not guarantee elimination of duplicates

Query: *Retrieve the name of the albums recorded by 'The Beatles' which have at least two different songs.*

```
PREFIX dbpedia, foaf, dc, mo

SELECT MODIFIER ?album_name

WHERE {
    dbpedia:The_Beatles foaf:made ?album .
    ?album dc:title ?album_name ;
         mo:track ?track1 .
         mo:track ?track2 .
     FILTER (?track1 != ?track2)
}
```

Results

MODIFIER= **DISTINCT**

?album
"Revolver"
"Sessions"
"Abbey Road"

MODIFIER= **REDUCED**

?album
"Revolver"
"Revolver"
"Revolver"
"Sessions"
"Abbey Road"
"Abbey Road"

Solution Modifier: ORDER

It is possible to combine the query with solution modifiers (`ORDER BY`, `LIMIT`, `OFFSET`) to get **subsets of results:**

- `ORDER BY (DESC/ASC)`: result set is shown in decreasing/ascending order
- `LIMIT`: limits the number of results returned by a query

Query *Create the `dc:creator` descriptions for the 10 most recent albums and their tracks recorded by 'The Beatles'.*

```
PREFIX dbpedia, foaf, mo, dc;

CONSTRUCT {
  ?album dc:creator dbpedia:The_Beatles .
  ?track dc:creator dbpedia:The_Beatles .}
WHERE {
  dbpedia:The_Beatles foaf:made ?album .
  ?album mo:track ?track ;
         dc:date ?date .
} ORDER BY DESC(?date)
  LIMIT 10
```

SPARQL - Agenda

- Basics
  - Introductory notions
  - The SELECT query form
  - Other query forms
- Advanced notions
  - CONSTRUCT queries
  - Filter & Order
  - **Aggregate & Group**
  - Union
  - Subqueries
  - Federated queries
  - Entailment regimes

Solution Modifier: GROUP BY

Aggregates

- Calculate aggregate values: `COUNT`, `SUM`, `MIN`, `MAX`, `AVG`, `GROUP_CONCAT` and `SAMPLE`
- In combination with the `GROUP BY` operator
- Prune at group level (cf. `FILTER`) using `HAVING`

Company	Amount	Year
ACME	$1250	2010
PRIME	$3000	2009
ABC	$2500	2009
ABC	$2800	2010
PRIME	$1950	2010
ACME	$2500	2009
ACME	$3100	2010
ABC	$1500	2009
ACME	$1250	2009
PRIME	$2350	2009
PRIME	$1850	2010

Q: How high are the total sales?

```
SELECT  (SUM(?val) AS ?total)
        WHERE {
            ?s a co:Sale .
            ?s co:amount ?val .
        }
```

```
:row1 a :Sale .
:row1 :company :ACME .
:row1 :amount 1250 .
:row1 :year 2010 .
```

?total
24050

Semantic Web for the Working Ontologist, p101-102

Solution Modifier: GROUP BY

DATA

Company	Amount	Year
ACME	$1250	2010
PRIME	$3000	2009
ABC	$2500	2009
ABC	$2800	2010
PRIME	$1950	2010
ACME	$2500	2009
ACME	$3100	2010
ABC	$1500	2009
ACME	$1250	2009
PRIME	$2350	2009
PRIME	$1850	2010

```
:row1 a :Sale .
:row1 :company :ACME .
:row1 :amount 1250 .
:row1 :year 2010 .
```

Q: How high are the sales per year?

```
SELECT ?year (SUM(?val) AS ?total)
        WHERE {
            ?s a co:Sale .
            ?s co:amount ?val .
            ?s co:year ?year .
        }
        GROUP BY ?year
```

?year	?total
2009	13100
210	10950

Q: Which companies generated a yearly sales higher than 5K and in which year?

```
SELECT ?year ?company (SUM(?val) AS ?total)
          WHERE {
              ?s a co:Sale .
              ?s co:amount ?val .
              ?s co:year ?year .
              ?s co:company ?company .
          }
          GROUP BY ?year ?company
          HAVING (?total > 5000)
```

HAVING is similar to FILTER but applies to the variables of the groups and is **outside** the graph pattern

?year	?company	?total
2009	PRIME	5350

Semantic Web for the Working Ontologist, p102-103

Solution Modifier: GROUP BY

Query: *Retrieve albums recorded by 'The Beatles' that have a duration higher than a given value.*

```
PREFIX dbpedia, foaf, mo

SELECT ?album (SUM(?track_duration) AS ?album_duration)
        WHERE {
           dbpedia:The_Beatles foaf:made   ?album .
           ?album mo:track    ?track .
           ?track mo:duration ?track_duration .
        } GROUP BY ?album
           HAVING (?album_duration > 3600000)
```

SPARQL - Agenda

- Basics
  - Introductory notions
  - The SELECT query form
  - Other query forms
- Advanced notions
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - **Union**
  - Subqueries
  - Federated queries
  - Entailment regimes

SPARQL UNION

- Allows the specification of alternatives (disjunctions)
- UNION = the set union of all solution bindings of the connected patterns

Query: *Which actors played in 'Giant' or 'Rebel without a cause' or both?*

```
PREFIX mo

SELECT ?actor
WHERE {
        {?actor mo:playedIn mo:Giant .}
        UNION
        {?actor mo:playedIn mo:RebelWithoutACause .}
       }
```

actor
Ann Doran
Carroll Baker
Elizabeth Taylor
James Dean
James Dean
Jim Backus
Mercedes McCambridge
Natalie Wood
Rock Hudson
Sal Mineo
Sal Mineo

Semantic Web for the Working Ontologist, p105-106

SPARQL - Agenda

- Basics
  - Introductory notions
  - The SELECT query form
  - Other query forms
- Advanced notions
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - **Subqueries**
  - Federated queries
  - Entailment regimes

Subqueries

```
PREFIX co

SELECT ?company
WHERE
{
{ SELECT ?company (SUM(?val) AS ?total09)
          WHERE {
             ?s a co:Sale .
             ?s co:amount ?val .
             ?s co:year 2009 .
             ?s co:company ?company .}
          GROUP BY ?company } .
{ SELECT ?company (SUM(?val) AS ?total10)
          WHERE {
             ?s a co:Sale .
             ?s co:amount ?val .
             ?s co:year 2010 .
             ?s co:company ?company .}
          GROUP BY ?company } .
FILTER (?total10 >?total09) .
}
```

Aggregate values can be ”named” using variables (keyword **AS**)

Query:

Select companies for which the total sales value in 2010 is larger than the total sales value in 2009.

?company
ACME

Subqueries

CONSTRUCT and GROUP BY can be combined

```
PREFIX co

CONSTRUCT {?company a co:PreferredCustomer .
           ?company co:totalSales ?total .}
WHERE
{ SELECT ?year ?company (SUM(?val) AS ?total)
           WHERE {
              ?s a co:Sale .
              ?s co:amount ?val .
              ?s co:year ?year .
              ?s co:company ?company .
           }
           GROUP BY ?year ?company
           HAVING (?total > 5000)
}
```

Query:

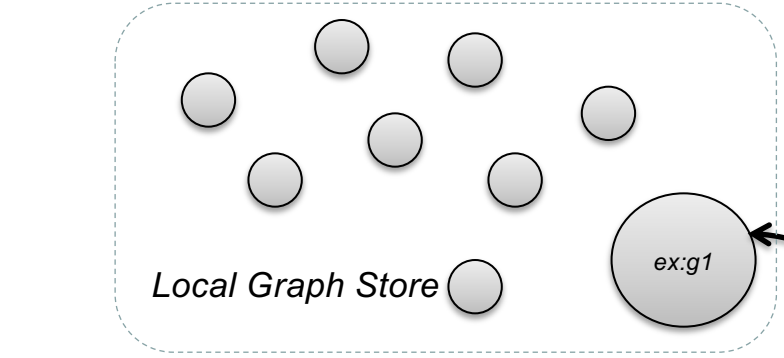
Companies for which the total sales value in any year is larger than 5000 are PreferredCustomers.

```
:PRIME a :PreferredCustomer .
:PRIME :totalSales 5350.00 .
```

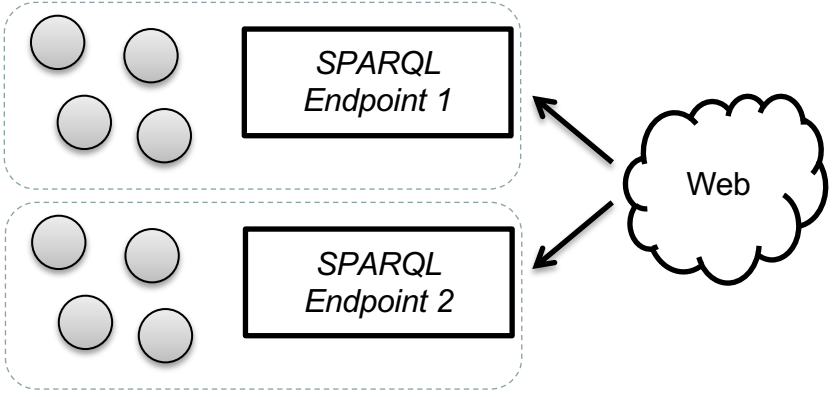
SPARQL - Agenda

- Basics
  - Introductory notions
  - The SELECT query form
  - Other query forms
- Advanced notions
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - **Federated queries**
  - Entailment regimes

Federated Query
(SPARQL 1.1)




```
PREFIX ex: <…>
SELECT …
FROM ex:g1
WHERE {
   … A …
   SERVICE <endpoint1> {
        … B …
   }
   SERVICE <endpoint2> {
        … C …
   }
}
```



https://www.w3.org/TR/sparql11-federated-query/
Slide adapted from: http://www.slideshare.net/LeeFeigenbaum/sparql-cheat-sheet

Federated Query Example

Find the birth dates of all of the actors in Star Trek: The Motion Picture

```
PREFIX movie: <http://data.linkedmdb.org/resource/movie/>
PREFIX dbpedia: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?actor_name ?birth_date
FROM <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf> # placeholder graph
WHERE {
 {
  SERVICE <http://data.linkedmdb.org/sparql> {
    <http://data.linkedmdb.org/resource/film/675> movie:actor ?actor .
    ?actor movie:actor_name ?actor_name
  }
  BIND(STRLANG(?actor_name, "en") AS ?actor_name_en)
 }
  SERVICE <http://dbpedia.org/sparql> {
    ?actor2 a foaf:Person ;
            foaf:name ?actor_name_en ;
            dbpedia:birthDate ?birth_date
 }
}
```

actor_name	birth_date
"William Shatner"	"1931-03-22" ^^<http://www.w3.org/2001/XMLSchema#date>
"Leonard Nimoy"	"1931-03-26" ^^<http://www.w3.org/2001/XMLSchema#date>
"DeForest Kelley"	"1920-01-20" ^^<http://www.w3.org/2001/XMLSchema#date>
"James Doohan"	"1920-03-03" ^^<http://www.w3.org/2001/XMLSchema#date>
"Nichelle Nichols"	"1932-12-28" ^^<http://www.w3.org/2001/XMLSchema#date>
"Stephen Collins"	"1947-10-01" ^^<http://www.w3.org/2001/XMLSchema#date>
"George Takei"	"1937-04-20" ^^<http://www.w3.org/2001/XMLSchema#date>

Example source: http://www.cambridgesemantics.com/semantic-university/sparql-by-example#%2851%29

SPARQL - Agenda

- Basics:
  - Introductory notions
  - The SELECT query form
  - Other query forms
- Advanced notions
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - Federated queries
  - **Entailment regimes**

Inference from Schema during Querying

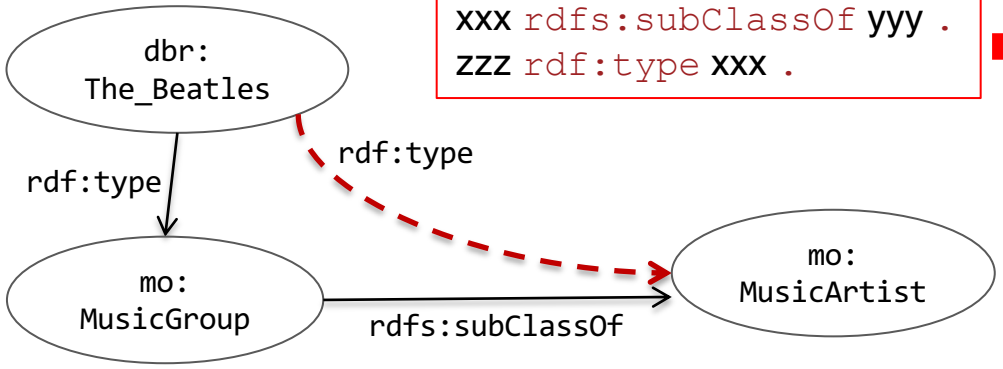
- A triple store can use the knowledge encoded in the schema to infer new facts

Schema:
```
mo:MusicGroup rdfs:subClassOf mo:MusicArtist .
```

Inferred facts:
```
mo:MusicGroup  a rdfs:Class .
mo:MusicArtist a rdfs:Class .
```

- Whether a triple store supports reasoning, influences the outcome of querying

Reasoning with – rdfs:subClassOf




```
xxx rdfs:subClassOf yyy .
zzz rdf:type xxx .
```

→

```
zzz rdf:type yyy .
```

Schema:

```
mo:MusicGroup rdfs:subClassOf mo:MusicArtist .
```

Query:

```
SELECT ?x
WHERE {?x a mo:MusicArtist.}
```

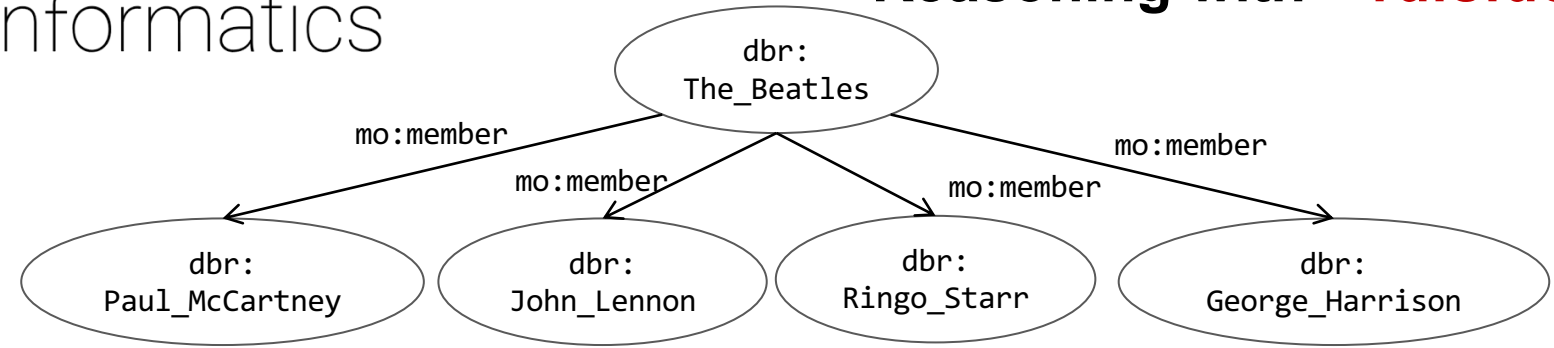
Result set with inference:

?x
dbr:The_Beatles …

Result set without inference:

?x

Reasoning with – rdfs:domain



Schema:

```
mo:member rdfs:domain mo:MusicGroup .
```

Query:

```
SELECT ?x
WHERE {?x a mo:MusicGroup.}
```

Result set with inference:

?x
dbr:The_Beatles …

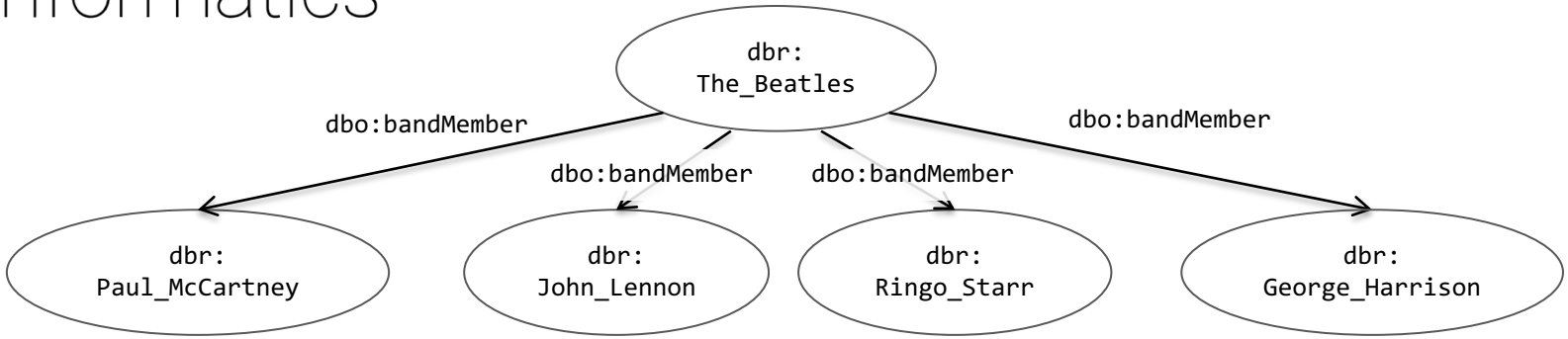
Result set without interference:

?x

aaa `rdfs:domain` xxx .
yyy aaa zzz .

→ yyy `rdf:type` xxx .

Reasoning with – rdfs:range



Schema:

```
dbo:bandMember rdfs:range foaf:Musician .
```

Query:

```
SELECT ?x
WHERE {?x a foaf:Musician .}
```

Result set with inference:

?x
dbr:Paul_McCartney
dbr:John_Lennon
dbr:Ringo_Starr
dbr:George_Harrison …

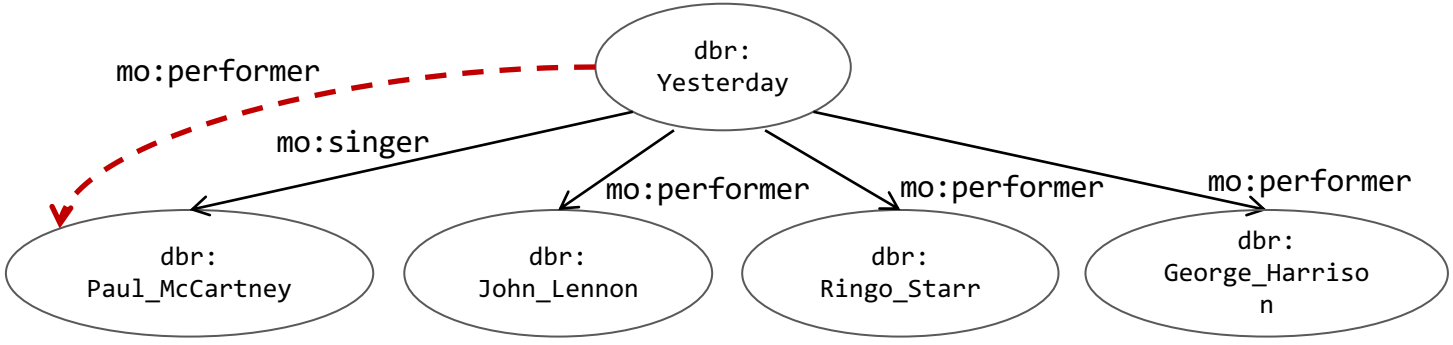
Result set without inference:

?x

aaa `rdfs:range` xxx .
yyy aaa zzz .

→ zzz `rdf:type` xxx .

Reasoning with – rdfs:subPropertyOf



Schema:

```
mo:singer rdfs:subPropertyOf mo:performer .
```

Query:

```
SELECT ?x
WHERE {dbr:Yesterday mo:performer ?x .}
```

Result set with inference:

?x
dbr:John_Lennon
dbr:Ringo_Starr
dbr:George_Harrison
dbr:Paul_McCartney

Result set without inference:

?x
dbr:John_Lennon
dbr:Ringo_Starr
dbr:George_Harrison

aaa `rdfs:subPropertyOf` bbb .
yyy aaa zzz .

→ yyy bbb zzz .

RDFS Entailment Patterns

	If S contains:	then S RDFS entails recognizing D:
rdfs1	any IRI aaa in D	aaa `rdf:type rdfs:Datatype .`
rdfs2	aaa `rdfs:domain` xxx . yyy aaa zzz .	yyy `rdf:type` xxx .
rdfs3	aaa `rdfs:range` xxx . yyy aaa zzz .	zzz `rdf:type` xxx .
rdfs4a	xxx aaa yyy .	xxx `rdf:type rdfs:Resource .`
rdfs4b	xxx aaa yyy.	yyy `rdf:type rdfs:Resource .`
rdfs5	xxx `rdfs:subPropertyOf` yyy . yyy `rdfs:subPropertyOf` zzz .	xxx `rdfs:subPropertyOf` zzz .
rdfs6	xxx `rdf:type rdf:Property .`	xxx `rdfs:subPropertyOf` xxx .
rdfs7	aaa `rdfs:subPropertyOf` bbb . xxx aaa yyy .	xxx bbb yyy .
rdfs8	xxx `rdf:type rdfs:Class .`	xxx `rdfs:subClassOf rdfs:Resource .`
rdfs9	xxx `rdfs:subClassOf` yyy . zzz `rdf:type` xxx .	zzz `rdf:type` yyy .
rdfs10	xxx `rdf:type rdfs:Class .`	xxx `rdfs:subClassOf` xxx .
rdfs11	xxx `rdfs:subClassOf` yyy . yyy `rdfs:subClassOf` zzz .	xxx `rdfs:subClassOf` zzz .
rdfs12	xxx `rdf:type rdfs:ContainerMembershipProperty .`	xxx `rdfs:subPropertyOf rdfs:member .`
rdfs13	xxx `rdf:type rdfs:Datatype .`	xxx `rdfs:subClassOf rdfs:Literal .`

S.. subgraph D.. Set of IRIs identifying datatypes

Source: https://www.w3.org/TR/2014/REC-rdf11-mt-20140225/#rdfs-interpretations

Updating RDF with SPARQL 1.1

SPARQL Query 1.1

- SPARQL 1.0 only allows data access (querying)
- SPARQL 1.1 introduces:

Query extensions

Aggregates, subqueries, negation, expressions in the `SELECT` clause, property paths, assignment, expanded set of functions and operators

Updates

`Insert, Delete, Delete/Insert`

`Create, Load, Clear, Drop`
`Copy, Move, Add`

Federation extension

Service, values, service variables (informative)

SPARQL Updates

SPARQL 1.1 provides data update operations:

- `INSERT (DATA)`: adds triples, given inline in the request, into a graph
- `DELETE` data: removes triples, given inline in the request, if the respective graphs contains those
- `DELETE/INSERT` data: uses in parallel `INSERT` and `DELETE`

SPARQL INSERT

INSERT + GP that specifies a set of RDF triples, given bindings for any variables that are included in the pattern

Insert "Peter Best" as a `formerMember` of `dbr:The_Beatles`

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

INSERT { dbr:The_Beatles dbr:formerMember ?x }
WHERE  { dbr:The_Beatles dbr:currentMember ?x .
         ?x foaf:name "Peter Best"
       }
```

SPARQL INSERT

Insert the following albums recorded by The Beatles into the graph http://musicbrainz.org/20130209-004702

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

INSERT DATA {
         GRAPH { <http://musicbrainz.org/20130302>
<http://musicbrainz.org/artist/b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d>
    foaf:made <http://musicbrainz.org/release/3a685770-7326-34fc-9f18-e5f5626f3dc5>
,
          <http://musicbrainz.org/release/cb6f8798-d51e-4fa5-a4d1-2c0602bfe1b6> .

<http://musicbrainz.org/release/3a685770-7326-34fc-9f18-e5f5626f3dc5>
         dc:title "Please Please Me".

<http://musicbrainz.org/release/cb6f8798-d51e-4fa5-a4d1-2c0602bfe1b6>
         dc:title "Something New". } }
```

SPARQL DELETE

Delete *all the information about the album "Casualities" of The Beatles.*

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

DELETE { ?album ?predicate ?object . }
WHERE {
http://musicbrainz.org/artist/b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d foaf:made
                    ?album .
?album dc:title "Casualities".
?album ?predicate ?object .}
```

SPARQL DELETE/INSERT

Delete the status of 'Peter Best' as current member of "The Beatles" and **insert** his status as former member of the band.

```
PREFIX dbr, dbo, foaf …

DELETE { dbr:The_Beatles dbo:currentMember ?x . }
INSERT { GRAPH <http://musicbrainz.org/20130209-004702> {
         dbr:The_Beatles dbo:formerBandMember ?x .}
       }
WHERE {
    dbr:The_Beatles dbo:currentMember ?x .
    ?x foaf:name "Peter Best" .}
```

SPARQL 1.1 Graph Update Operations

SPARQL 1.1 provides graph update operations:

- **CREATE** creates an empty graph in the Graph Store
- **LOAD** reads the content of a document into a graph in the Graph Store
- **CLEAR** removes all triples in one or more graphs
- **DROP** removes the graph from the Graph Store
- Other operations: **COPY**, **MOVE**, **ADD**

SPARQL CREATE

- Creates a new named graph
- Can be used with the **DEFAULT** and **ALL** keywords

```
CREATE GRAPH <http://musicbrainz.org/20130302>
```

- **LOAD**: An RDF graph can be loaded from a URL

```
LOAD <http://xmlns.com/foaf/spec/20100809.rdf>
```

```
LOAD <http://xmlns.com/foaf/spec/20100809.rdf>
        INTO <http://xmlns.com/foaf/0.1/>
```

Named graph

- **LOAD** can be used with the **SILENT** keyword (suppresses errors)

SPARQL CLEAR and DROP

```
CLEAR GRAPH <http://musicbrainz.org/20130302>
```

- **CLEAR** removes all triples in the graph (emptied, but not deleted!)
- The graph(s) can be specified with the following keywords: `DEFAULT`, `NAMED`, `ALL`, `GRAPH`
- Can be used with the `SILENT` keyword

```
DROP GRAPH <http://musicbrainz.org/20130302>
```

- **DROP** removes the given graph from the graph store, including its content
- Can be used with the `DEFAULT` and `ALL` keywords

SPARQL COPY, MOVE, ADD

- COPY … TO …

```
COPY GRAPH <http://musicbrainz.org/20130302>
TO GRAPH <http://musicbrainz.org/20130303>
```

- MOVE … TO …

```
MOVE GRAPH <http://musicbrainz.org/temp>
TO GRAPH <http://musicbrainz.org/20130303>
```

- ADD … TO …

```
ADD GRAPH <http://musicbrainz.org/20130302>
TO GRAPH <http://musicbrainz.org/20130303>
```

Summary

- SPARQL basics
  - Introductory notions
  - The SELECT query
  - Other query types
- Advanced SPARQL
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - Federated queries
  - Entailment regimes
- Updates with SPARQL

TU WIEN Informatics

188.399 Introduction to Semantic Systems

(2.0 VU / 3.0 ECTS)

2023W

Katja Hose
Emanuel Sallinger

iss@dbai.tuwien.ac.at

SPARQL - Agenda

- Basics
  - Introductory notions
  - The SELECT query form
  - Other query forms
- Advanced notions
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - **Optional**
  - Federated queries
  - Entailment regimes

OPTIONAL

```
PREFIX ex: <http://ex.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o ?c
WHERE {
  ?s foaf:name ?o.
  OPTIONAL { ?s ex:country ?c }.
}
```

```
ex:Anzhelika%20Sidorova  rdf:type foaf:Person.
ex:Sandi%20Morris        rdf:type foaf:Person.
ex:Katerina%20Stefanidi  rdf:type foaf:Person.
ex:Holly%20Bradshaw      rdf:type foaf:Person.
ex:Alysha%20Newman       rdf:type foaf:Person.
ex:Angelica%20Bengtsson  rdf:type foaf:Person.

ex:Anzhelika%20Sidorova  ex:score "4.95"^^xsd:decimal.
ex:Sandi%20Morris        ex:score "4.90"^^xsd:decimal.
ex:Katerina%20Stefanidi  ex:score "4.85"^^xsd:decimal.
ex:Holly%20Bradshaw      ex:score "4.80"^^xsd:decimal.
ex:Alysha%20Newman       ex:score "4.80"^^xsd:decimal.
ex:Angelica%20Bengtsson  ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova  foaf:name "Anzhelika Sidorova"@en.
ex:Sandi%20Morris        foaf:name "Sandi Morris"@en.
ex:Katerina%20Stefanidi  foaf:name "Katerina Stefanidi"@en.
ex:Holly%20Bradshaw      foaf:name "Holly Bradshaw"@en.
ex:Alysha%20Newman       foaf:name "Alysha Newman"@en.
ex:Angelica%20Bengtsson  foaf:name "Angelica Bengtsson"@en.

ex:Anzhelika%20Sidorova  ex:country <http://ex.com/RU>.
ex:Sandi%20Morris        ex:country <http://ex.com/US>.
ex:Katerina%20Stefanidi  ex:country <http://ex.com/EL>.
ex:Holly%20Bradshaw      ex:country <http://ex.com/UK>.
ex:Alysha%20Newman       ex:country <http://ex.com/CA>.
ex:Angelica%20Bengtsson  ex:country <http://ex.com/SE>.
```

TU WIEN Informatics

OPTIONAL

```
PREFIX ex: <http://ex.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o ?c
WHERE {
  ?s foaf:name ?o.
  OPTIONAL { ?s ex:country ?c }.
}
```

?s	?o	?c
ex:Alysha%20Newman	"Alysha Newman"@en	ex:CA
ex:Angelica%20Bengtsson	"Angelica Bengtsson"@en	ex:SE
ex:Anzhelika%20Sidorova	"Anzhelika Sidorova"@en	ex:RU
ex:Holly%20Bradshaw	"Holly Bradshaw"@en	ex:UK
ex:Katerina%20Stefanidi	"Katerina Stefanidi"@en	ex:EL
ex:Sandi%20Morris	"Sandi Morris"@en	

```
ex:Anzhelika%20Sidorova  rdf:type foaf:Person.
ex:Sandi%20Morris        rdf:type foaf:Person.
ex:Katerina%20Stefanidi  rdf:type foaf:Person.
ex:Holly%20Bradshaw      rdf:type foaf:Person.
ex:Alysha%20Newman       rdf:type foaf:Person.
ex:Angelica%20Bengtsson rdf:type foaf:Person.

ex:Anzhelika%20Sidorova  ex:score "4.95"^^xsd:decimal.
ex:Sandi%20Morris        ex:score "4.90"^^xsd:decimal.
ex:Katerina%20Stefanidi  ex:score "4.85"^^xsd:decimal.
ex:Holly%20Bradshaw      ex:score "4.80"^^xsd:decimal.
ex:Alysha%20Newman       ex:score "4.80"^^xsd:decimal.
ex:Angelica%20Bengtsson ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova  foaf:name "Anzhelika Sidorova"@en.
ex:Sandi%20Morris        foaf:name "Sandi Morris"@en.
ex:Katerina%20Stefanidi  foaf:name  "Katerina Stefanidi"@en.
ex:Holly%20Bradshaw      foaf:name "Holly Bradshaw"@en.
ex:Alysha%20Newman       foaf:name "Alysha Newman"@en.
ex:Angelica%20Bengtsson foaf:name "Angelica Bengtsson"@en.

ex:Anzhelika%20Sidorova  ex:country <http://ex.com/RU>.
ex:Sandi%20Morris        ex:country <http://ex.com/US>.
ex:Katerina%20Stefanidi  ex:country <http://ex.com/EL>.
ex:Holly%20Bradshaw      ex:country <http://ex.com/UK>.
ex:Alysha%20Newman       ex:country <http://ex.com/CA>.
ex:Angelica%20Bengtsson ex:country <http://ex.com/SE>.
```

Adapted from Anastasia Dimou (course on “Data & Knowledge on the Web“)

TU WIEN Informatics

OPTIONAL

```
PREFIX ex: <http://ex.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o ?c
WHERE {
  ?s foaf:name ?o.
        ?s ex:country ?c  .
}
```

?s	?o	?c
ex:Alysha%20Newman	"Alysha Newman"@en	ex:CA
ex:Angelica%20Bengtsson	"Angelica Bengtsson"@en	ex:SE
ex:Anzhelika%20Sidorova	"Anzhelika Sidorova"@en	ex:RU
ex:Holly%20Bradshaw	"Holly Bradshaw"@en	ex:UK
ex:Katerina%20Stefanidi	"Katerina Stefanidi"@en	ex:EL
ex:Sandi%20Morris	"Sandi Morris"@en	

```
ex:Anzhelika%20Sidorova  rdf:type foaf:Person.
ex:Sandi%20Morris        rdf:type foaf:Person.
ex:Katerina%20Stefanidi  rdf:type foaf:Person.
ex:Holly%20Bradshaw      rdf:type foaf:Person.
ex:Alysha%20Newman       rdf:type foaf:Person.
ex:Angelica%20Bengtsson rdf:type foaf:Person.

ex:Anzhelika%20Sidorova  ex:score "4.95"^^xsd:decimal.
ex:Sandi%20Morris        ex:score "4.90"^^xsd:decimal.
ex:Katerina%20Stefanidi  ex:score "4.85"^^xsd:decimal.
ex:Holly%20Bradshaw      ex:score "4.80"^^xsd:decimal.
ex:Alysha%20Newman       ex:score "4.80"^^xsd:decimal.
ex:Angelica%20Bengtsson ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova  foaf:name "Anzhelika Sidorova"@en.
ex:Sandi%20Morris        foaf:name "Sandi Morris"@en.
ex:Katerina%20Stefanidi  foaf:name  "Katerina Stefanidi"@en.
ex:Holly%20Bradshaw      foaf:name "Holly Bradshaw"@en.
ex:Alysha%20Newman       foaf:name "Alysha Newman"@en.
ex:Angelica%20Bengtsson foaf:name "Angelica Bengtsson"@en.

ex:Anzhelika%20Sidorova  ex:country <http://ex.com/RU>.
ex:Sandi%20Morris        ex:country <http://ex.com/US>.
ex:Katerina%20Stefanidi  ex:country <http://ex.com/EL>.
ex:Holly%20Bradshaw      ex:country <http://ex.com/UK>.
ex:Alysha%20Newman       ex:country <http://ex.com/CA>.
ex:Angelica%20Bengtsson ex:country <http://ex.com/SE>.
```

Adapted from Anastasia Dimou (course on “Data & Knowledge on the Web“)

TU WIEN Informatics

OPTIONAL

```
PREFIX ex: <http://ex.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o ?c
WHERE {
  ?s foaf:name ?o.
          ?s ex:country ?c  .
}
```

?s	?o	?c
ex:Alysha%20Newman	"Alysha Newman"@en	ex:CA
ex:Angelica%20Bengtsson	"Angelica Bengtsson"@en	ex:SE
ex:Anzhelika%20Sidorova	"Anzhelika Sidorova"@en	ex:RU
ex:Holly%20Bradshaw	"Holly Bradshaw"@en	ex:UK
ex:Katerina%20Stefanidi	"Katerina Stefanidi"@en	ex:EL

```
ex:Anzhelika%20Sidorova  rdf:type foaf:Person.
ex:Sandi%20Morris        rdf:type foaf:Person.
ex:Katerina%20Stefanidi  rdf:type foaf:Person.
ex:Holly%20Bradshaw      rdf:type foaf:Person.
ex:Alysha%20Newman       rdf:type foaf:Person.
ex:Angelica%20Bengtsson rdf:type foaf:Person.

ex:Anzhelika%20Sidorova  ex:score "4.95"^^xsd:decimal.
ex:Sandi%20Morris        ex:score "4.90"^^xsd:decimal.
ex:Katerina%20Stefanidi  ex:score "4.85"^^xsd:decimal.
ex:Holly%20Bradshaw      ex:score "4.80"^^xsd:decimal.
ex:Alysha%20Newman       ex:score "4.80"^^xsd:decimal.
ex:Angelica%20Bengtsson ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova  foaf:name "Anzhelika Sidorova"@en.
ex:Sandi%20Morris        foaf:name "Sandi Morris"@en.
ex:Katerina%20Stefanidi  foaf:name  "Katerina Stefanidi"@en.
ex:Holly%20Bradshaw      foaf:name "Holly Bradshaw"@en.
ex:Alysha%20Newman       foaf:name "Alysha Newman"@en.
ex:Angelica%20Bengtsson foaf:name "Angelica Bengtsson"@en.

ex:Anzhelika%20Sidorova  ex:country <http://ex.com/RU>.
ex:Sandi%20Morris        ex:country <http://ex.com/US>.
ex:Katerina%20Stefanidi  ex:country <http://ex.com/EL>.
ex:Holly%20Bradshaw      ex:country <http://ex.com/UK>.
ex:Alysha%20Newman       ex:country <http://ex.com/CA>.
ex:Angelica%20Bengtsson ex:country <http://ex.com/SE>.
```

Adapted from Anastasia Dimou (course on “Data & Knowledge on the Web“)

OPTIONAL

```
PREFIX ex: <http://ex.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o ?c ?sc
WHERE { ?s foaf:name ?o.
        OPTIONAL { ?s ex:country ?c }.
        OPTIONAL { ?s ex:score ?sc }.
}
```

```
ex:Anzhelika%20Sidorova  rdf:type foaf:Person.
ex:Sandi%20Morris        rdf:type foaf:Person.
ex:Katerina%20Stefanidi  rdf:type foaf:Person.
ex:Holly%20Bradshaw      rdf:type foaf:Person.
ex:Alysha%20Newman       rdf:type foaf:Person.
ex:Angelica%20Bengtsson  rdf:type foaf:Person.

ex:Anzhelika%20Sidorova  ex:score "4.95"^^xsd:decimal.
ex:Sandi%20Morris        ex:score "4.90"^^xsd:decimal.
ex:Katerina%20Stefanidi  ex:score "4.85"^^xsd:decimal.
ex:Holly%20Bradshaw      ex:score "4.80"^^xsd:decimal.
ex:Alysha%20Newman       ex:score "4.80"^^xsd:decimal.
ex:Angelica%20Bengtsson  ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova  foaf:name "Anzhelika Sidorova"@en.
ex:Sandi%20Morris        foaf:name "Sandi Morris"@en.
ex:Katerina%20Stefanidi  foaf:name "Katerina Stefanidi"@en.
ex:Holly%20Bradshaw      foaf:name "Holly Bradshaw"@en.
ex:Alysha%20Newman       foaf:name "Alysha Newman"@en.
ex:Angelica%20Bengtsson  foaf:name "Angelica Bengtsson"@en.

ex:Anzhelika%20Sidorova  ex:country <http://ex.com/RU>.
ex:Sandi%20Morris        ex:country <http://ex.com/US>.
ex:Katerina%20Stefanidi  ex:country <http://ex.com/EL>.
ex:Holly%20Bradshaw      ex:country <http://ex.com/UK>.
ex:Alysha%20Newman       ex:country <http://ex.com/CA>.
ex:Angelica%20Bengtsson  ex:country <http://ex.com/SE>.
```

TU WIEN Informatics

OPTIONAL

```
PREFIX ex: <http://ex.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o ?c ?sc
WHERE { ?s foaf:name ?o.
        OPTIONAL { ?s ex:country ?c }.
        OPTIONAL { ?s ex:score ?sc }.
}
```

?s	?o	?c	?sc
ex:Anzhelika%20Sidorova	"Anzhelika Sidorova"@en	ex:RU	"4.95"^^xsd:decimal
ex:Sandi%20Morris	"Sandi Morris"@en		"4.90"^^xsd:decimal
ex:Katerina%20Stefanidi	"Katerina Stefanidi"@en	ex:EL	"4.85"^^xsd:decimal
ex:Holly%20Bradshaw	"Holly Bradshaw"@en	ex:UK	"4.80"^^xsd:decimal
ex:Alysha%20Newman	"Alysha Newman"@en	ex:CA	"4.80"^^xsd:decimal
ex:Angelica%20Bengtsson	"Angelica Bengtsson"@en	ex:SE	"4.80"^^xsd:decimal

```
ex:Anzhelika%20Sidorova rdf:type foaf:Person.
ex:Sandi%20Morris rdf:type foaf:Person.
ex:Katerina%20Stefanidi rdf:type foaf:Person.
ex:Holly%20Bradshaw rdf:type foaf:Person.
ex:Alysha%20Newman rdf:type foaf:Person.
ex:Angelica%20Bengtsson rdf:type foaf:Person.

ex:Anzhelika%20Sidorova ex:score "4.95"^^xsd:decimal.
ex:Sandi%20Morris ex:score "4.90"^^xsd:decimal.
ex:Katerina%20Stefanidi ex:score "4.85"^^xsd:decimal.
ex:Holly%20Bradshaw ex:score "4.80"^^xsd:decimal.
ex:Alysha%20Newman ex:score "4.80"^^xsd:decimal.
ex:Angelica%20Bengtsson ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova foaf:name "Anzhelika Sidorova
ex:Sandi%20Morris foaf:name "Sandi Morris"@en.
ex:Katerina%20Stefanidi foaf:name "Katerina Stefanidi"
ex:Holly%20Bradshaw foaf:name "Holly Bradshaw"@
ex:Alysha%20Newman foaf:name "Alysha Newman"@
ex:Angelica%20Bengtsson foaf:name "Angelica Bengtsson

ex:Anzhelika%20Sidorova ex:country <http://ex.com/RU>
ex:Sandi%20Morris ex:country <http://ex.com/US>
ex:Katerina%20Stefanidi ex:country <http://ex.com/EL>
ex:Holly%20Bradshaw ex:country <http://ex.com/UK>
ex:Alysha%20Newman ex:country <http://ex.com/CA>
ex:Angelica%20Bengtsson ex:country <http://ex.com/SE>.
```

TU WIEN Informatics

OPTIONAL

```
PREFIX ex: <http://ex.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o ?c ?sc
WHERE { ?s foaf:name ?o.
        OPTIONAL { ?s ex:country ?c .
                   ?s ex:score ?sc }.
}
```

?s	?o	?c	?sc
ex:Anzhelika%20Sidorova	"Anzhelika Sidorova"@en	ex:RU	"4.95"^^xsd:decimal
ex:Sandi%20Morris	"Sandi Morris"@en		"4.90"^^xsd:decimal
ex:Katerina%20Stefanidi	"Katerina Stefanidi"@en	ex:EL	"4.85"^^xsd:decimal
ex:Holly%20Bradshaw	"Holly Bradshaw"@en	ex:UK	"4.80"^^xsd:decimal
ex:Alysha%20Newman	"Alysha Newman"@en	ex:CA	"4.80"^^xsd:decimal
ex:Angelica%20Bengtsson	"Angelica Bengtsson"@en	ex:SE	"4.80"^^xsd:decimal

```
ex:Anzhelika%20Sidorova rdf:type foaf:Person.
ex:Sandi%20Morris rdf:type foaf:Person.
ex:Katerina%20Stefanidi rdf:type foaf:Person.
ex:Holly%20Bradshaw rdf:type foaf:Person.
ex:Alysha%20Newman rdf:type foaf:Person.
ex:Angelica%20Bengtsson rdf:type foaf:Person.

ex:Anzhelika%20Sidorova ex:score "4.95"^^xsd:decimal.
ex:Sandi%20Morris ex:score "4.90"^^xsd:decimal.
ex:Katerina%20Stefanidi ex:score "4.85"^^xsd:decimal.
ex:Holly%20Bradshaw ex:score "4.80"^^xsd:decimal.
ex:Alysha%20Newman ex:score "4.80"^^xsd:decimal.
ex:Angelica%20Bengtsson ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova foaf:name "Anzhelika Sidorova
ex:Sandi%20Morris foaf:name "Sandi Morris"@en.
ex:Katerina%20Stefanidi foaf:name "Katerina Stefanidi"
ex:Holly%20Bradshaw foaf:name "Holly Bradshaw"@
ex:Alysha%20Newman foaf:name "Alysha Newman"@
ex:Angelica%20Bengtsson foaf:name "Angelica Bengtsson

ex:Anzhelika%20Sidorova ex:country <http://ex.com/RU>
ex:Sandi%20Morris ex:country <http://ex.com/US>
ex:Katerina%20Stefanidi ex:country <http://ex.com/EL>
ex:Holly%20Bradshaw ex:country <http://ex.com/UK>
ex:Alysha%20Newman ex:country <http://ex.com/CA>
ex:Angelica%20Bengtsson ex:country <http://ex.com/SE>.
```

Adapted from Anastasia Dimou (course on "Data & Knowledge on the Web")

TU WIEN Informatics

OPTIONAL

```
PREFIX ex: <http://ex.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?s ?o ?c ?sc
WHERE { ?s foaf:name ?o.
        OPTIONAL { ?s ex:country ?c .
                   ?s ex:score ?sc }.
}
```

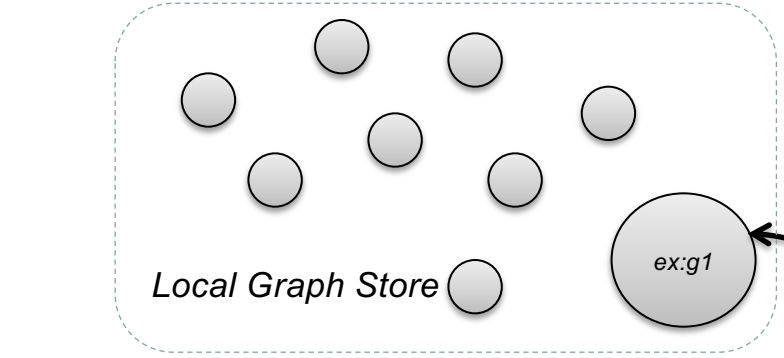
?s	?o	?c	?sc
ex:Anzhelika%20Sidorova	"Anzhelika Sidorova"@en	ex:RU	"4.95"^^xsd:decimal
ex:Sandi%20Morris	"Sandi Morris"@en		
ex:Katerina%20Stefanidi	"Katerina Stefanidi"@en	ex:EL	"4.85"^^xsd:decimal
ex:Holly%20Bradshaw	"Holly Bradshaw"@en	ex:UK	"4.80"^^xsd:decimal
ex:Alysha%20Newman	"Alysha Newman"@en	ex:CA	"4.80"^^xsd:decimal
ex:Angelica%20Bengtsson	"Angelica Bengtsson"@en	ex:SE	"4.80"^^xsd:decimal

```
ex:Anzhelika%20Sidorova  rdf:type foaf:Person.
ex:Sandi%20Morris        rdf:type foaf:Person.
ex:Katerina%20Stefanidi  rdf:type foaf:Person.
ex:Holly%20Bradshaw      rdf:type foaf:Person.
ex:Alysha%20Newman       rdf:type foaf:Person.
ex:Angelica%20Bengtsson  rdf:type foaf:Person.

ex:Anzhelika%20Sidorova  ex:score "4.95"^^xsd:decimal.
ex:Sandi%20Morris        ex:score "4.90"^^xsd:decimal.
ex:Katerina%20Stefanidi  ex:score "4.85"^^xsd:decimal.
ex:Holly%20Bradshaw      ex:score "4.80"^^xsd:decimal.
ex:Alysha%20Newman       ex:score "4.80"^^xsd:decimal.
ex:Angelica%20Bengtsson  ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova  foaf:name "Anzhelika Sidorova
ex:Sandi%20Morris        foaf:name "Sandi Morris"@en.
ex:Katerina%20Stefanidi  foaf:name  "Katerina Stefanidi"
ex:Holly%20Bradshaw      foaf:name "Holly Bradshaw"@
ex:Alysha%20Newman       foaf:name "Alysha Newman"@
ex:Angelica%20Bengtsson  foaf:name "Angelica Bengtsson

ex:Anzhelika%20Sidorova  ex:country <http://ex.com/RU>
ex:Sandi%20Morris        ex:country <http://ex.com/US>
ex:Katerina%20Stefanidi  ex:country <http://ex.com/EL>
ex:Holly%20Bradshaw      ex:country <http://ex.com/UK>
ex:Alysha%20Newman       ex:country <http://ex.com/CA>
ex:Angelica%20Bengtsson  ex:country <http://ex.com/SE>.
```

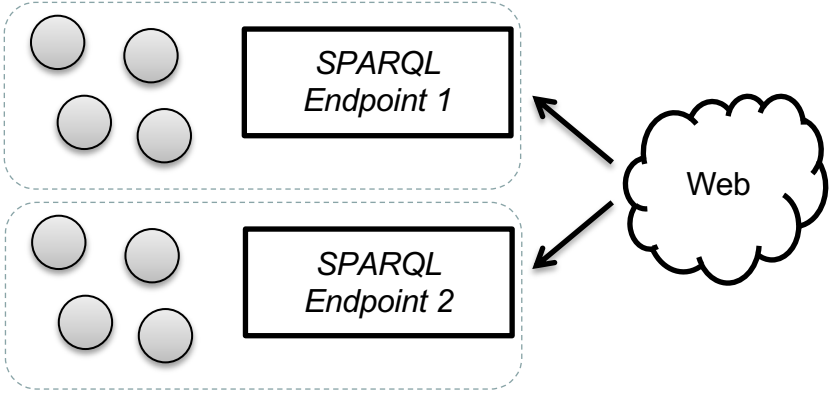
SPARQL - Agenda

- Basics
  - Introductory notions
  - The SELECT query form
  - Other query forms
- Advanced notions
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - Optional
  - **Federated queries**
  - Entailment regimes

Federated Query
(SPARQL 1.1)




```
PREFIX ex: <…>
SELECT …
FROM ex:g1
WHERE {
   … A …
   SERVICE <endpoint1> {
        … B …
   }
   SERVICE <endpoint2> {
        … C …
   }
}
```



https://www.w3.org/TR/sparql11-federated-query/
Slide adapted from: http://www.slideshare.net/LeeFeigenbaum/sparql-cheat-sheet

Federated Query Example

Find the birth dates of all of the actors in Star Trek: The Motion Picture

```
PREFIX movie: <http://data.linkedmdb.org/resource/movie/>
PREFIX dbpedia: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?actor_name ?birth_date
FROM <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf> # placeholder graph
WHERE {
 {
  SERVICE <http://data.linkedmdb.org/sparql> {
    <http://data.linkedmdb.org/resource/film/675> movie:actor ?actor .
    ?actor movie:actor_name ?actor_name
  }
  BIND(STRLANG(?actor_name, "en") AS ?actor_name_en)
 }
  SERVICE <http://dbpedia.org/sparql> {
    ?actor2 a foaf:Person ;
            foaf:name ?actor_name_en ;
            dbpedia:birthDate ?birth_date
 }
}
```

?actor_name	? birth_date
"William Shatner"	"1931-03-22" ^^<http://www.w3.org/2001/XMLSchema#date>
"Leonard Nimoy"	"1931-03-26" ^^<http://www.w3.org/2001/XMLSchema#date>
"DeForest Kelley"	"1920-01-20" ^^<http://www.w3.org/2001/XMLSchema#date>
"James Doohan"	"1920-03-03" ^^<http://www.w3.org/2001/XMLSchema#date>
"Nichelle Nichols"	"1932-12-28" ^^<http://www.w3.org/2001/XMLSchema#date>
"Stephen Collins"	"1947-10-01" ^^<http://www.w3.org/2001/XMLSchema#date>
"George Takei"	"1937-04-20" ^^<http://www.w3.org/2001/XMLSchema#date>

Example source: http://www.cambridgesemantics.com/semantic-university/sparql-by-example#%2851%29

SPARQL - Agenda

- Basics:
  - Introductory notions
  - The SELECT query form
  - Other query forms
- Advanced notions
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - Optional
  - Federated queries
  - **Entailment regimes**

Inference from Schema during Querying

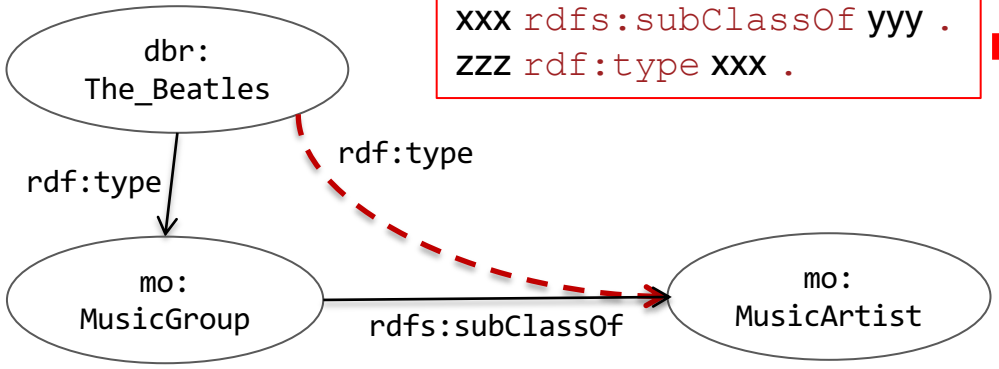
- A triple store can use the knowledge encoded in the schema to infer new facts

Schema:
```
mo:MusicGroup rdfs:subClassOf mo:MusicArtist .
```

Inferred facts:
```
mo:MusicGroup  a rdfs:Class .
mo:MusicArtist a rdfs:Class .
```

- Whether a triple store supports reasoning, influences the outcome of querying

Reasoning with – rdfs:subClassOf




```
xxx rdfs:subClassOf yyy .
zzz rdf:type xxx .
```

→

```
zzz rdf:type yyy .
```

Schema:

```
mo:MusicGroup rdfs:subClassOf mo:MusicArtist .
```

Query:

```
SELECT ?x
WHERE {?x a mo:MusicArtist.}
```

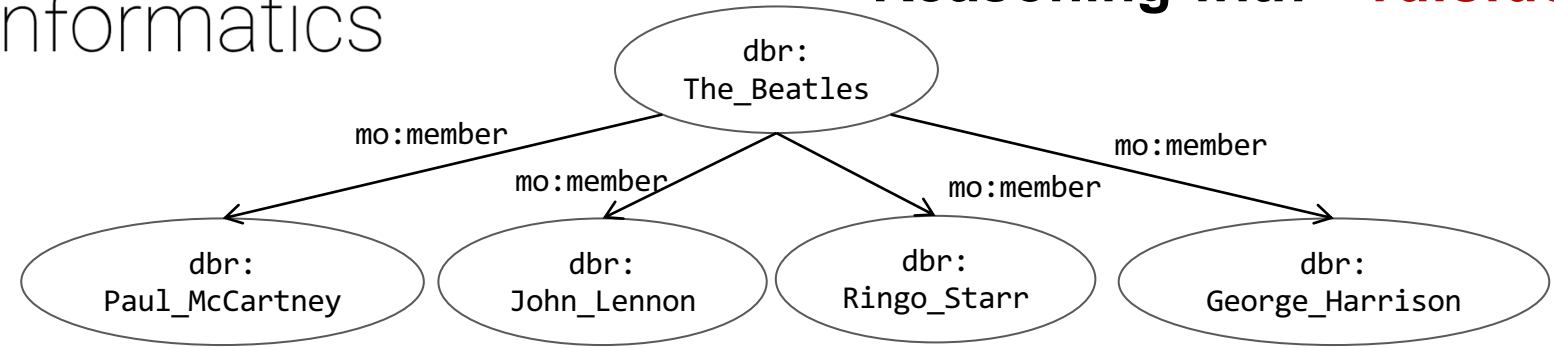
Result set with inference:

?x
dbr:The_Beatles …

Result set without inference:

?x

Reasoning with – rdfs:domain



Schema:

```
mo:member rdfs:domain mo:MusicGroup .
```

Query:

```
SELECT ?x
WHERE {?x a mo:MusicGroup.}
```

Result set with inference:

?x
dbr:The_Beatles …

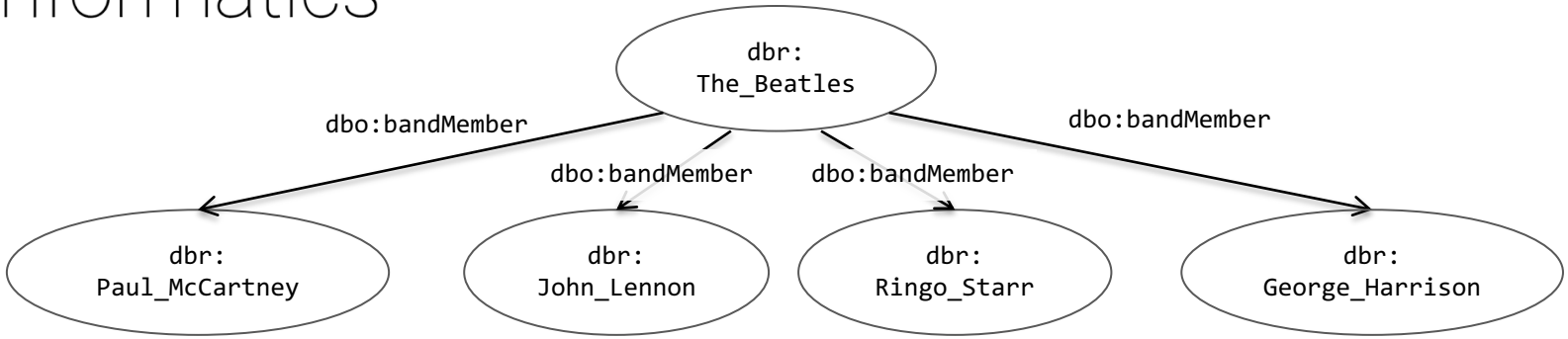
Result set without interference:

?x

aaa `rdfs:domain` xxx .
yyy aaa zzz .

→ yyy `rdf:type` xxx .

Reasoning with – rdfs:range



Schema:

```
dbo:bandMember rdfs:range foaf:Musician .
```

Query:

```
SELECT ?x
WHERE {?x a foaf:Musician .}
```

Result set with inference:

?x
dbr:Paul_McCartney
dbr:John_Lennon
dbr:Ringo_Starr
dbr:George_Harrison …

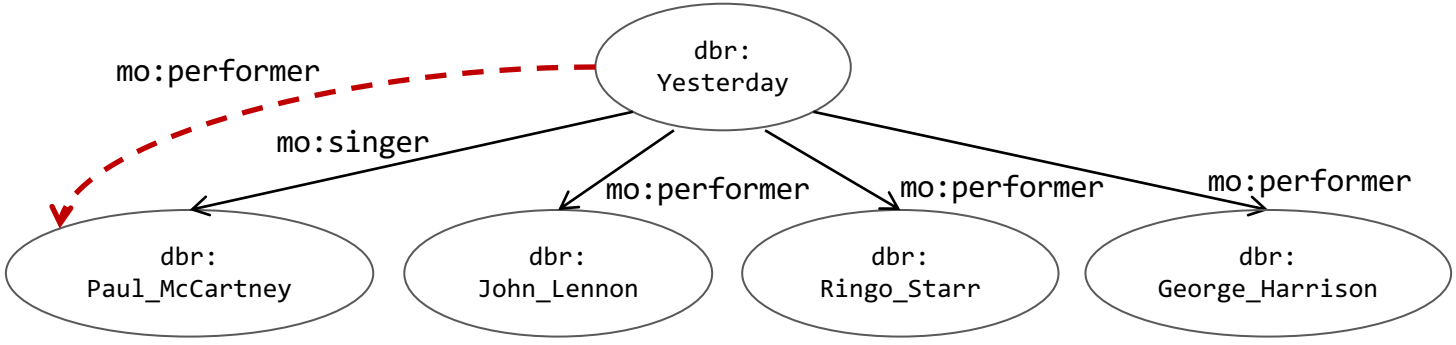
Result set without inference:

?x

aaa `rdfs:range` xxx .
yyy aaa zzz .

→ zzz `rdf:type` xxx .

Reasoning with – rdfs:subPropertyOf



Schema:

```
mo:singer rdfs:subPropertyOf mo:performer .
```

Query:

```
SELECT ?x
WHERE {dbr:Yesterday mo:performer ?x .}
```

Result set with inference:

?x
dbr:John_Lennon
dbr:Ringo_Starr
dbr:George_Harrison
dbr:Paul_McCartney

Result set without inference:

?x
dbr:John_Lennon
dbr:Ringo_Starr
dbr:George_Harrison

aaa `rdfs:subPropertyOf` bbb .
yyy aaa zzz .

→ yyy bbb zzz .

RDFS Entailment Patterns

	If S contains:	then S RDFS entails recognizing D:
rdfs1	any IRI aaa in D	aaa `rdf:type rdfs:Datatype .`
rdfs2	aaa `rdfs:domain` xxx . yyy aaa zzz .	yyy `rdf:type` xxx .
rdfs3	aaa `rdfs:range` xxx . yyy aaa zzz .	zzz `rdf:type` xxx .
rdfs4a	xxx aaa yyy .	xxx `rdf:type rdfs:Resource .`
rdfs4b	xxx aaa yyy.	yyy `rdf:type rdfs:Resource .`
rdfs5	xxx `rdfs:subPropertyOf` yyy . yyy `rdfs:subPropertyOf` zzz .	xxx `rdfs:subPropertyOf` zzz .
rdfs6	xxx `rdf:type rdf:Property .`	xxx `rdfs:subPropertyOf` xxx .
rdfs7	aaa `rdfs:subPropertyOf` bbb . xxx aaa yyy .	xxx bbb yyy .
rdfs8	xxx `rdf:type rdfs:Class .`	xxx `rdfs:subClassOf rdfs:Resource .`
rdfs9	xxx `rdfs:subClassOf` yyy . zzz `rdf:type` xxx .	zzz `rdf:type` yyy .
rdfs10	xxx `rdf:type rdfs:Class .`	xxx `rdfs:subClassOf` xxx .
rdfs11	xxx `rdfs:subClassOf` yyy . yyy `rdfs:subClassOf` zzz .	xxx `rdfs:subClassOf` zzz .
rdfs12	xxx `rdf:type rdfs:ContainerMembershipProperty .`	xxx `rdfs:subPropertyOf rdfs:member .`
rdfs13	xxx `rdf:type rdfs:Datatype .`	xxx `rdfs:subClassOf rdfs:Literal .`

S.. subgraph D.. Set of IRIs identifying datatypes

Source: https://www.w3.org/TR/2014/REC-rdf11-mt-20140225/#rdfs-interpretations

Updating RDF with SPARQL 1.1

SPARQL Query 1.1

- SPARQL 1.0 only allows data access (querying)
- SPARQL 1.1 introduces:

Query extensions

Aggregates, subqueries, negation, expressions in the `SELECT` clause, property paths, assignment, expanded set of functions and operators

Updates

`Insert, Delete, Delete/Insert`

`Create, Load, Clear, Drop`
`Copy, Move, Add`

Federation extension

Service, values, service variables (informative)

SPARQL Updates

SPARQL 1.1 provides data update operations

- `INSERT (DATA)`: adds triples, given inline in the request, into a graph
- `DELETE (DATA)`: removes triples, given inline in the request, if the respective graphs contains those
- `DELETE/INSERT (DATA)`: uses in parallel `INSERT` and `DELETE`

SPARQL INSERT

INSERT + graph pattern specifying a set of RDF triples, given bindings for any variables that are included in the pattern

Insert "Peter Best" as a `formerMember` of `dbr:The_Beatles`

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

INSERT { dbr:The_Beatles dbr:formerMember ?x }
WHERE  { dbr:The_Beatles dbr:currentMember ?x .
         ?x foaf:name "Peter Best"
       }
```

SPARQL INSERT

Insert the following albums recorded by The Beatles into the graph http://musicbrainz.org/20130209-004702

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

INSERT DATA {
        GRAPH { <http://musicbrainz.org/20130302>
<http://musicbrainz.org/artist/b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d>
   foaf:made <http://musicbrainz.org/release/3a685770-7326-34fc-9f18-e5f5626f3dc5> ,
         <http://musicbrainz.org/release/cb6f8798-d51e-4fa5-a4d1-2c0602bfe1b6> .

<http://musicbrainz.org/release/3a685770-7326-34fc-9f18-e5f5626f3dc5>
        dc:title "Please Please Me".

<http://musicbrainz.org/release/cb6f8798-d51e-4fa5-a4d1-2c0602bfe1b6>
        dc:title "Something New". } }
```

SPARQL DELETE

Delete all the information about the album "Casualities" of The Beatles.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

DELETE { ?album ?predicate ?object . }
WHERE {
http://musicbrainz.org/artist/b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d foaf:made
                    ?album .
?album dc:title "Casualities".
?album ?predicate ?object .}
```

SPARQL DELETE/INSERT

Delete the status of 'Peter Best' as current member of "The Beatles" and **insert** his status as former member of the band.

```
PREFIX dbr, dbo, foaf …

DELETE { dbr:The_Beatles dbo:currentMember ?x . }
INSERT { GRAPH <http://musicbrainz.org/20130209-004702> {
         dbr:The_Beatles dbo:formerBandMember ?x .}
       }
WHERE {
    dbr:The_Beatles dbo:currentMember ?x .
    ?x foaf:name "Peter Best" .}
```

SPARQL 1.1 Graph Update Operations

SPARQL 1.1 provides graph update operations

- **CREATE** creates an empty graph in the Graph Store
- **LOAD** reads the content of a document into a graph in the Graph Store
- **CLEAR** removes all triples in one or more graphs
- **DROP** removes the graph from the Graph Store
- Other operations: **COPY**, **MOVE**, **ADD**

SPARQL CREATE

- Creates a new named graph
- Can be used with the **DEFAULT** and **ALL** keywords

```
CREATE GRAPH <http://musicbrainz.org/20130302>
```

- **LOAD**: An RDF graph can be loaded from a URL

```
LOAD <http://xmlns.com/foaf/spec/20100809.rdf>
```

```
LOAD <http://xmlns.com/foaf/spec/20100809.rdf>
        INTO <http://xmlns.com/foaf/0.1/>
```

Named graph

- **LOAD** can be used with the **SILENT** keyword (suppresses errors)

SPARQL CLEAR and DROP

```
CLEAR GRAPH <http://musicbrainz.org/20130302>
```

- **CLEAR** removes all triples from the graph (emptied, but not deleted!)
- The graph(s) can be specified with the following keywords `DEFAULT`, `NAMED`, `ALL`, `GRAPH`
- Can be used with the `SILENT` keyword

```
DROP GRAPH <http://musicbrainz.org/20130302>
```

- **DROP** removes the given graph from the graph store, including its content
- Can be used with the `DEFAULT` and `ALL` keywords

SPARQL COPY, MOVE, ADD

- COPY … TO …

```
COPY GRAPH <http://musicbrainz.org/20130302>
TO GRAPH <http://musicbrainz.org/20130303>
```

- MOVE … TO …

```
MOVE GRAPH <http://musicbrainz.org/temp>
TO GRAPH <http://musicbrainz.org/20130303>
```

- ADD … TO …

```
ADD GRAPH <http://musicbrainz.org/20130302>
TO GRAPH <http://musicbrainz.org/20130303>
```

Summary

- SPARQL basics
  - Introductory notions
  - The SELECT query
  - Other query types
- Advanced SPARQL
  - CONSTRUCT queries
  - Filter & Order
  - Aggregate & Group
  - Union
  - Subqueries
  - Optional
  - Federated queries
  - Entailment regimes
- Updates with SPARQL

TU WIEN Informatics

188.399 Introduction to Semantic Systems

(2.0 VU / 3.0 ECTS)

2023W

Katja Hose
Emanuel Sallinger

iss@dbai.tuwien.ac.at

Learning Goals

- SHACL – Shapes Constraint Language

RDF Quality and SHACL

SHACL Shapes Constraint Language

Language for validating RDF graphs against a set of **conditions**

W3C recommendation since July 2017: https://www.w3.org/TR/shacl/

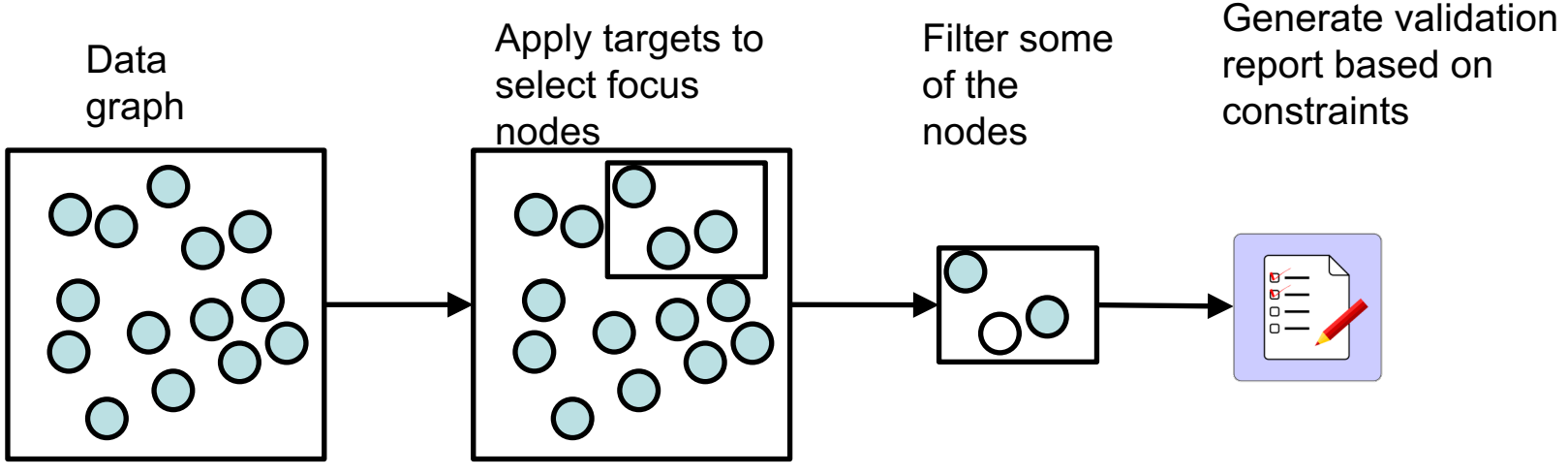
data graph: RDF graph to be validated against a shapes graph

shapes graph: RDF graphs with **conditions** provided as shapes and other constructs descriptions of the data graphs that do satisfy these conditions

SHACL Shapes Constraint Language

Language for validating RDF graphs against a set of conditions

W3C recommendation since July 2017



Example

Shape graph

```
ex:PersonShape a sh:NodeShape ;
sh:targetClass ex:Person ;
sh:property [
  a sh:PropertyShape;
  sh:path ex:ssn;
  sh:minCount 1 ] .
```

sh:minCount specifies the minimum number of value nodes that satisfy the condition.

Data graph

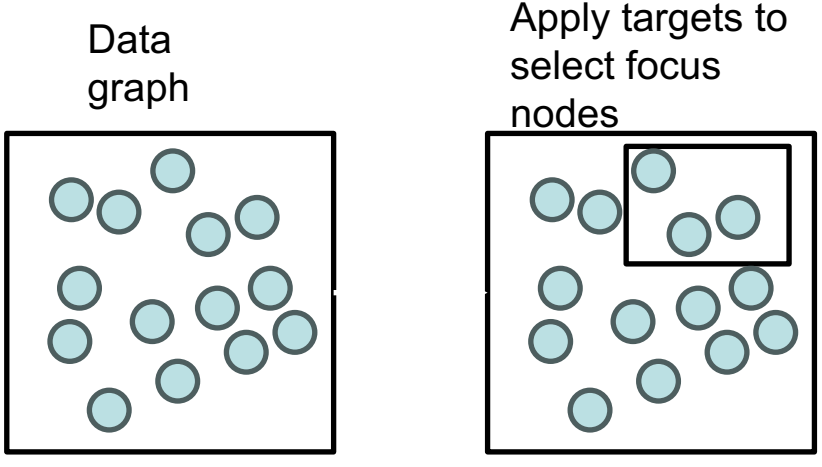
```
ex:Alice rdf:type ex:Person .
ex:Alice ex:ssn "987-65-432A" .

ex:Bob rdf:type ex:Person .
ex:Bob ex:ssn "123-45-6789" .
ex:Bob ex:ssn "124-35-6789" .

ex:Calvin rdf:type ex:Person .
ex:Calvin ex:birthDate "1971-07-07"^^xsd:date .
ex:Calvin ex:worksFor ex:UntypedCompany .
```

Targets in SHACL

- **Targets** specify what RDF graph nodes are validated against a shape
- During validation, we refer to the targets as **focus nodes**

Targets in SHACL Core

Value	Description
targetNode	directly specified nodes
targetClass	All instances of a class
targetSubjectOf	All subjects of a specific property
targetObjectOf	All objects of a specific property



Targets and focus nodes - examples

Target definitions in a shape graph

ex:PersonShape sh:targetNode ex:Alice .

ex:PersonShape sh:targetClass ex:Person .

ex:PersonShape sh:targetSubjectsOf ex:ssn .

ex:PersonShape sh:targetObjectsOf ex:worksFor .

Data Graph

ex:Alice rdf:type ex:Person .
ex:Alice ex:ssn "987-65-432A".

ex:Bob rdf:type ex:Person .
ex:Bob ex:ssn "123-45-6789” .
ex:Bob ex:ssn "124-35-6789” .

ex:Calvin rdf:type ex:Person .
ex:Calvin ex:birthDate "1971-07-07"^^xsd:date .
ex:Calvin ex:worksFor ex:UntypedCompany .

Targets and focus nodes - examples

Shape Graph

ex:PersonShape rdf:type sh:NodeShape .

ex:PersonShape sh:targetNode **ex:Alice** .

ex:PersonShape sh:targetClass ex:Person .

ex:PersonShape sh:targetSubjectsOf ex:ssn .

ex:PersonShape sh:targetObjectsOf ex:worksFor .

Data Graph

ex:Alice rdf:type ex:Person .

ex:Alice ex:ssn "987-65-432A".

ex:Bob rdf:type ex:Person .

ex:Bob ex:ssn "123-45-6789" .

ex:Bob ex:ssn "124-35-6789" .

ex:Calvin rdf:type ex:Person .

ex:Calvin ex:birthDate "1971-07-07"^^xsd:date .

ex:Calvin ex:worksFor ex:UntypedCompany .

Targets and focus nodes - examples

Shape Graph

ex:PersonShape rdf:type sh:NodeShape .
ex:PersonShape sh:targetNode **ex:Alice** .

ex:PersonShape rdf:type sh:NodeShape .
ex:PersonShape sh:targetClass ex:Person .

ex:PersonShape sh:targetSubjectsOf ex:ssn .

ex:PersonShape sh:targetObjectsOf ex:worksFor .

Data Graph

ex:Alice rdf:type ex:Person .
ex:Alice ex:ssn "987-65-432A".

ex:Bob rdf:type ex:Person .
ex:Bob ex:ssn "123-45-6789" .
ex:Bob ex:ssn "124-35-6789" .

ex:Calvin rdf:type ex:Person .
ex:Calvin ex:birthDate "1971-07-07"^^xsd:date .
ex:Calvin ex:worksFor ex:UntypedCompany .

Targets and focus nodes - examples

Shape Graph

ex:PersonShape rdf:type sh:NodeShape .
ex:PersonShape sh:targetNode **ex:Alice** .

ex:PersonShape rdf:type sh:NodeShape .
ex:PersonShape sh:targetClass **ex:Person** .

ex:PersonShape rdf:type sh:NodeShape .
ex:PersonShape sh:targetSubjectsOf ex:ssn .

ex:PersonShape sh:targetObjectsOf ex:worksFor .

Data Graph

ex:Alice rdf:type ex:Person .
ex:Alice ex:ssn "987-65-432A".

ex:Bob rdf:type ex:Person .
ex:Bob ex:ssn "123-45-6789" .
ex:Bob ex:ssn "124-35-6789" .

ex:Calvin rdf:type ex:Person .
ex:Calvin ex:birthDate "1971-07-07"^^xsd:date .
ex:Calvin ex:worksFor ex:UntypedCompany .

Targets and focus nodes - examples

Shape Graph

```
ex:PersonShape rdf:type sh:NodeShape .
ex:PersonShape sh:targetNode ex:Alice .

ex:PersonShape rdf:type sh:NodeShape .
ex:PersonShape sh:targetClass ex:Person .

ex:PersonShape rdf:type sh:NodeShape .
ex:PersonShape sh:targetSubjectsOf ex:ssn .

ex:PersonShape rdf:type sh:NodeShape .
ex:PersonShape sh:targetObjectsOf ex:worksFor .
```

Data Graph

```
ex:Alice rdf:type ex:Person .
ex:Alice ex:ssn "987-65-432A".

ex:Bob rdf:type ex:Person .
ex:Bob ex:ssn "123-45-6789" .
ex:Bob ex:ssn "124-35-6789" .

ex:Calvin rdf:type ex:Person .
ex:Calvin ex:birthDate "1971-07-07"^^xsd:date .
ex:Calvin ex:worksFor ex:UntypedCompany .
```

Targets and focus nodes - examples

Shape Graph

```
ex:PersonShape rdf:type sh:NodeShape .
ex:PersonShape sh:targetNode ex:Alice .

ex:PersonShape rdf:type sh:NodeShape .
ex:PersonShape sh:targetClass ex:Person .

ex:PersonShape rdf:type sh:NodeShape .
ex:PersonShape sh:targetSubjectsOf ex:ssn .

ex:PersonShape rdf:type sh:NodeShape .
ex:PersonShape sh:targetObjectsOf ex:worksFor .
```

Data Graph

```
ex:Alice rdf:type ex:Person .
ex:Alice ex:ssn "987-65-432A".

ex:Bob rdf:type ex:Person .
ex:Bob ex:ssn "123-45-6789" .
ex:Bob ex:ssn "124-35-6789" .

ex:Calvin rdf:type ex:Person .
ex:Calvin ex:birthDate "1971-07-07"^^xsd:date .
ex:Calvin ex:worksFor ex:UntypedCompany .
```

SHACL shapes

Node Shapes specify constraints that need to be met by the focus nodes

ex:PersonShape rdf:type sh:NodeShape

Property Shapes specify constraints on the values of particular properties/paths of the focus nodes

ex:PersonPropertyShape a sh:PropertyShape

Property shapes

Shape Graph

```
ex:PersonShape rdf:type sh:NodeShape .
ex:PersonShape sh:targetClass ex:Person .
ex:PersonShape sh:property [
    rdf:type sh:PropertyShape ;
    sh:path ex:ssn ] .
```

What does this shape check?

Data Graph

```
ex:Alice rdf:type ex:Person .
ex:Alice ex:ssn "987-65-432A".

ex:Bob rdf:type ex:Person .
ex:Bob ex:ssn "123-45-6789” .
ex:Bob ex:ssn "124-35-6789” .

ex:Calvin rdf:type ex:Person .
ex:Calvin ex:birthDate "1971-07-07"^^xsd:date .
ex:Calvin ex:worksFor ex:UntypedCompany .
```

Property shapes

Shape Graph

```
ex:PersonShape rdf:type sh:NodeShape .
ex:PersonShape sh:targetClass ex:Person .
ex:PersonShape sh:property [
    rdf:type sh:PropertyShape ;
    sh:path ex:ssn ] .
```

How can we extend this example to check that ex:Person can have at most 1 value for the property ex:ssn?

Data Graph

```
ex:Alice rdf:type ex:Person .
ex:Alice ex:ssn "987-65-432A".

ex:Bob rdf:type ex:Person .
ex:Bob ex:ssn "123-45-6789” .
ex:Bob ex:ssn "124-35-6789” .

ex:Calvin rdf:type ex:Person .
ex:Calvin ex:birthDate "1971-07-07"^^xsd:date .
ex:Calvin ex:worksFor ex:UntypedCompany .
```

Property shapes

Shape Graph

```
ex:PersonShape rdf:type sh:NodeShape .
ex:PersonShape sh:targetClass ex:Person .
ex:PersonShape sh:property [
    rdf:type sh:PropertyShape ;
    sh:path ex:ssn ;
    sh:maxCount 1 ].
```

Data Graph

```
ex:Alice rdf:type ex:Person .
ex:Alice ex:ssn "987-65-432A".

ex:Bob rdf:type ex:Person .
ex:Bob ex:ssn "123-45-6789” .
ex:Bob ex:ssn "124-35-6789” .

ex:Calvin rdf:type ex:Person .
ex:Calvin ex:birthDate "1971-07-07"^^xsd:date .
ex:Calvin ex:worksFor ex:UntypedCompany .
```

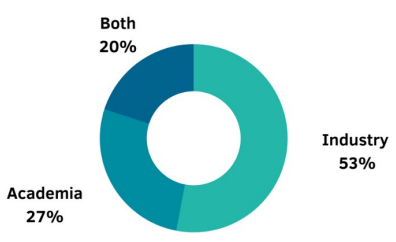
SHACL Core constraints

Type	Constraint
Cardinality	minCount, maxCount
Values	node, in, hasValue
Value range	minInclusive, maxInclusive, minExclusive, maxExclusive
String-based	minLength, maxLength, pattern, languageIn, uniqueLang
Logical constraints	not, and, or, xone
Closed shapes	closed, ignoredProperties
Property pair constraints	equals, disjoint, lessThan, lessThanOrEquals
Non-validating constraints	name, description, order, group, defaultValue
…	…

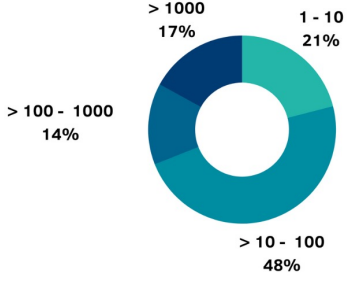
SHACL in the wild

Online survey

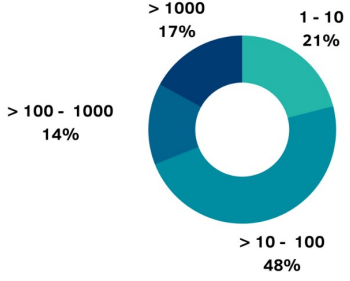
- Common practice to create shapes manually
- Automatic extraction often has scalability issues
- Automatic extraction results in many spurious shapes



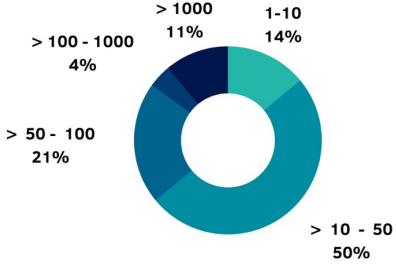
(a) Area of occupation



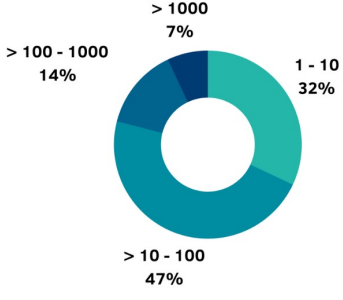
(b) Size of graph (Number of triples)



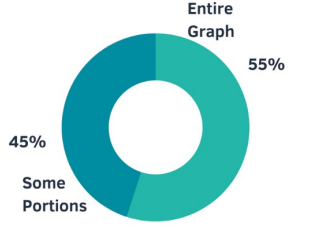
(c) Size of ontology (Number of classes)



(d) Distinct # of properties in the graph



(e) Number of generated shapes



(f) Scope of the generated shapes

K. Rabbani, M. Lissandrini, K. Hose: SHACL and ShEx in the Wild: A Community Survey on Validating Shapes Generation and Adoption. TheWebConf-2022

Summary

- SHACL constraints can help improve the quality of knowledge graphs
- Shapes can be used to define a broad range of constraints
- Validation reports can be used to increase the quality of an RDF graph

TU WIEN Informatics

Knowledge Graph Creation

Emanuel Sallinger
Katja Hose

Option 1: Template-based on-the-fly generation

Typical web application architecture:

- Data resides in back-end databases
- Front-end translates entries via templates into HTML, JSON,.. representations

Use, e.g., JSON-LD (cf. RDF notations) to impose semantics on JSON documents by providing a context that identifies properties and structural elements

→ can be interpreted as plain JSON document or as RDF graphs and triples

```
{
  "@context": "http://schema.org/",
  "@id": "http://dbpedia.org/resource/Tim_Berners-Lee",
  "givenName": "Tim",
  "knows": [{
     "@id": "http://dbpedia.org/resource/Ted_Nelson",
     "givenName": "Ted"
  }]
}
```

=

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

dbr:Ted_Nelson      schema:givenName   "Ted".
dbr:Tim_Berners-Lee schema:givenName   "Tim";
                    schema:knows   dbr:Ted_Nelson.
```

But: not always sufficient..

- Remodeling data to RDF on the fly is tricky
- Does not create links between resources
- Link to external concepts?
- Reuse of external identifiers?

JSON-LD Specification: http://json-ld.org/spec/latest/json-ld/
Example: https://en.wikipedia.org/wiki/JSON-LD
Template-based generation: http://rubenverborgh.github.io/WebFundamentals/linked-data-publishing/

RDF Notations: JSON-LD

- **J**ava**S**cript **O**bject **N**otation for **L**inked **D**ata
- W3C recommendation
- Imposes structure on JSON with @context annotations (often external vocabularies)

```
{"@context": {
    "name": "http://xmlns.com/foaf/0.1/name",
    "homepage": {
      "@id": "http://xmlns.com/foaf/0.1/workplaceHomepage",
      "@type": "@id"
    },
    "Person": "http://xmlns.com/foaf/0.1/Person"
  },
  "@id": "http://me.example.com",
  "@type": "Person",
  "name": "John Smith",
  "homepage": "http://www.example.com/"
}
```

Specification: http://json-ld.org/spec/latest/json-ld/
Example from https://en.wikipedia.org/wiki/JSON-LD

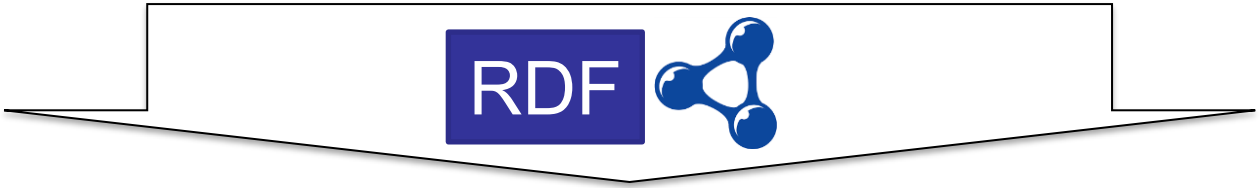
Option 2: Mapping and Transformation Tools

Tabular data

Databases

Structured Data

Text



Adapted from EUCLID learning materials – "ProvidingLinked Data"

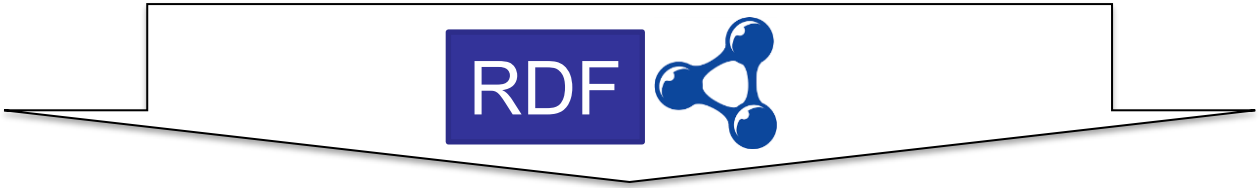
TU WIEN Informatics

Knowledge Graph Creation
Tabular Data to RDF: OpenRefine

Emanuel Sallinger
Katja Hose

Mapping and Transformation Tools

Tabular data



Adapted from EUCLID learning materials – "ProvidingLinked Data"

Example Tabular Data


```
The Beatles, 250 million
Elvis Presley, 203.3 million
Michael Jackson, 157.4 million
Madonna, 160.1 million
Led Zeppelin, 135.5 million
Queen, 90.5 million
```


```
{
  "artist": {
    "class": "artist",
    "name": "The Beatles"
  },
  "rank": 1,
  "value": 250 million
},
…
```

HTML tables

Artist	Country of origin	Period active	Release-year of first charted record	Total certified units (from available markets)[Notes]
The Beatles	United Kingdom	1960–1970[4]	1962[4]	**Total available certified units: 250 million**[show]
Elvis Presley	United States	1954–1977[28]	1954[28]	**Total available certified units: 203.3 million**[show]
Michael Jackson[Note 2]	United States	1964–2009[32]	1971[32]	**Total available certified units: 157.4 million**[show]
Madonna	United States	1979–present[44]	1982[44]	**Total available certified units: 160.1 million**[show]
Led Zeppelin	United Kingdom	1968–1980[50]	1969[50]	**Total available certified units: 135.5 million**[show]
Queen	United Kingdom	1971–present[53]	1973[53]	**Total available certified units: 90.5 million**[show]

http://en.wikipedia.org/wiki/List_of_best-selling_music_artists

Extraction tools: OpenRefine

- Tool for **messy data**
- May look like a spreadsheet program, but
  - NOT for entering new data one cell at a time
  - NOT for accounting
- Key feature: **apply transformations** over many existing cells in bulk to
  - clean up data
  - integrate data from multiple sources
  - perform entity reconciliation (against SPARQL endpoint)
  - convert data (e.g., RDF plugin)
  - *… among other things*

Using OpenRefine for KG Population

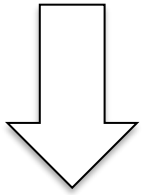
1. Import messy input data, transform it into a table, and clean it

	B	C	E	F	G	I	J
1	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Meta_score	Director
2	The Shawshank Redemption	1994	142 min	Drama	9.3	80	Frank Darabont
3	The Godfather	1972	175 min	Crime, Drama	9.2	100	Francis Ford Coppola
4	The Dark Knight	2008	152 min	Action, Crime, Drama	9	84	Christopher Nolan
5	The Godfather: Part II	1974	202 min	Crime, Drama	9	90	Francis Ford Coppola
6	12 Angry Men	1957	96 min	Crime, Drama	9	96	Sidney Lumet

2. (optional) Apply entity reconciliation to interlink with existing data sets.

3. Define the structure of the RDF output.

4. Export the data into some RDF syntax.



```
<http://example-instance.org/The%20Shawshank%20Redemption> a mv:Movie;
              mv:hasName "The Shawshank Redemption";
              mv:hasReleaseDate "1994-01-01T00:00Z";
              mv:hasGenre "Drama";
              mv:hasMovieDirector <http://example-instance.org/Frank%20Darabont> .

<http://example-instance.org/Frank%20Darabont> a mv:MovieDirector;
              mv:hasName "Frank Darabont" .
```

Step 1 - Data Transformation

Typical steps:

- Group and explore data items
- Delete columns or rows based on filter condition
- Split columns into several columns based on condition
- Modify messy data items with **GREL**, a powerful expression language
- Replay steps from a previous Refine project

	Series_Title	Released_Year	Runtime	Genre
1.	The Shawshank Redemption	1994	142 min	Drama
2.	The Godfather	1972	175 min	Crime, Drama
3.	The Dark Knight	2008	152 min edit	Action, Crime, Drama
4.	The Godfather: Part II	1974	202 min	Crime, Drama

Transform column "Released_Year" to date and execute String processing to get value of the "Runtime" as integer

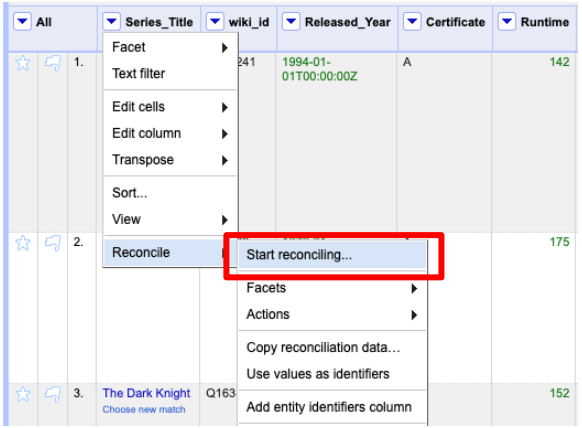
	Series_Title	Released_Year	Runtime	Genre
1.	The Shawshank Redemption	1994-01-01T00:00:00Z	142	Drama
2.	The Godfather	1972-01-01T00:00:00Z	175	Crime, Drama
3.	The Dark Knight	2008-01-01T00:00:00Z	152	Action, Crime, Drama
4.	The Godfather: Part II	1974-01-01T00:00:00Z	202	Crime, Drama

Split column "Genre" -> to create several "Genres"

	Series_Title	Released_Year	Runtime	Genre 1	Genre 2	Genre 3
1.	The Shawshank Redemption	1994-01-01T00:00:00Z	142	Drama		
2.	The Godfather	1972-01-01T00:00:00Z	175	Crime	Drama	
3.	The Dark Knight	2008-01-01T00:00:00Z	152	Action	Crime	Drama
4.	The Godfather: Part II	1974-01-01T00:00:00Z	202	Crime	Drama	

GREL = Google Refine Expression Language

Step 2 - Entity Reconciliation

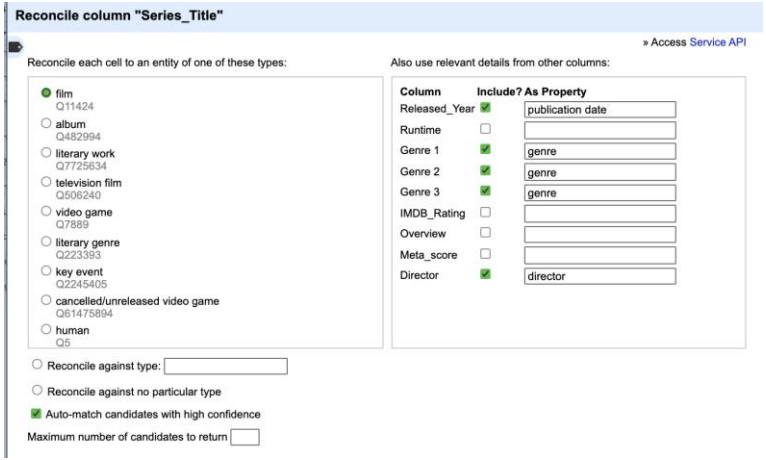


Reconciliation: process of matching text names to the corresponding semantic entities in another knowledge graph.

Typical steps:

- Define a reconciliation service (e.g. Wikidata)
- Select specific types to reconcile against (e.g., Film)
- Start reconciling a column against the service

	Series_Title	Wiki_ID	Released_Year
1.	The Shawshank Redemption Choose new match	Q172241	1994-01-01T00:00:00Z
2.	The Godfather Choose new match	Q47703	1972-01-01T00:00:00Z
3.	The Dark Knight Choose new match	Q163872	2008-01-01T00:00:00Z
4.	The Godfather Part II Choose new match	Q184768	1974-01-01T00:00:00Z



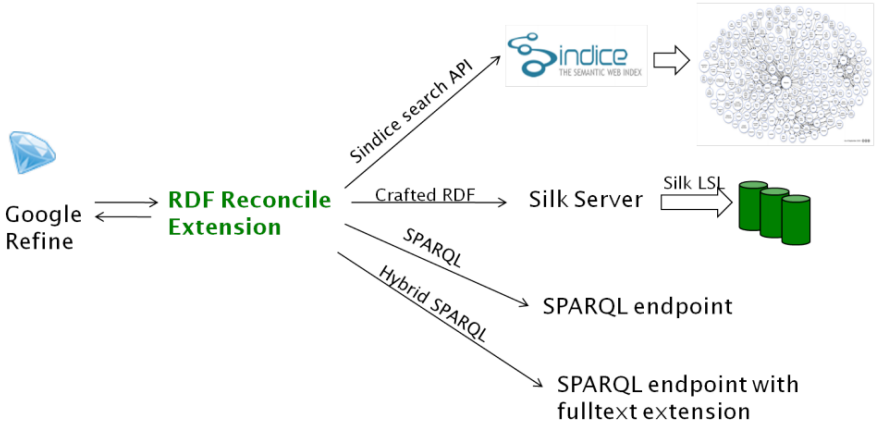
Adapted from EUCLID learning materials – "ProvidingLinked Data"

OpenRefine Reconciliation Extension

Several reconciliation services can be accessed from OpenRefine

☆ 👎	39.	The Pianist ☑ ☑ The Pianist (100) ☑ ☑ The Pianist (100) ☑ ☑ The Pianist (100) ☑ ☑ The Passionate Pianist (67) ☑ ☑ Create new item Search for match	2002-01-01T00:00:00Z
☆ 👎	40.	Gladiator ☑ ☑ Gladiator (100) ☑ ☑ The Gladiator (100) ☑ ☑ Gladiator (100) ☑ ☑ The Gladiator (100) ☑ ☑ Gladiators Seven (90) ☑ ☑ Gladiators of Rome (67) ☑ ☑ Create new item Search for match	2000-01-01T00:00:00Z
☆ 👎	41.	American History X Choose new match	1998-01-01T00:00:00Z
☆ 👎	42.	The Usual Suspects Choose new match	1995-01-01T00:00:00Z
☆ 👎	43.	Léon: The Professional Choose new match	1994-01-01T00:00:00Z
☆ 👎	44.	The Lion King ☑ ☑ The Lion King (100) ☑ ☑ The Lion King (100) ☑ ☑ The Lion King II: Simba's Pride (93) ☑ ☑ The Lion King 1½ (93) ☑ ☑ Leo the Lion: King of the Jungle (59) ☑ ☑ Create new item Search for match	1994-01-01T00:00:00Z

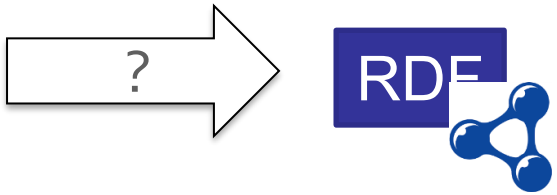
OpenCorporates service reconciles company names against legal entities database:
http://vimeo.com/17924204



Re-using Cool URIs: Entity Reconciliation Against LOD Hubs,
Fadi Maali, Richard Cyganiak, Vassilios Peristeras, Proceedings of the Linked Data on the Web Workshop 2011 (LDOW2011)

How to Generate RDF?

	Series_Title	Released_Year	Runtime	Genre 1	Genre 2	Genre 3
1.	The Shawshank Redemption	1994-01-01T00:00:00Z	142	Drama		
2.	The Godfather	1972-01-01T00:00:00Z	175	Crime	Drama	
3.	The Dark Knight	2008-01-01T00:00:00Z	152	Action	Crime	Drama
4.	The Godfather: Part II	1974-01-01T00:00:00Z	202	Crime	Drama	



RDF Refine:

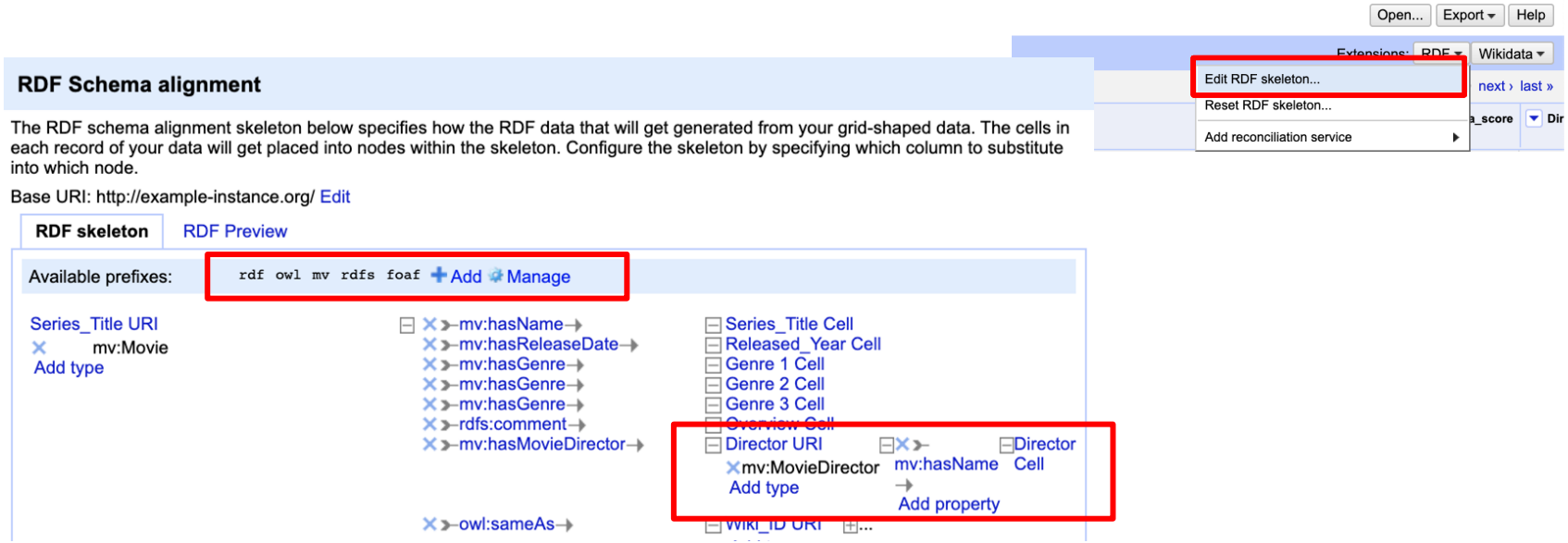
- Extension of OpenRefine to support RDF Generation
- Originally developed by INSIGHT – Galway
- Now further developed by open source communities

RDF Refine facilitate RDF export with the help of RDF skeletons

- Define the vocabulary and graph structure of the RDF serialization
- Export to Turtle or RDF/XML

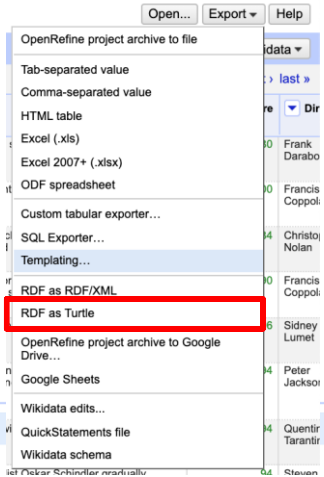
Step 3 - Define RDF Skeletons

An RDF skeleton defines the structure of the RDF triples that are exported



Adapted from EUCLID learning materials – "ProvidingLinked Data"

Step 4 - Export RDF Graph Data



Adapted from EUCLID learning materials – "ProvidingLinked Data"

Further Pointers

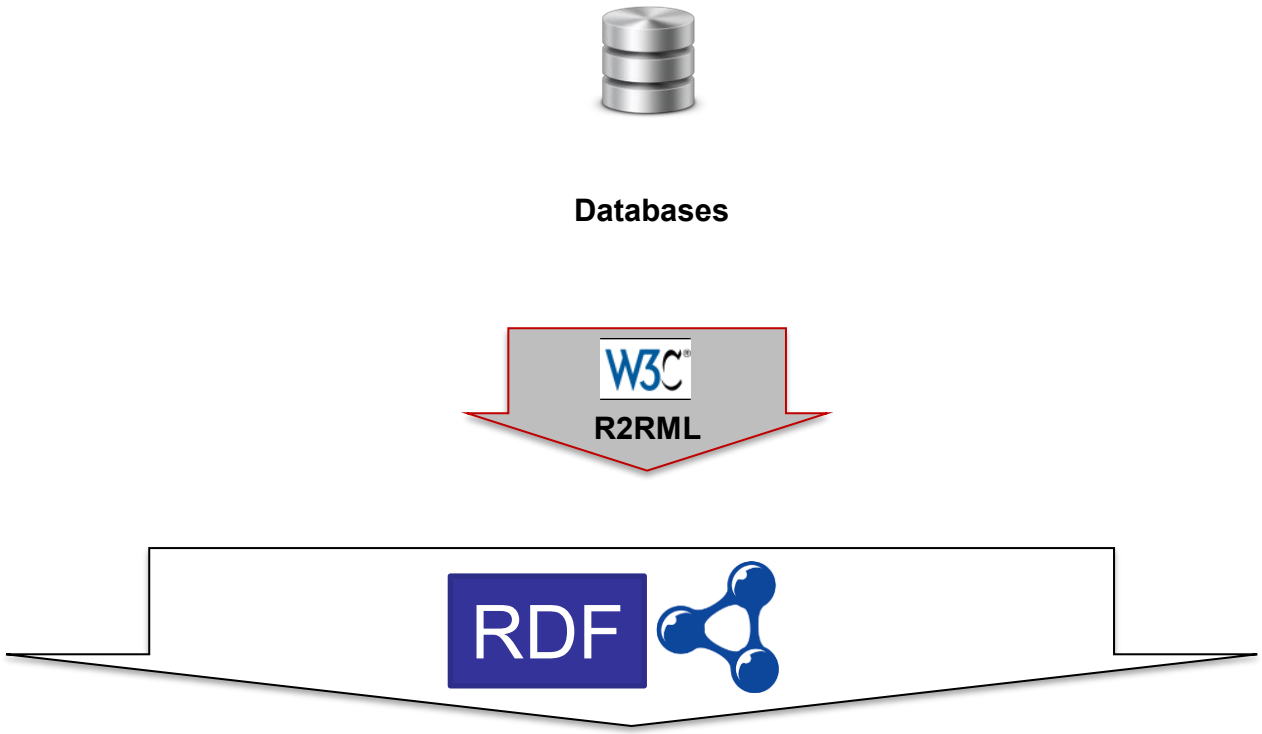
- Open Refine: https://openrefine.org/
- RDF Refine extension: https://github.com/stkenny/grefine-rdf-extension
- An example project with RDF skeletons: https://bit.ly/iss_openrefine_example

TU WIEN Informatics

Knowledge Graph Creation
Relational Data to RDF: R2RML

Emanuel Sallinger
Katja Hose

Extracting Relational Data: R2RML



Adapted from EUCLID learning materials – "ProvidingLinked Data"

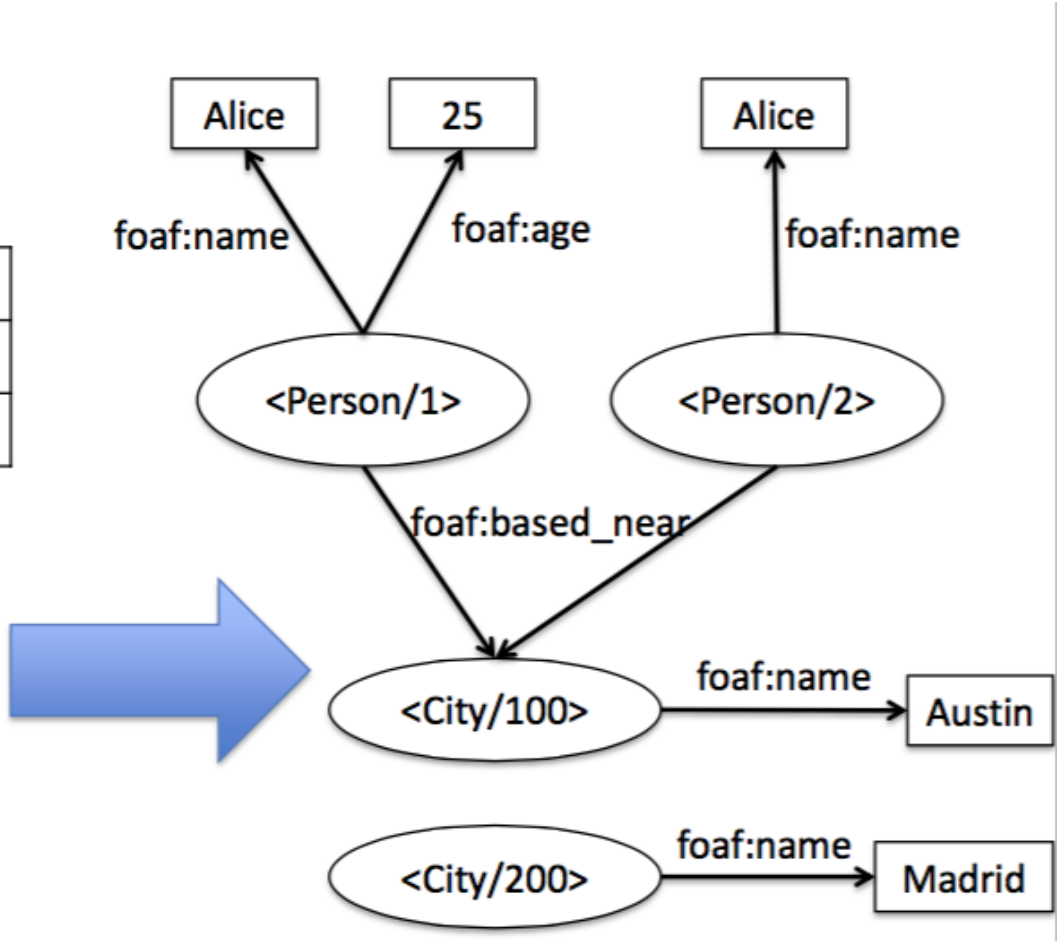
Goal: Transform Databases to RDF

Person

ID	NAME	AGE	CID
1	Alice	25	100
2	Bob	NULL	100

City

CID	NAME
100	Austin
200	Madrid

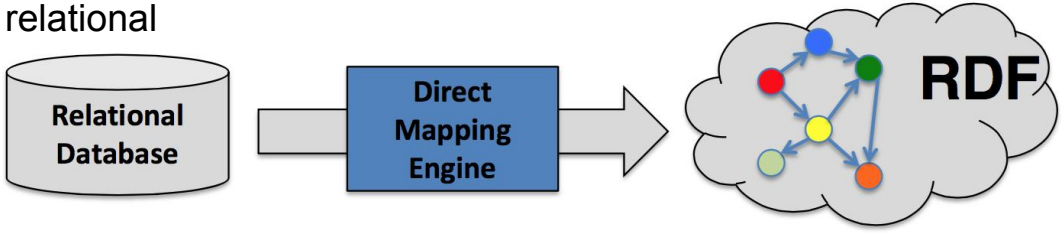


Source: ISWC 2013 Tutorial on RDB2RDF

W3C RDB2RDF

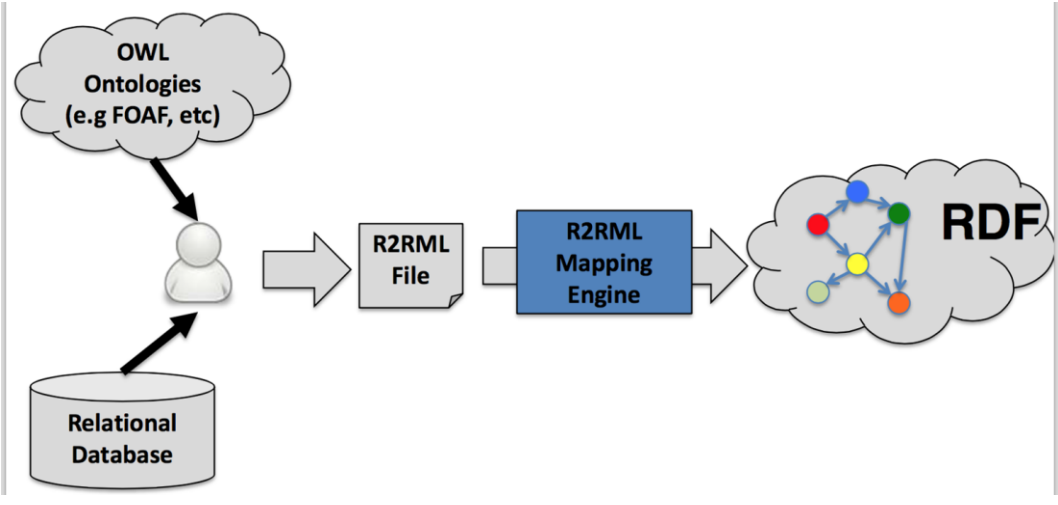
Direct mapping: directly exposes data as RDF

- Does not allow for vocabulary mapping
- No interlinking (unless URIs used in relational data)



R2RML, the RDB to RDF mapping language

- Allows vocabulary mapping (subject, predicate and object maps with class options)
- Allows interlinking – URIs can be constructed

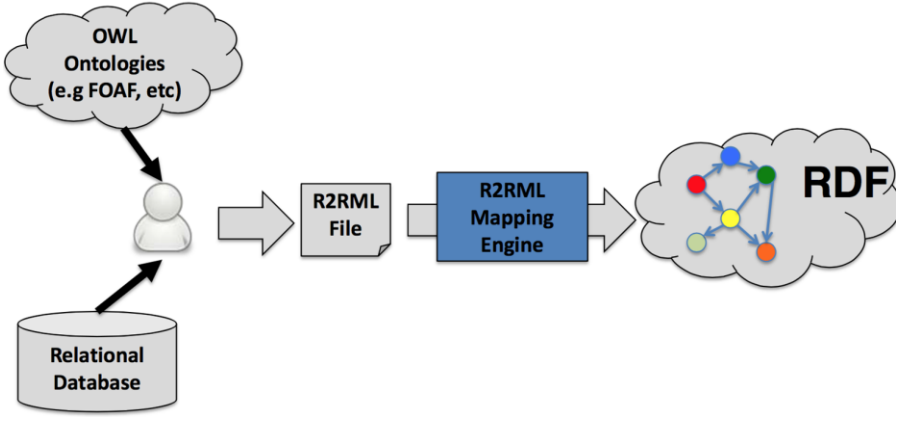


http://www.w3.org/2001/sw/rdb2rdf/

Source: ISWC 2013 Tutorial on RDB2RDF

R2RML Mapping Language – Basic Structure

- Each mapping called Triples Map "`rr:TriplesMap`"
- Consist of
  - The definition of the data source "`rr:LogicalTable`"
  - One rule to generate the subject of the triples "`rr:SubjectMap`"
  - Zero or more rules to generate the predicate and object of the triples "`rr:PredicateObjectMap`"

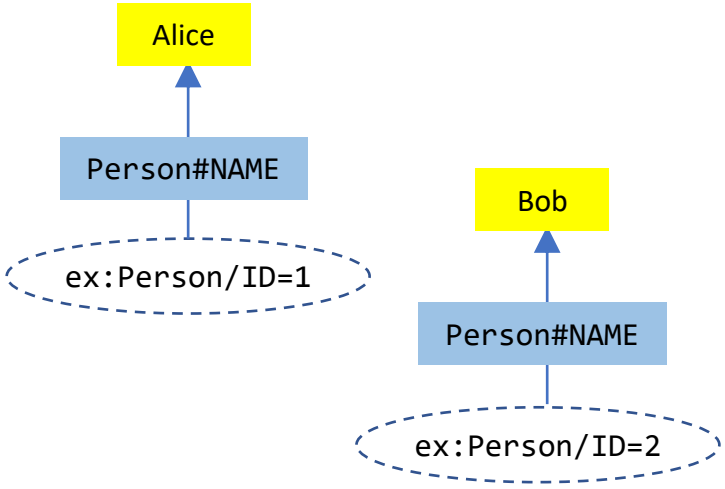


Direct Mapping as R2RML

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .

<TriplesMap1>
   a rr:TriplesMap;

   rr:logicalTable [ rr:tableName “Person”];

   rr:subjectMap [
      rr:template "http://www.ex.com/Person/ID={ID}";
      rr:class <http://www.ex.com/Person>
   ];

  rr:predicateObjectMap [
     rr:predicate <http://www.ex.com/Person#NAME> ;
     rr:objectMap [rr:column ”NAME" ]
].
```

Person

ID	NAME	AGE	CID
1	Alice	25	100
2	Bob	NULL	100



Source: ISWC 2013 Tutorial on RDB2RDF

Direct Mapping as R2RML

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .

<TriplesMap1>
   a rr:TriplesMap;

   rr:logicalTable [ rr:tableName “Person”];

   rr:subjectMap [
      rr:template "http://www.ex.com/Person/ID={ID}";
      rr:class <http://www.ex.com/Person>
   ];

   rr:predicateObjectMap [
      rr:predicate <http://www.ex.com/Person#NAME> ;
      rr:objectMap [rr:column ”NAME" ]
]
```

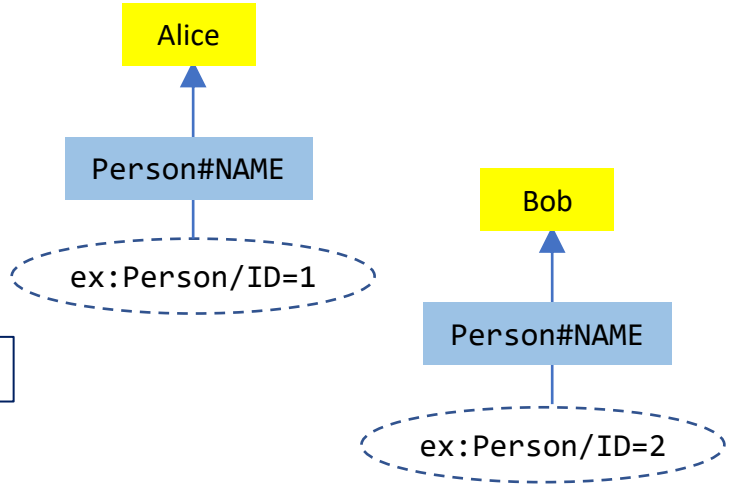
What is being mapped?

How to generate the subject URI?

How to generate the predicate and object? (optional)

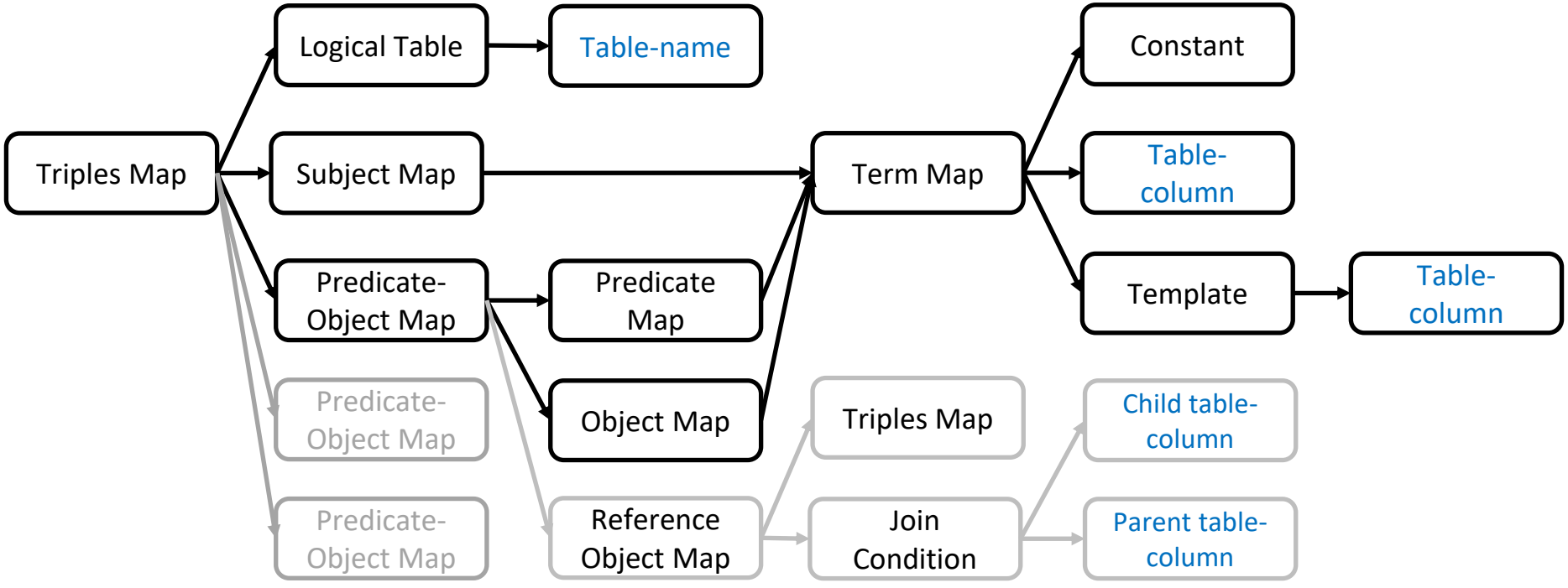
Person

ID	NAME	AGE	CID
1	Alice	25	100
2	Bob	NULL	100



Source: ISWC 2013 Tutorial on RDB2RDF

R2RML Mapping Definition



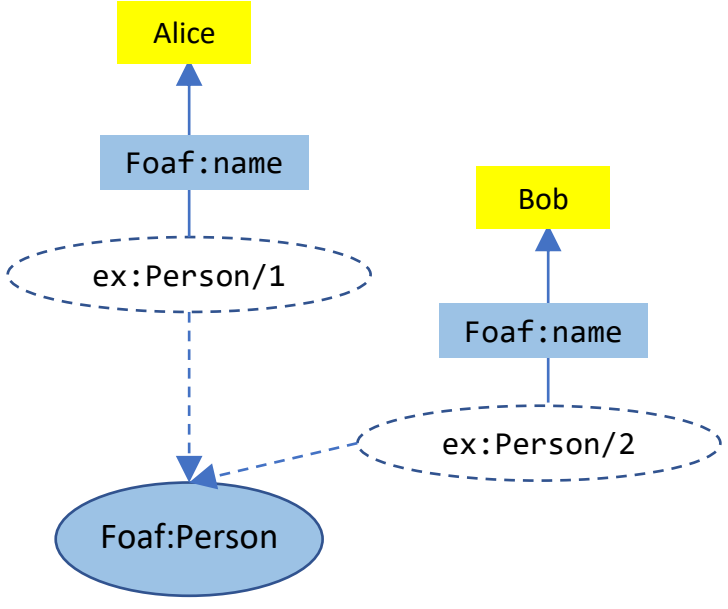
Dimou et al. - Mapping Hierarchical Sources into RDF using the RML Mapping Language
http://www.slideshare.net/andimou/rml-hierarchical-sourcesshort

Customizing R2RML Mapping

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

<TriplesMap1> a rr:TriplesMap;
  rr:logicalTable [ rr:tableName ”Person”];

  rr:subjectMap [
    rr:template "http://www.ex.com/Person/{ID}";
    rr:class    foaf:Person
  ];

  rr:predicateObjectMap [
    rr:predicate foaf:name;
    rr:objectMap [rr:column "NAME" ]
  ];
.
```

Person

ID	NAME	AGE	CID
1	Alice	25	100
2	Bob	NULL	100

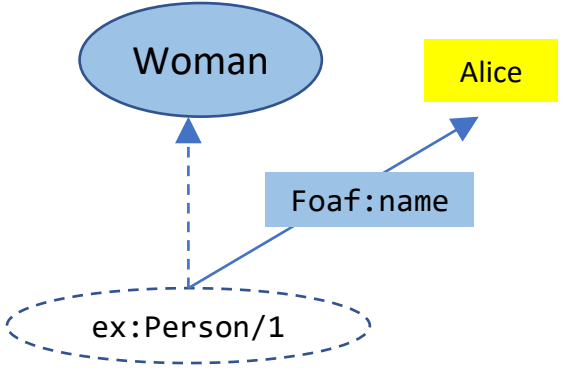


Customizing R2RML Source

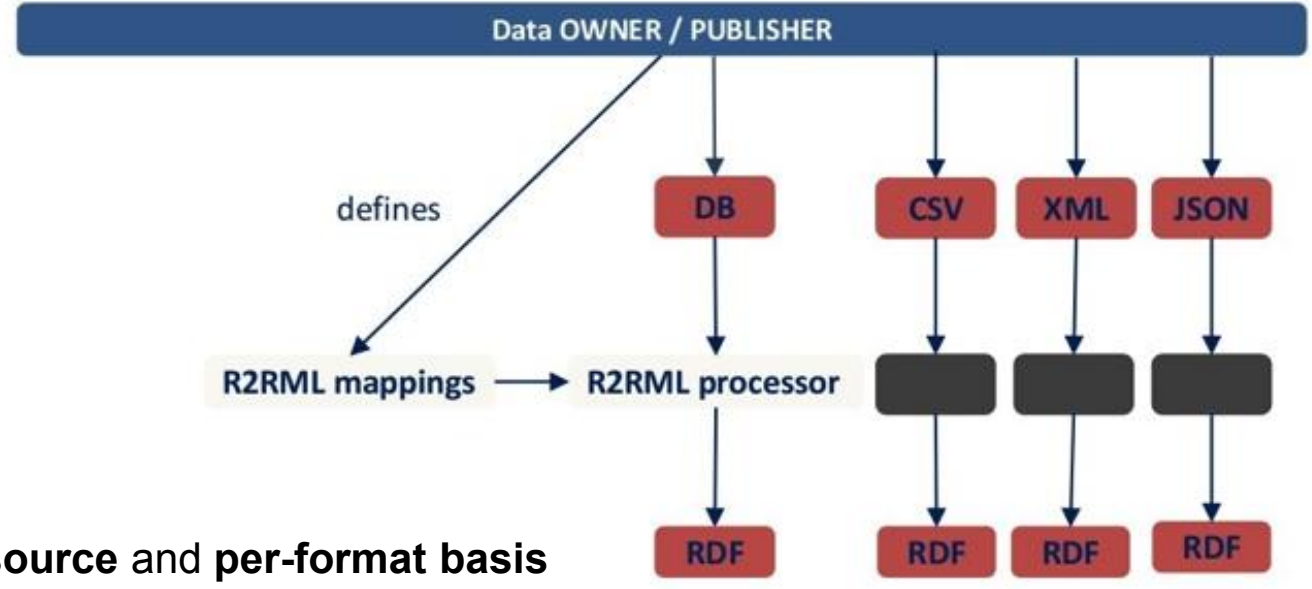
```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

<TriplesMap1>
  a rr:TriplesMap;
  rr:logicalTable [ rr:sqlQuery
        """ SELECT ID, NAME
        FROM Person WHERE gender = "F" """];
  rr:subjectMap [
    rr:template "http://www.ex.com/Person/{ID}";
    rr:class <http://www.ex.com/Woman>
  ];
  rr:predicateObjectMap [
    rr:predicate foaf:name;
    rr:objectMap [rr:column "NAME" ]
  ] .
```

Person

ID	NAME	GENDER
1	Alice	F
2	Bob	M



R2RML is good, but how about other formats?



- Mapping data **per-source** and **per-format basis**
- Mappings are **tied to implementations and not interoperable** across different implementations
- No **uniform way to describe mappings** of heterogeneous resources
  - **Mapping definitions can't be reused** for data in the same or different formats

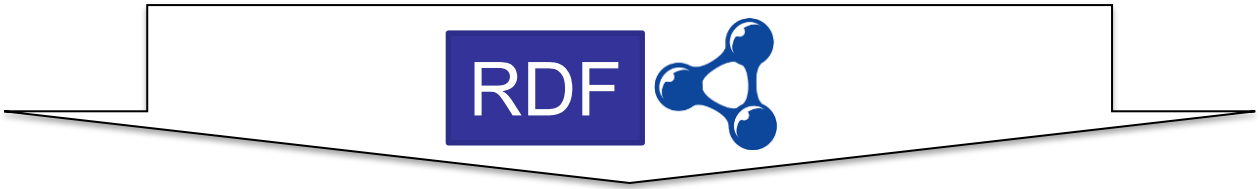
TU WIEN Informatics

Knowledge Graph Creation
Structured Data to RDF: RML

Emanuel Sallinger
Katja Hose

Extracting Structured Data: RML

Structured Data



Adapted from EUCLID learning materials – "ProvidingLinked Data"

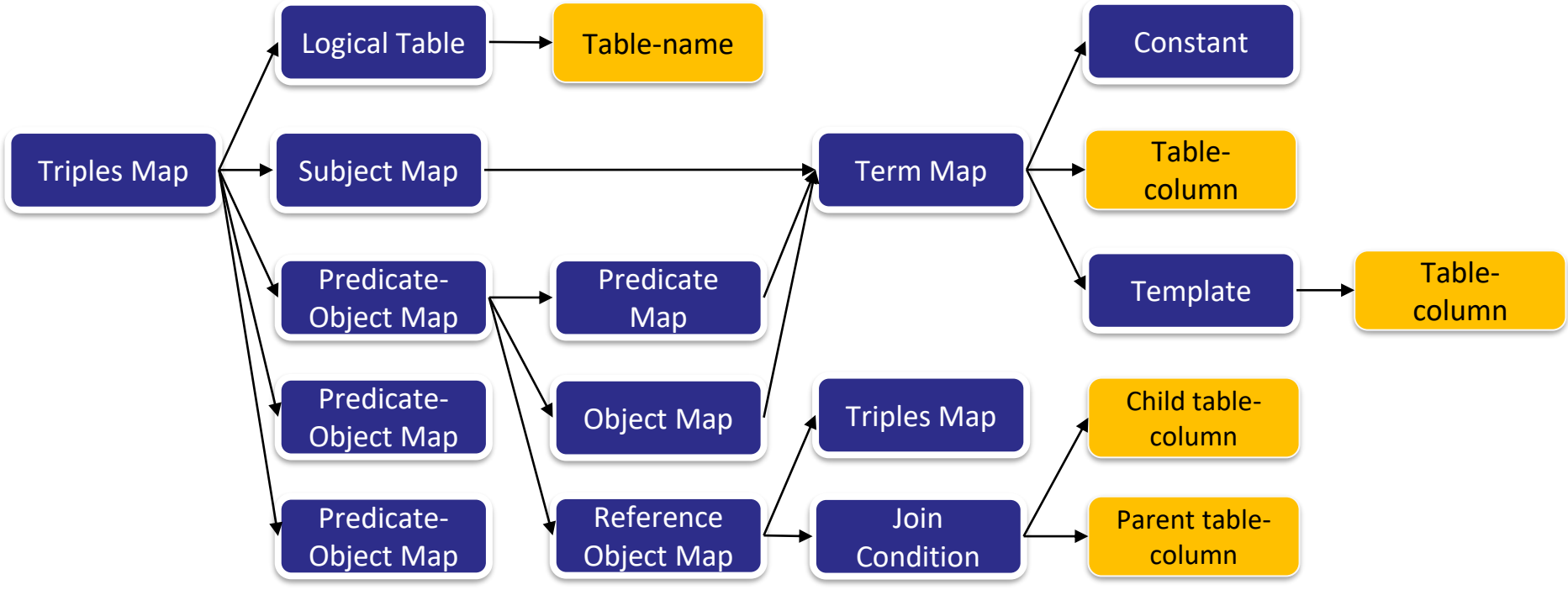
RDF Mapping Language (RML)

- A generic, scalable mapping language defined to express rules that map data in heterogeneous structures and serializations to the RDF data model.
- A superset of R2RML.

Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. "**RML: a generic language for integrated RDF mappings of heterogeneous data**." In *LDOW*. 2014.

Source: http://semweb.mmlab.be/rml/index.html

From R2RML to RML



Dimou et al. - Mapping Hierarchical Sources into RDF using the RML Mapping Language
http://www.slideshare.net/andimou/rml-hierarchical-sourcesshort

From R2RML to RML

R2RML	Generic mapping language
Logical Table	Logical Source (CSV, XML, JSON)
Table Name	Source name / URI
Column	???
per row iteration	???

References to values of heterogeneous resources

```
<PendingOrders>...
  <Order id="398">
    <Product>
      <Id>AE5982</Id>
      <Name>Samsonite DeLux 45</Name>
    </Product>
  </Order>...
<PendingOrders>
```

XPath for XML

Reference:

"Order@Id"

Iterator:

"/PendingOrders /Order"

```
{ ... ,
"ProductInStock" :
 { "ID": "567",
   "Name": "Samsonite DeLux 45",
   "type": "suitcase",
 }, ...
}
```

JSONPath for JSON

Reference:

"$. ProductInStock.ID"

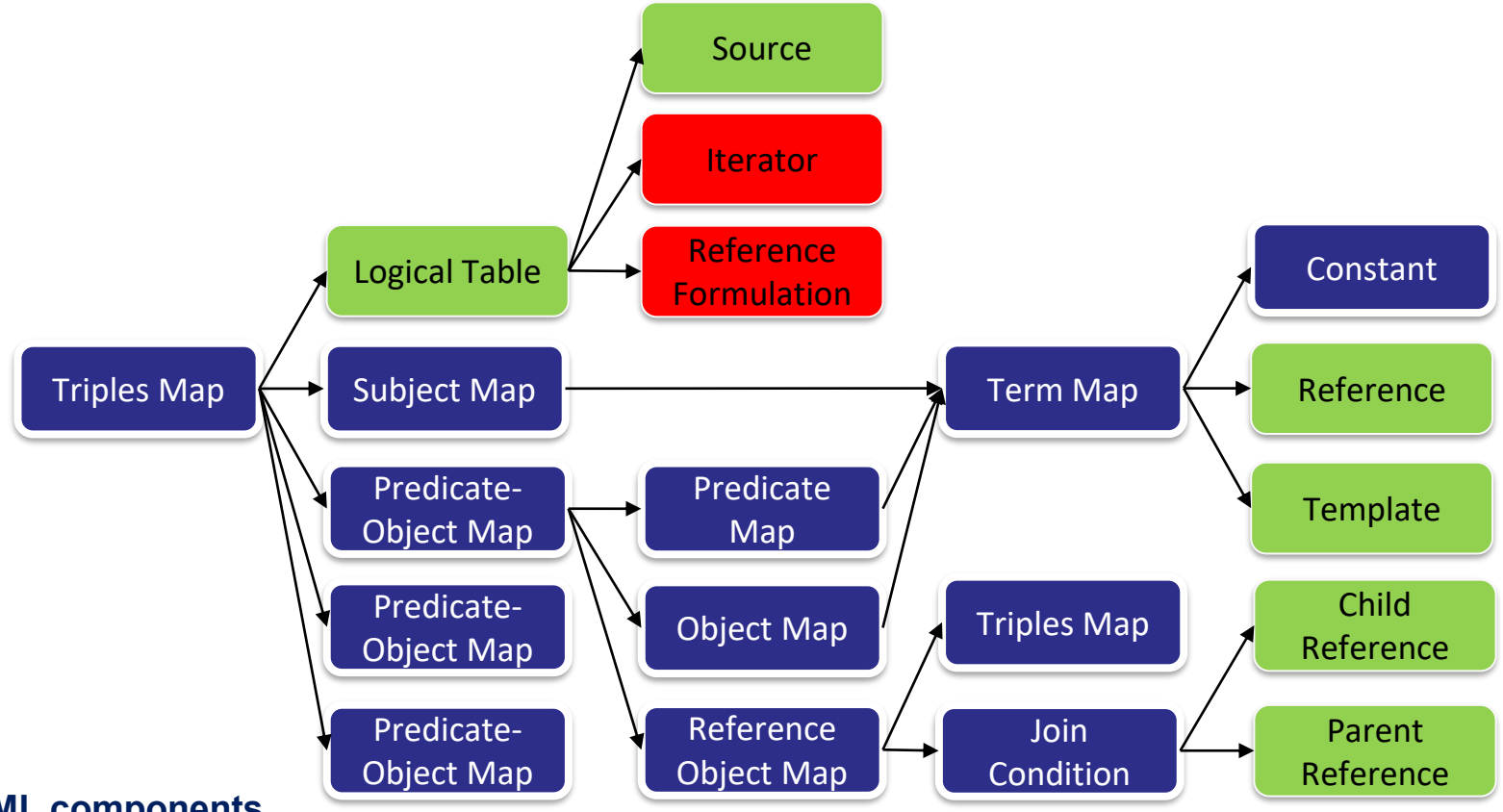
Iterator:

"$.ProductInStock"

From R2RML to RML

R2RML	~~R2~~RML
Logical Table	Logical Source (CSV, XML, JSON)
Table Name	Source name / URI
Column	Reference (defined Reference Formulation)
per row iteration	defined Iterator

From R2RML to RML



R2RML components
R2RML extended components
RML specific components

Dimou et al. - Mapping Hierarchical Sources into RDF using the RML Mapping Language
http://www.slideshare.net/andimou/rml-hierarchical-sourcesshort

RML Basics

Input in JSON format (People.json)

```
[ ...   { "Title": "Multimedia Lab",
          "People": [ { "name": "Anastasia",  "surname": "Dimou" },
                      { "name": "Ruben", "surname": "Verborgh" } ],
        { "Title": "Media and ICT",
          "People": [ ... ] } ,  ... ]
```

Output RDF representation

```
ex:Anastasia_Dimou a ex:Person ;
  ex:name "Anastasia".

ex:Ruben_Verborgh a ex:Person ;
  ex:name "Ruben".
```

RML Logical Source

```
<#PersonMapping>
  rml:logicalSource [
    rml:source "People.json" ;
    rml:referenceFormulation ql:JSONPath ;
    rml:iterator "$.[*].People" ;
  ]
.
```

Logical Source Mapping contains:

(i) a reference to input source(s)

(ii) the referenceFormulation to specify how to refer to the data (e.g., ql:CSV, ql:CSS3, ql:JSONPath, rr:SQL2008 and ql:Xpath) and

(iii) the iterator that specifies how to iterate over the data.

Source: http://semweb.mmlab.be/rml/index.html

RML Basics

Input in JSON format (People.json)

```
[ ...   { "Title": "Multimedia Lab",
          "People": [ { "name": "Anastasia",  "surname": "Dimou" },
                      { "name": "Ruben", "surname": "Verborgh" } ],
        { "Title": "Media and ICT",
          "People": [ ... ] } ,  ... ]
```

Output RDF representation

```
ex:Anastasia_Dimou a ex:Person ;
  ex:name "Anastasia".

ex:Ruben_Verborgh a ex:Person ;
  ex:name "Ruben".
```

RML Subject Map

```
<#PersonMapping>
  rr:subjectMap [
    rr:template "http://ex.com/Person/{name}_{surname}" ;
    rr:class ex:Person ;
  ]
.
```

The Subject Map consists of:

(i) the URI pattern that defines how each triple's subject is generated and

(ii) (optionally) its type.

The references to the input data occurs using valid references according to reference formulation specified at the Logical Source

Source: http://semweb.mmlab.be/rml/index.html

RML Basics

Input in JSON format (People.json)

```
[ ...   { "Title": "Multimedia Lab",
          "People": [ { "name": "Anastasia",  "surname": "Dimou" },
                      { "name": "Ruben", "surname": "Verborgh" } ],
        { "Title": "Media and ICT",
          "People": [ ... ] } ,  ... ]
```

Output RDF representation

```
ex:Anastasia_Dimou a ex:Person ;
  ex:name "Anastasia".

ex:Ruben_Verborgh a ex:Person ;
  ex:name "Ruben".
```

Predicate Object Map

```
<#PersonMapping>
  rr:predicateObjectMap [
    rr:predicate ex:name ;
    rr:objectMap [ rml:reference "name" ] ;
  ]
.
```

Triples are generated using Predicate Object Maps consisting of:

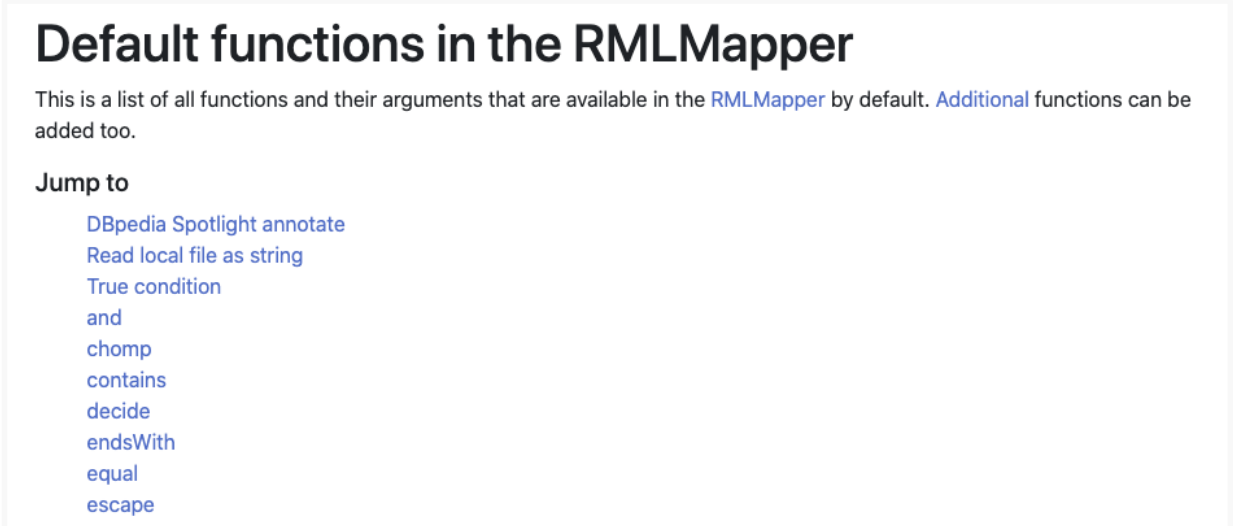
(i) A Predicate Map specifying how the triple's predicate is generated.

(ii) An Object Map specifying how the triple's object(s) are generated.

Source: http://semweb.mmlab.be/rml/index.html

Looks promising.. But can I operate on values?

- Custom functions can be added into RML engine
- However, the procedure for providing custom functions is implementation specific,
  - standard function specs are currently NOT defined
  - e.g., RML-Mapper: https://rml.io/docs/rmlmapper/default-functions/



Default functions in the RMLMapper

This is a list of all functions and their arguments that are available in the RMLMapper by default. Additional functions can be added too.

Jump to

- DBpedia Spotlight annotate
- Read local file as string
- True condition
- and
- chomp
- contains
- decide
- endsWith
- equal
- escape

Wait.. are there multiple RML implementations?

- Yes! → https://rml.io/implementation-report/

Name	Version	Test date	Contact	Web page
RMLMapper	4.9.0	2020-09-17	Anastasia Dimou	https://github.com/rmlio/rmlmapper-java
CARML	0.3.0	2020-09-15	Pano Maria	https://github.com/carml/carml
RocketRML	1.0.6	2019-06-28	Umutcan Simsek	https://github.com/semantifyit/RocketRML
SDM-RDFizer	3.2	2020-06-08	Maria-Esther Vidal	https://github.com/SDM-TIB/SDM-RDFizer
RMLStreamer	2.0.0	2020-06-10	Anastasia Dimou	https://github.com/RMLio/RMLStreamer
Chimera	2.1	2021-02-18	Mario Scrocca	https://github.com/cefriel/chimera

Still.. Writing RML mapping in turtle is cumbersome for some people...

Introducing YARRRML – RML rules in YAML

- YARRRML is a human readable text-based representation for RDF Graph generation rules (i.e., RML/R2RML).
  - A subset of [YAML].
  - Spec: https://rml.io/yarrrml/spec/

- A convenient way to write (R2)RML mappings

- YARRRML will be translated into (R2)RML mappings before executed by the appropriate engine.

Translation from RML to YARRRML

RML

```
… (prefixes ommitted)
map:map_person_000 a rr:TriplesMap ;
  rml:logicalSource [
    rml:source "data.json" ;
    rml:referenceFormulation ql:JSONPath ;
    rml:iterator "$.persons[*]" ;
  ] ;
  rr:subjectMap [
    rr:template "http://example.com/{firstname}" ;
    rr:class foaf:Person ;
  ];
  rr:predicateObjectMap [
    rr:predicate foaf:name ;
    rr:objectMap [
      rml:reference "firstname" ;
      rr:termType rr:Literal ;
  ]; ];
.
```

YARRRML

```
prefixes:
  ex: "http://example.com/"

mappings:
  person:
    sources:
      - ['data.json~jsonpath', '$.persons[*]']
    s: http://example.com/$(firstname)
    po:
      - [a, foaf:Person]
      - [foaf:name, $(firstname)]
```

Further Pointers

- RML demo using caRML (https://github.com/carml/carml/)
  - Example implementation available at: https://github.com/fekaputra/carmlizer
  - CLI tool: https://github.com/netage/carml-cli

- RML using YARRRML + Matey (on-the-fly)
  - Matey: https://rml.io/yarrrml/matey/

… and some final notes on RML

- Handling of some **details on RML mappings might be vary** between different engines
- Before implementing (complex) custom functions, **consider pre-processing the data** → might be more efficient due to the limitation of the RML / Query Engines (JSONPath / XPath)
- Before (or during) the RML mapping creations for JSON/XML, **consider using online JSONPath or XPath editors** for checking the correctness of iterator/queries.
- **YARRRML** simplified the creation of RML mappings; however, be wary that you need time to learn it (on top of learning RML basics)
- For you that hate to write in turtle/yaml, **RML editor** (https://rml.io/tools/rmleditor/) might be something for you
- Recently, OntoText (the company behind GraphDB) have forked OpenRefine + RDFRefine into one of its (free) product called **Ontotext-Refine**, which is a great alternative to the original OpenRefine (https://www.ontotext.com/products/ontotext-refine/)
- For **tabular data, consider using OpenRefine + RDF refine** before jumping into RML → OpenRefine is much more agile and powerful for tabular data (pre-)processing.

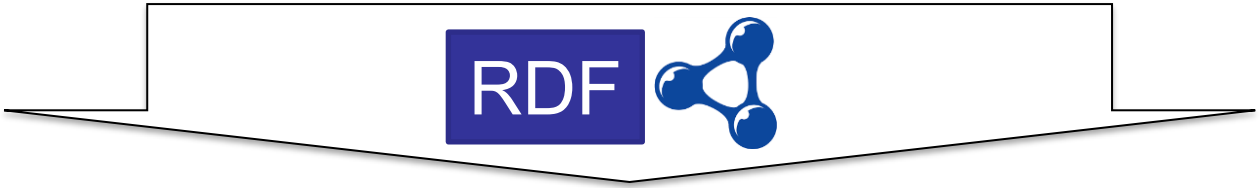
TU WIEN Informatics

Knowledge Graph Creation
Text to RDF

Emanuel Sallinger
Katja Hose

Mapping and Transformation Tools

Text



Adapted from EUCLID learning materials – "ProvidingLinked Data"

Document Annotation

Document Annotation with explicit semantics, e.g. semantic entities

Example for Linked Data Based Document Annotation

http://scihi.org/breaking-new-grounds-in-cinematography-stanley-kubrick/

Document Annotation allows Advanced Exploration

Stanley Kubrick

15 Recommended Articles:

- #1 Breaking New Grounds in Cinematography – Stanley Kubrick
- #2 Benoît Mandelbrot and the Beauty of Mathematics
- #3 Benoît Mandelbrot and the Beauty of Mathematics
- #4 Now I am become Death, the Destroyer of Worlds
- #5 Charles Proteus Steinmetz and the Alternating Current

Stanley Kubrick

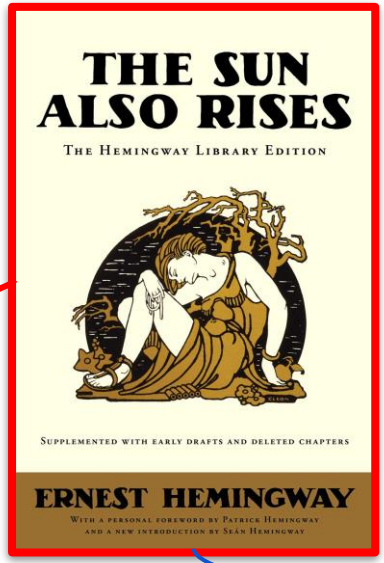
Stanley Kubrick (/ˈkuːbrɪk/; July 26, 1928 – March 7, 1999) was an American film director, screenwriter, producer, cinematographer, and editor who did much of his work in the United Kingdom. He is regarded as one of the great filmmakers. He was part of the New Hollywood wave of filmmaking. His films, typically adaptations of novels or short stories, are noted for their "dazzling" and unique cinematography, attention to detail in the service of realism, and the evocative use of music. Kubrick's films covered a variety of genres, including war, crime, literary adaptations, romantic and black comedies, horror, epic and science fiction. Kubrick

DBpedia: Stanley Kubrick

Semantic Annotation

- The term “**annotation**” implies to **attach data to some other piece of data**. It establishes a **(typed) relation between the annotated data and the annotating data**
- The term “**annotation**” can denote both the **process of annotating** and the **result of that process**
- **Formal Model:**
  - An **annotation** A is a tuple **(a_s, a_p, a_o, a_c)**, where
    - **a_s** is the **subject** of the annotation (the annotated data)
    - **a_o** is the **object** of the annotation (the annotating data)
    - **a_p** is the **predicate** (the annotation relation) that defines the type of relationship between a_s and a_o, and
    - **a_c** is the **context** in which the annotation is made.

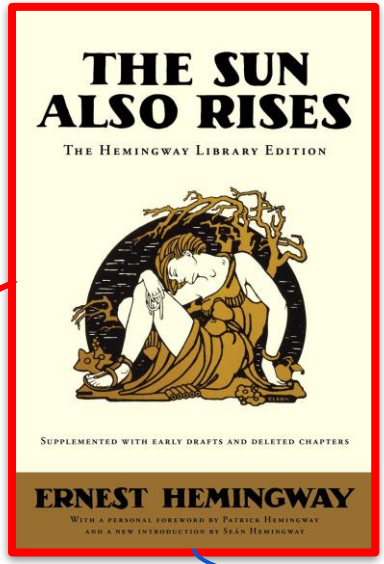
Semantic Annotation



- **Formal Model:**
  - An **annotation** A is a tuple **(a_s, a_p, a_o, a_c)**, where
    - a_s is the **subject** of the annotation (the annotated data)
    - a_o is the **object** of the annotation (the annotating data)
    - a_p is the **predicate** (the annotation relation) that defines the type of relationship between a_s and a_o, and
    - a_c is the **context** in which the annotation is made.

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

:9787532717071 dbo:author dbr:Ernest_Hemingway .
```

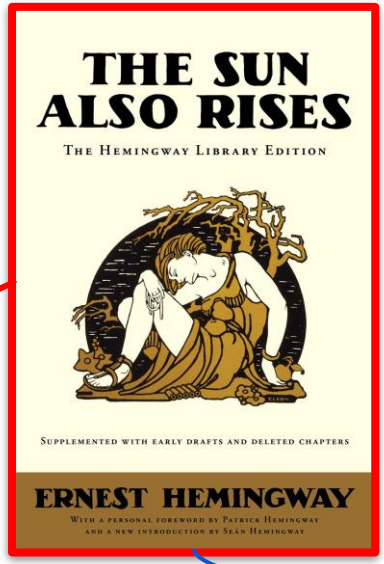
Semantic Annotation



- **Formal Model:**
  - An **annotation** A is a tuple **(a_s, a_p, a_o, a_c)**, where
    - **a_s** is the **subject** of the annotation (the annotated data)
    - **a_o** is the **object** of the annotation (the annotating data)
    - **a_p** is the **predicate** (the annotation relation) that defines the type of relationship between a_s and a_o, and
    - **a_c** is the **context** in which the annotation is made.

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

:9787532717071 dbo:author dbr:Ernest_Hemingway .
```

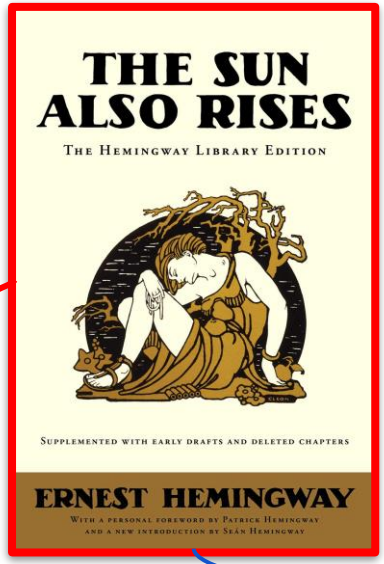
Semantic Annotation



- **Formal Model:**
  - An **annotation** A is a tuple **(a_s, a_p, a_o, a_c)**, where
    - a_s is the **subject** of the annotation (the annotated data)
    - a_o is the **object** of the annotation (the annotating data)
    - a_p is the **predicate** (the annotation relation) that defines the type of relationship between a_s and a_o, and
    - a_c is the **context** in which the annotation is made.

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

:9787532717071 dbo:author dbr:Ernest_Hemingway .
```

Semantic Annotation



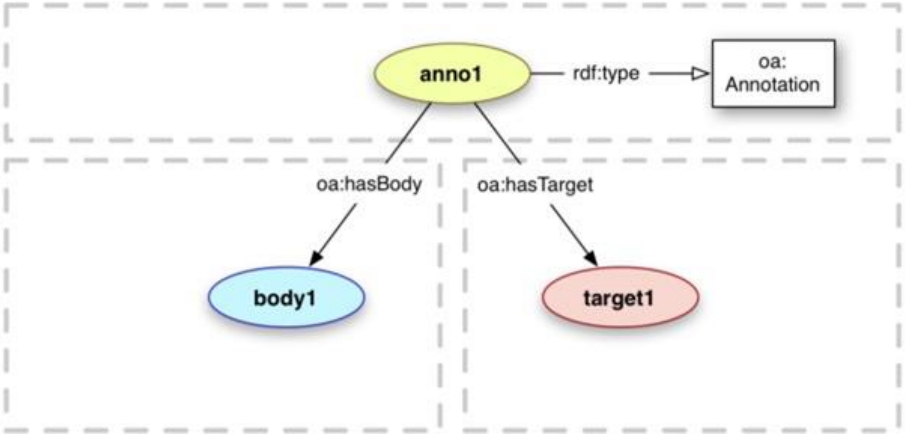
- **Formal Model:**
  - An **annotation** A is a tuple **(a_s, a_p, a_o, a_c)**, where
    - a_s is the **subject** of the annotation (the annotated data)
    - a_o is the **object** of the annotation (the annotating data)
    - a_p is the **predicate** (the annotation relation) that defines the type of relationship between a_s and a_o, and
    - a_c is the **context** in which the annotation is made.

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

:9787532717071 dbo:author dbr:Ernest_Hemingway .
```

- `created at Dec 11, 2017`
- `created by Lysander07`
- `...`

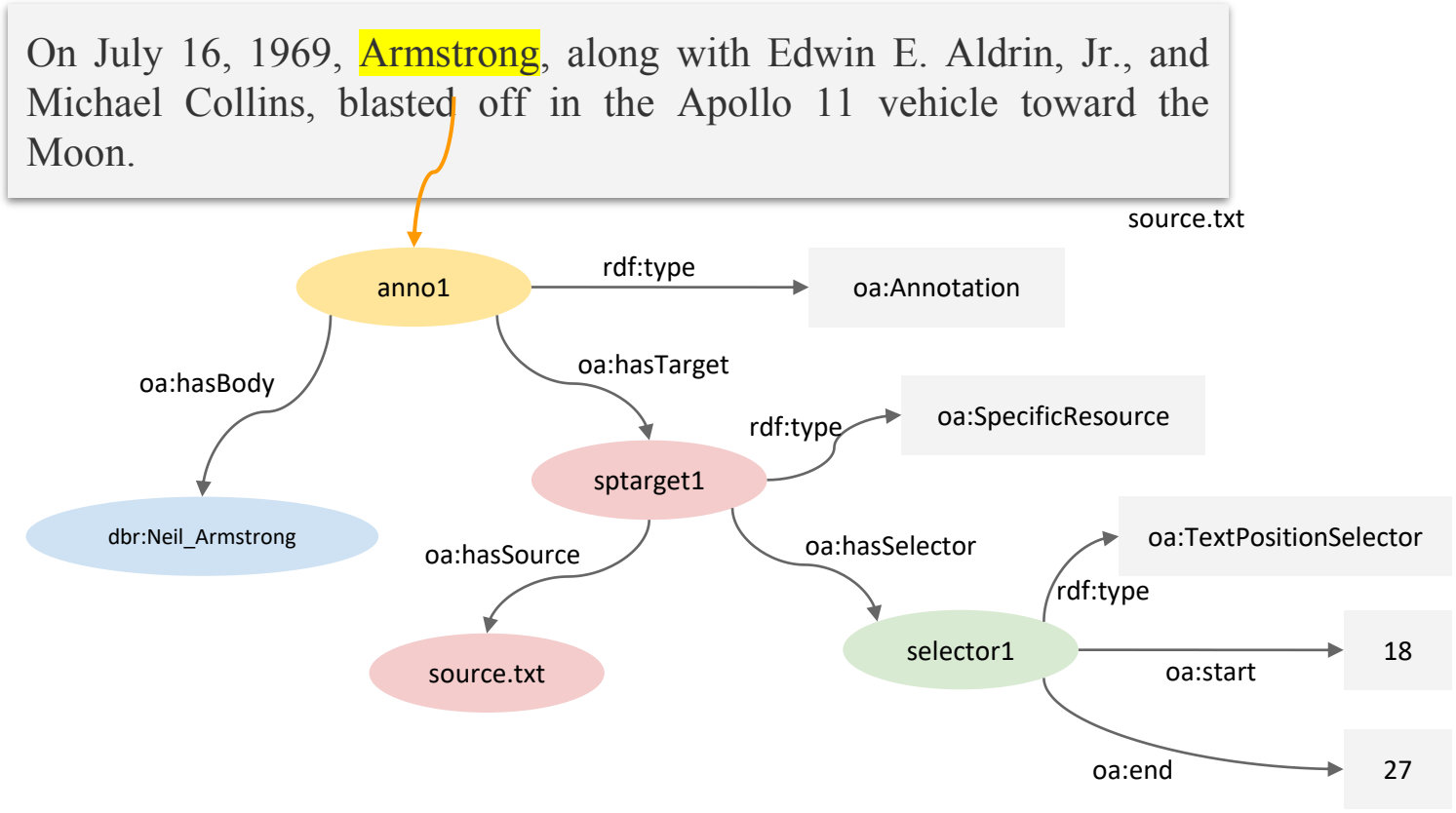
Ontologies for Semantic Annotation

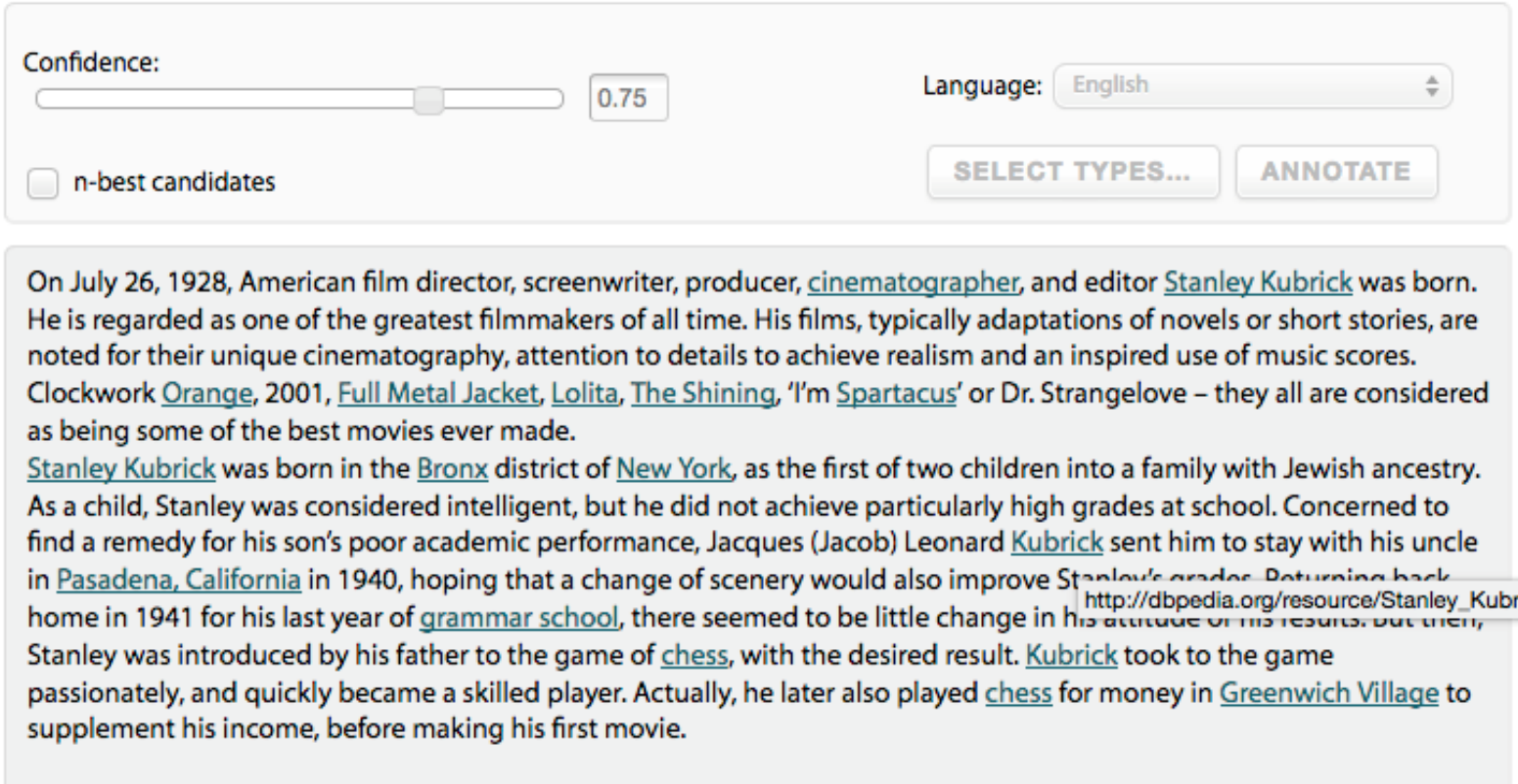
- The **Open Annotation Ontology** is a W3C Standard (2017) [1]
  - "...specifies an interoperable framework for creating associations between related resources, annotations, using a methodology that conforms to the Architecture of the World Wide Web."



- The **Body** provides the information which is annotating the **Target**

Open Annotation Ontology Example

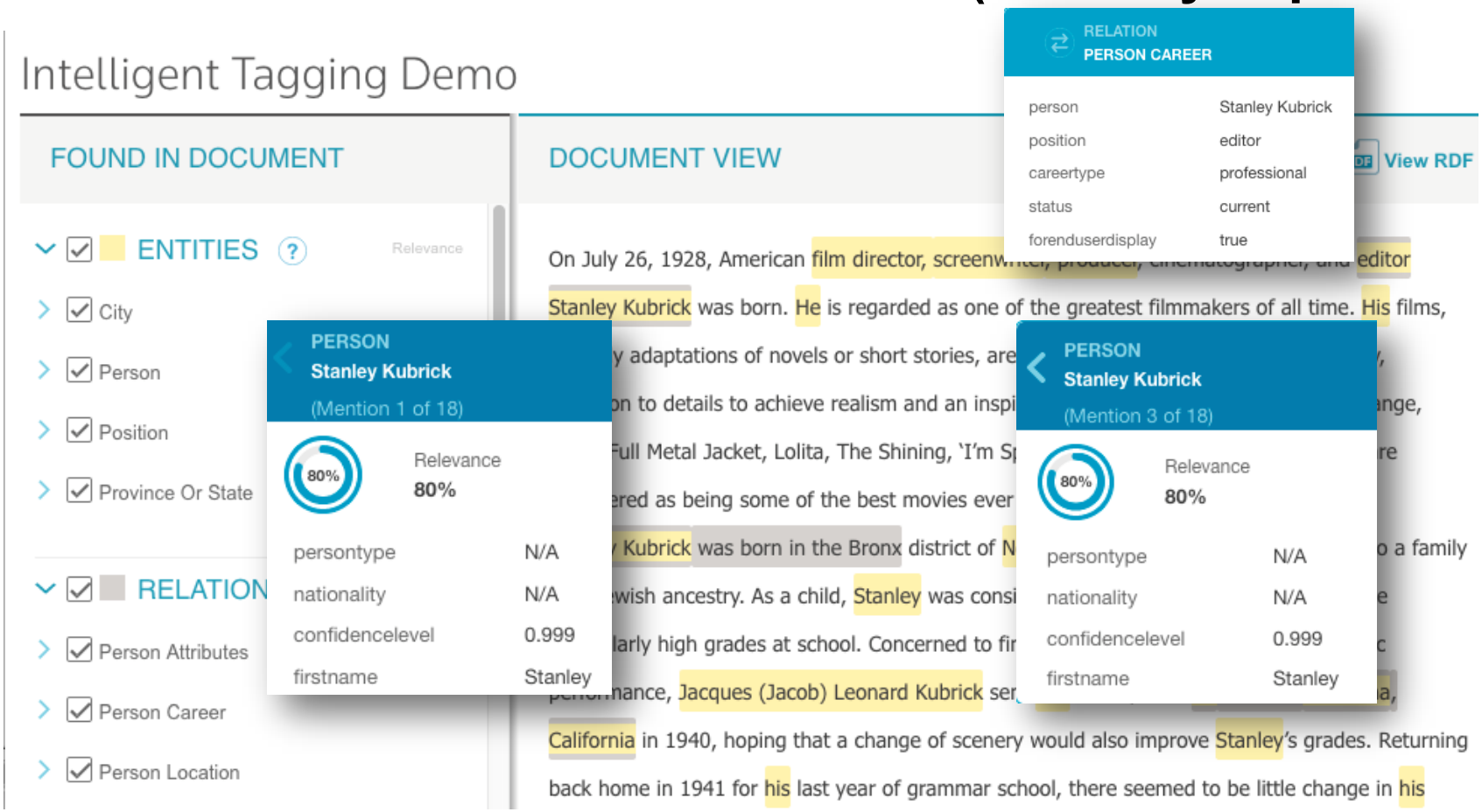




- Linking unstructured information sources to the LOD cloud through DBpedia
- Web Service, Java/Scala API, jQuery plugin, Java and PHP clients
- Currently supporting more than 15 languages!

Source: https://demo.dbpedia-spotlight.org/

PermIDs (formerly: OpenCalais)



- NLP and machine learning to extract entities, facts, and events
- Commercial product (https://www.refinitiv.com/en)

Source: https://permid.org/onecalaisViewer

GATE

- Common problem with general purpose, open-domain semantic annotation tools
- Best results require bespoke customisation

General Architecture for Text Engineering

- Open-source framework and development environment
- Started 1996, large developer community
- Used worldwide by many organisations to build bespoke solutions; e.g. Press Association and the National Archive
- Information Extraction in many languages

http://www.gate.ac.uk/

GATE Cloud NLP Services

Home | Services | Dashboard

Show only items tagged: Basque (1) | Biomed (1) | Bulgarian (1) | Catalan (1) | Chunker (1) | Croatian (1) | Custom (2) | Czech (1) | Danish (1) | Dutch (2) | English (19) | Environment (2) | Estonian (1) | Finnish (1) | French (4) | German (5) | Greek (1) | Indonesian (1) | Latvian (1) | Measurements (2) | Morphology (1) | Named Entity (23) | OpenNLP (3) | Opinion Mining (4) | Part-of-Speech (23) | Polish (1) | Politics (2) | Portugese (1) | Romanian (2) | Russian (3) | Server (2) | Slovak (1) | Slovenian (1) | SoBigData (3) | Spanish (3) | Summarization (2) | Swedish (1) | Term Recognition (2) | Twitter (14) | Veracity (1) | Welsh (1)

English Named Entity Recognizer

Identify names of *persons*, *locations*, *organizations*, as well as *money amounts*, *time and date expressions* in English texts automatically.

1,200 free requests / day
Larger batches **GBP0.80 / CPU hour**

English Named Entity Recognizer for Tweets

Analyse tweets for names of *persons*, *locations*, *organizations* and other entities. Also performs normalization of abbreviations and common shorthands ("brb", "gr8", "2day",etc.).

1,200 free requests / day
Larger batches **GBP0.80 / CPU hour**

Twitter Collector

Collect tweets, view tweet statistics, and store results in your dashboard for further analysis.

GBP0.05 / CPU hour

BioYODIE Named Entity Disambiguation

Identifies biomedical named entities of various types, and disambiguates them against UMLS.

800 free requests / day
Batch processing not available

YODIE Named Entity Disambiguation (English)

Identifies named entities of various types in English documents, and disambiguates them against DBpedia.

800 free requests / day
Batch processing not available

YODIE Named Entity Disambiguation (German)

Identifies named entities of various types in German documents, and disambiguates them against DBpedia.

800 free requests / day
Batch processing not available

YODIE Named Entity Disambiguation

Annotation types: Mention

On July 26, 1928, American film director, screenwriter, producer, cinematographer, and editor Stanley Kubrick was born. He is regarded as one of the greatest filmmakers of all time. His films, typically adaptations of novels or short stories, are noted for their unique cinematography, attention to details to achieve realism and an inspired use of music scores. Clockwork Orange, 2001, Full Metal Jacket, Lolita, The Shining, 'I'm Spartacus' or Dr. Strangelove – they all are considered as being some of the best movies ever made. Stanley Kubrick was born in the Bronx district of New York, as the first of two children into a family with Jewish ancestry. As a child, Stanley was considered intelligent, but he did not achieve particularly high grades at school. Concerned to find a remedy for his son's poor academic performance, Jacques (Jacob) Leonard Kubrick sent him to stay with his uncle in Pasadena, California in 1940, hoping that a change of scenery would also improve Stanley's grades. Returning back home in 1941 for his last year of grammar school, there seemed to be little change in his attitude or his results. But then, Stanley was introduced by his father to the game of chess, with the desired result. Kubrick took to the game passionately, and quickly became a skilled player. Actually, he later also played chess for money in Greenwich Village to supplement his income, before making his first movie.

Annotations at this location

Mention	
confidence	0.965
dbpInterestingClasses	dbpedia-owl:Person
dbpSpecificClasses	dbpedia-owl:Actor
inst	http://dbpedia.org/resource/Stanley_Kubrick

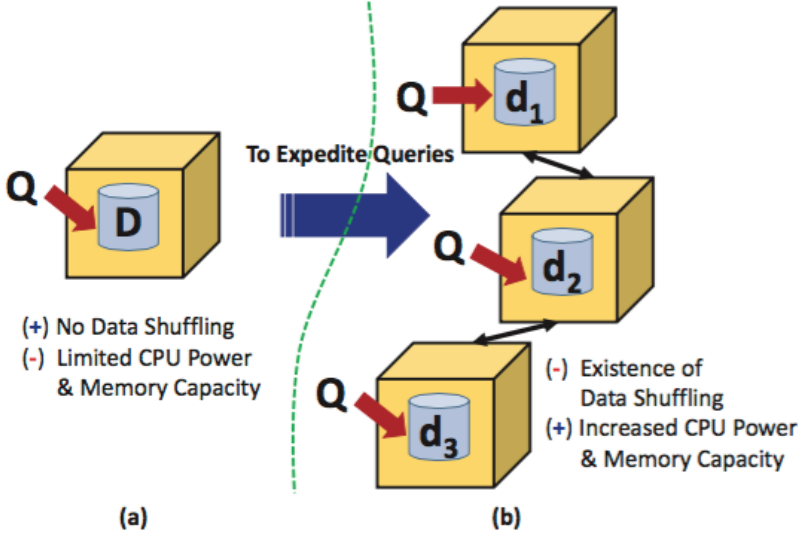
TU WIEN Informatics

Knowledge Graph Creation
RDF Storage

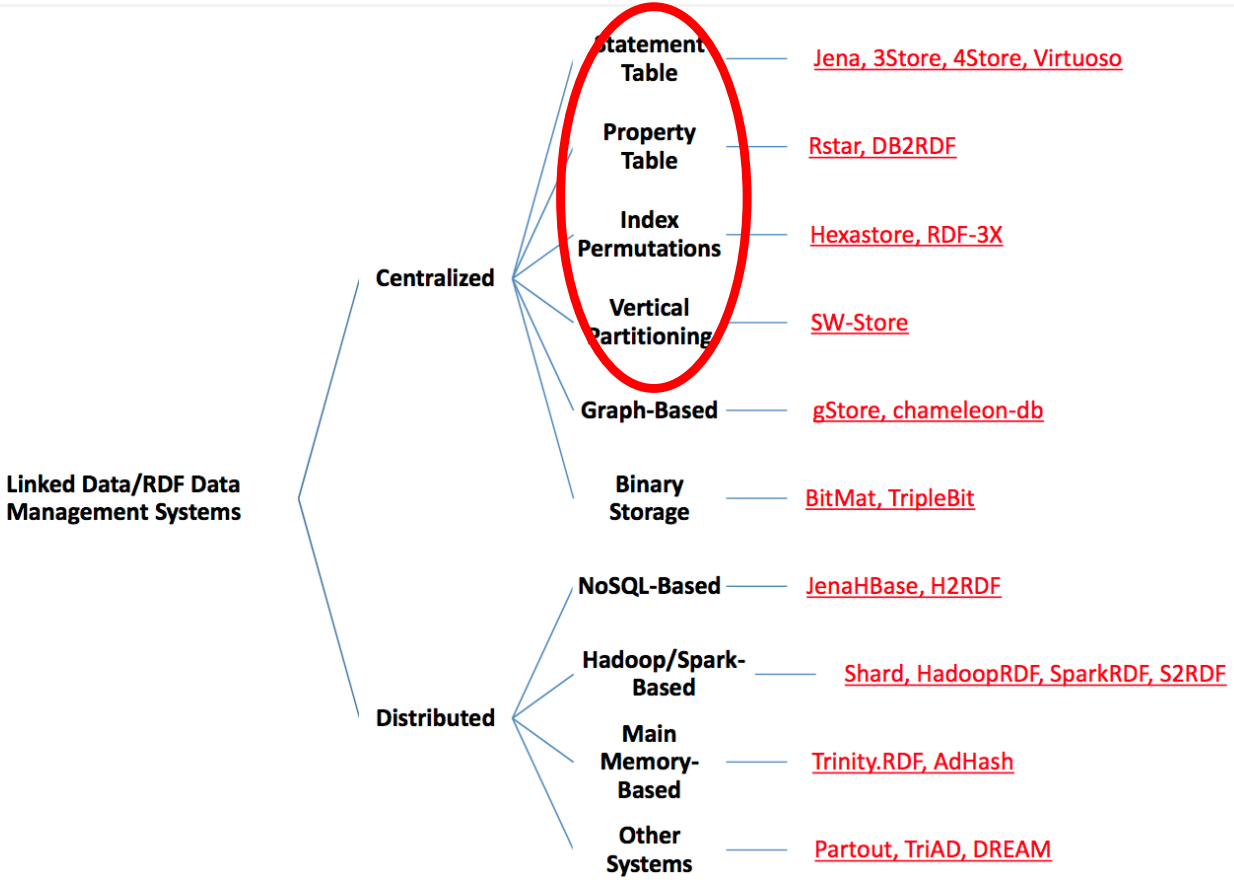
Emanuel Sallinger
Katja Hose

RDF Storage Systems

- Centralized systems (a):
  - storage and processing of RDF data on a single node
- Distributed systems (b):
  - storage and processing of RDF data on a multiple nodes



A Taxonomy of RDF Storage Systems



Statement Table

Triple statements are directly stored in a table-like structure as a linearized list of triples.

Subject	Predicate	Object
Product12345	rdf:type	bsbm:Product
Product12345	rdfs:label	Canon Ixus 2010
Product12345	bsbm:producer	bsbm-inst:Producer1234
...	...	...
Producer1234	rdf:label	Canon
Producer1234	foaf:homepage	http://www.canon.com
...	...	...

Example systems: Jena SDB, 3Store, 4Store, Virtuoso.

Property Table

Cluster commonly accessed nodes in the graph together in a single table to avoid the expensive cost of many self-join operations on the large statement table encoding the RDF data

Product Property Table

Subject	Type	Label	NumericProperty1	aaa
Product12345	bsbm:Product	Canon Ixus 2010	NULL	...
...	...	...	...	...

Left-Over Triples

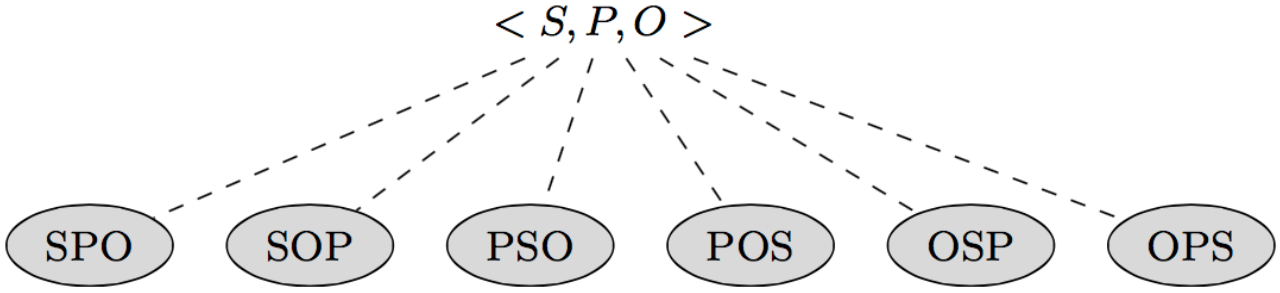
Subject	Predicate	Object
Producer1234	foaf:homepage	http://www.canon.com
...	...	...

- Example systems: DB2RDF, Jena2 SDB.

Index Permutations

All possible combinations the three components of triples is indexed and materialized

Rationale: any query can be answered using the available indices so that it allows fast access to all parts of the triples by sorted lists and fast merge-joins



- Example systems: Hexastore, RDF-3x.

Vertical Partitioning

Creates a table for each unique property in the dataset

- Containing subjects and objects connected by the property
- For multi-valued attributes, each distinct value is listed in a successive row
- Tables are indexed by subject for quick retrieval

<rdf:type>

Subject	Object
Product12345	bsbm:Product

<rdfs:label>

Subject	Object
Product12345	Canon Ixus 2010
Producer1234	Canon

<aaa>

Subject	Object
uuu	xxx
...	...

- Example systems: SW-Store.

TU WIEN Informatics

188.399 Introduction to Semantic Systems

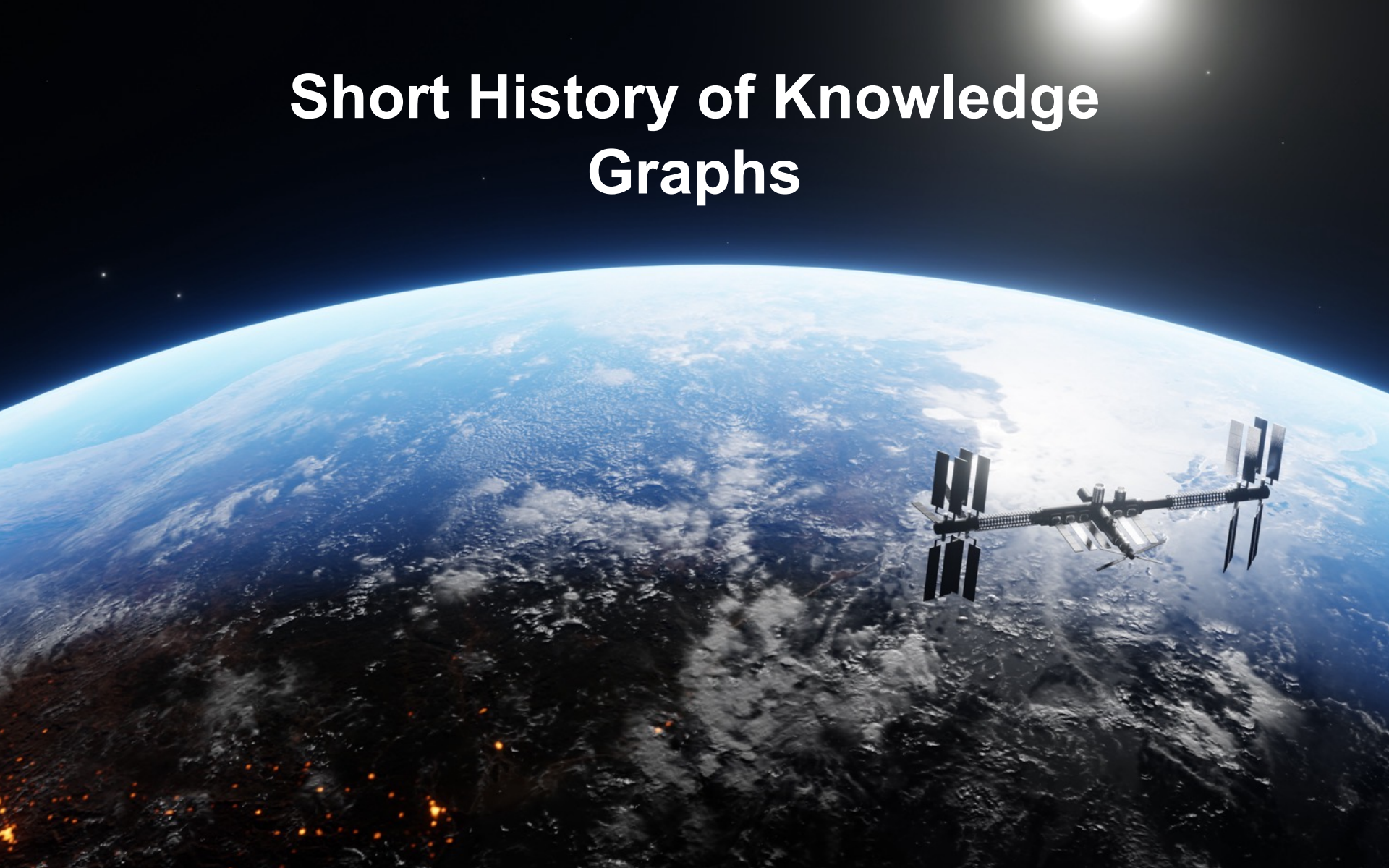
(2.0 VU / 3.0 ECTS)

2023W

Katja Hose
Emanuel Sallinger

iss@dbai.tuwien.ac.at

Short History of Knowledge Graphs



How the journey began…

- Expressing meaning
- Knowledge representation
- Ontologies
- Agents
- Evolution of knowledge

The Semantic Web

A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities

by TIM BERNERS-LEE, JAMES HENDLER and ORA LASSILA

The entertainment system was belting out the Beatles' "We Can Work It Out" when the phone rang. When Pete answered, his phone turned the sound down by sending a message to all the other *local* devices that had a *volume control*. His sister, Lucy, was on the line from the doctor's office: "Mom needs to see a specialist and then has to have a series of physical therapy sessions. Biweekly or something. I'm going to have my agent set up the appointments." Pete immediately agreed to share the chauffeuring. At the doctor's office, Lucy instructed her Semantic Web agent through her handheld Web browser. The agent promptly retrieved information about Mom's *prescribed treatment* from the doctor's agent, looked up several lists of *providers*, and checked for the ones *in-plan* for Mom's insurance within a *20-mile radius* of her *home* and with a *rating* of *excellent* or *very good* on trusted rating services. It then began trying to find a match between available *appointment times* (supplied by the agents of individual providers through their Web sites) and Pete's and Lucy's busy schedules. (The emphasized keywords indicate terms whose semantics, or meaning, were defined for the agent through the Semantic Web.)

In a few minutes the agent presented them with a plan. Pete didn't like it—University Hospital was all the way across town from Mom's place, and he'd be driving back in the middle of rush hour. He set his own agent to redo the search with stricter preferences about *location* and *time*. Lucy's agent, having *complete trust* in Pete's agent in the context of the present task, automatically assisted by supplying access certificates and shortcuts to the data it had already sorted through.

Killer app: Agents

- Retrieve information about prescribed treatment/therapy
- Look for providers/specialists with good ratings in proximity
- Schedule times according to multiple busy schedules
- Share information with other agents and “negotiate”

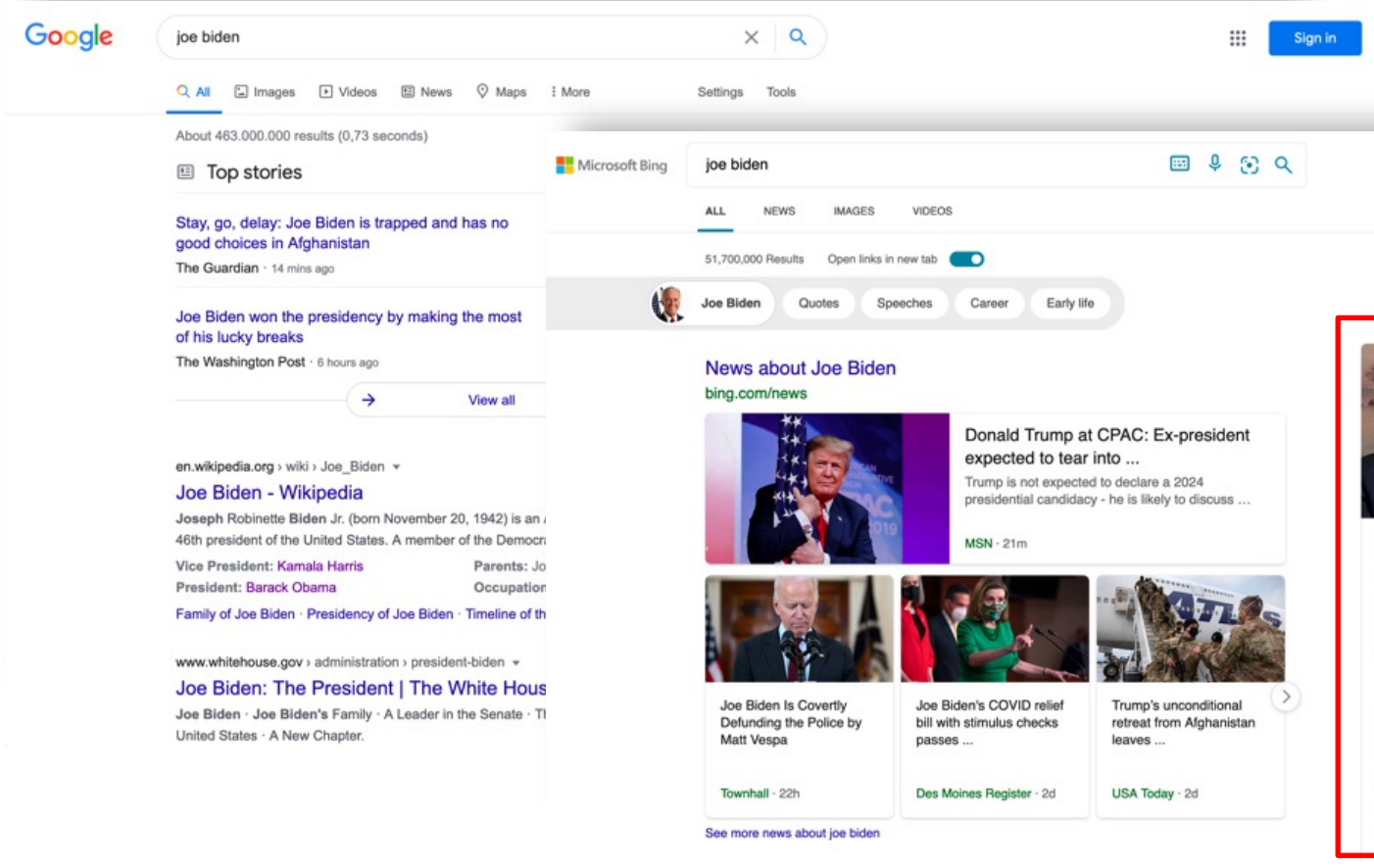
Scientific American, 2001

https://sit.instructure.com/courses/22632/files/2992212?module_item_id=462878

“A really important thing about data is: the more things you have to connect together, the more powerful it is.”

Sir Tim Berners-Lee on Linked Data at TED 2009

Search engines



Knowledge graphs in industry

> Knowledge graphs provide the structured data and factual knowledge that drive many products and make them more intelligent and "magical".

N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson, J. Taylor: Industry-scale knowledge graphs: lessons and challenges. Commun. ACM 62(8): 36-43 (2019)

Common characteristics of the knowledge graphs.

	Data model	Size of the graph	Development stage
Microsoft	The types of entities, relations, and attributes in the graph are defined in an ontology.	~2 billion primary entities, ~55 billion facts	Actively used in products
Google	Strongly typed entities, relations with domain and range inference	1 billion entities, 70 billion assertions	Actively used in products
Facebook	All of the attributes and relations are structured and strongly typed, and optionally indexed to enable efficient retrieval, search, and traversal.	~50 million primary entities, ~500 million assertions	Actively used in products
eBay	Entities and relation, well-structured and strongly typed	Expect around 100 million products, >1 billion triples	Early stages of development and deployment
IBM	Entities and relations with evidence information associated with them.	Various sizes. Proven on scales documents >100 million, relationships >5 billion, entities >100 million	Actively used in products and by clients

TU WIEN Informatics

Knowledge graph adoption

Open knowledge graphs

1.9B Facts

210M Facts

52M Facts

http://linkedlifedata.com/sources.html

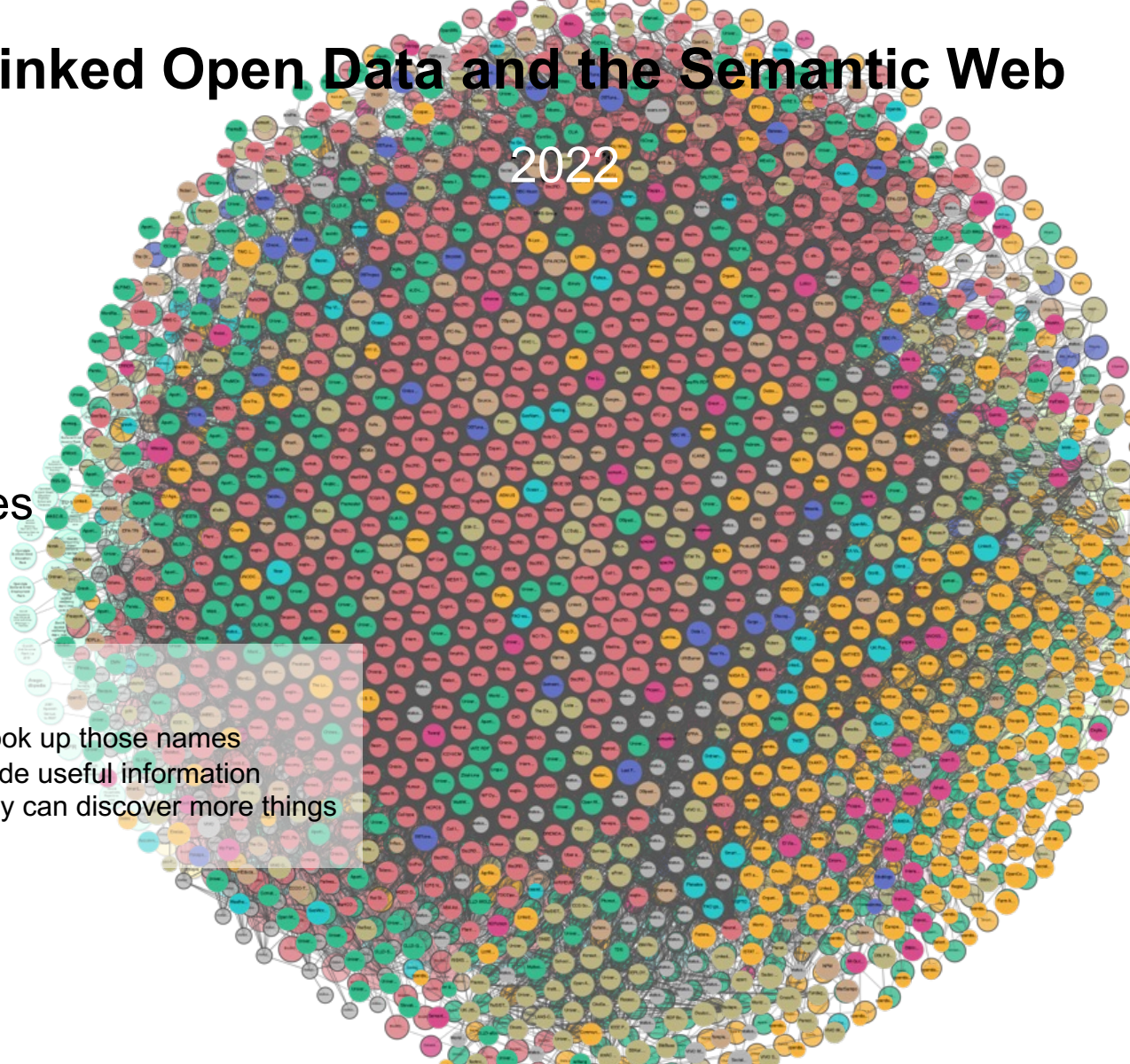
6.7B Facts

https://pubchem.ncbi.nlm.nih.gov/docs/rdf

132B Facts

TU WIEN Informatics

Linked Open Data and the Semantic Web



- Constantly growing
- Ca. 1255 datasets
- Ca. 400 billion triples

Linked Data principles

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names
3. When someone looks up a URI, provide useful information
4. Include links to other URIs so that they can discover more things

https://lod-cloud.net/

TU WIEN Informatics

Take back control of your data

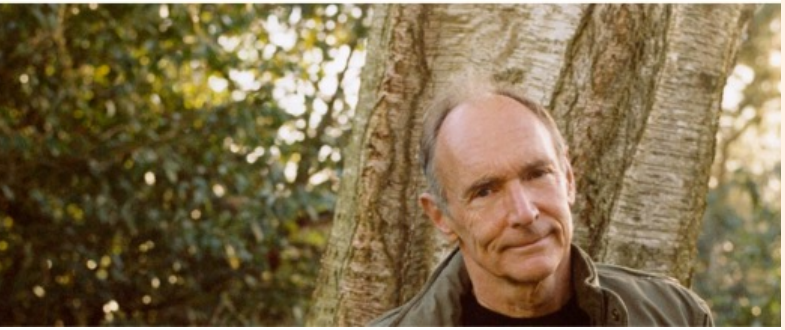
The New York Times

SUBSCRIBE NOW

Digital Privacy | How to Read Privacy Labels | How to Protect Your Digital Life | 10 Simple Tips | How Apps Collect Your Data

He Created the Web. Now He's Out to Remake the Digital World.

Tim Berners-Lee wants to put people in control of their personal data. He has technology and a start-up pursuing that goal. Can he succeed?



Tim Berners Lee's startup Inrupt releases Solid privacy platform for enterprises

Ron Miller @ron_miller / 6:01 AM GMT+1 • November 9, 2020

Comme

FINANCIAL TIMES

HOME WORLD US COMPANIES TECH MARKETS CLIMATE OPINION WORK & CAREERS LIFE & ARTS HOW TO SPEND IT

Technology + Add to myFT

NHS signs up for Tim Berners-Lee pilot to reinvent web

BBC, NatWest and government of Flanders also test new Solid servers from Inrupt

Save

Tim Berners-Lee: "There is a feeling, a zeitgeist, that change is really overdue" © AFP/Getty Images

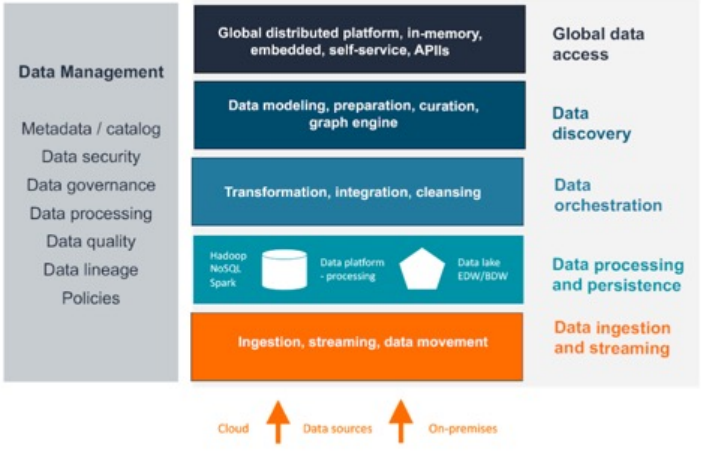
John Thornhill in London NOVEMBER 9 2020

118

Inrupt, the company founded by Tim Berners-Lee to promote a new data architecture for the web, has launched its first enterprise-ready Solid servers

TU WIEN Informatics

Data fabrics & semantic data lakes



A knowledge graph built on semantics and graph data models fits into the data orchestration and discovery layers in a data fabric.

CAMBRIDGE SEMANTICS

Data Fabrics: The Killer Use Case for Knowledge Graphs

Posted by Barbara Petrocelli on Mar 5, 2021 10:54:44 AM

Tweet | Share | Like | Share

This article explores the role of knowledge graphs in data fabrics. If you want to learn more about this topic, please join Forrester Analyst Noel Yuhanna and Cambridge Semantics Co-Founder Ben Szekely at an interactive discussion on this topic. Learn more here.

In a data fabric, knowledge graphs connect siloed data and simplify complicated data.

"A knowledge graph in a data fabric solves the data integration task and bridges the gap between disparate corporate datasets and the actual business requirements.

It allows business users to model, explore and discover how datasets are connected in a human readable model where there is little difference between the conceptual and the physical machine readable models."

Source: Demystifying the Data Fabric | 9.17.20 | Gartner Group

The Data Fabric is an architecture for modern data management that anticipates the need to connect data across the enterprise at speed and scale.

Knowledge graph is central to the success of any Data Fabric as an agile integration overlay that blends complex and diverse data into easy-to-consume data products for end-users, data scientists, and applications.

Neurosymbolic AI

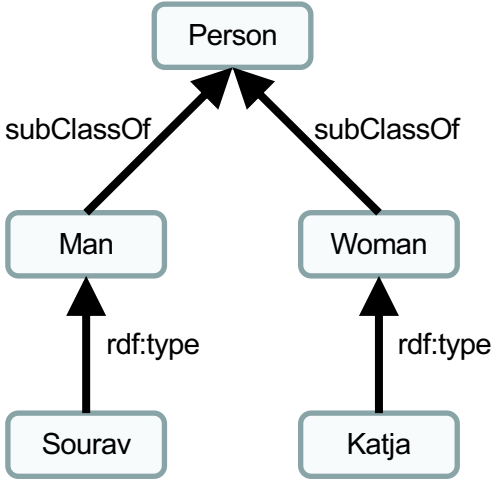
Pure Logic Reasoning

Pure Learning

Neurosymbolic AI

Pure Logic Reasoning

Pure Learning



(Katja, rdf:type, Woman) ∧ (Woman, subClassOf, Person)
→ (Katja, rdf:type, Person)

Neurosymbolic AI

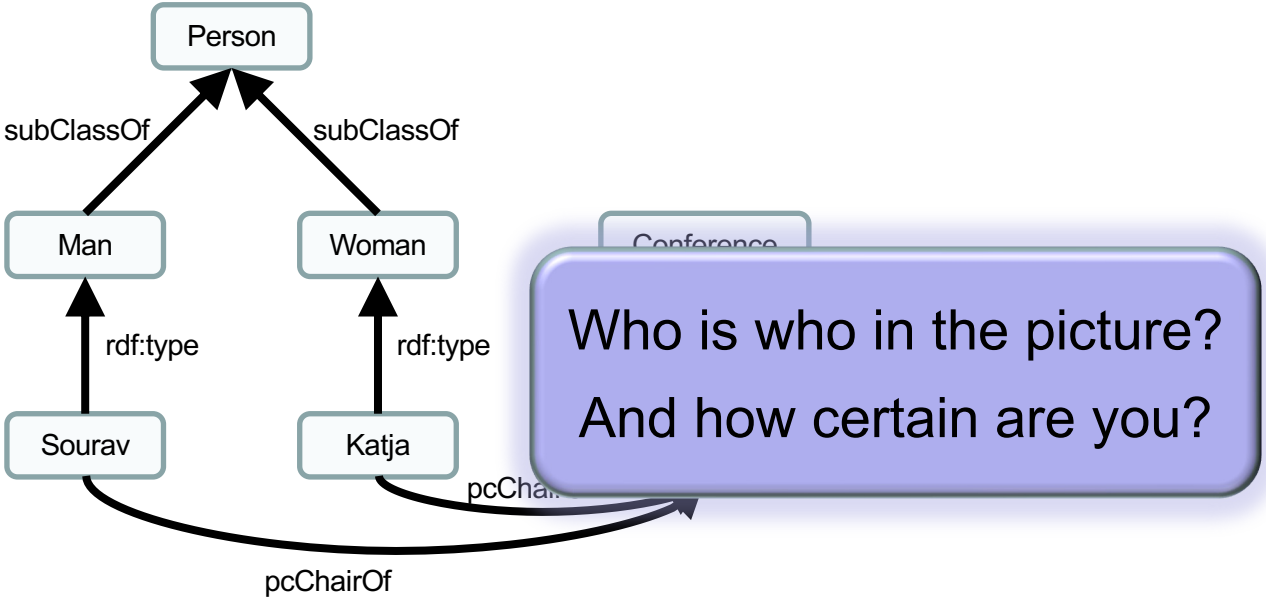
Pure Logic Reasoning

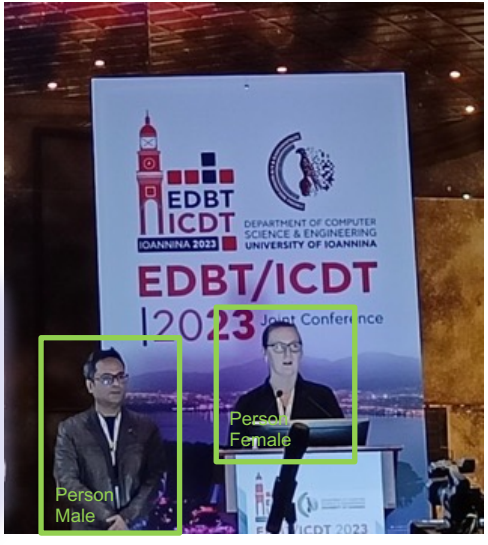
Pure Learning

PC chairs opening EDBT 2023

role conference

Neurosymbolic AI





EDBT 2023 has two PC chairs*
These are Katja and Sourav
Katja is a Woman, Sourav is a Man
Men and Women are Persons

The PC chairs of EDBT 2023 are (likely) on the picture
One is (likely) a male person, the other one (likely) female

* Partial Completeness Assumption (PCA)
Galárraga, Teflioudi, Hose, Suchanek. Fast Rule Mining in Ontological Knowledge Bases with AMIE+, VLDBJ 2015

Data and Knowledge Engineering

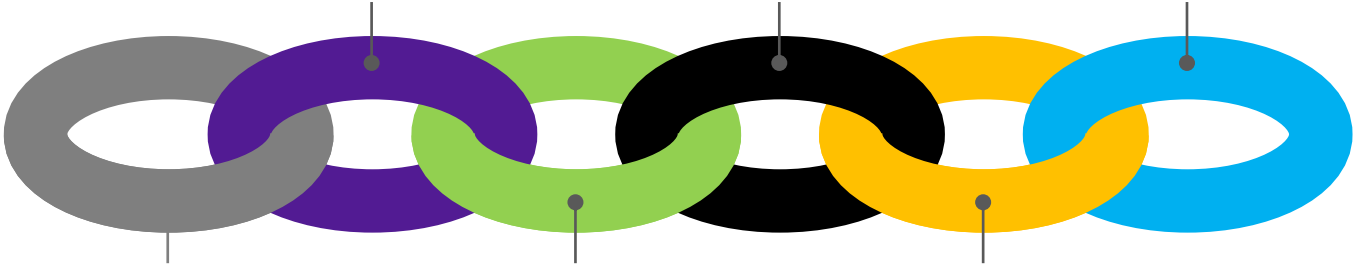
TU WIEN Informatics

Knowledge Graph Value Chain

Data Management

Data Analytics

Machine Learning

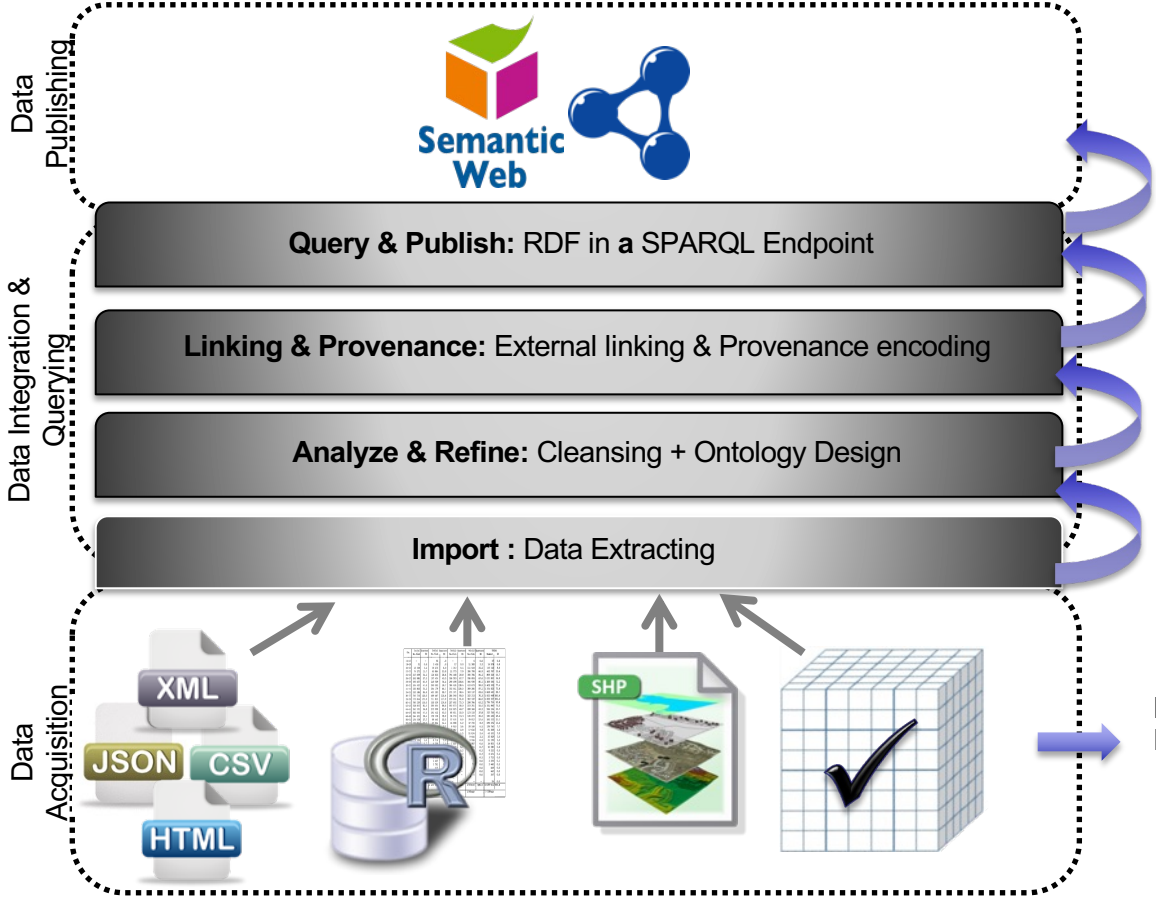


Extraction and Integration

Query Processing and Optimization

Provenance, Evolution, Metadata, and Privacy

Semantic data warehousing



Task: Integrate data and updates into a knowledge graph & enable efficient querying and analytics

Challenges: Statistical knowledge graphs require novel data modeling, management, and query optimization techniques

Existing standards: RDF Cube (QB), QB4OLAP, Provenance (PROV-O), RDF-star

Multidimensional Data Cubes

Nath, Romero, Pedersen, Hose
High-Level ETL for Semantic Data Warehouses. In: SWJ 2021

Gür, Pedersen, Hose, Midtgaard
Multidimensional Enrichment of Spatial RDF Data for SOLAP. In: SWJ 2021

Provenance and Explainability

Ok Google, what's the capital of Denmark?

The capital of Denmark is Aalborg.

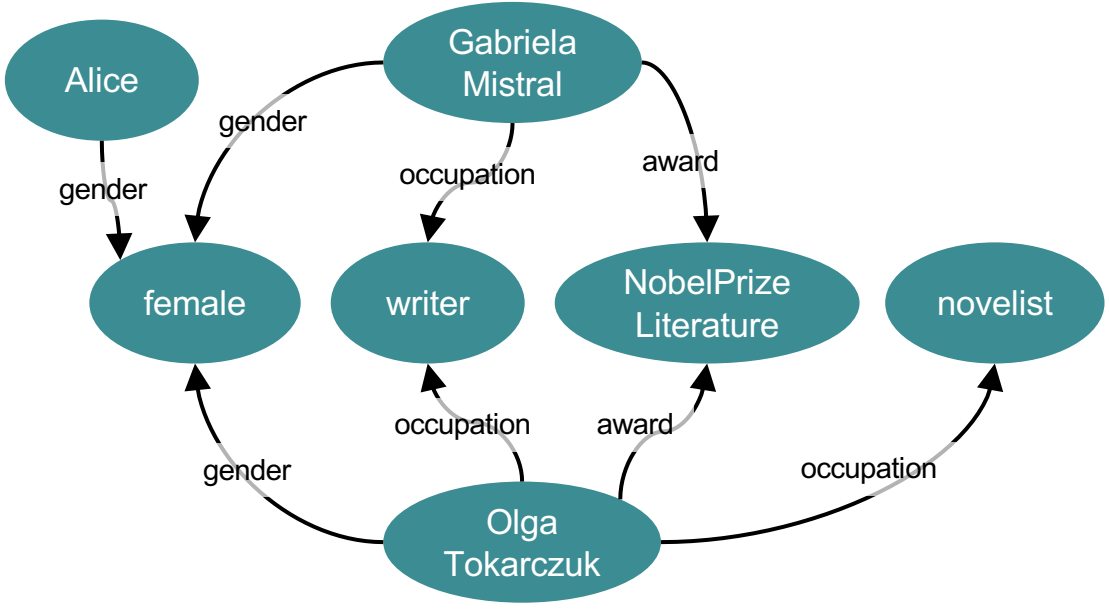
Is there a war in Ukraine?

No, there's a special military operation though.

???

Provenance and Explainability

Example Knowledge Graph

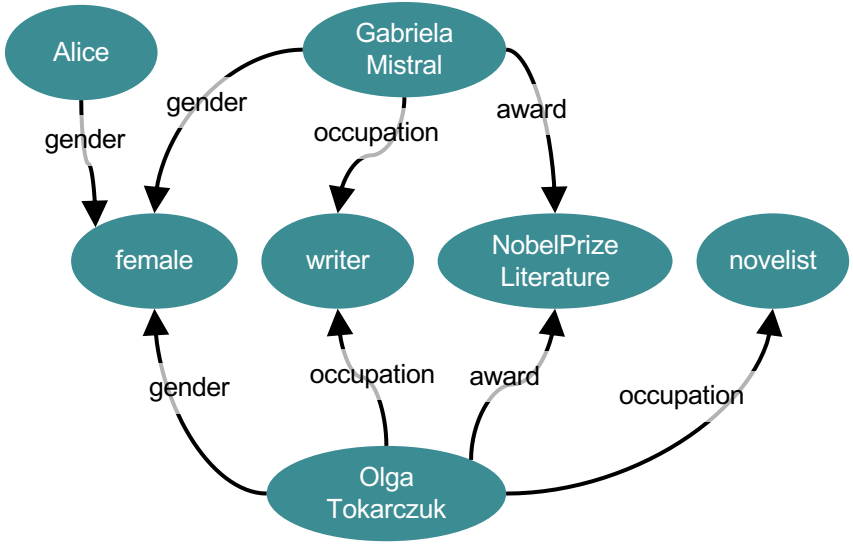


Daniel Hernández, Luis Galárraga, Katja Hose
Computing How-Provenance for SPARQL Queries via Query Rewriting
In: Proc. VLDB Endow. 14(13), 2021, pp. 3389-3401

TU WIEN Informatics

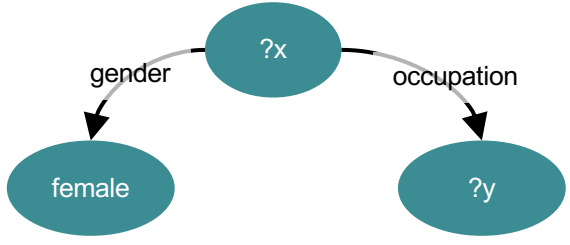
How-provenance: How are query answers generated?

Example Query



Which women have occupations?

```
SELECT ?x
WHERE {
  ?x gender ‘female‘ .
  ?x occupation ?y
}
```



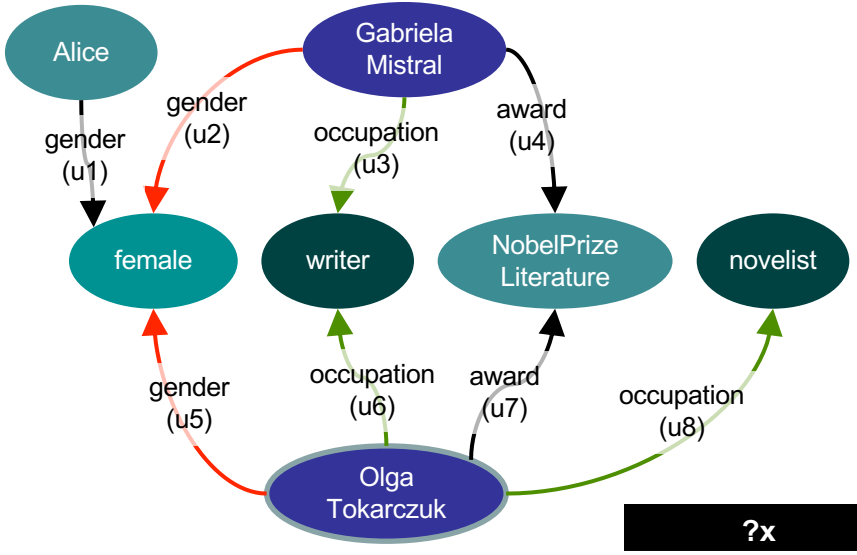
Answers

?x
GabrielaMistral
OlgaTokarczuk

Daniel Hernández, Luis Galárraga, Katja Hose
Computing How-Provenance for SPARQL Queries via Query Rewriting
In: Proc. VLDB Endow. 14(13), 2021, pp. 3389-3401

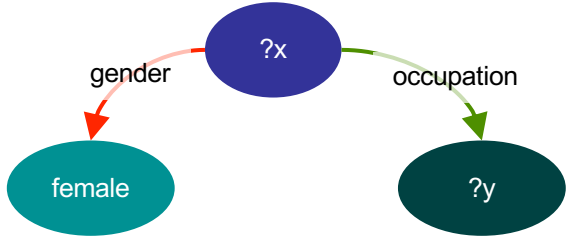
How-provenance: How are query answers generated?

Adding a unique identifier per edge (triple) enables us to track provenance.



Which women have occupations?

```
SELECT ?x
WHERE {
  ?x gender ‘female‘ .
  ?x occupation ?y
}
```



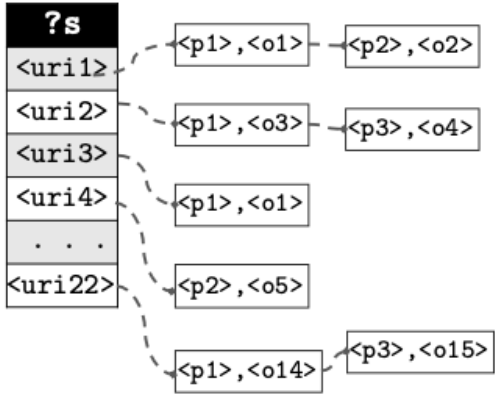
Answers

?x	?x	nce
GabrielaMistral	GabrielaMistral	ı3
OlgaTokarczuk	OlgaTokarczuk	u5 ⊗ u8)

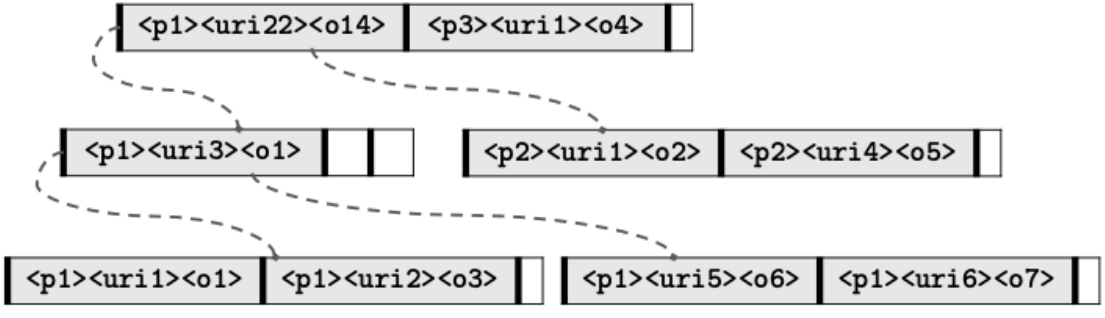
Daniel Hernández, Luis Galárraga, Katja Hose
Computing How-Provenance for SPARQL Queries via Query Rewriting
In: Proc. VLDB Endow. 14(13), 2021, pp. 3389-3401

Storage optimization

?s	?p	?o
<uri1>	<p1>	<o1>
<uri1>	<p2>	<o2>
<uri2>	<p1>	<o3>
<uri2>	<p3>	<o4>
<uri3>	<p1>	<o1>
<uri4>	<p2>	<o5>
. . .	. . .	. . .
<uri22>	<p1>	<o14>
<uri22>	<p3>	<o15>



?s	<p1>	<p2>	<p3>	...
<uri1>	<o1>	<o2>	-	...
<uri2>	<o3>	-	<o4>	...
<uri3>	<o1>	-	-	...
<uri4>	-	<o5>	-	...
...	...	...	...	...
<uri22>	<o14>	-	<o15>	...



Sagi, Lissandrini, Pedersen, Hose:
A Design Space for RDF Data Representation Systems.
In: VLDBJ 2022

Knowledge graph management and querying

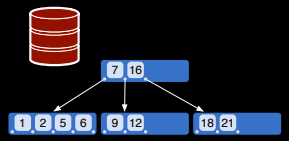
KG Management

dump

centralized

Tentris [Bigerl et al. 2020]
Tunable-LSH [Aluç et al. 2019]
MARVEL [Ibragimov et al. 2016]
RDFox [Nenov et al. 2015]
chameleon-db [Aluç…2013]
TripleBit [Yuan et al. 2013]
gStore [Zou et al. 2011]
dipLODocusRDF [Wylot…2011]
BitMat [Atre et al. 2009]
…

runtime



client/server

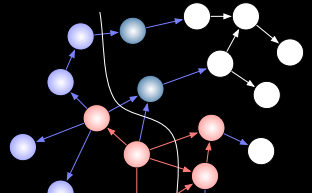
WiseKG [Azzam et al. 2021]
SPF [Aebeloe et al. 2020]
SmartKG [Azzam et al. 2020]
SaGe [Minier et al 2019]
brTPF [Hartig et al. 2016]
TPF [Verborgh et al. 2016]
…

throughput

distributed/parallel

DP2RPQ [Wang et al. 2020]
gStore-D2 [Peng et al. 2019]
WORQ [Madkour et al. 2018]
S2RDF [Wu et al. 2017]
DiploCloud [Wylot et al. 2016]
CliqueSquare [Goasdoué…2015]
Partout [Galárraga et al. 2014]
WARP [Hose et al. 2013]
TriAD [Gurajada et al. 2014]
H2RDF [Papailiou et al. 2013]
SHAPE [Lee et al. 2013]
EAGRE [Zhang et al. 2013]
Trinity.RDF [Zeng et al. 2013]
H-RDF-3X [Huang et al. 2011]
HadoopRDF [Husain et al. 2011]
SHARD [Rohloff et al. 2011]
…

runtime



federated

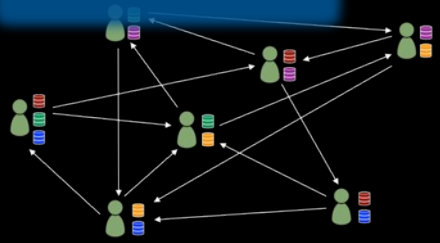
MultiQuery [Peng et al. 2021]
Odyssey [Montoya et al 2017]
Lusail [Mansour et al. 2017]
SemaGrow [Charalambidis…2015]
CoDA [Ibragimov et al. 2015]
HiBiSCuS [Saleem et al.2014]
FedX [Schwarte et al. 2011]
SPLENDID [Görlitz et al. 2011]
Anapsid [Acosta et al.2011|
DARQ [Quilitz et al. 2008]
…

runtime

P2P

Lothbrok (Aebeloe et al. 2023)
ColChain [Aebeloe et al. 2021]
PIQNIC [Aebeloe et al. 2019]
Cyclades [Folz et al. 2016]
RDFpeers [Cai et al. 2014]
ATLAS [Kaoudi et al.2010]
Unistore [Karnstedt et al.2007]
Bibster [Haase et al. 2004]
InfoQuilt [Arumugam…2002]
…

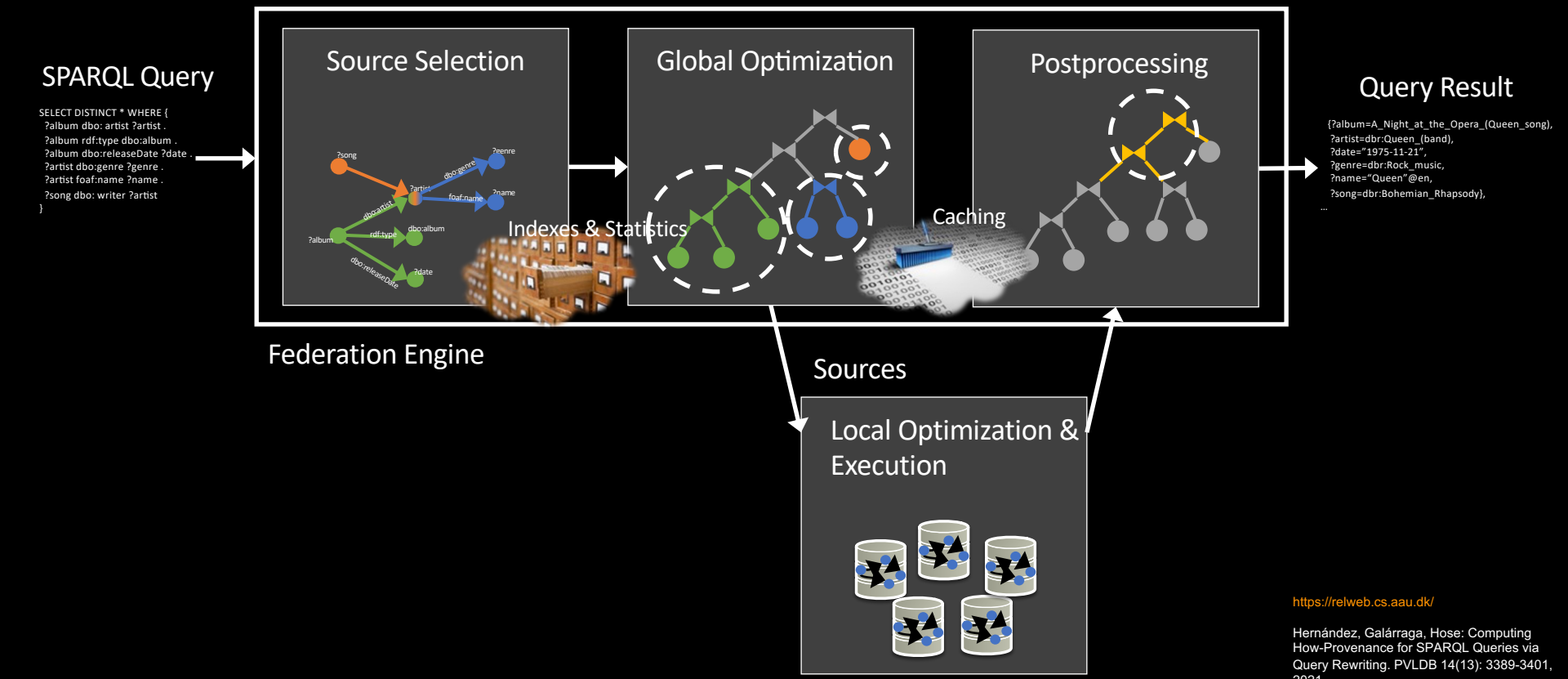
availability



streams

…

Federated architectures

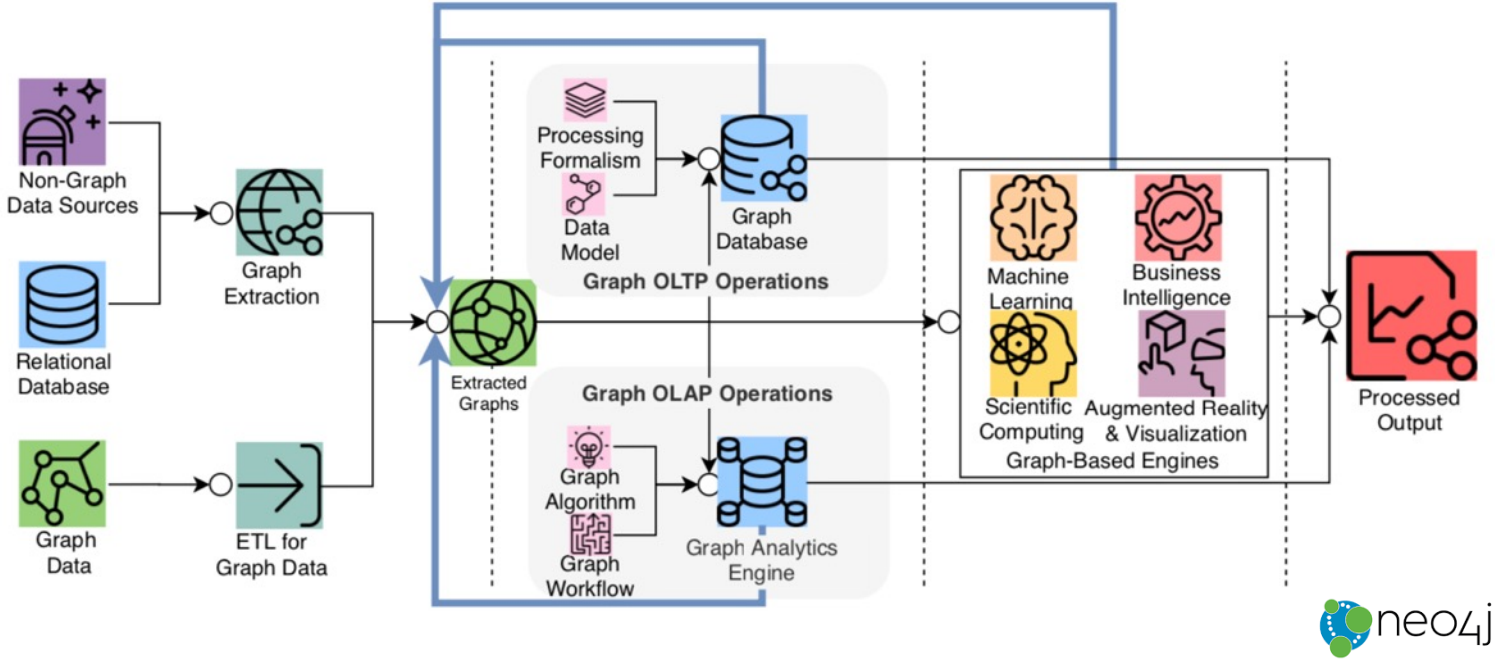


https://relweb.cs.aau.dk/

Hernández, Galárraga, Hose: Computing How-Provenance for SPARQL Queries via Query Rewriting. PVLDB 14(13): 3389-3401, 2021

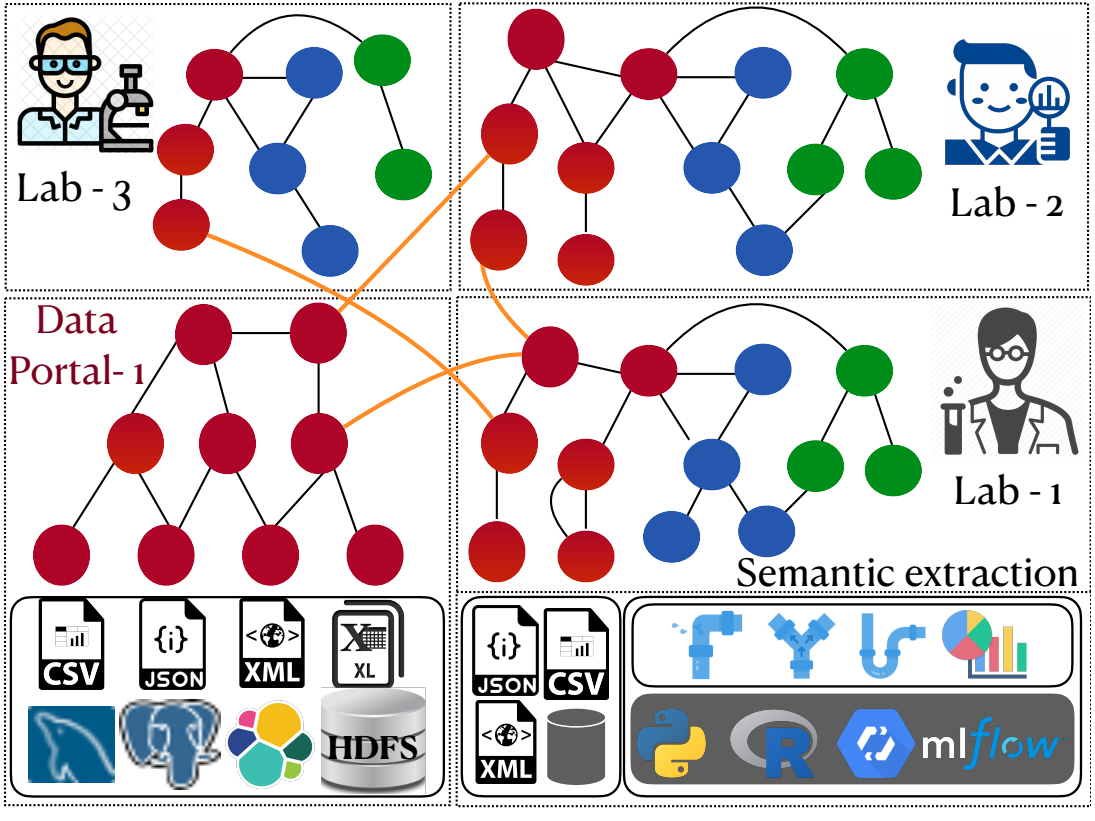
Azzam, Aebeloe, Montoya, Keles, Polleres, Hose: WiseKG: Balanced Access to Web Knowledge Graphs. TheWebConf, 1422-1434, 2021

Graph frameworks and ecosystems



Sakr et al.: The Future is Big Graphs!
A Community View on Graph Processing Systems.
In: CACM 2021

Federated Data Science



Mansour, Srinivas, Hose.
Federated Data Science to Break Down Silos. In: SIGMOD Rec. 2021

Use Cases

Use Cases

Manufacturing and Production

Machine Learning

Sustainability and Circular Economy

Medicine and Bio Science

Media and Entertainment

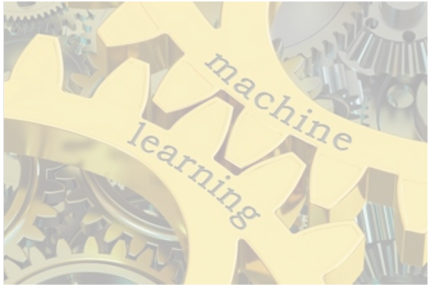
Finance and Economics

Transportation

Use Cases

Manufacturing and Production

Machine Learning



Sustainability and Circular Economy

Medicine and Bio Science

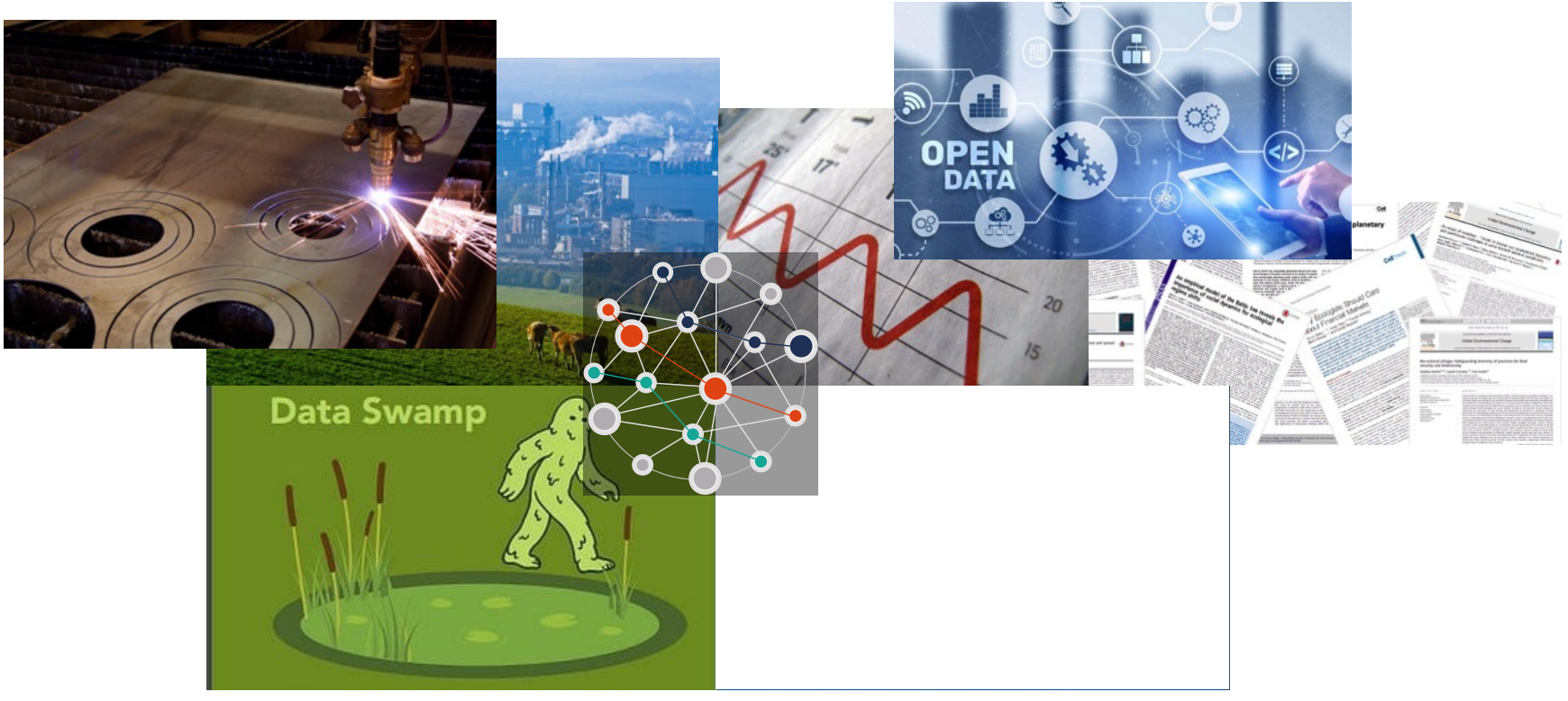
Media and Entertainment

Finance and Economics

Transportation

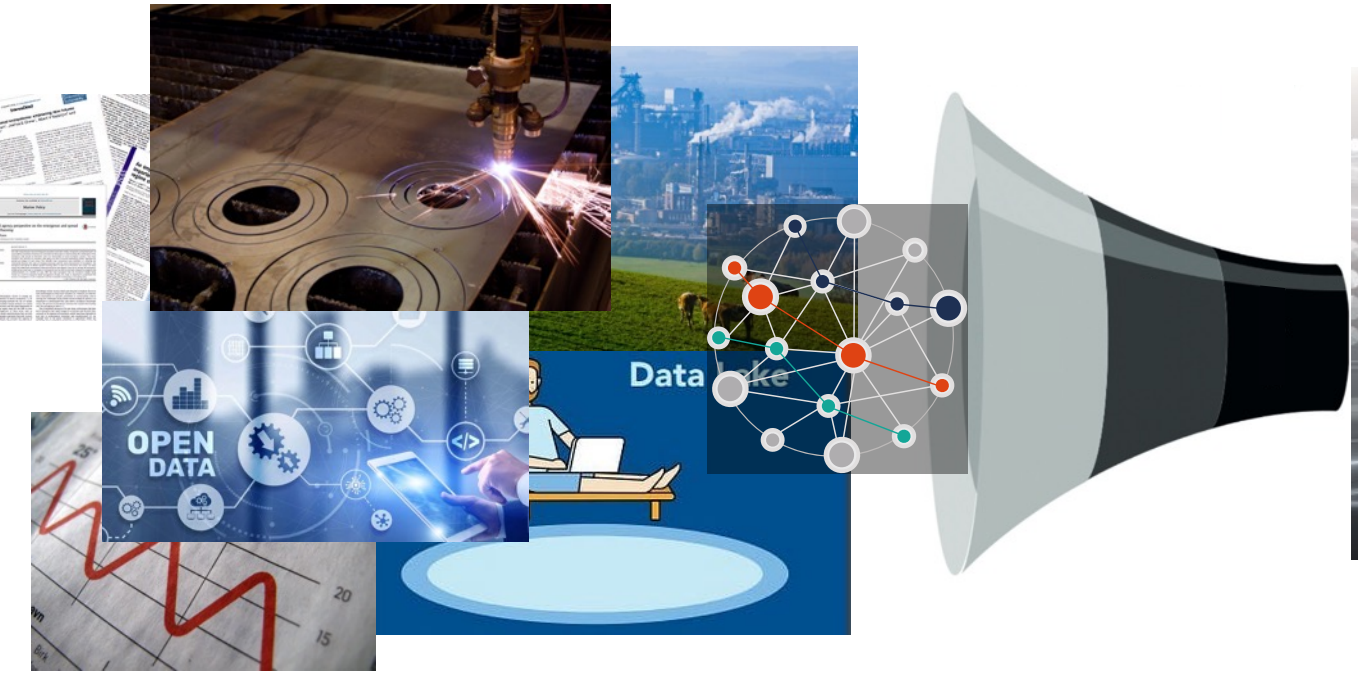
TU WIEN Informatics

Data Integration & Contextual AI for sustainable production

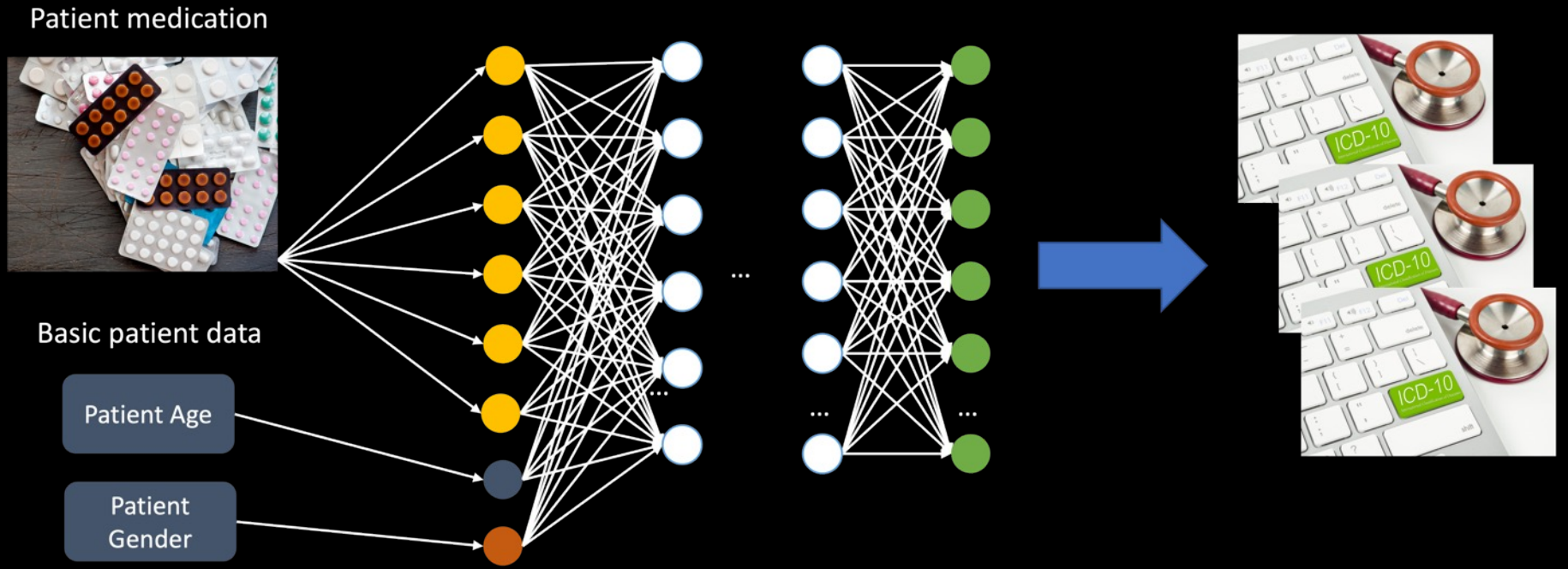


TU WIEN Informatics

Data Integration & Contextual AI for sustainable production



Patient data and electronic health records

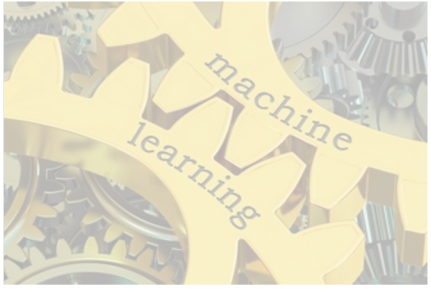


E. R. Hansen, T. Sagi, K. Hose, G. Y. H. Lip, T. B. Larsen, F. Skjøth: Assigning diagnosis codes using medication history. Artificial Intelligence in Medicine, 2022

Use Cases

Manufacturing and Production

Machine Learning



Sustainability and Circular Economy

Medicine and Bio Science

Media and Entertainment

Finance and Economics

Transportation

Digitally supported environmental assessments

https://dreamsproject.dk

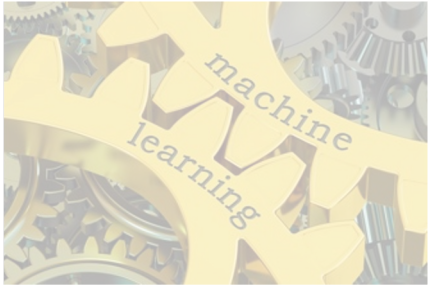
K. Hose, J. Bjerva: Fra korrelation til kausalitet: ML, NLP og relation extraction skal forstå miljørapporter. In: ING/DATATECH Synspunkt, 2021

TU WIEN Informatics

Use Cases

Manufacturing and Production

Machine Learning



Sustainability and Circular Economy

Medicine and Bio Science

Media and Entertainment

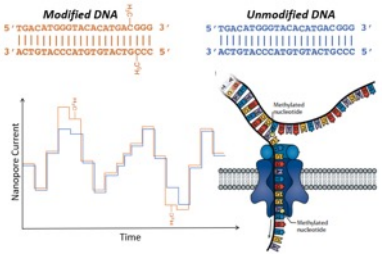
Finance and Economics

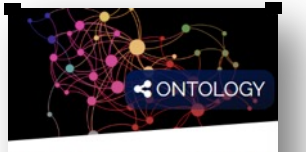
Transportation

TU WIEN Informatics

Integrating Heterogeneous Data

Genome recovery



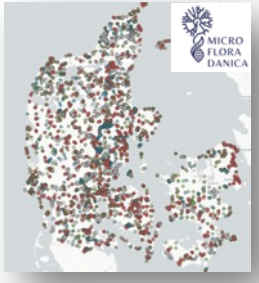


The network of biological classes describing the current best representation of the "universe" of biology: the molecular functions, cellular locations, and processes gene products may carry out.

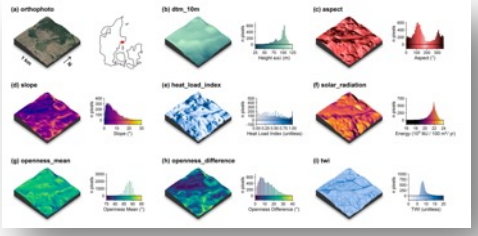
Gene function

MICRO FLORA DANICA

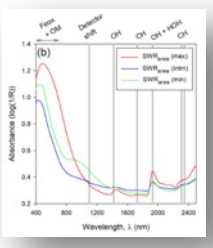
Metadata



Satellite & drone imaging



Soil properties

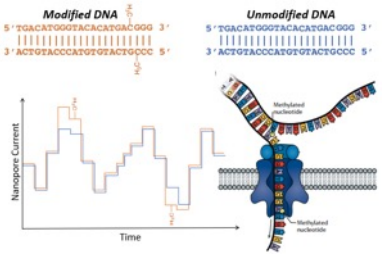


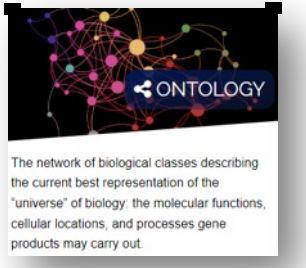
https://darkmatter.aau.dk/

TU WIEN Informatics

Integrating Heterogeneous Data

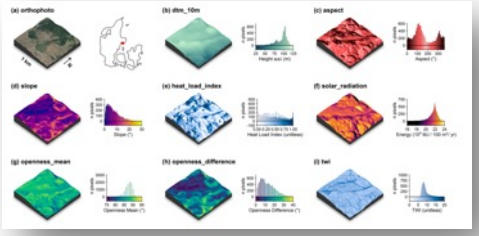
Genome recovery





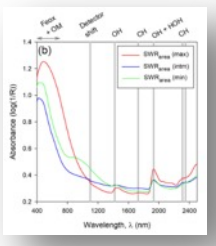
Gene function

Satellite & drone imaging



MICRO FLORA DANICA

Soil properties



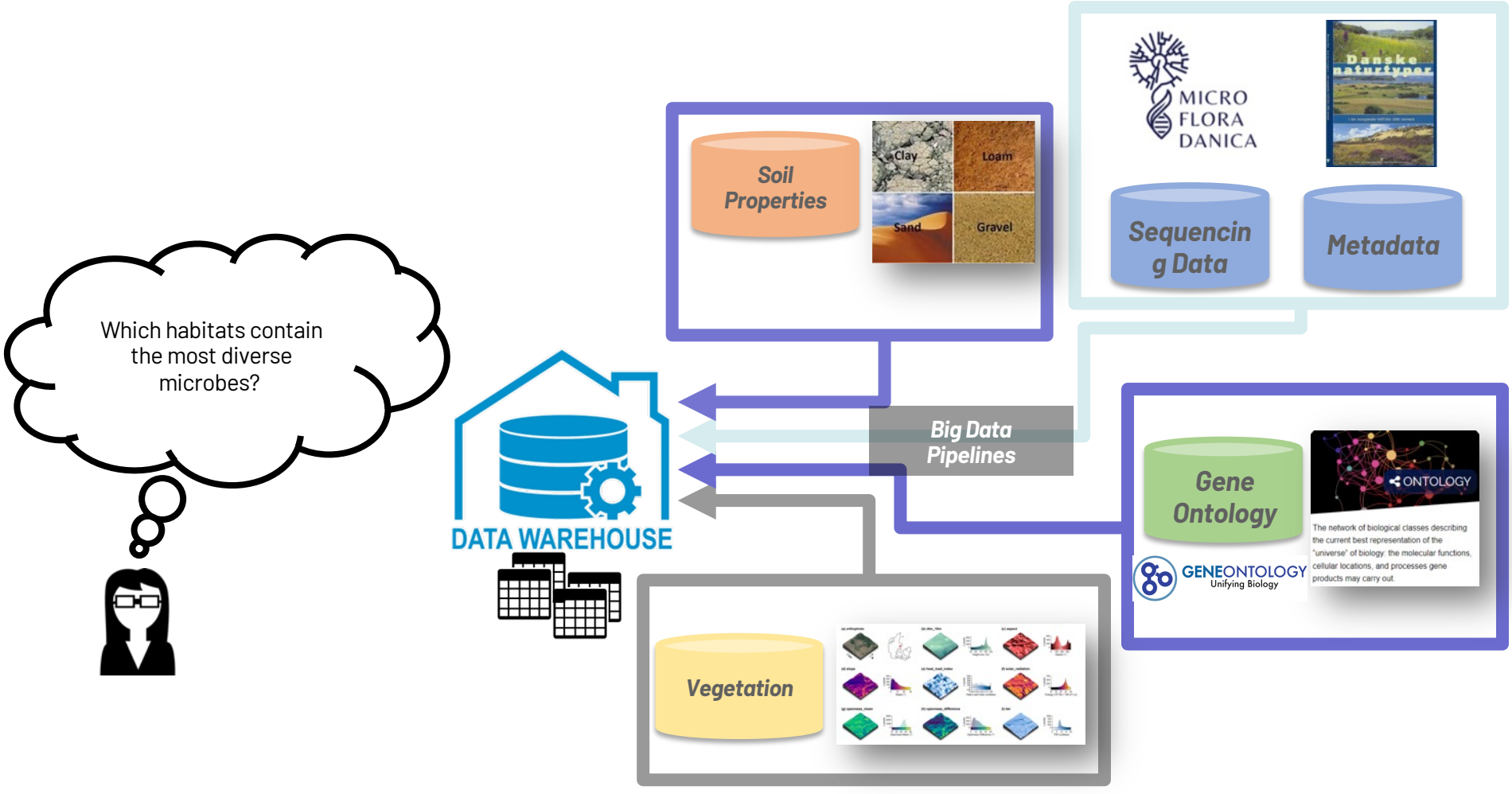
Metadata

External ontologies,
legacy databases,
Open Data,
FAIR data, etc.

https://darkmatter.aau.dk/

TU WIEN Informatics

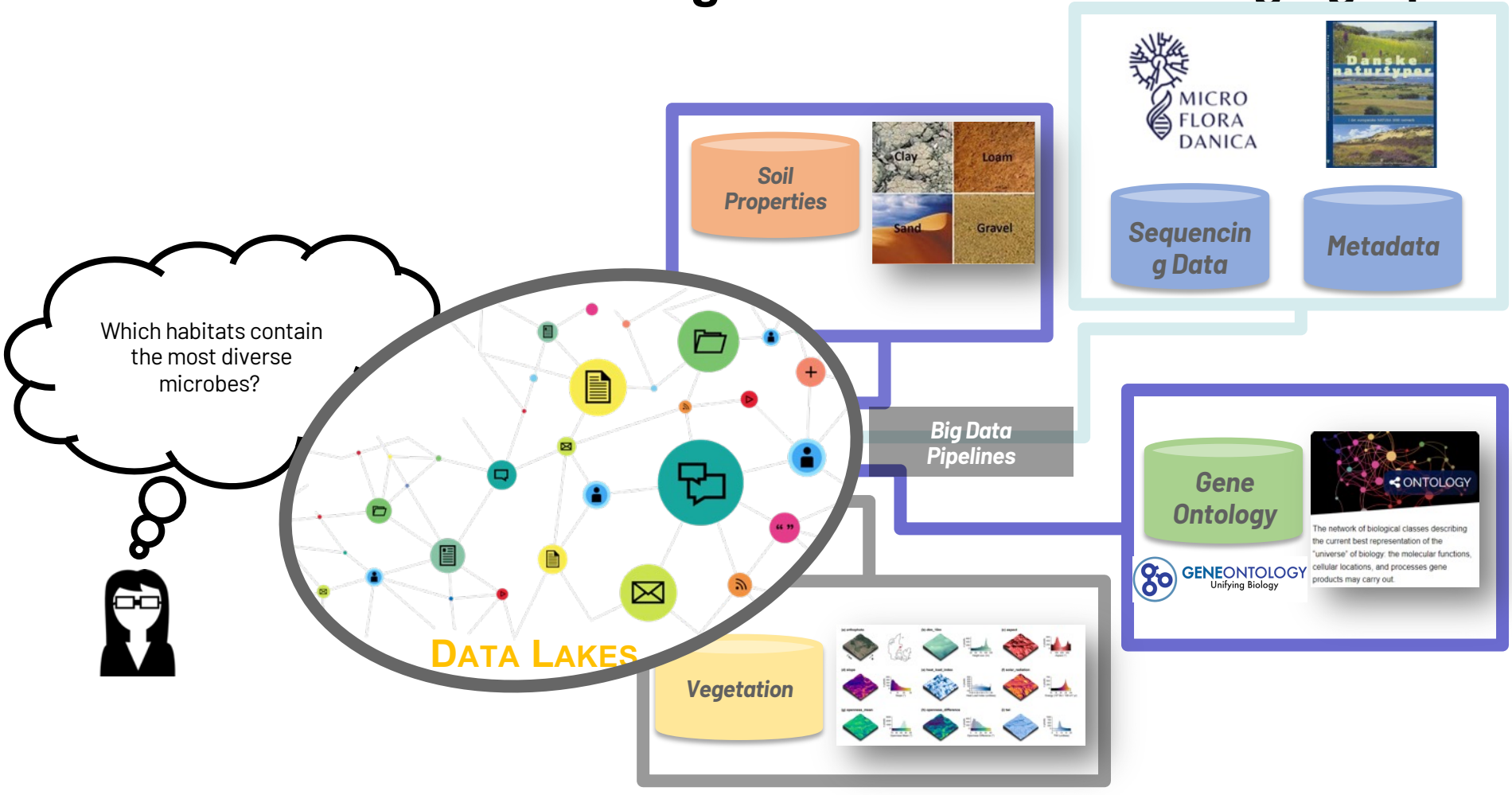
Traditional approach: data warehouses



https://darkmatter.aau.dk/

TU WIEN Informatics

New approach: data lakes and data fabrics integrate data in a knowledge graph



https://darkmatter.aau.dk/

New approach: data lakes and data fabrics integrate data in a knowledge graph

https://darkmatter.aau.dk/

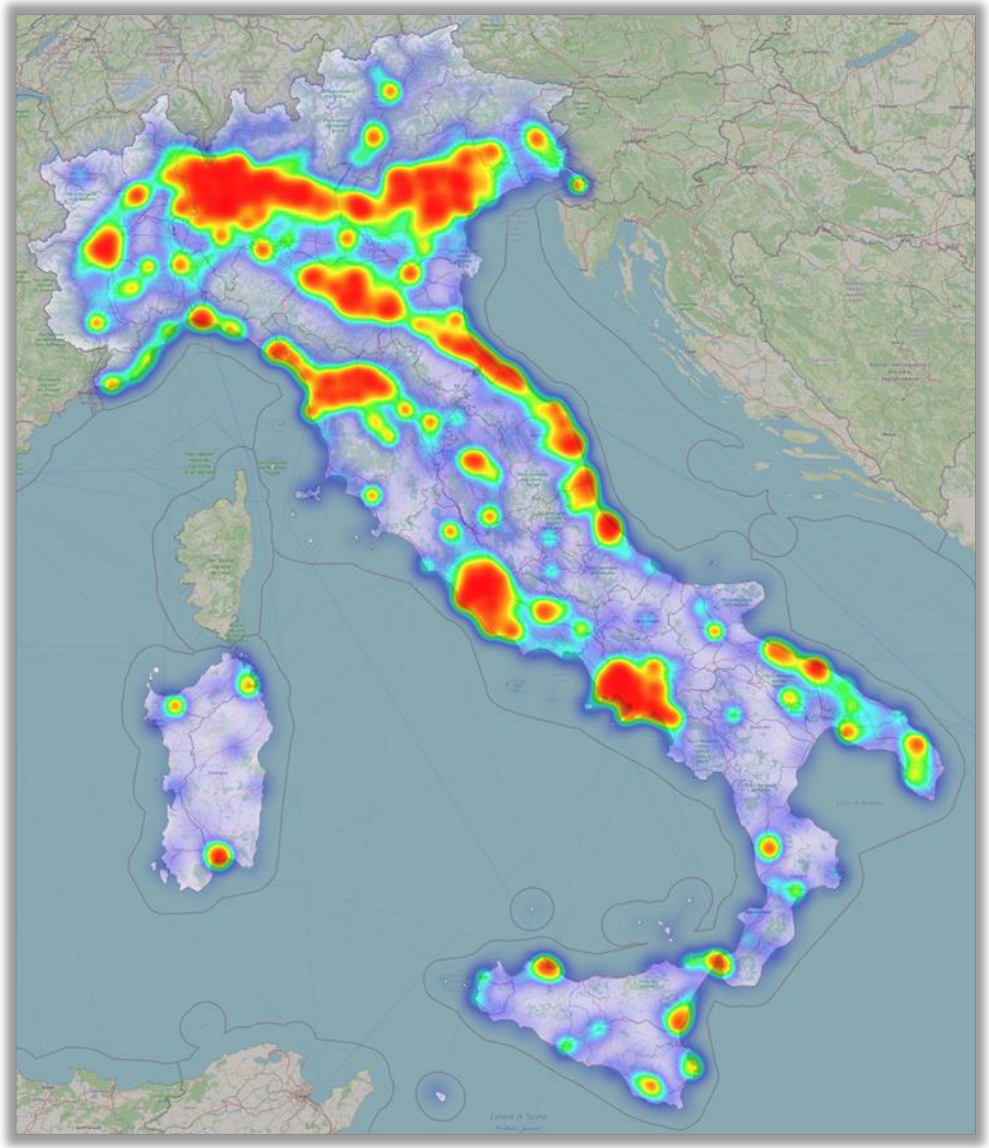
Knowledge Graphs

Real-World Application Teaser

Teaser

Emanuel Sallinger

Economic Impact of Crises



Hostile Takeovers

REUTERS

Business Markets World Politics TV More

BUSINESS NEWS MARCH 26, 2020 / 11:36 AM / 2 MONTHS AGO

EU leaders to shield strategic firms from hostile interest amid crisis

Francesco Guarascio, Gabriela Baczynska 4 MIN READ

BRUSSELS (Reuters) - European Union leaders will on Thursd
healthcare, infrastructure and other firms seen as having strate
foreign takeovers, draft EU summit conclusions show.

Anti-Money Laundering

FINANCIAL TIMES

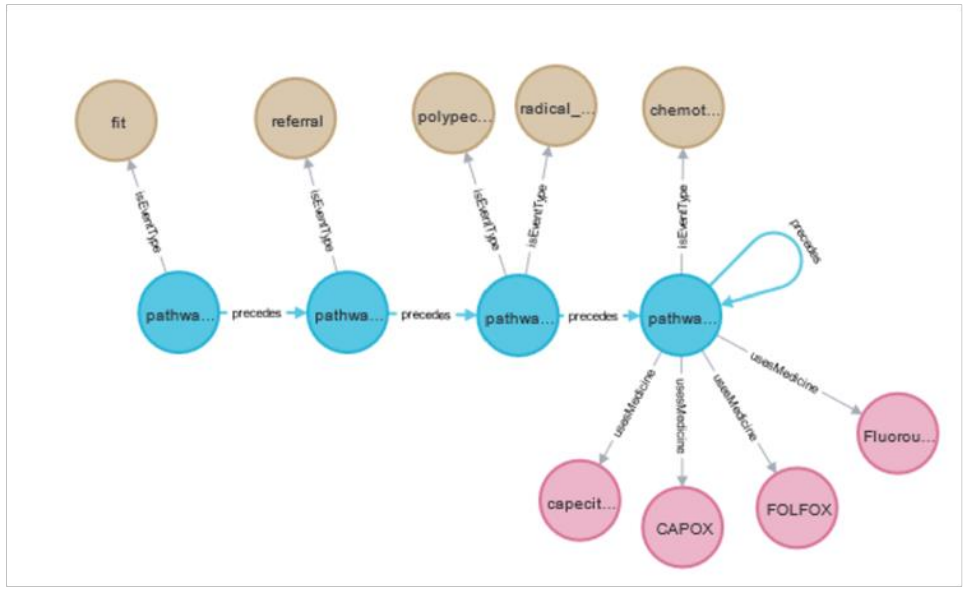
Danske Bank AS + Add to myFT

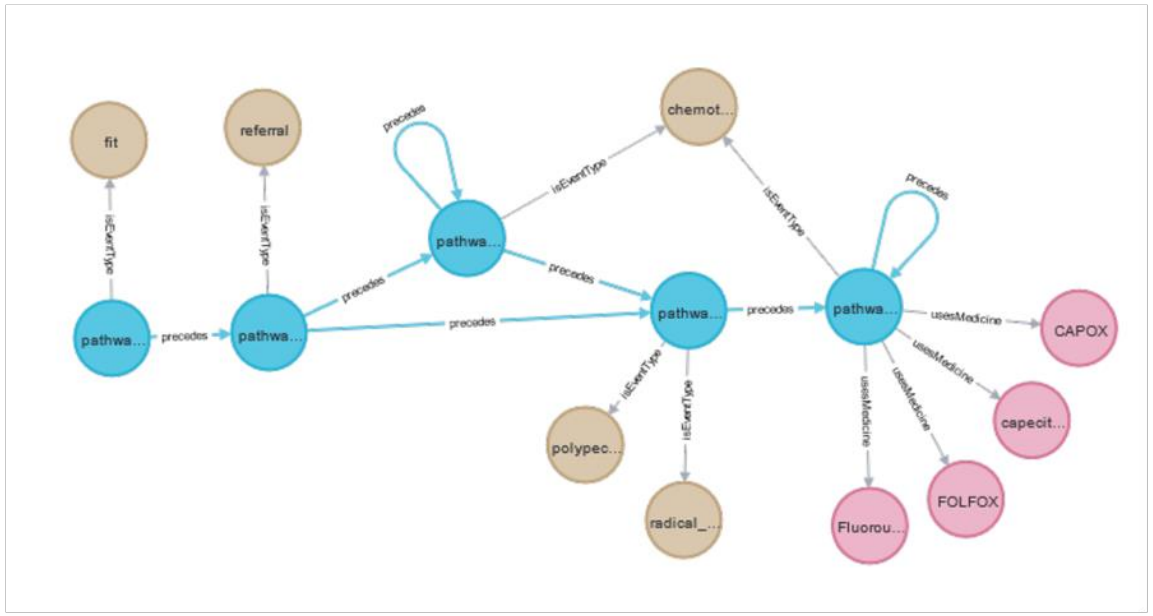
Danske Bank chairman ousted by main shareholder after scandal

Maersk family brings in new blood to stabilise lender in wake of €200bn money laundering

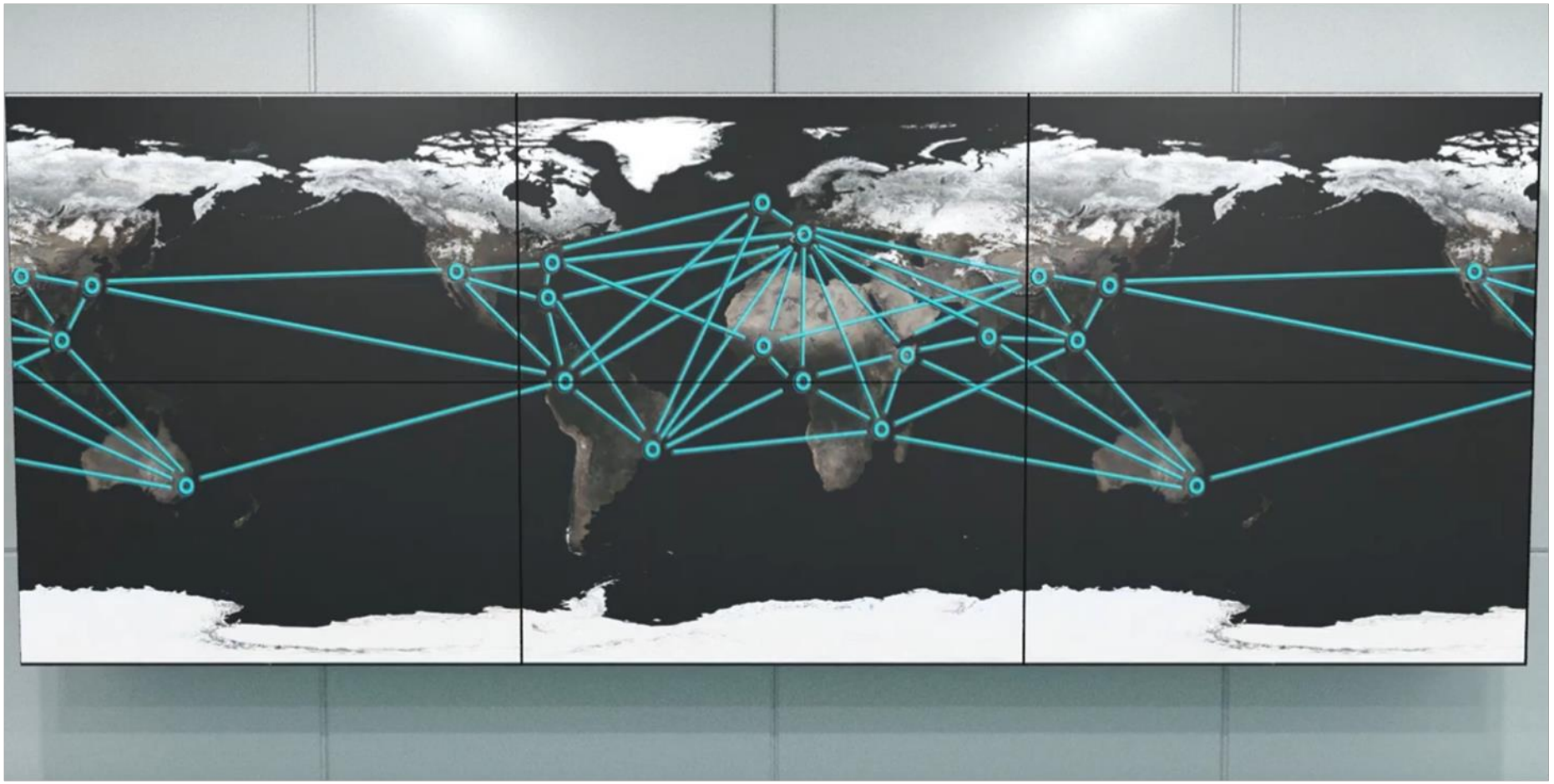
€200bn money laundering

Cancer Pathways





Supply Chains



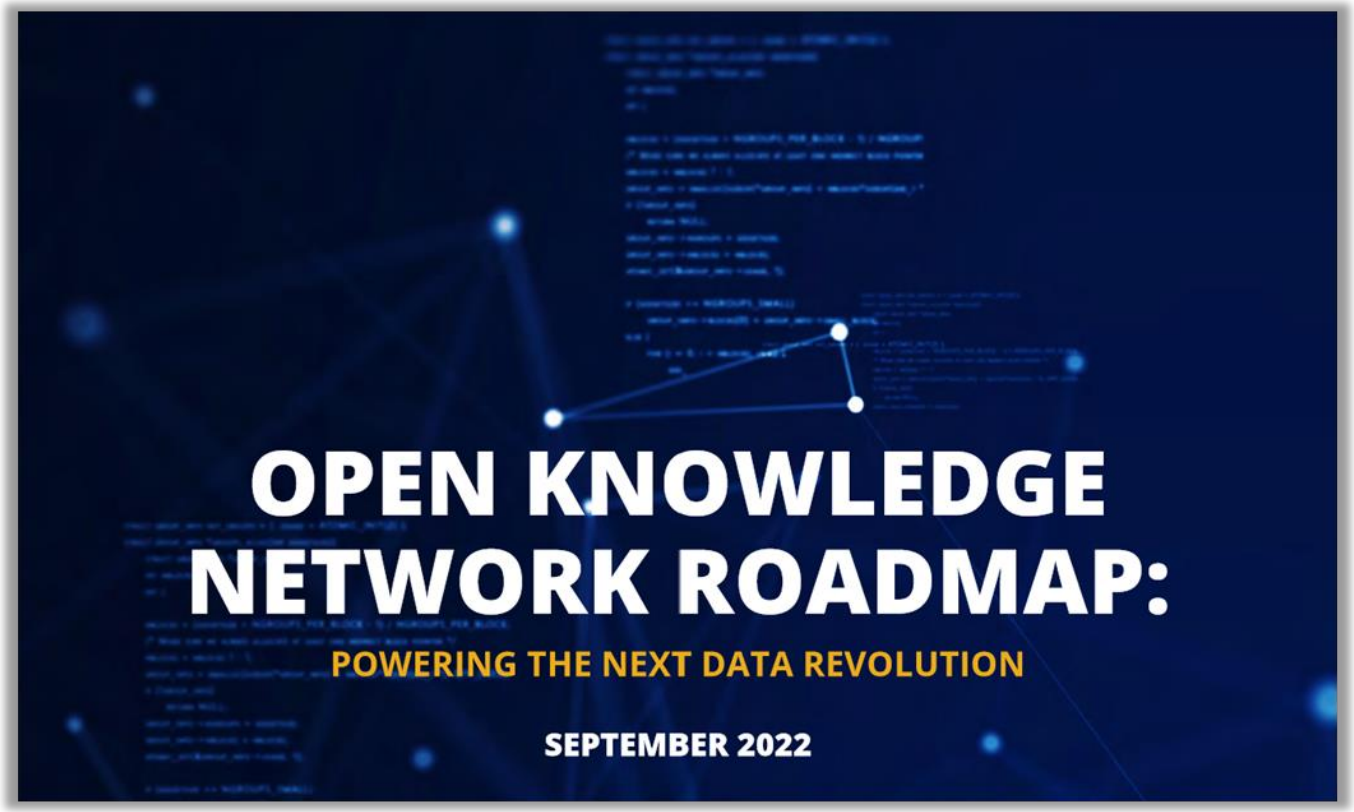
Media Intelligence

Meltwater

UNIVERSITY OF OXFORD

TU WIEN

Open Knowledge Network
Innovation Sprint



Baby Formula Supply Chain Issue

Individual

Supply Chain
+ Finance

Image source: https://www.newsweek.com/

PPP / US CARES Effectiveness

Small Business

COVID Impact
+ Finance

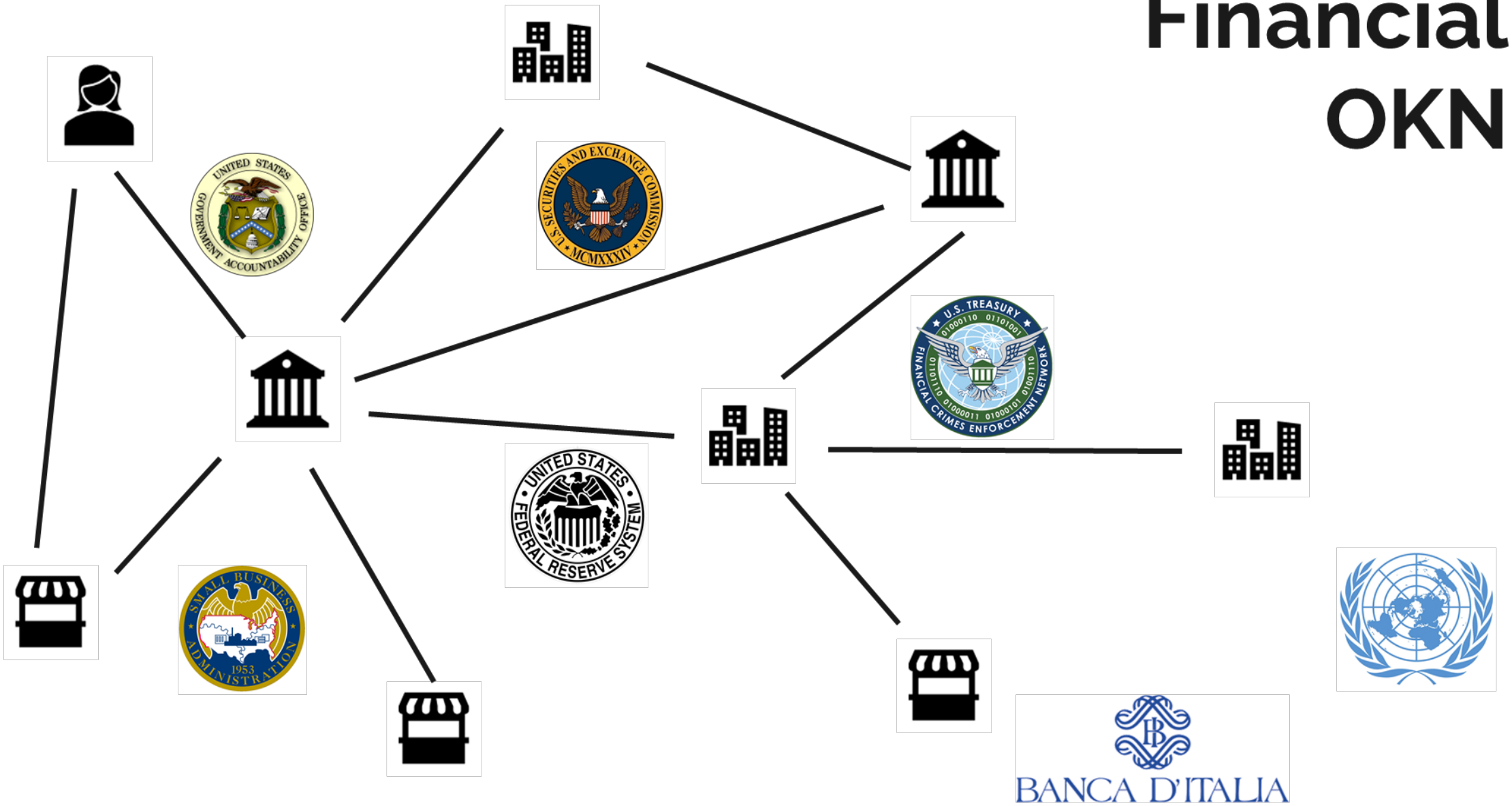
Image source: https://shutterstock.com/

Climate Finance

Environmental Models
+ Finance

Image source: https://www.lse.ac.uk/

Financial OKN



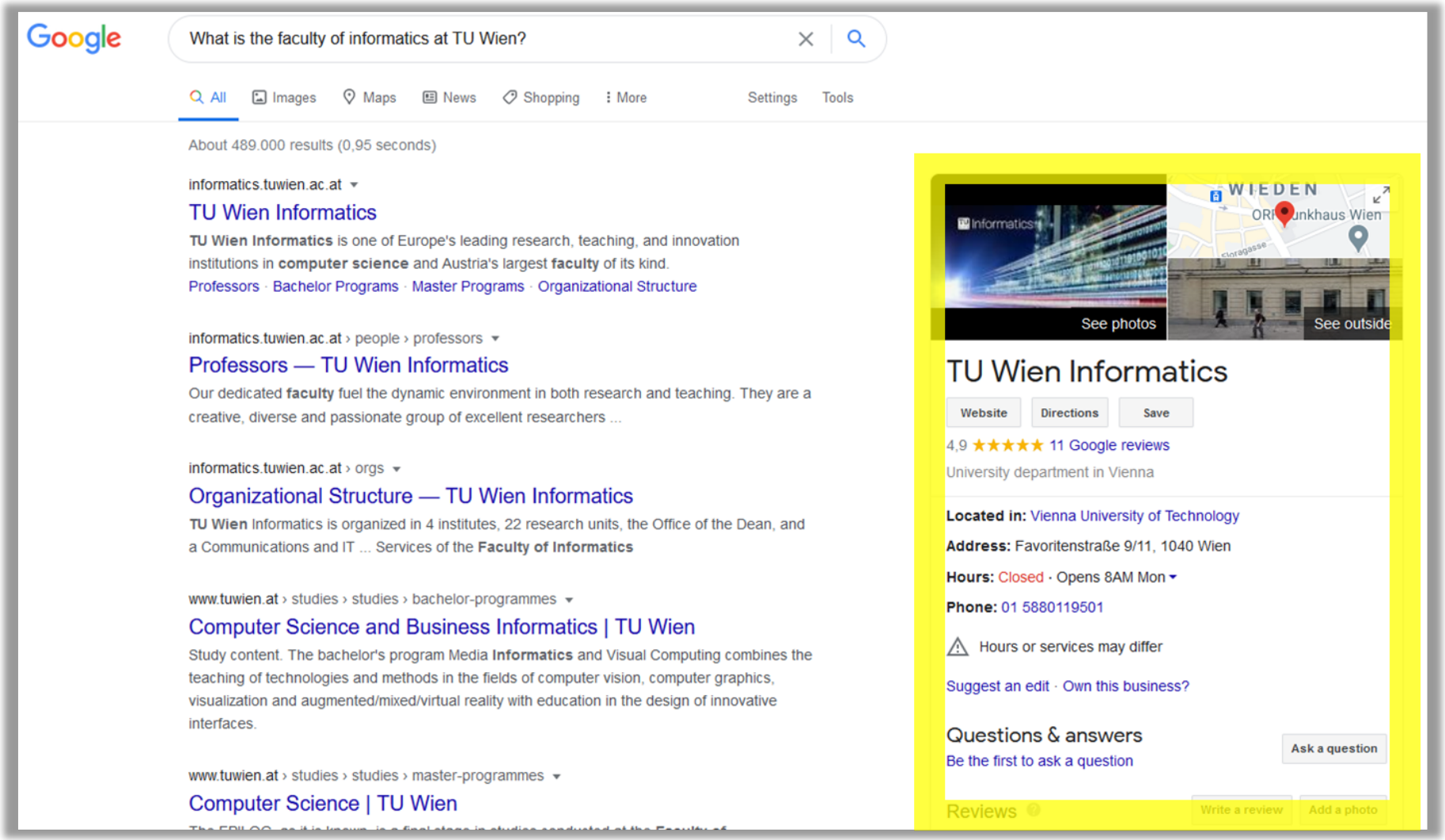
Course Teaser

"192.116 Knowledge Graphs"

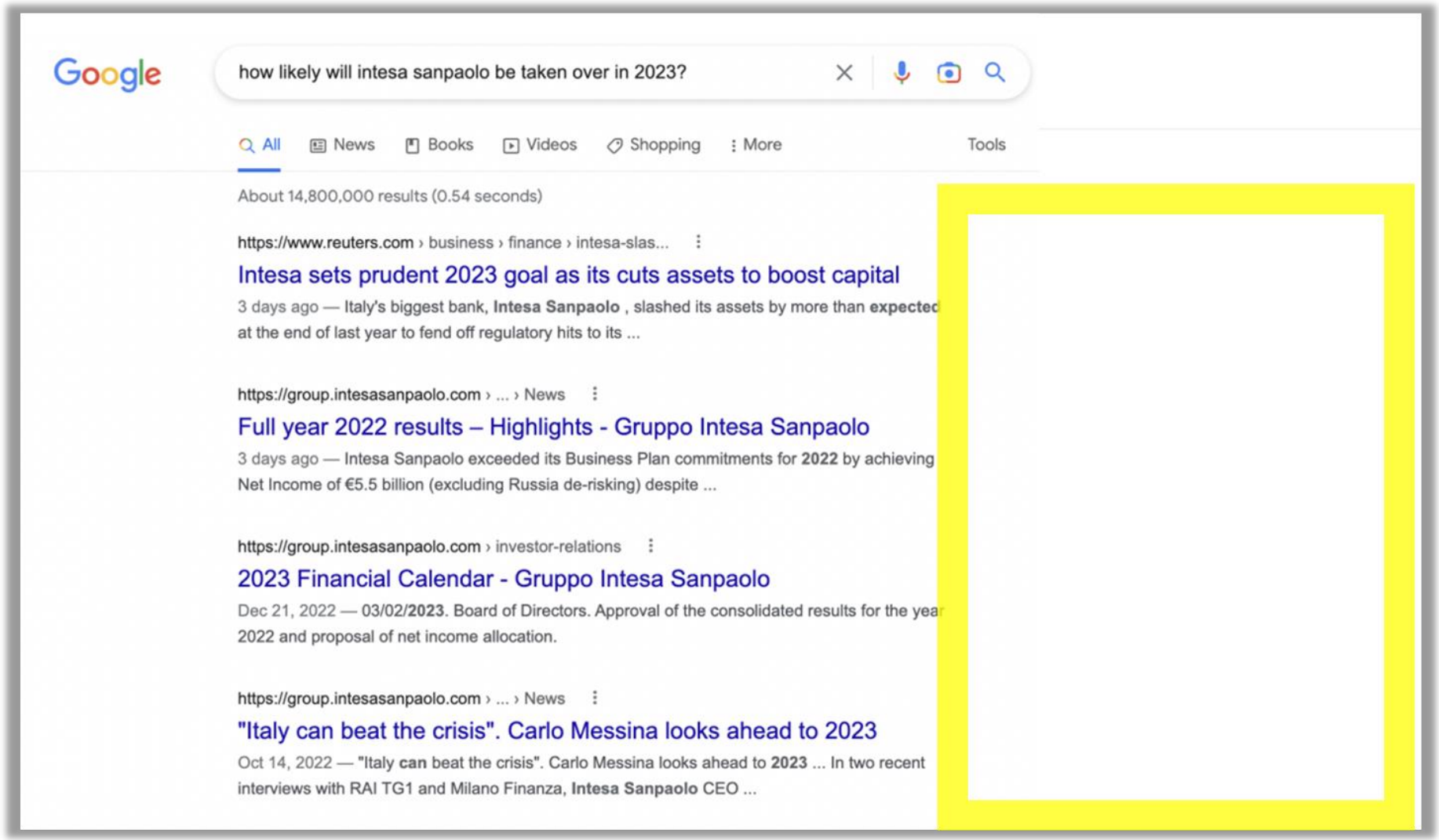
Teaser

Emanuel Sallinger

“What is the faculty of informatics at TU Wien”?



“How likely will the largest bank in Italy be taken over in 2023?"



“How likely will the largest bank in Italy be taken over in 2023?"

how likely will intesa sanpaolo be taken over in 2023?

It is difficult to predict with certainty whether Intesa Sanpaolo will be taken over in 2023 Takeovers are typically the result of a complex set of factors, including the strategic goals of the potential acquirer, the financial performance of the target company, and regulatory approval.

That being said, Intesa Sanpaolo is a large and well-established banking group, and it would likely take a significant offer from a well-capitalized acquirer to successfully complete a takeover. Additionally, the bank's strategic importance to the Italian economy would likely be a major consideration for regulators and government officials.

Therefore, it is not likely that Intesa Sanpaolo will be taken over in 2023 but it can't be completely ruled out. As the year progresses, it is important to keep an eye on any developments in the merger and acquisition market that might affect the bank's future.

Knowledge Graphs

1. The **technology** used by Google and others
2. A meeting place of **databases**, **data science** and **Artificial Intelligence** research
3. A **skillset** to solve fascinating problems

Anti-Money Laundering

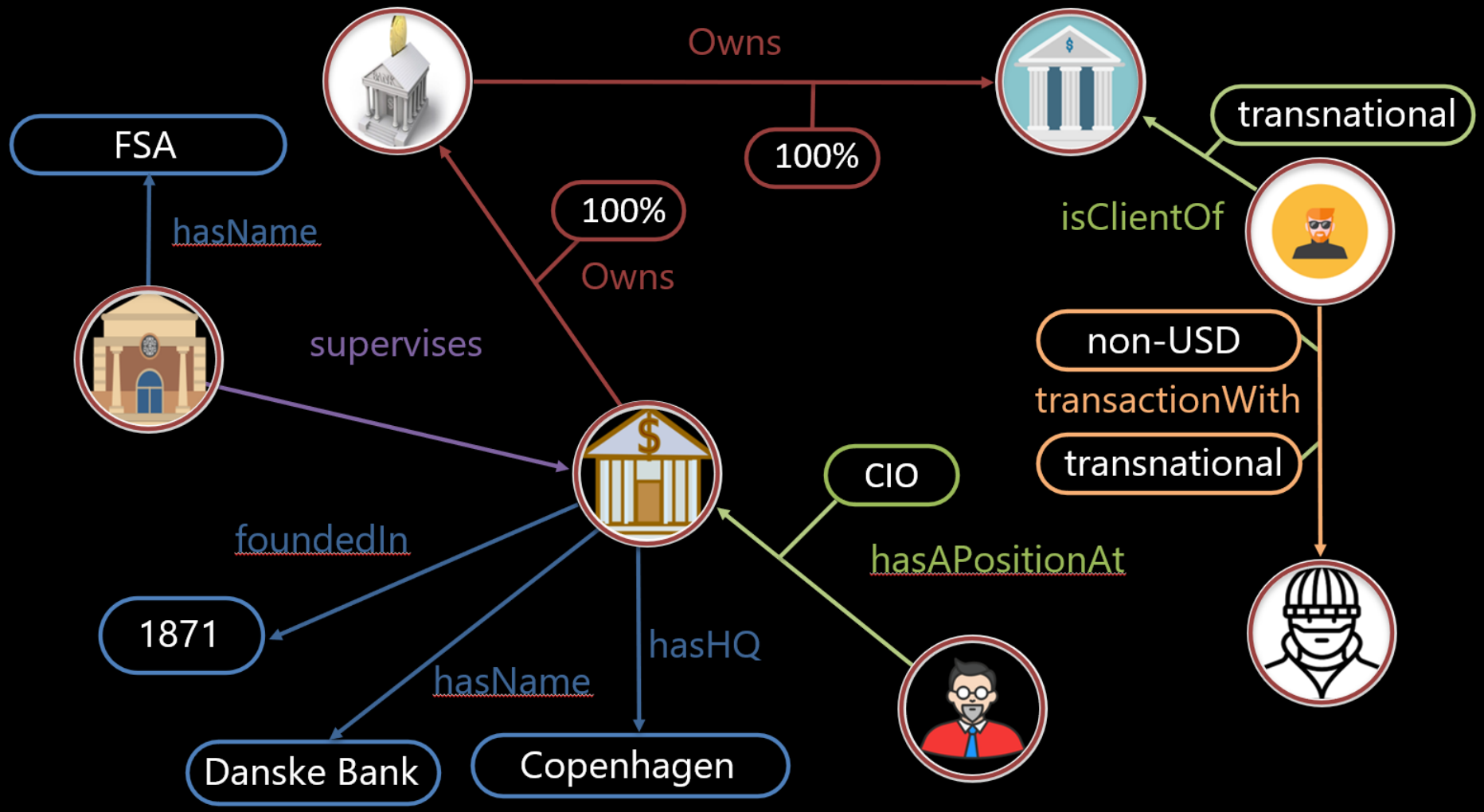
FINANCIAL TIMES

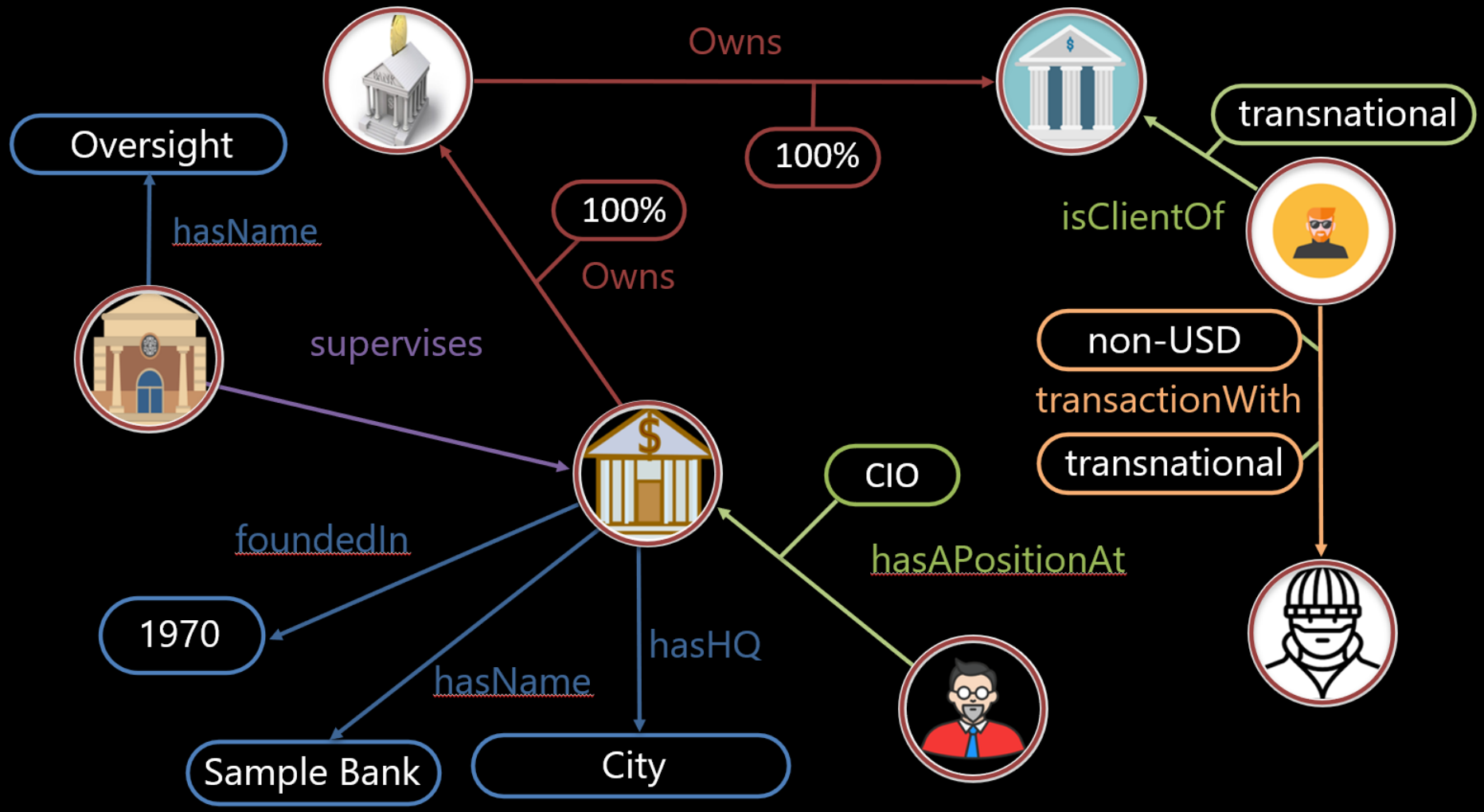
Danske Bank AS + Add to myFT

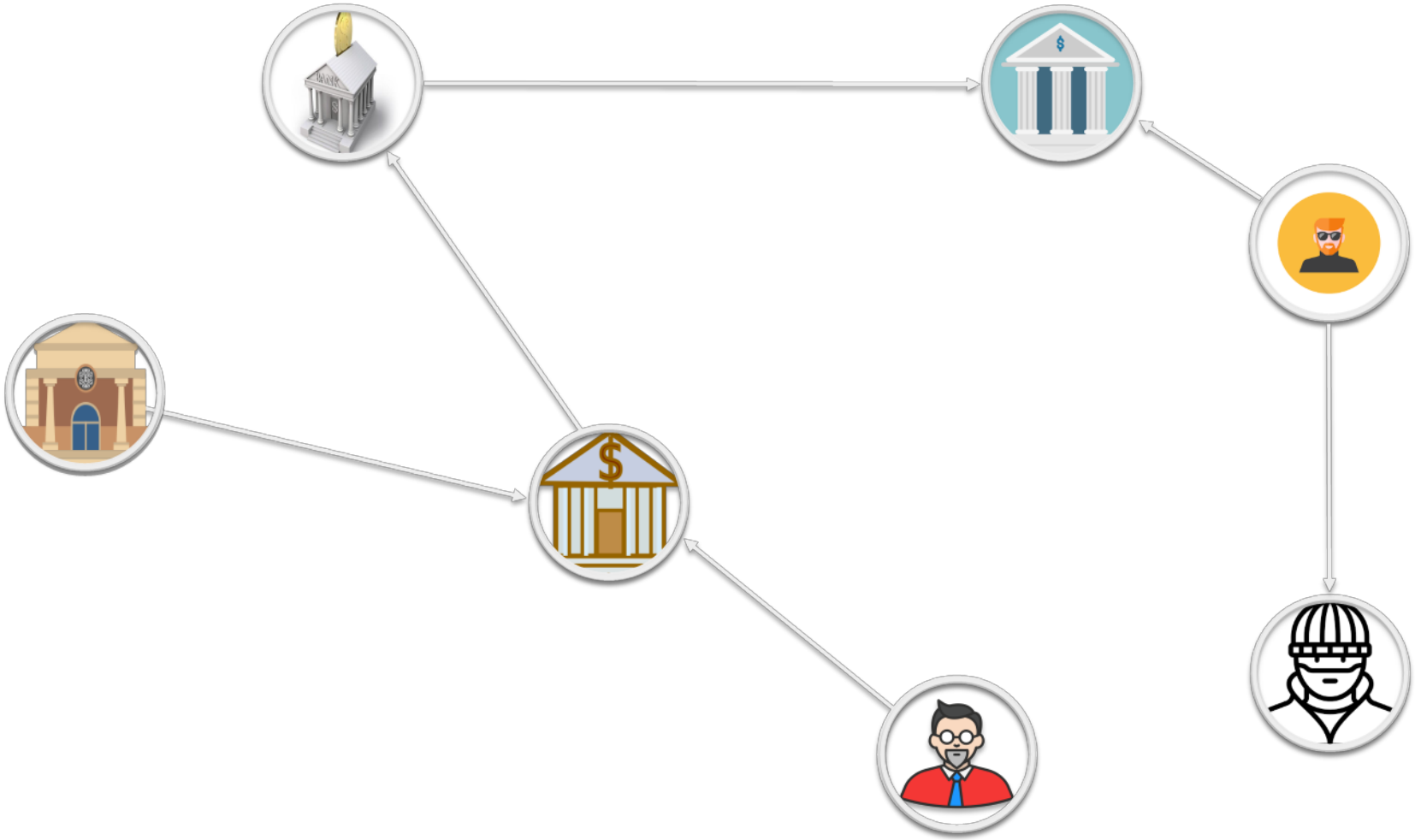
Danske Bank chairman ousted by main shareholder after scandal

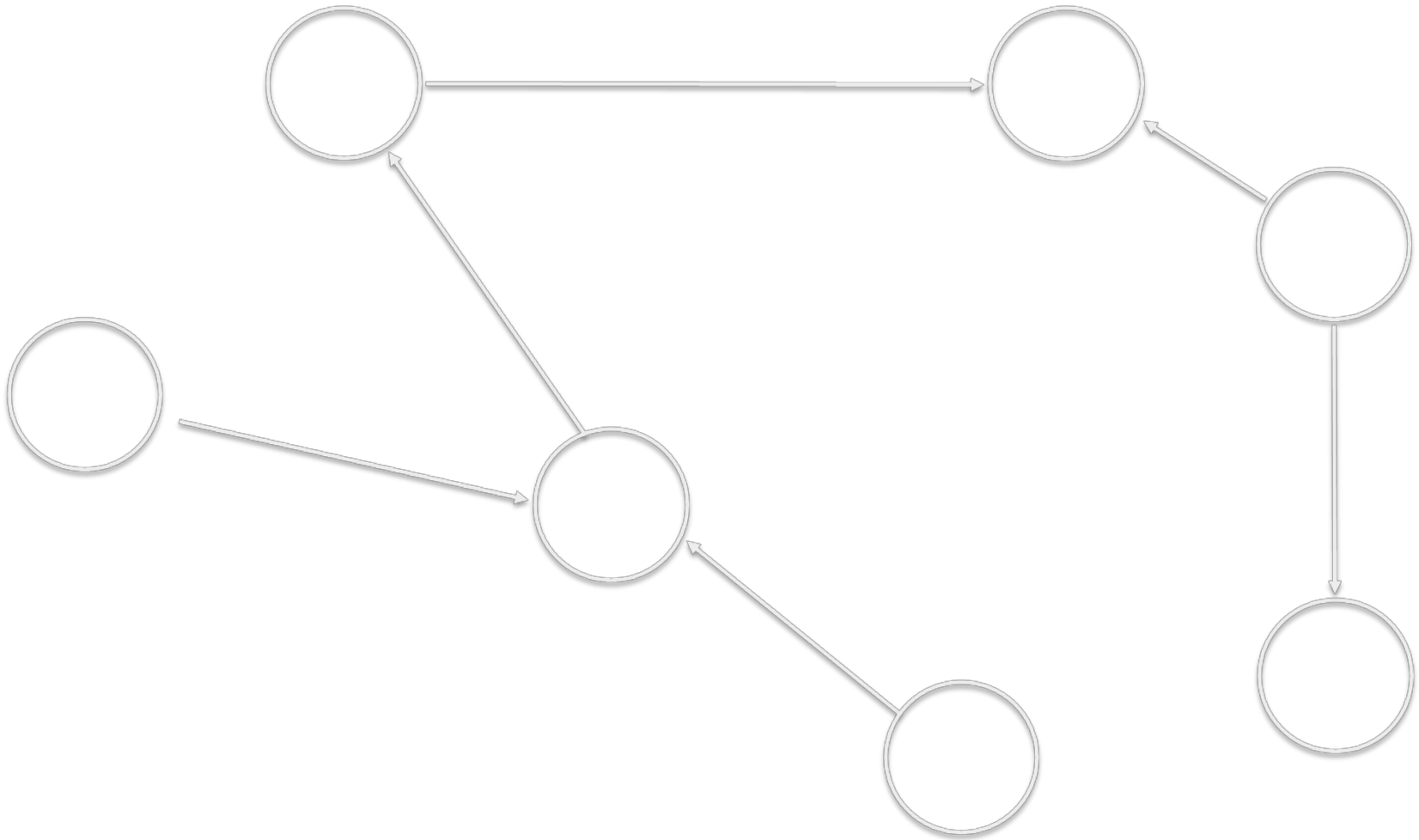
Maersk family brings in new blood to stabilise lender in wake of €200bn money laundering

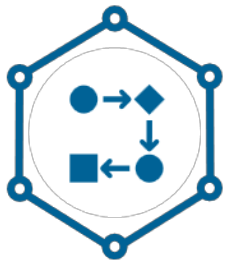
€200bn money laundering


Owns
100%
FSA
hasName
100%
Owns
supervises
transnational
isClientOf
non-USD
transactionWith
transnational
CIO
hasAPositionAt
foundedIn
1871
hasName
hasHQ
Danske Bank
Copenhagen


Owns
100%
Oversight
hasName
100%
Owns
transnational
isClientOf
supervises
non-USD
transactionWith
transnational
CIO
hasAPositionAt
foundedIn
1970
hasName
hasHQ
Sample Bank
City

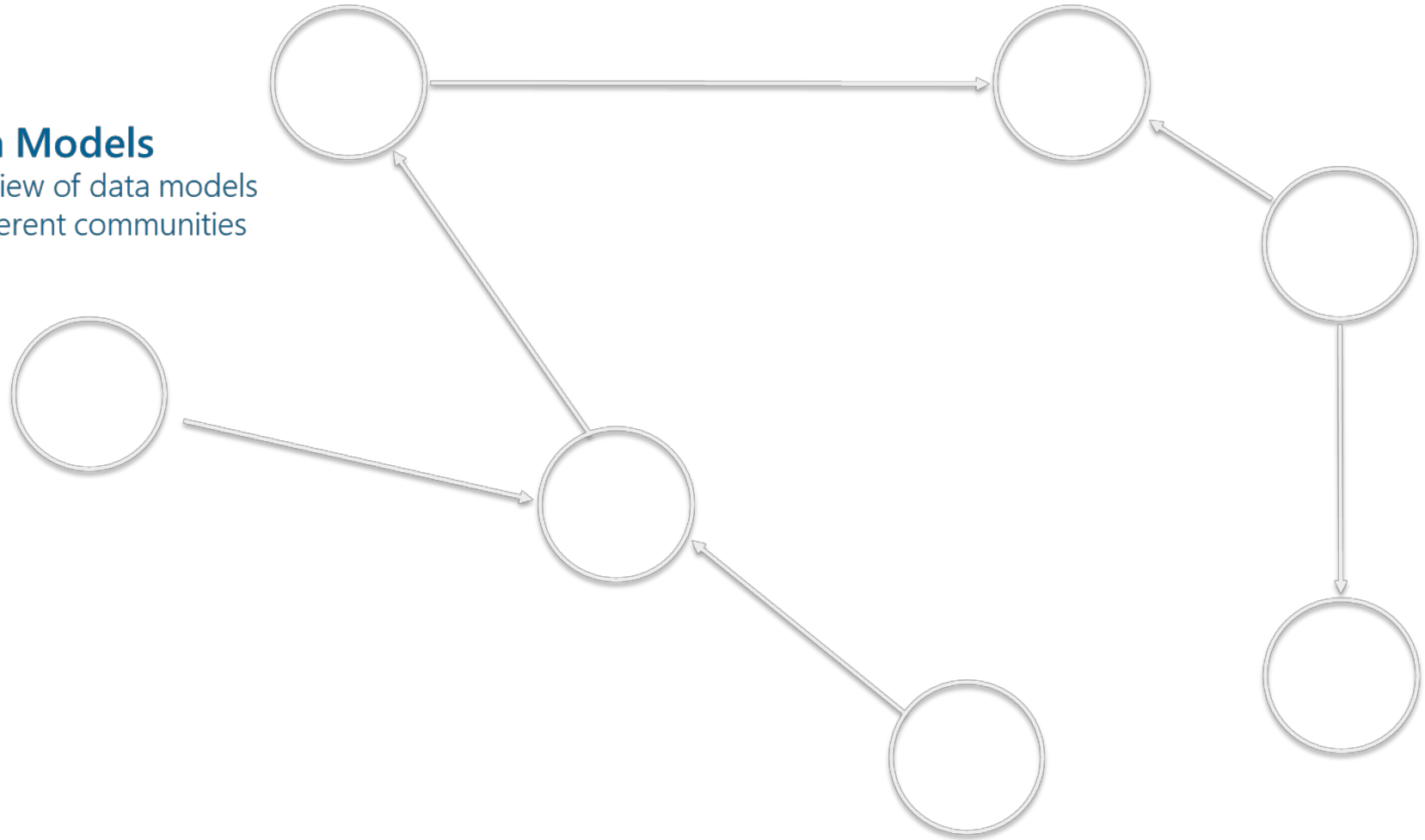


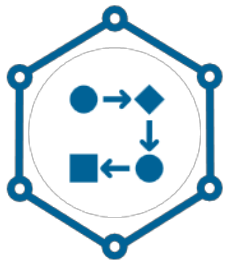




Data Models

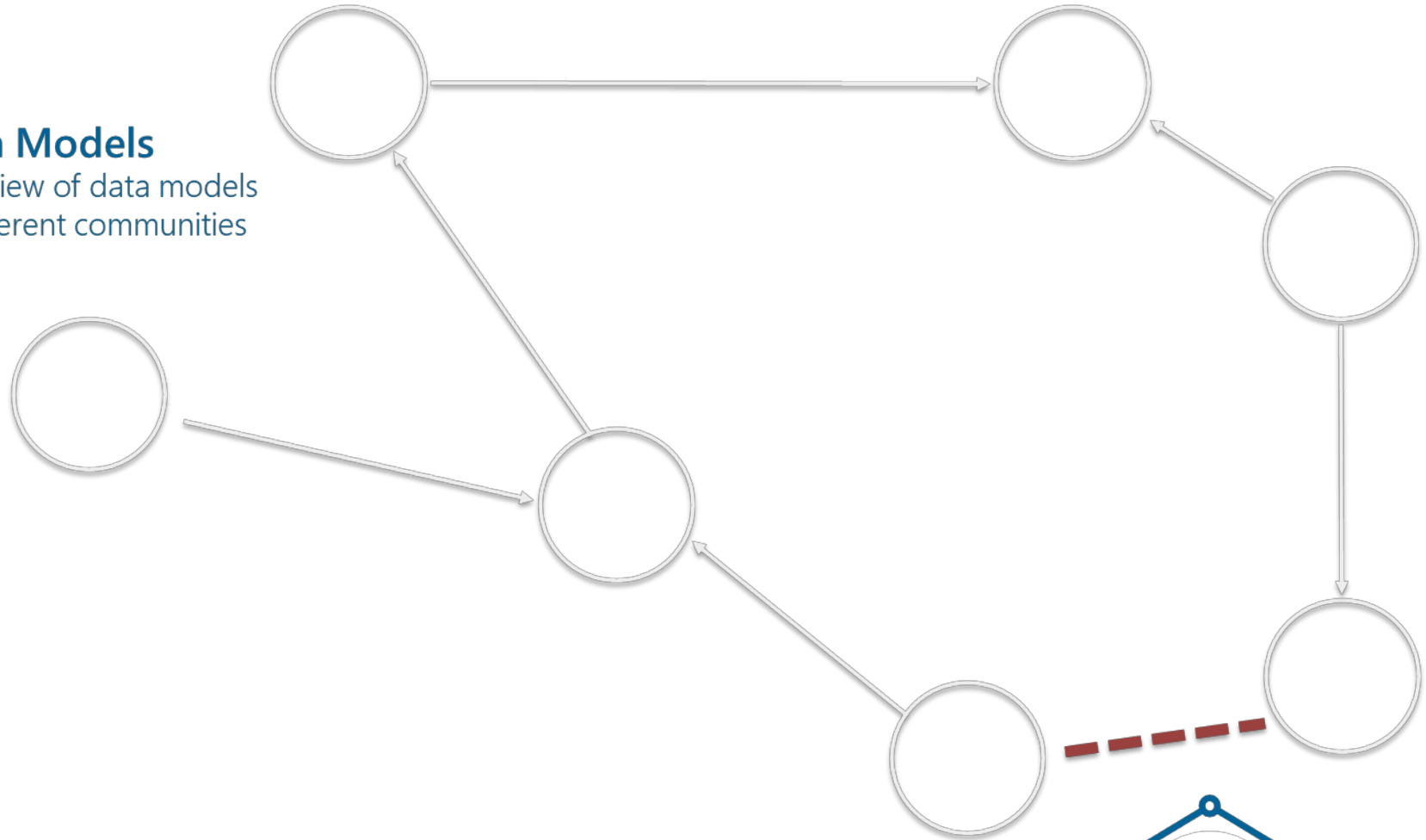
Overview of data models
in different communities





Data Models

Overview of data models in different communities

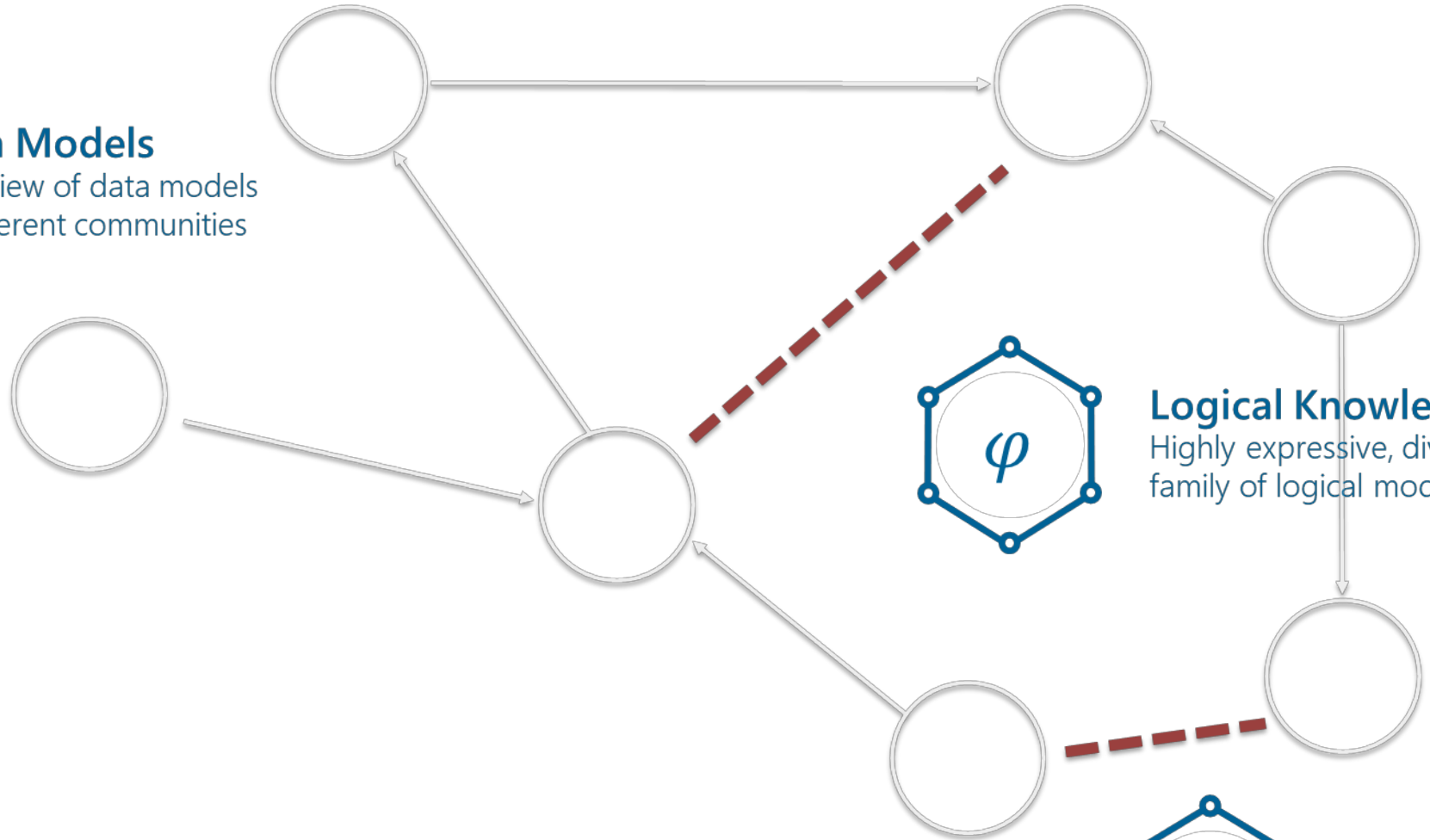


KG Embeddings

Widely-applied, large family of ML models

Data Models

Overview of data models
in different communities

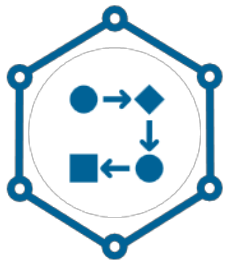


Logical Knowledge in KGs

Highly expressive, diverse
family of logical models.

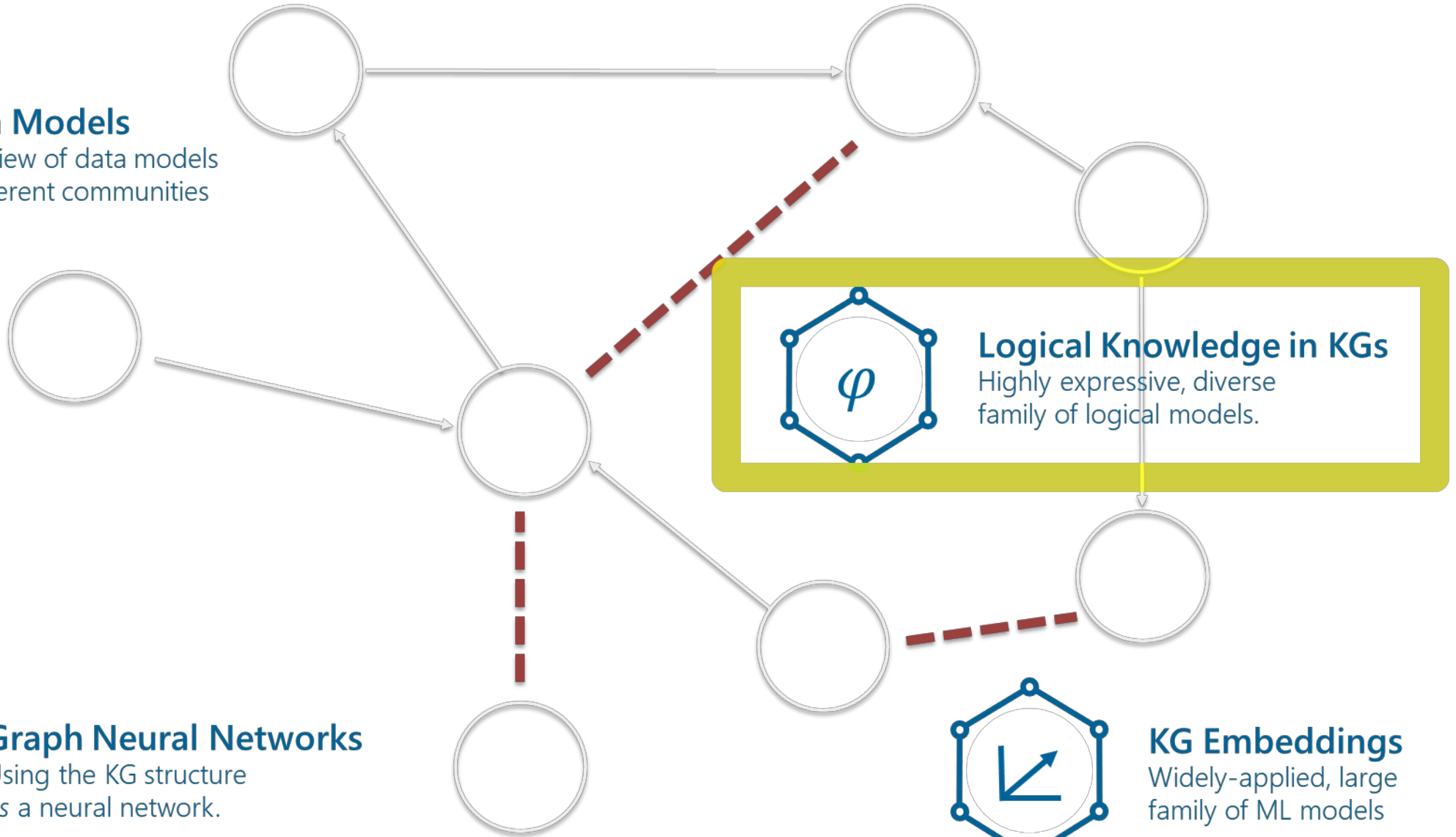
KG Embeddings

Widely-applied, large
family of ML models



Data Models

Overview of data models
in different communities



Logical Knowledge in KGs

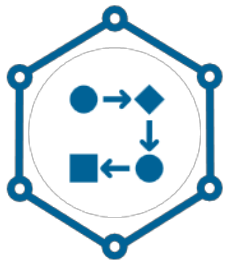
Highly expressive, diverse
family of logical models.

Graph Neural Networks

Using the KG structure
as a neural network.

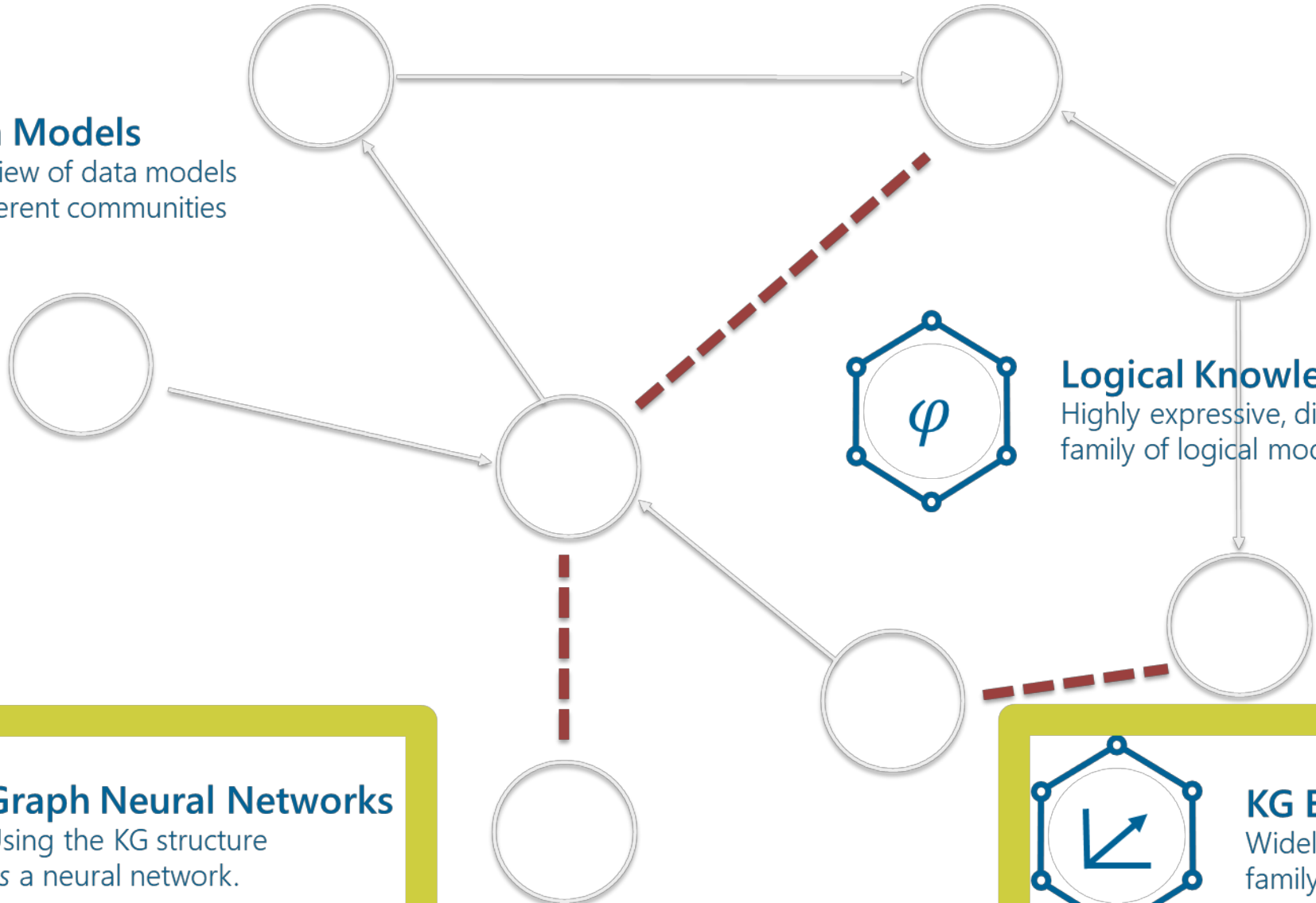
KG Embeddings

Widely-applied, large
family of ML models



Data Models

Overview of data models in different communities



Logical Knowledge in KGs

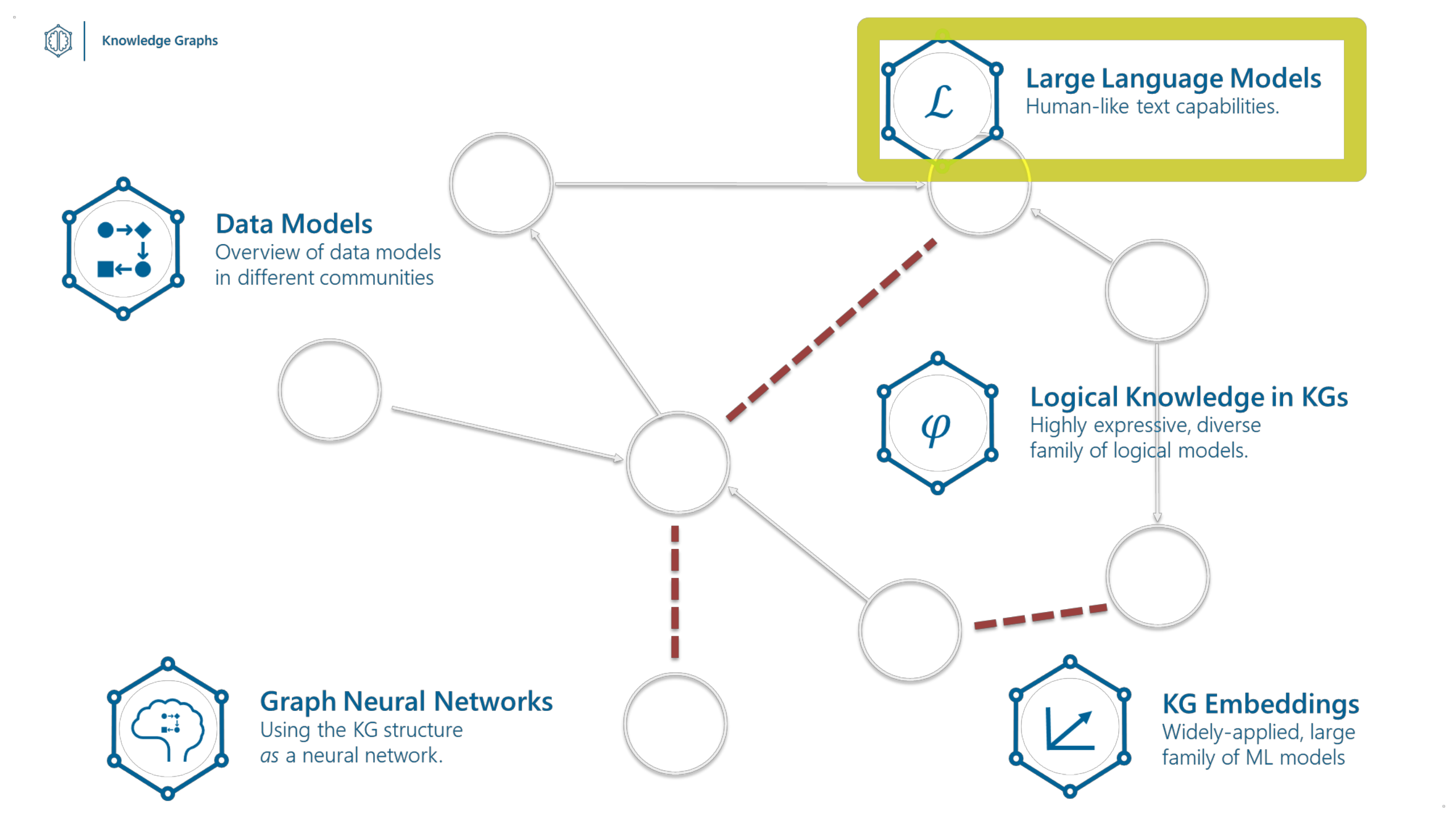
Highly expressive, diverse family of logical models.

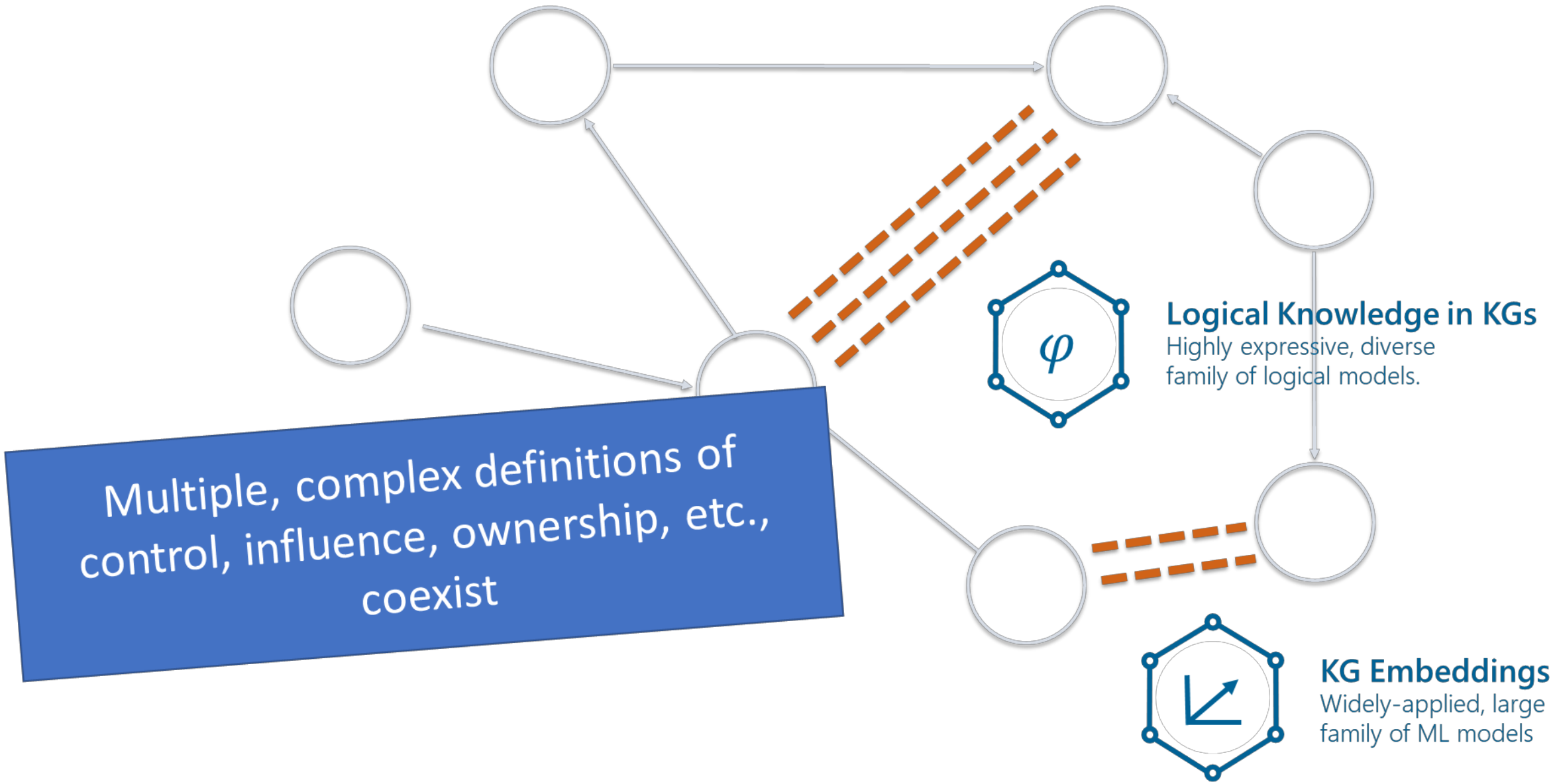
Graph Neural Networks

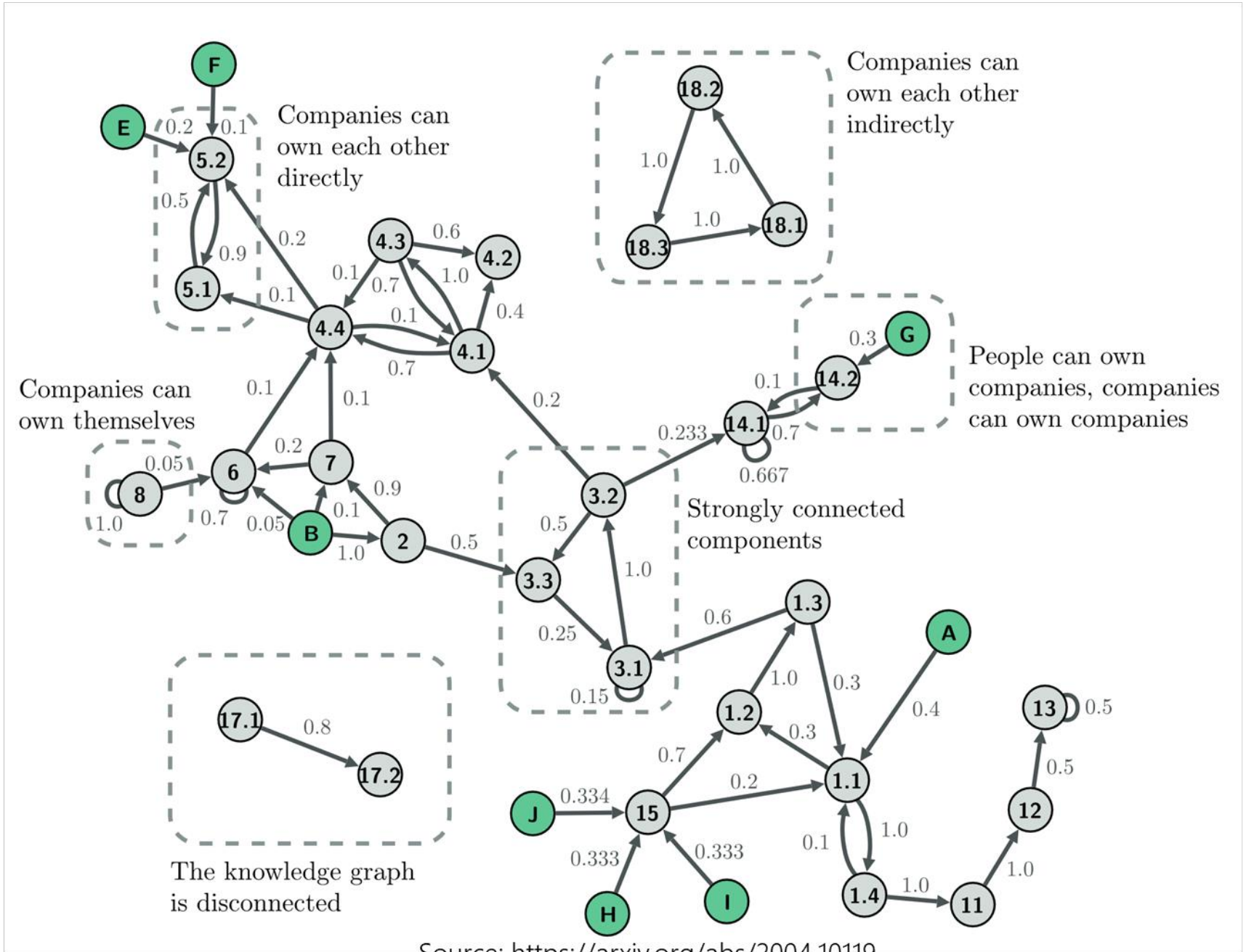
Using the KG structure *as* a neural network.

KG Embeddings

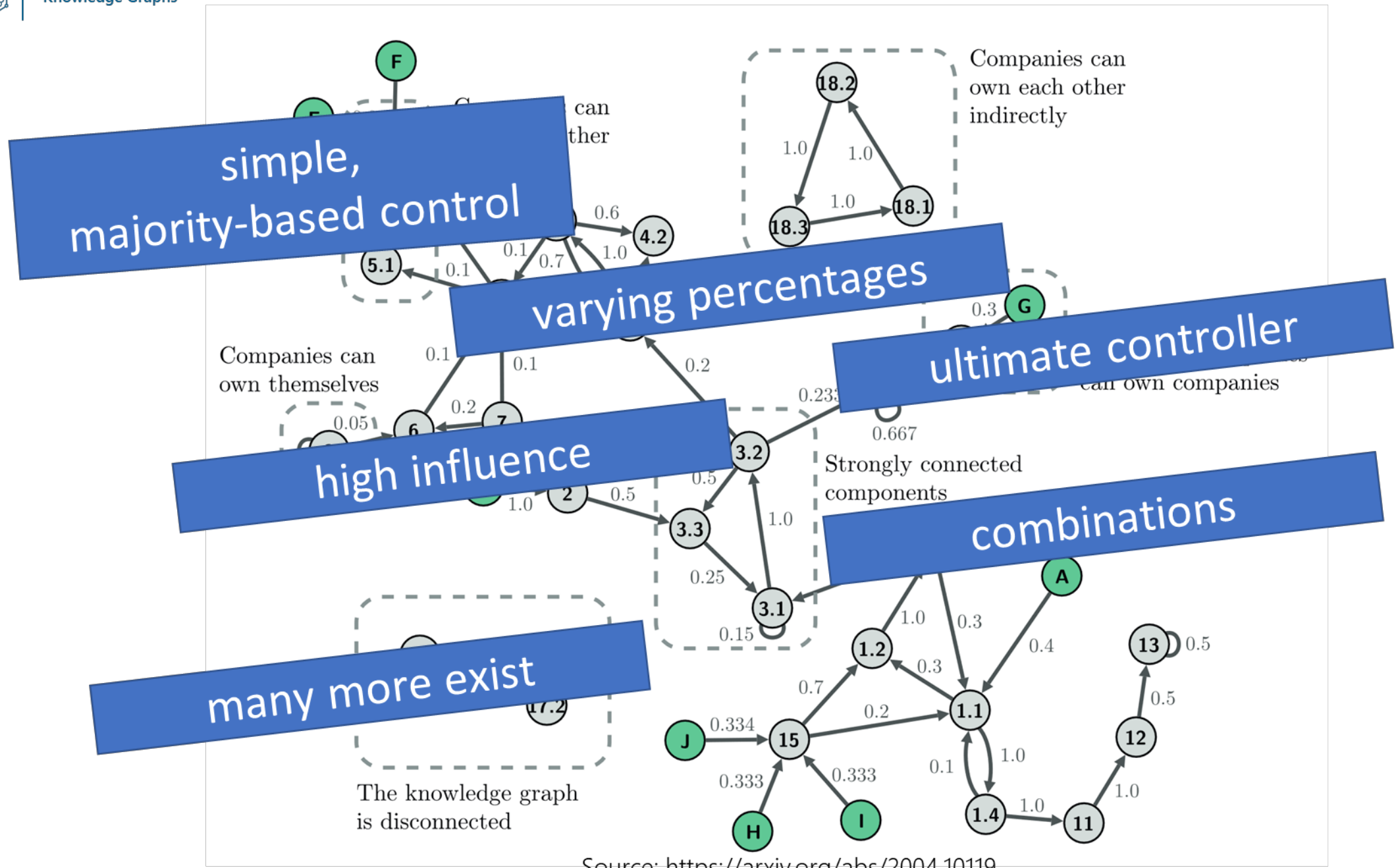
Widely-applied, large family of ML models


Large Language Models
Human-like text capabilities.
Data Models
Overview of data models
in different communities
Logical Knowledge in KGs
Highly expressive, diverse
family of logical models.
Graph Neural Networks
Using the KG structure
as a neural network.
KG Embeddings
Widely-applied, large
family of ML models


Multiple, complex definitions of control, influence, ownership, etc., coexist
Logical Knowledge in KGs
Highly expressive, diverse family of logical models.
KG Embeddings
Widely-applied, large family of ML models



Source: https://arxiv.org/abs/2004.10119



Source: https://arxiv.org/abs/2004.10119

Knowledge Graphs

Research and Pushing the Boundaries

Teaser

Emanuel Sallinger

2016-2019 Selected Highlights

LANGUAGE

Data Wrangling for Big Data: Towards a Lingua Franca for Data Wrangling

LOGIC

Swift Logic for Big Data and Knowledge Graphs

PROJECT

The VADA Architecture for Cost-Effective Data Wrangling

MEDI 2018

DATA SCIENCE

Data Science with Vadalog: Bridging Machine Learning and Reasoning

SYSTEM

The Vadalog System: Datalog-based Reasoning for Knowledge Graphs

Journal of Big Data 2019

PROJECT ARCHITECTURE

VADA: an architecture for end user informed data preparation

SPACE EFFICIENCY

The Space-Efficient Core of Vadalog

ENTERPRISE AI

Knowledge Graphs and Enterprise AI: The Promise of an Enabling Technology

Datalog 2.0 2019

RECOMMENDER SYSTEMS

Feature Engineering and Explainability with Vadalog: A Recommender Systems Application

2020 | Selected Highlights

EMERGENCY RESPONSE
(not peer-reviewed)

COVID-19: LOCKDOWN

COVID-19 and Company Knowledge Graphs: Assessing Golden Powers and Economic Impact of Selective Lockdown via AI Reasoning

EDBT ICDT COPENHAGEN 2020

ENTERPRISE AI IN PRACTICE

Weaving Enterprise Knowledge Graphs: The Case of Company Ownership Graphs

Declarative AI 2020

COVID-19: TAKEOVERS

Reasoning on Company Takeovers during the COVID-19 Crisis with Knowledge Graphs

Declarative AI 2020

PROBABILISTIC

Reasoning Under Uncertainty in Knowledge Graphs

Declarative AI 2020

MONEY LAUNDERING

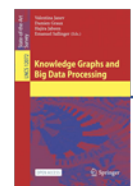
Rule-based Anti-Money Laundering in Financial Intelligence Units: Experience and Vision

KNOWLEDGE GRAPHS

Knowledge Graphs: The Layered Perspective

BOOK

Knowledge Graphs and Big Data Processing



KG EMBEDDINGS

Reasoning in Knowledge Graphs: An Embeddings Spotlight

2021 Selected Highlights

CONFIDENTIALITY

Financial Data Exchange with Statistical Confidentiality: A Reasoning-based Approach

COMPANY CONTROL

Distributed Company Control in Company Shareholding Graphs

TEMPORAL

Monotonic Aggregation for Temporal Datalog

HARMFUL JOINS

Eliminating Harmful Joins in Warded Datalog+/-.

INDUSTRIAL BLOCKCHAIN

Rule-based Blockchain Knowledge Graphs: Declarative AI for Solving Industrial Blockchain Challenges

JOINS

Traversing Knowledge Graphs with Good Old (and New) Joins

HYBRID AI

Augmenting Logic-based Knowledge Graphs: The Case of Company Graphs

BLOCKCHAIN VISION

Blockchains as Knowledge Graphs - Blockchains for Knowledge Graphs

2022 Selected Highlights

DATA SCIENCE

Data science with Vadalog: Knowledge Graphs with machine learning and reasoning in practice

CS AND ECONOMY

Reasoning on company takeovers: From tactic to strategy

AI ARCHITECTURE

Vadalog: A modern architecture for automated reasoning with large knowledge graphs

KG DESIGN

Model-Independent Design of Knowledge Graphs - Lessons Learnt From Complex Financial Graphs

38th IEEE International Conference on Data Engineering

TEMPORAL

iTemporal: An Extensible Generator of Temporal Benchmarks

SHY AND WARDED

On the Relationship between Shy and Warded Datalog+/-

RULE LEARNING

Rule Learning over Knowledge Graphs with Genetic Logic Programming

EGDs

Exploiting the Power of Equality-generating Dependencies in Ontological Reasoning

DATA SCIENCE

Data Science with Vadalog: Bridging Machine Learning and Reasoning

HYBRID AI

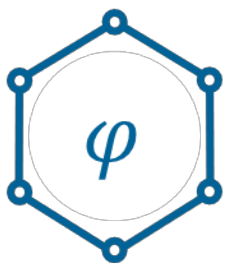
Augmenting Logic-based Knowledge Graphs: The Case of Company Graphs

RULE LEARNING

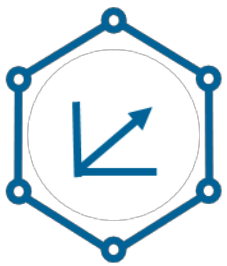
Rule Learning over Knowledge Graphs with Genetic Logic Programming

PATTERN AWARENESS

Pattern-Aware and Noise-Resilient Embedding Models



Logical Knowledge



Knowledge Graph Embeddings

Graph Neural Networks

Representations

Logical and ML-based representations for KGs.

- **KG Embeddings** — Widely-applied, large family of ML models.
- **Logical Knowledge in KGs** — Highly expressive, diverse family of logical models.
- **Graph Neural Networks** — Using the KG structure *as* a neural network.
- **Data Models** — Overview of data models in different communities.

Systems

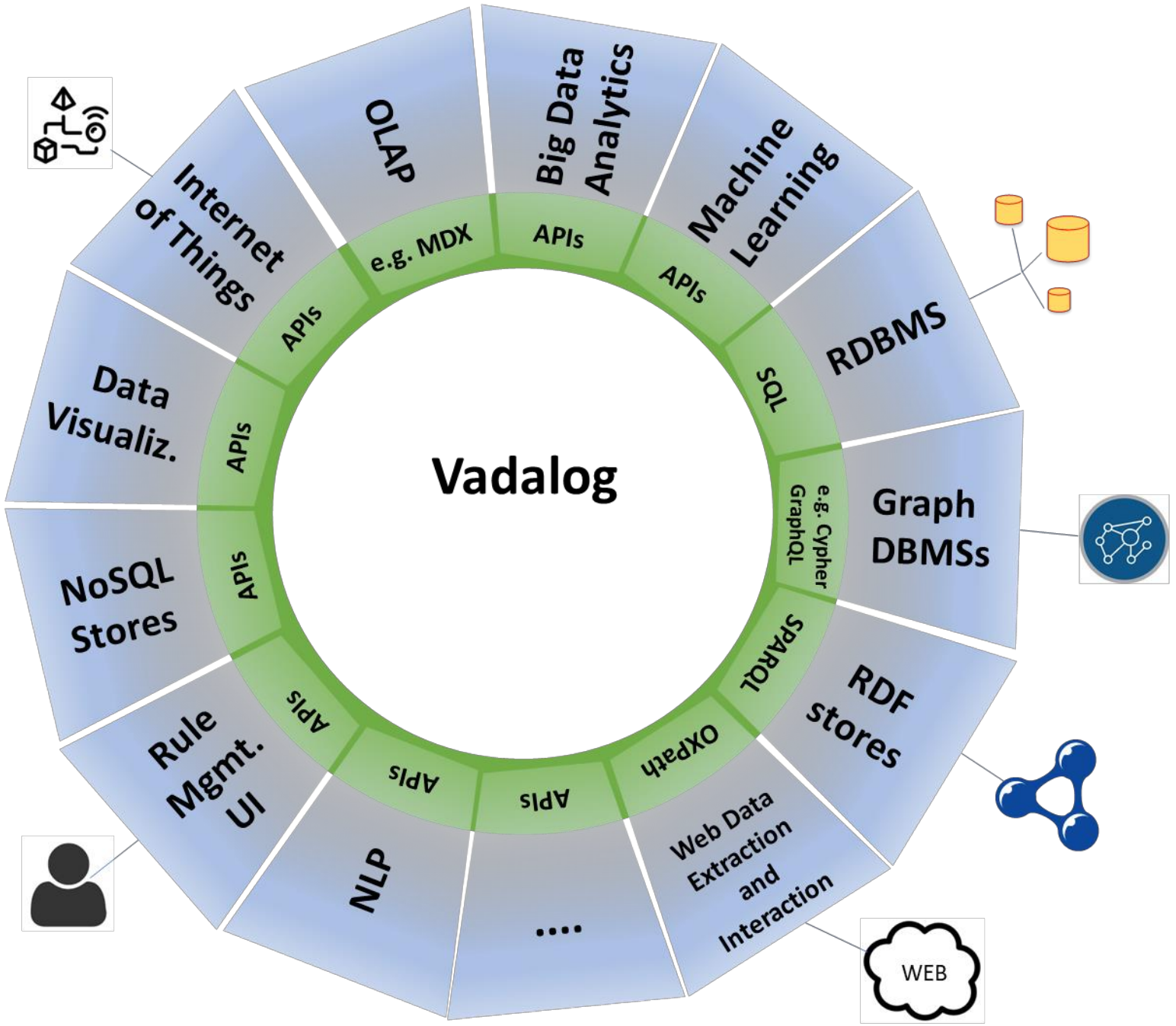
Systems to bring KGs into practice.

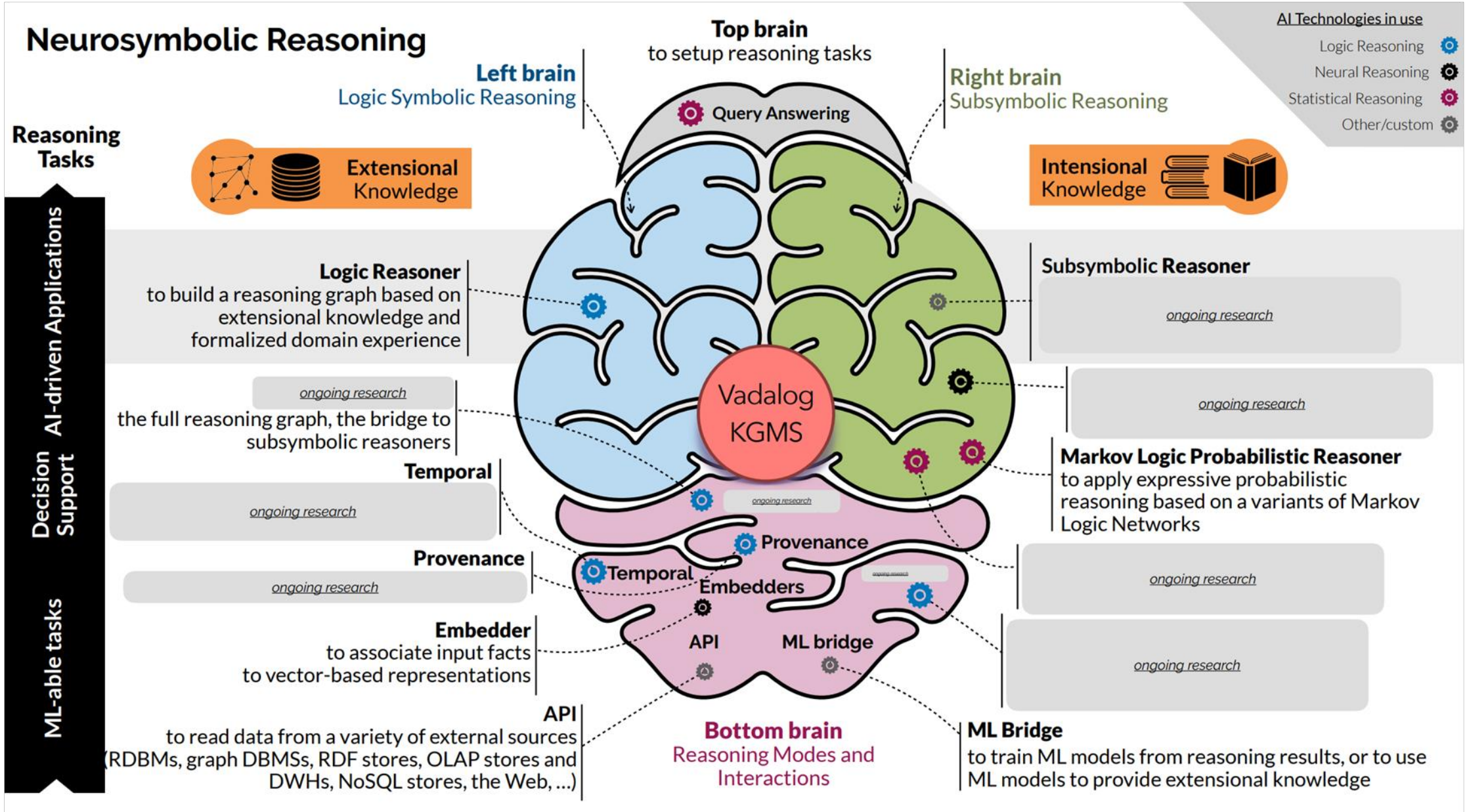
- **Architectures** — The big picture of building IT architectures for KGs.
- **Scalable Reasoning** — Making use of the knowledge in the KG.
- **KG Creation** — How to create a KG from heterogeneous data?
- **KG Evolution** — How to update, correct and complete a KG?

Applications

Real-world applications of KGs.

- **Real-World Applications** — Overview of diverse applications.
- **Financial KGs** — Concrete applications in finance and economics.
- **Services** — Which service to provide based on KGs?
- **Connections** — ... between KGs, AI, ML and Data Science.


Vadalog
OLAP
Big Data Analytics
Machine Learning
RDBMS
Graph DBMSs
RDF stores
Web Data Extraction and Interaction
....
NLP
Rule Mgmt. UI
NoSQL Stores
Data Visualiz.
Internet of Things
APIs
e.g. MDX
APIs
APIs
SQL
e.g. Cypher GraphQL
SPARQL
OXPath
APIs
APIs
APIs
APIs
APIs
WEB

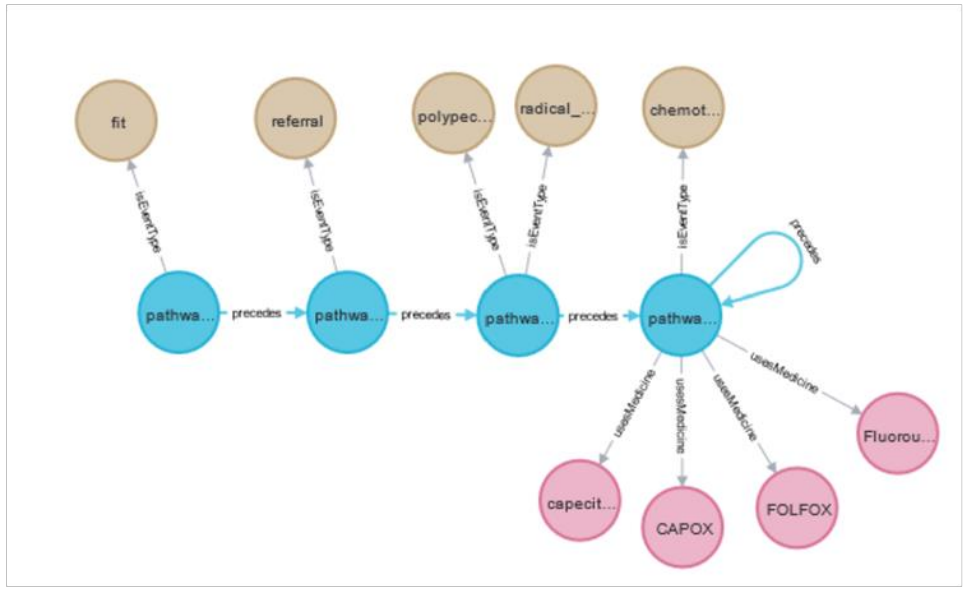

Neurosymbolic Reasoning
Top brain
to setup reasoning tasks
Left brain
Logic Symbolic Reasoning
Right brain
Subsymbolic Reasoning
AI Technologies in use
Logic Reasoning
Neural Reasoning
Statistical Reasoning
Other/custom
Reasoning Tasks
Query Answering
Extensional Knowledge
Intensional Knowledge
AI-driven Applications
Decision Support
ML-able tasks
Logic Reasoner
to build a reasoning graph based on extensional knowledge and formalized domain experience
Subsymbolic Reasoner
ongoing research
ongoing research
the full reasoning graph, the bridge to subsymbolic reasoners
ongoing research
Temporal
ongoing research
Markov Logic Probabilistic Reasoner
to apply expressive probabilistic reasoning based on a variants of Markov Logic Networks
Vadalog KGMS
ongoing research
Provenance
Temporal
Embedders
Provenance
ongoing research
ongoing research
Embedder
to associate input facts to vector-based representations
API
ML bridge
ongoing research
API
to read data from a variety of external sources (RDBMs, graph DBMSs, RDF stores, OLAP stores and DWHs, NoSQL stores, the Web, ...)
Bottom brain
Reasoning Modes and Interactions
ML Bridge
to train ML models from reasoning results, or to use ML models to provide extensional knowledge

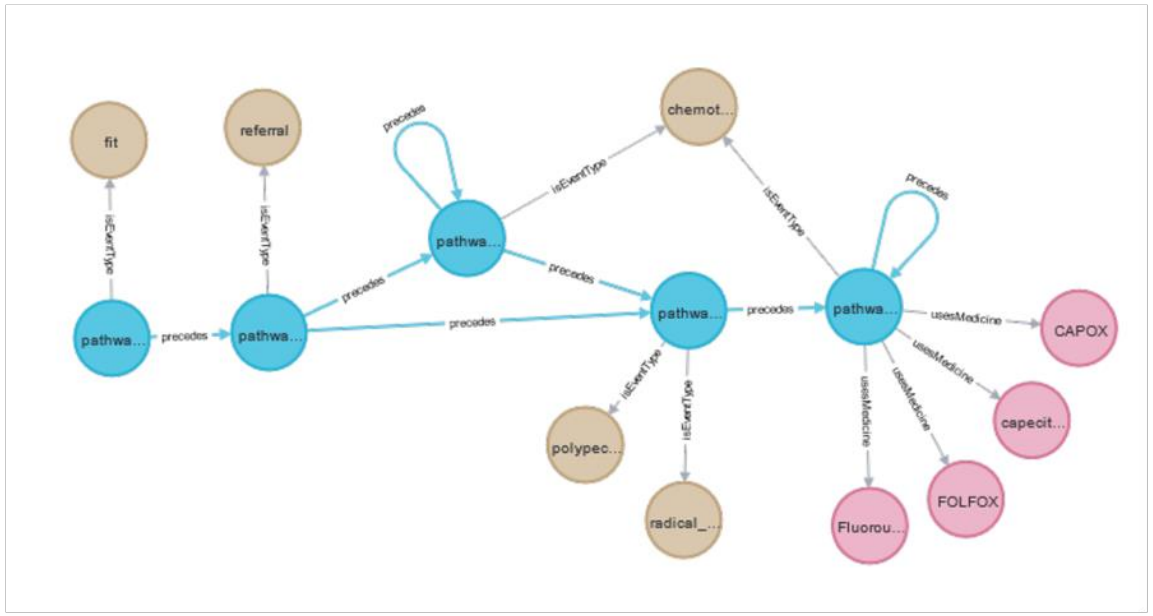
Knowledge Graphs
Real-World Applications – Diving Deeper

Teaser

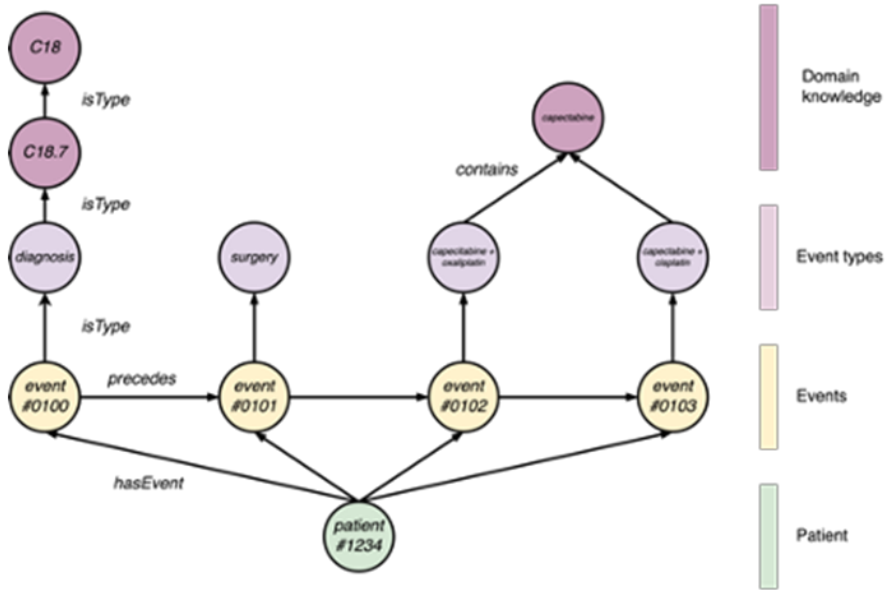
Emanuel Sallinger

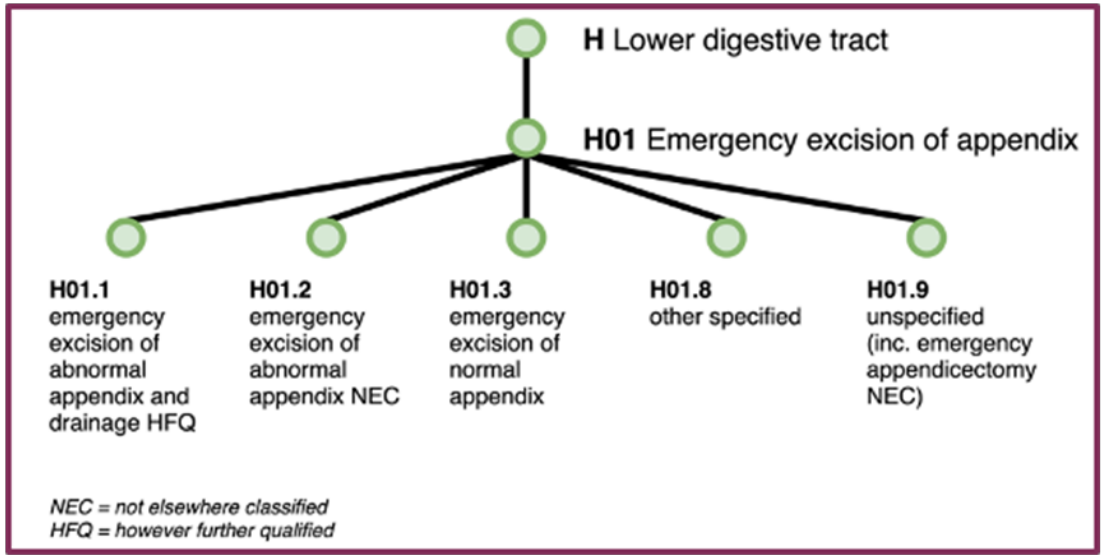
Cancer Pathways



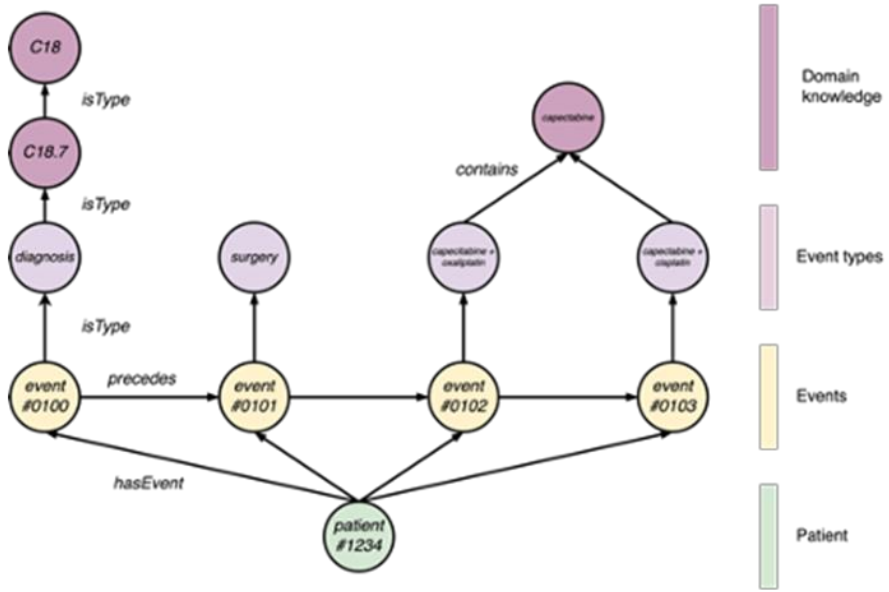


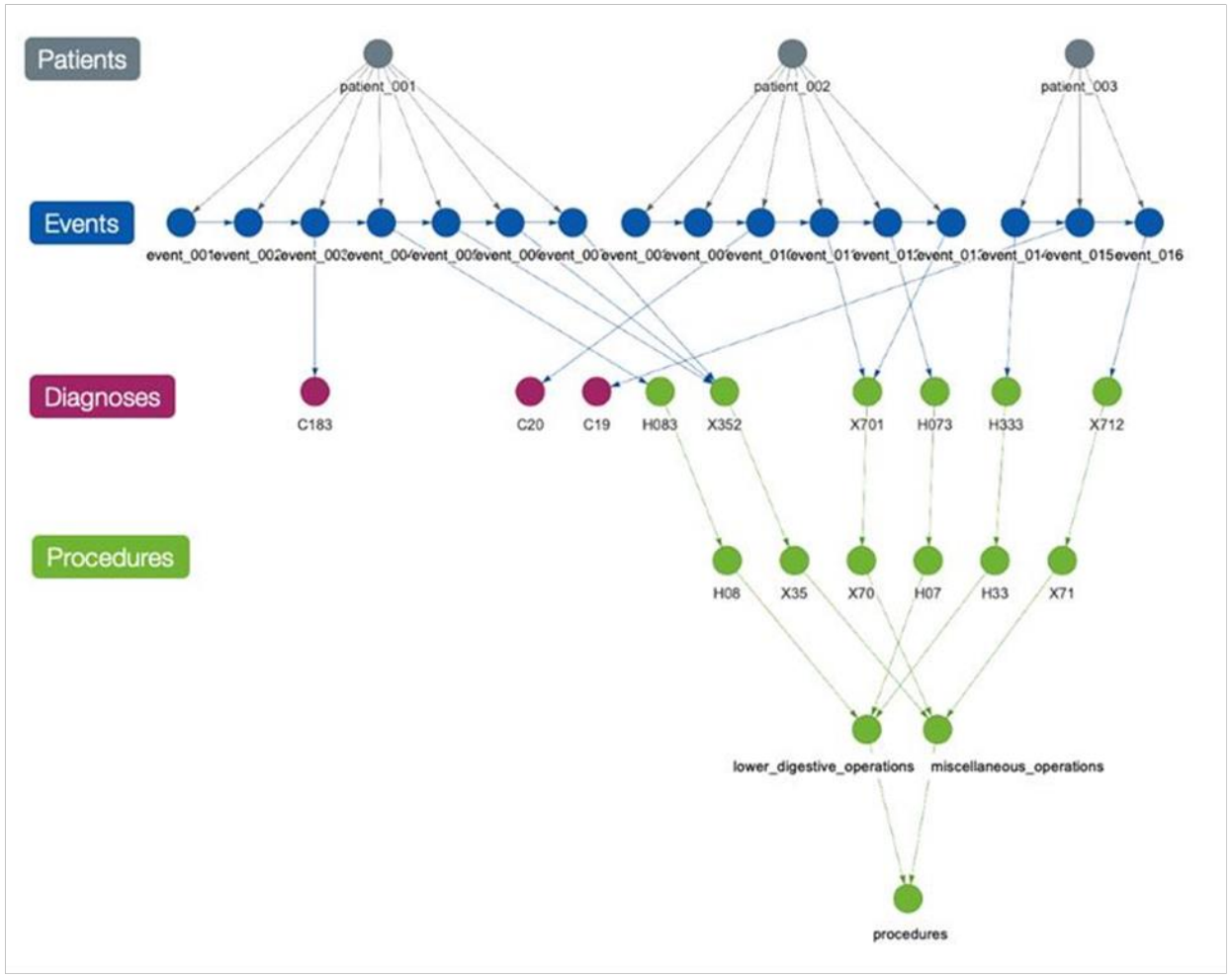
Cancer Pathways



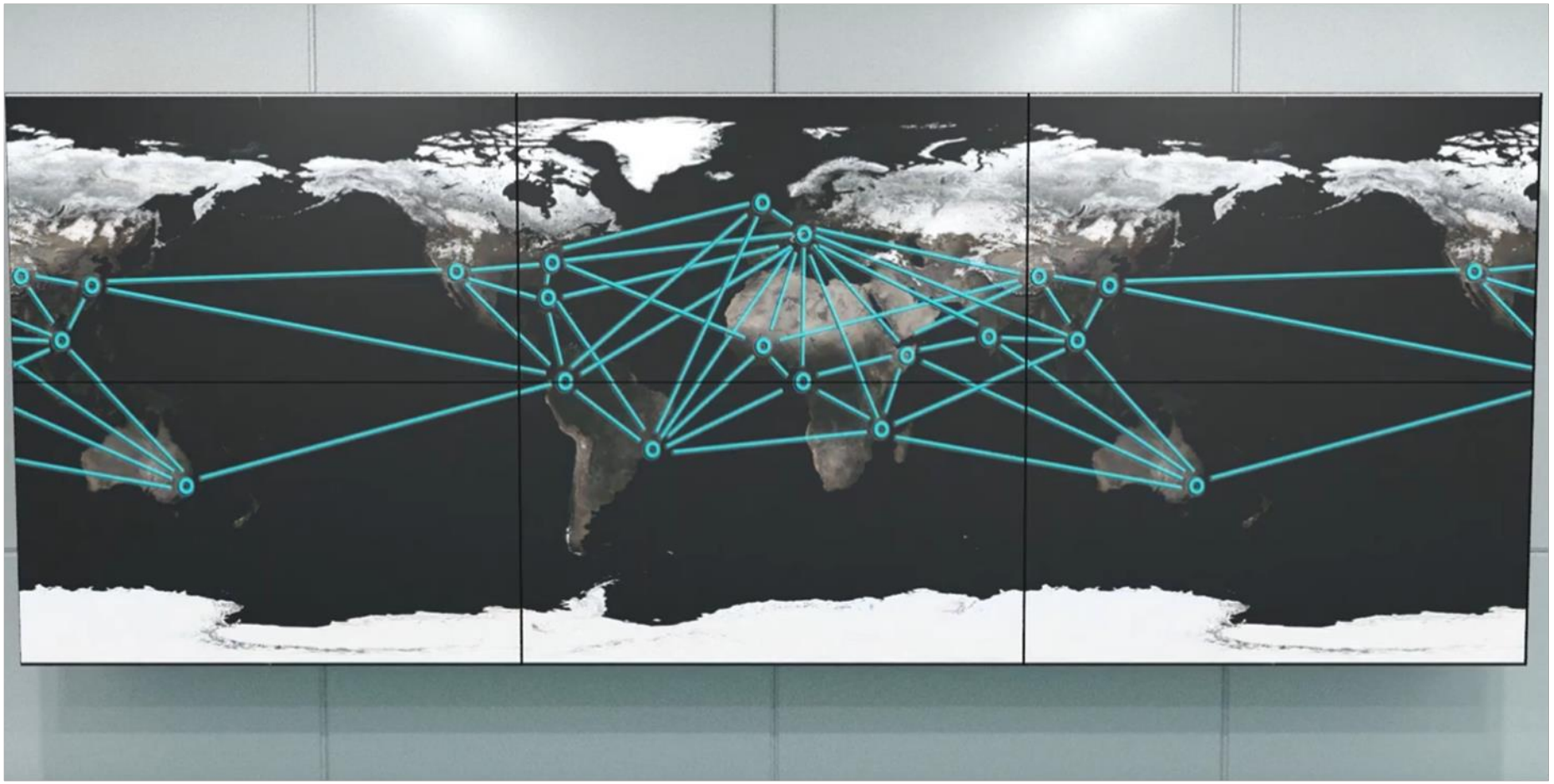


Cancer Pathways

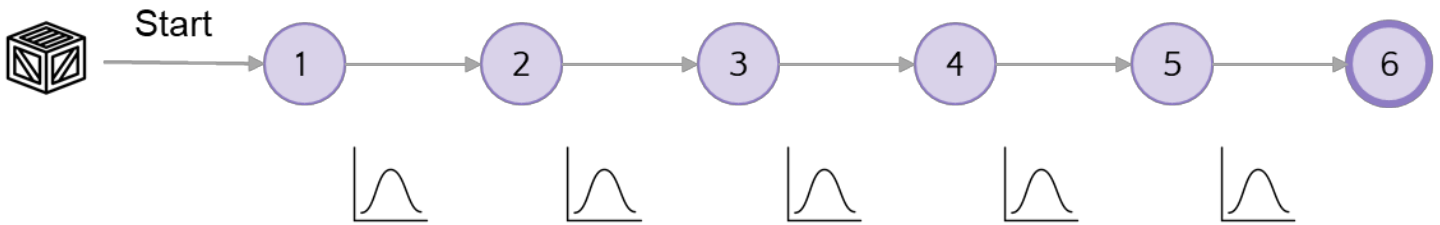




Supply Chains

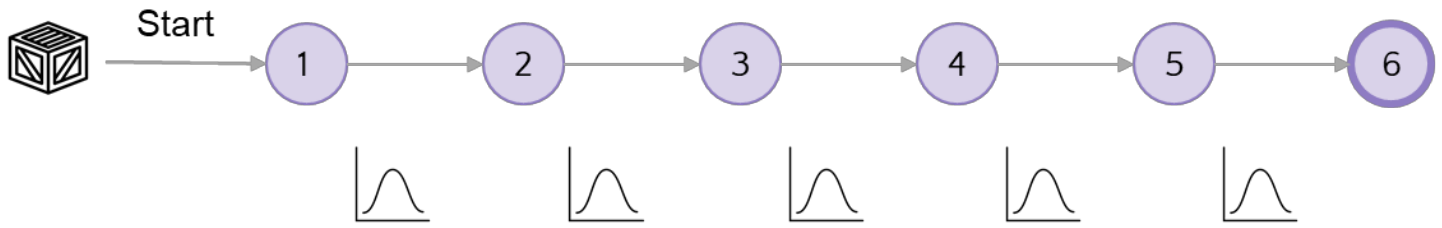


Supply Chains



Historic Data,
Machine Learning

Supply Chains

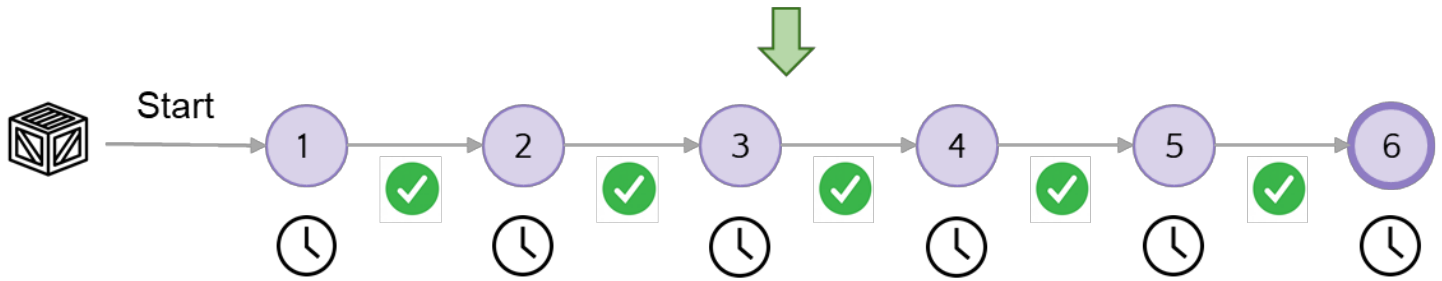


Historic Data,
Machine Learning

→ **Rules:** Express what you want, not how you want it done.

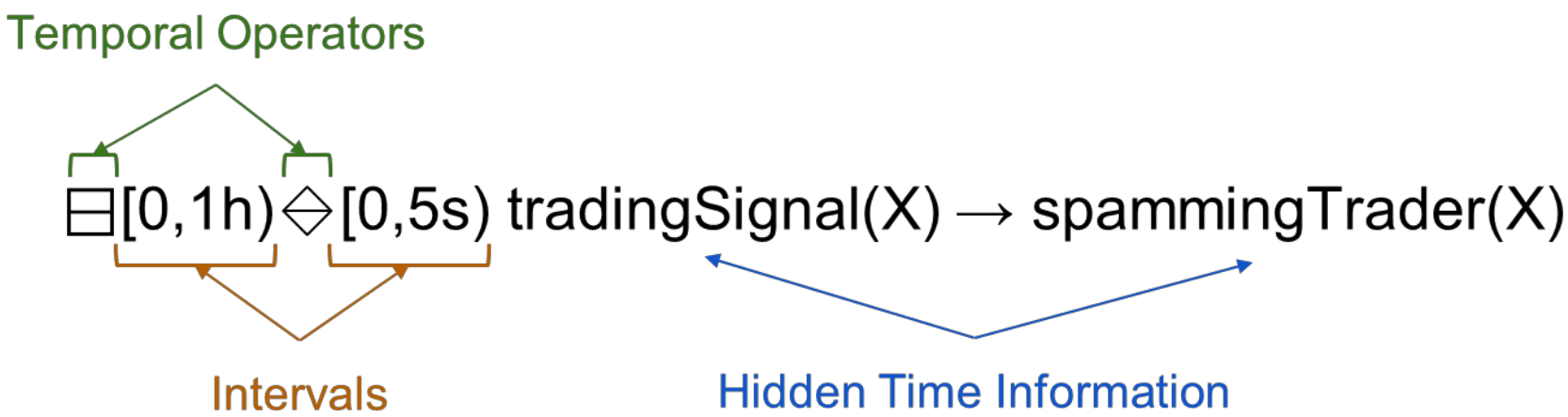
```
path(X, Y, Time) :-
     start(X, T0),
     section(X, Y),
     Time = T0 + stat:mean(X, Y)
```

```
path(X, Z, Time) :-
     path(X, Y, T1),
     section(Y, Z),
     Time = T + stat:mean(Y, Z).
```



Temporal: DatalogMTL

Datalog extended with operators from the Metric Temporal Logic

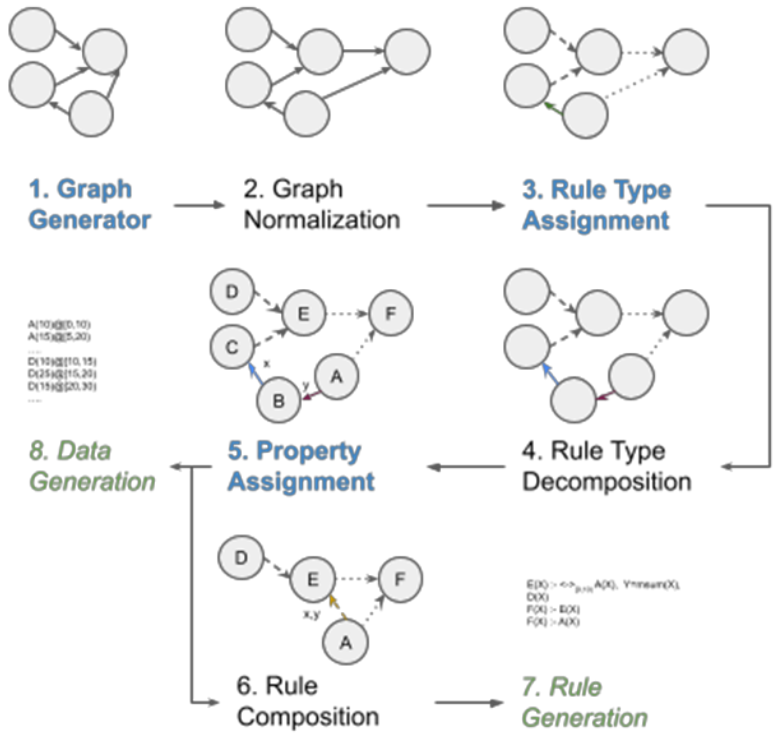


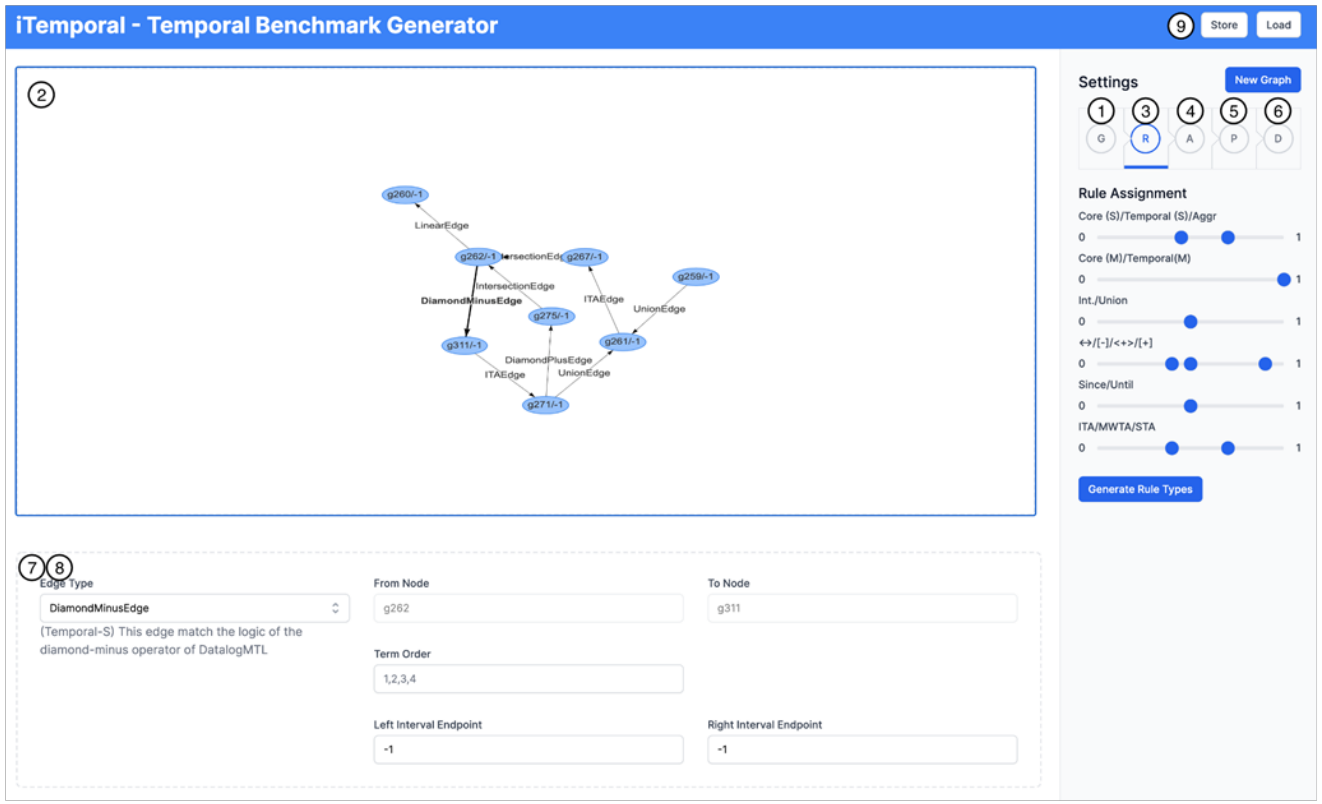
iTemporal

Temporal Benchmark Suite

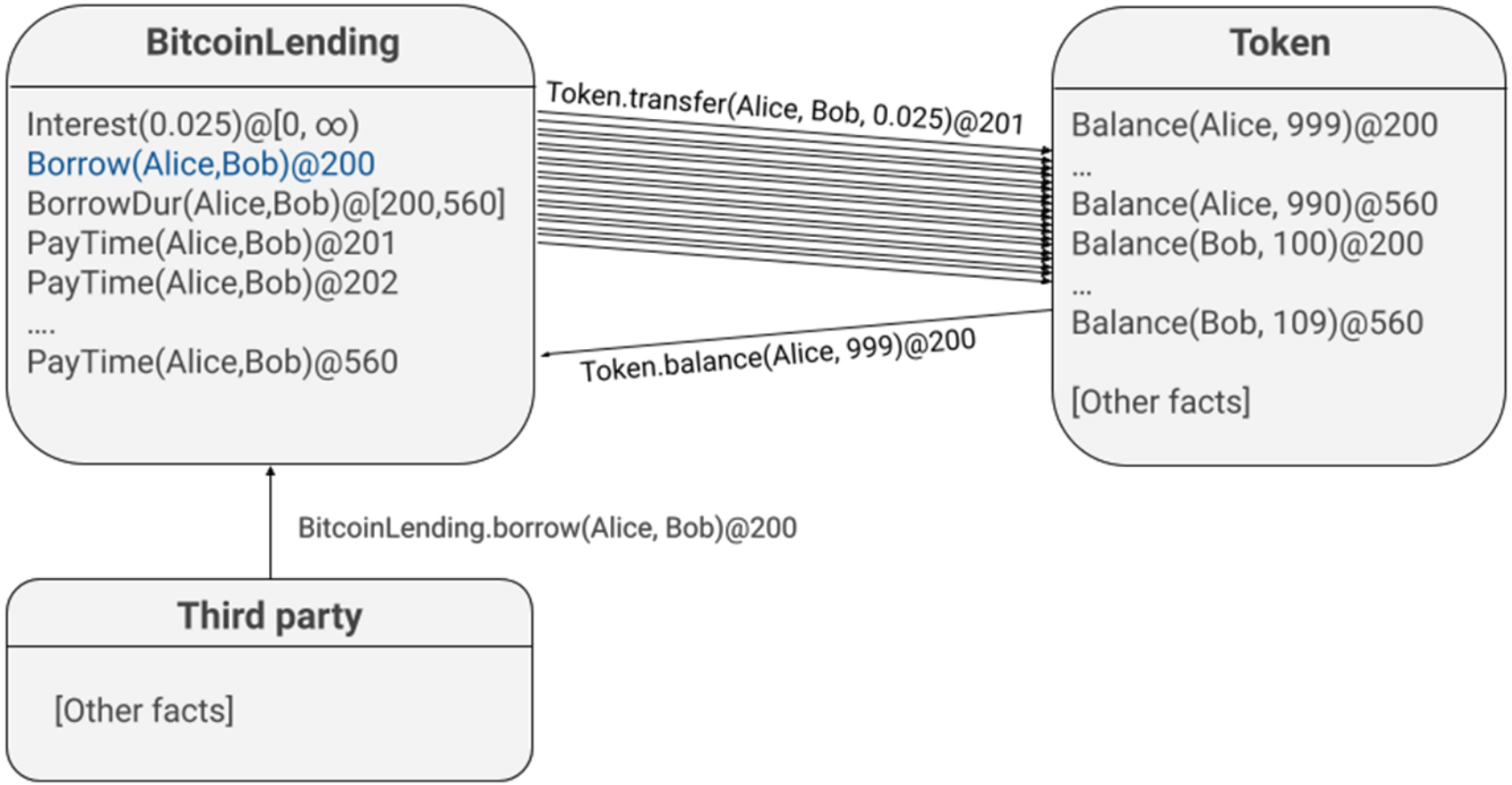
TEMPORAL

iTemporal: An Extensible Generator of Temporal Benchmarks

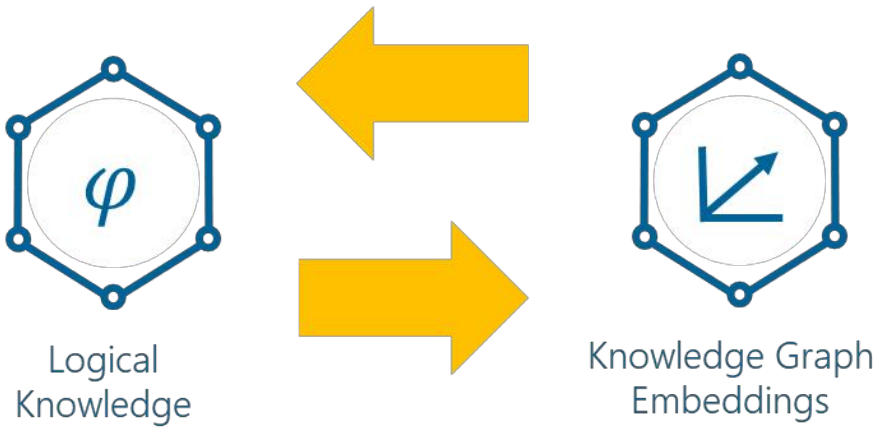


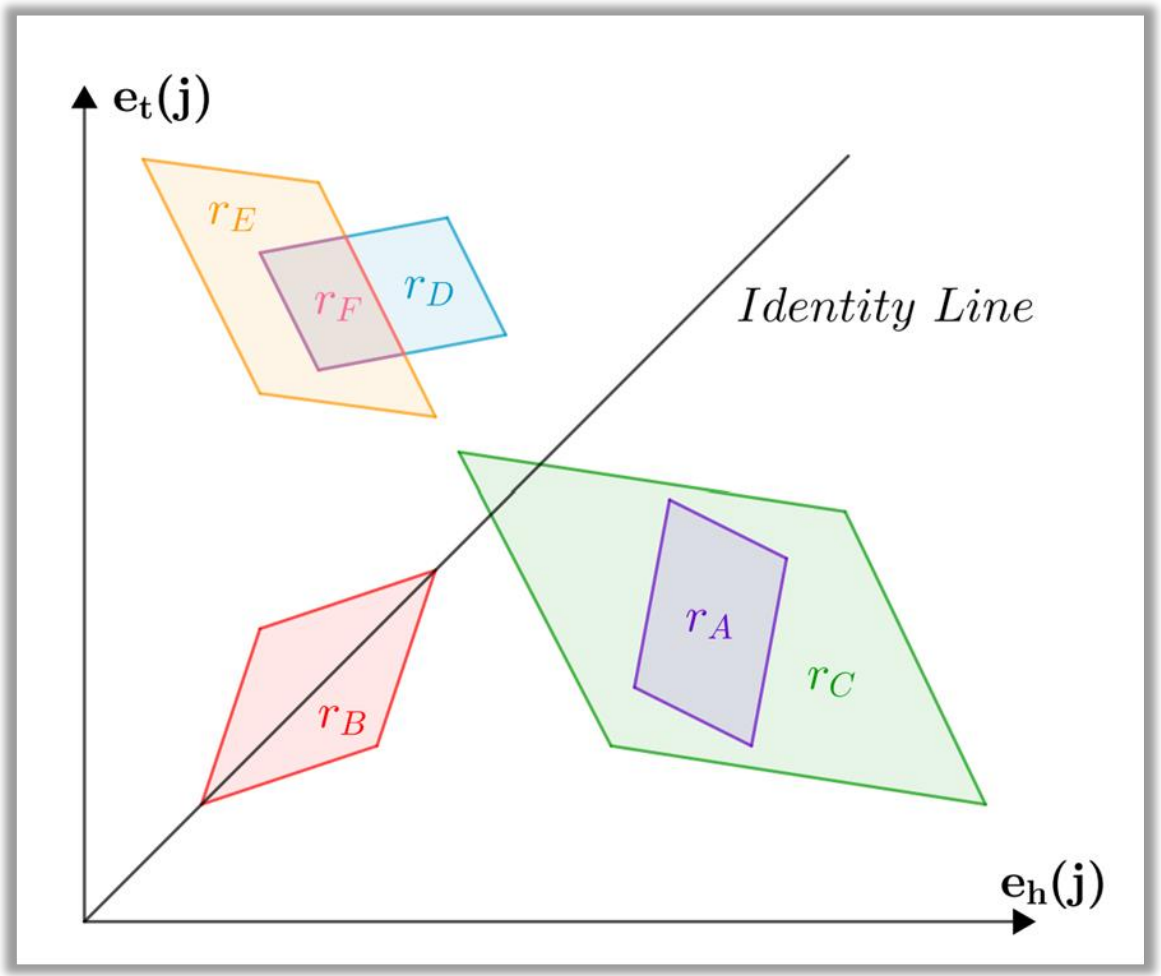


Modelling Smart Contracts with Temporal Datalog



“ExpressivE”: Expressive Knowledge Graph Embeddings





Media Intelligence

Meltwater

UNIVERSITY OF OXFORD

TU WIEN