



Ontology Engineering

Knut Hinkelmann

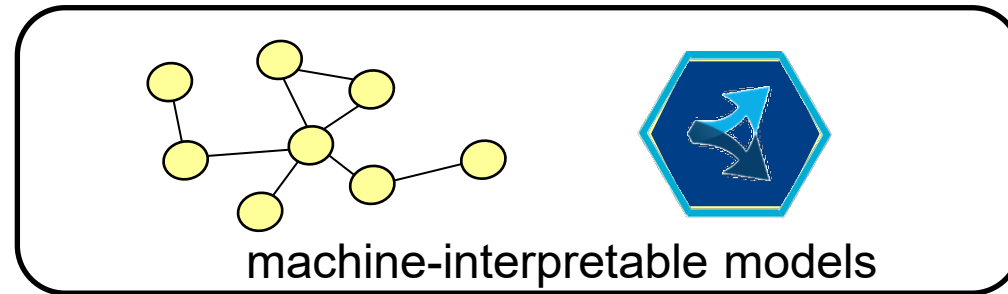


Knowledge-Representation and Reasoning

Reasoning/Inference



Knowledge Base



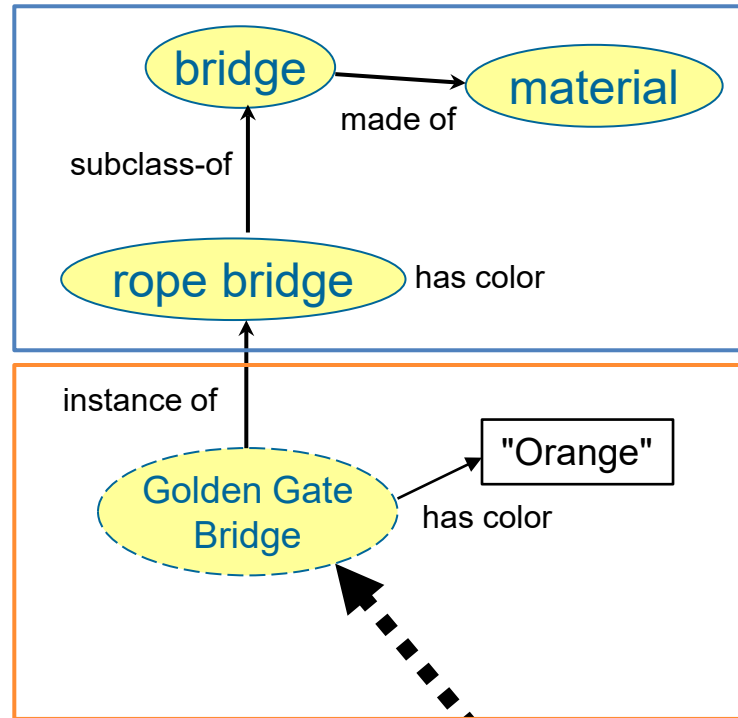
Reality



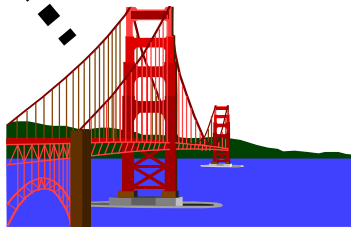
An Ontology – very informal

An ontology is a formal explicit description of concepts in a domain of discourse

- An **ontology** consists of
 - ◆ Concepts (Classes),
 - ◆ Relationships (Object Properties) between concepts
 - ◆ Attributes (Data Properties) of concepts
 - ◆ Constraints that hold between/for the concepts,
- An ontology together with a set of individual instances constitutes a **knowledge base**

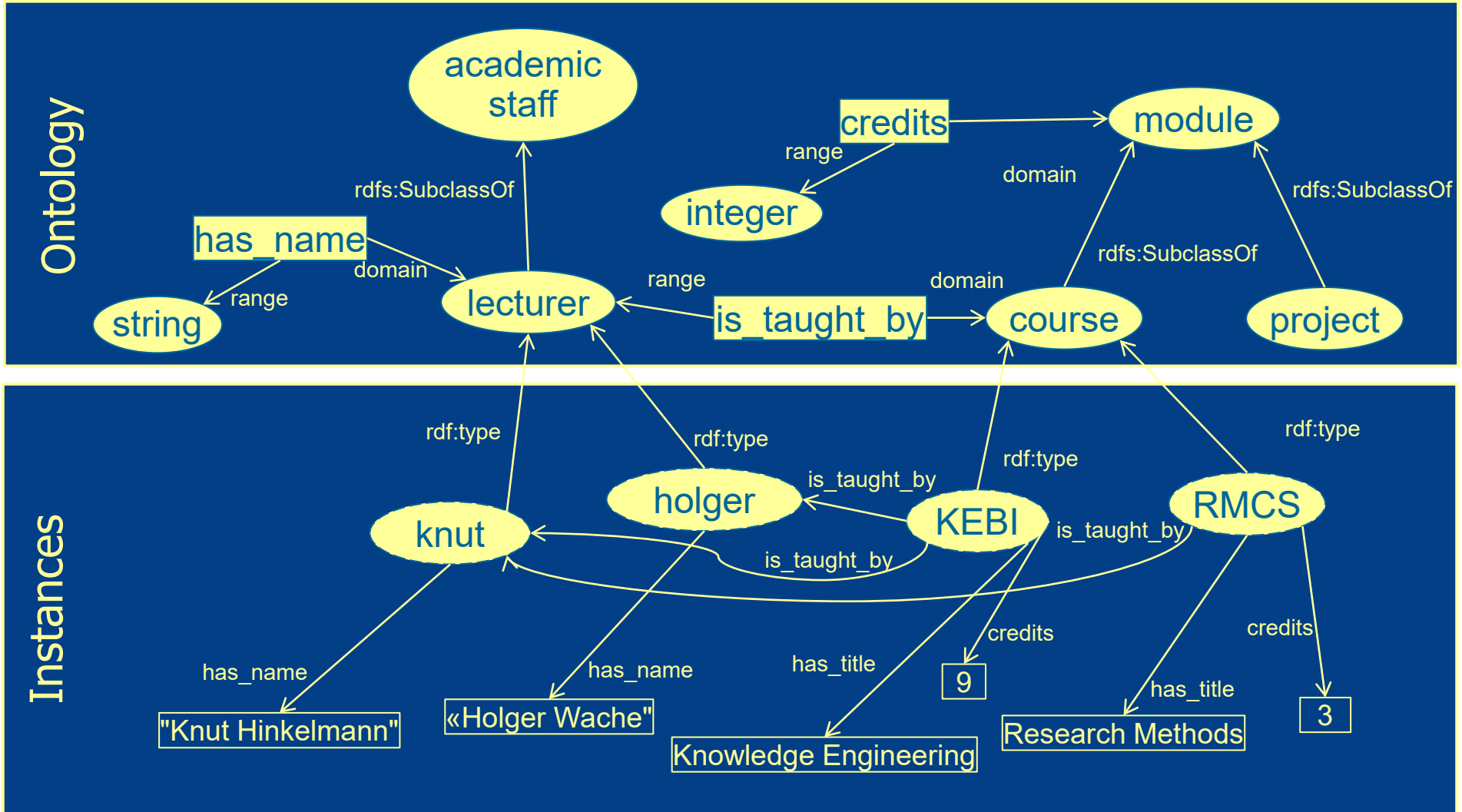


real object



ontology engineering
is
knowledge engineering

Example of an Ontology



Ontology Representation Formalisms

■ Representations of Ontologies

◆ ***RDF(S)***

← Our focus


◆ OWL

◆ Neo4J

◆ ...

Tools: Examples of Programming Libraries

EasyRDF for PHP: <https://www.easyrdf.org/>



**EASY
RDF**

A PHP library designed to make it easy to **consume** and **produce** RDF.

Designed for use in mixed teams of experienced and inexperienced RDF developers. Written in **PSR-12** compliant PHP and tested extensively using **PHPUnit**.

[Getting Started >](#)

Apache Jena for Java: <https://jena.apache.org/>



Apache Jena

A free and open source Java framework for building **Semantic Web** and **Linked Data** applications.

[> Get started now!](#) [Download](#)

RDFLib for Python: <https://rdflib.readthedocs.io/en/stable/>




Table of Contents

- rdflib 5.0.0
- Getting started
- In depth
- Reference
- For developers
- The Code
- Further help

rdflib 5.0.0

RDFLib is a pure Python package for working with RDF. RDFLib contains useful APIs for working with RDF, including:

- **Parsers & Serializers**
 - for RDF/XML, N3, NTriples, N-Quads, Turtle, TriX, RDFa and Microdata
 - and JSON-LD, via a plugin module
- **Store implementations**
 - for in-memory and persistent RDF storage - Berkeley DB
- **Graph interface**
 - to a single graph
 - or a conjunctive graph (multiple Named Graphs)
 - or a dataset of graphs
- **SPARQL 1.1 implementation**
 - supporting both Queries and Updates

RDF

RDF API

Interact with the core API to create and read **Resource Description Framework (RDF)** graphs. Serialise your triples using popular formats such as **RDF/XML** or **Turtle**.

ARQ (SPARQL)

Query your RDF data using ARQ, a **SPARQL 1.1** compliant engine. ARQ supports remote federated queries and free text search.

Triple store

TDB

Persist your data using TDB, a native high performance triple store. TDB supports the full range of Jena APIs.

Fuseki

Expose your triples as a SPARQL end-point accessible over HTTP. Fuseki provides REST-style interaction with your RDF data.

OWL

Ontology API

Work with models, RDFS and the **Web Ontology Language (OWL)** to add extra semantics to your RDF data.

Inference API

Reason over your data to expand and check the content of your triple store. Configure your own inference rules or use the built-in OWL and RDFS reasoners.

Tool: Ontology Engineering

<https://protege.stanford.edu/>



PRODUCTS

SUPPORT

COMMUNITY

ABOUT

A free, open-source ontology editor and framework for building intelligent systems

Protégé is supported by a strong community of academic, government, and corporate users, who use Protégé to build knowledge-based solutions in areas as diverse as biomedicine, e-commerce, and organizational modeling.

DOWNLOAD NOW

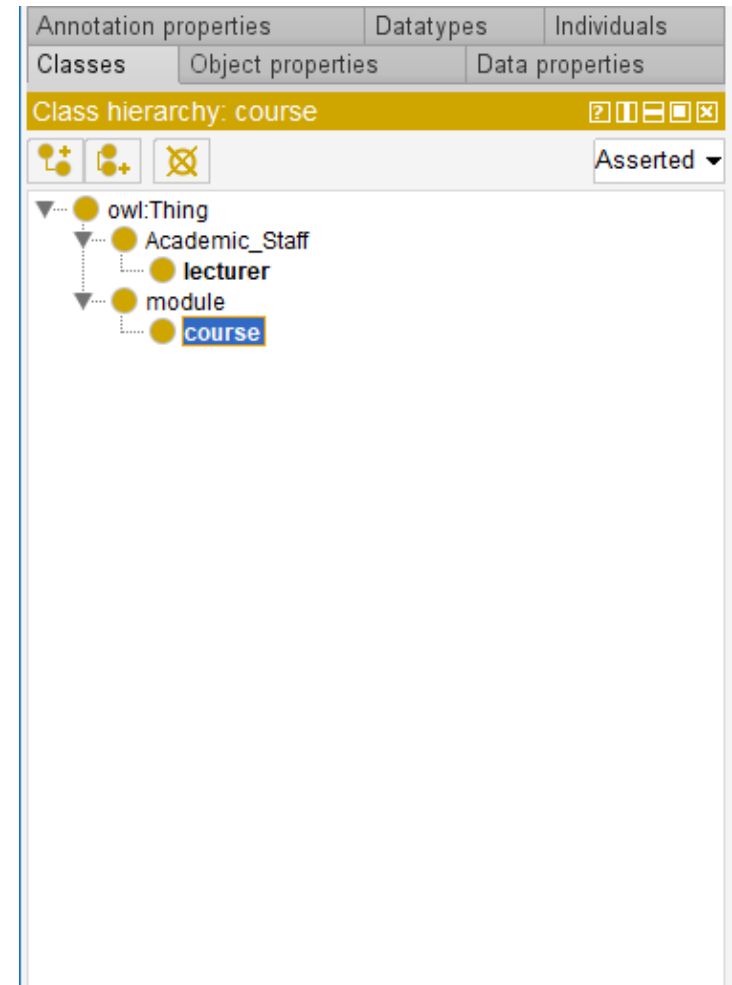
USE WEBPROTÉGÉ

Creating an Ontology

- Defining classes in the ontology
- Arranging the classes in a taxonomic (subclass-superclass) hierarchy
- Defining properties and describing allowed values for the properties
- Creating instances and filling the values for properties

Define Classes and Class Hierarchy

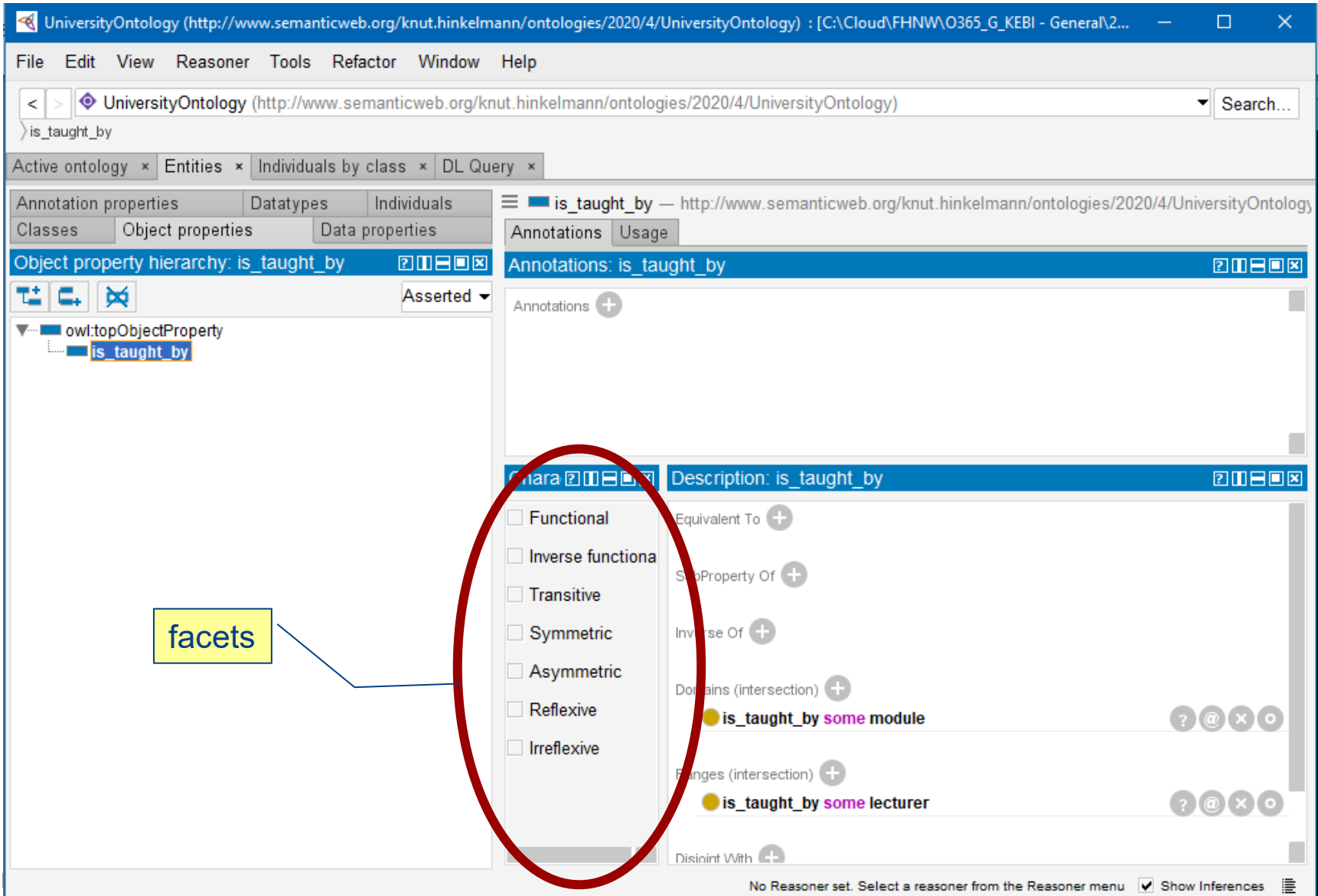
- There are several approaches
 - ◆ Top-down: Start with the most general concept, and work your way down
 - ◆ Bottom-up: Start with the most specific, and work your way up
 - ◆ Combination



Define Properties of Classes

- Describe the internal structure of concepts
 - ◆ Data Properties: Attributes
 - Range are data types like String, Integer, ...
 - ◆ Object Properties: Relations to other concepts
 - Range are Classes
- Describe facets: Characteristics of Properties
- Inheritance to Subclasses

Object Property



UniversityOntology (http://www.semanticweb.org/knut.hinkelmann/ontologies/2020/4/UniversityOntology) : [C:\Cloud\FHNW\O365_G_KEBI - General\2...

File Edit View Reasoner Tools Refactor Window Help

UniversityOntology (http://www.semanticweb.org/knut.hinkelmann/ontologies/2020/4/UniversityOntology) Search...

is_taught_by

Active ontology x Entities x Individuals by class x DL Query x

Annotation properties Datatypes Individuals

Classes Object properties Data properties

Object property hierarchy: is_taught_by

- owl:topObjectProperty
 - is_taught_by

Annotations Usage

Annotations: is_taught_by

Annotations +

Description: is_taught_by

- Functional
- Inverse functiona
- Transitive
- Symmetric
- Asymmetric
- Reflexive
- Irreflexive

Equivalent To +

SubProperty Of +

Inverse Of +

Domains (intersection) +

- is_taught_by some module

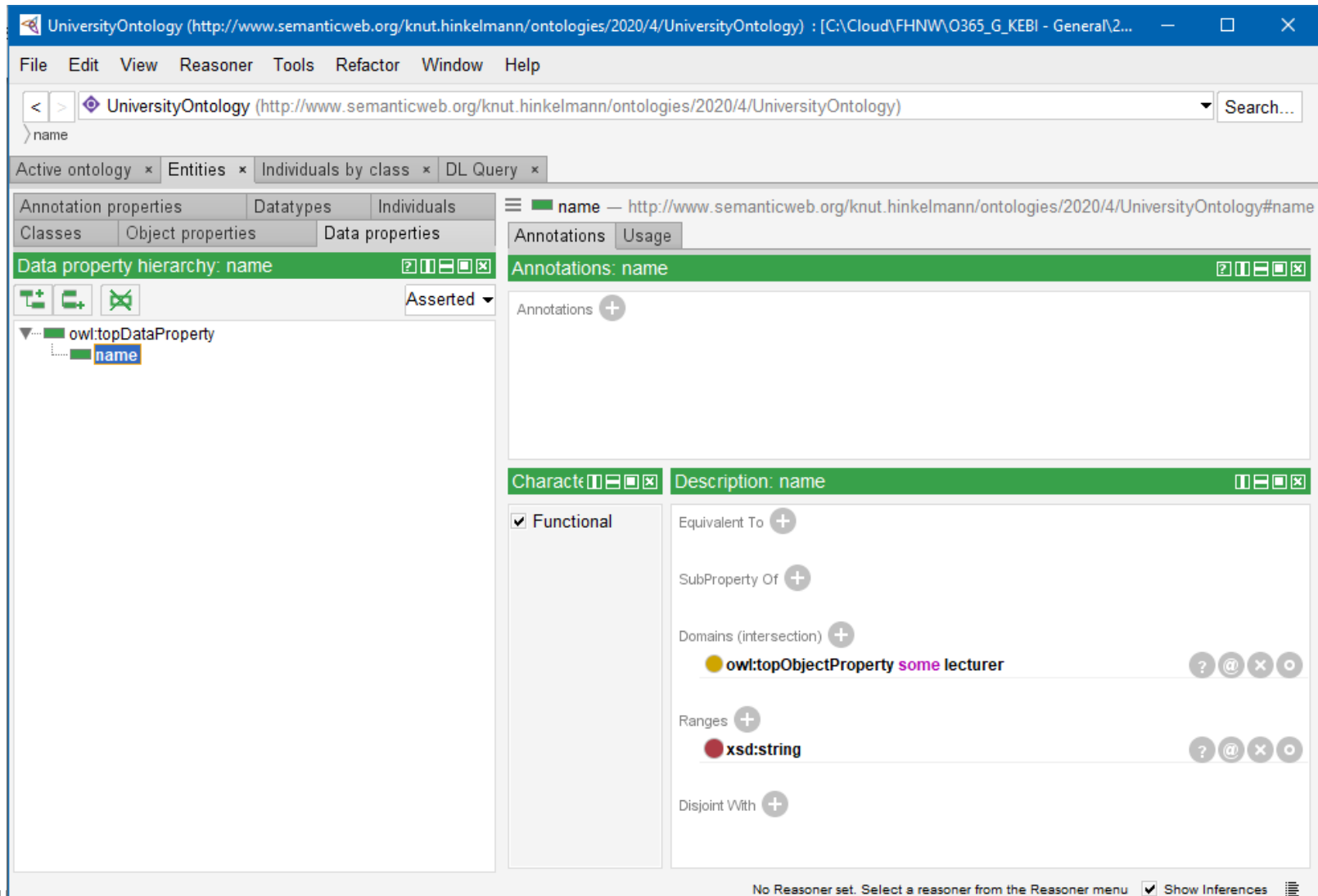
Ranges (intersection) +

- is_taught_by some lecturer

Disjoint With +

No Reasoner set. Select a reasoner from the Reasoner menu Show Inferences

Data Property

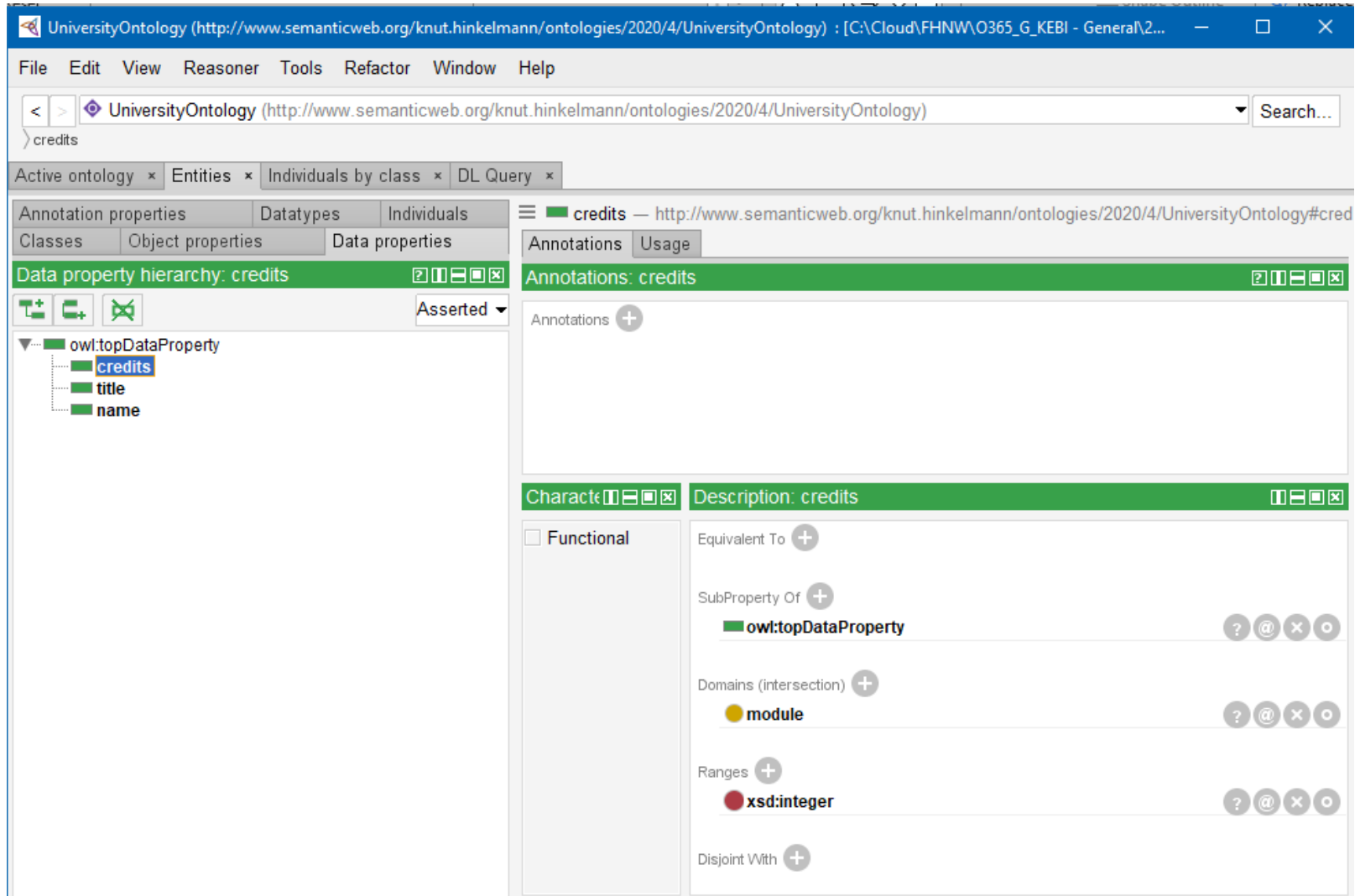


The screenshot shows the Protege interface for editing the 'name' data property in the 'UniversityOntology'.

- Left Panel:** Shows the 'Data property hierarchy: name' with 'owl:topDataProperty' expanded to show 'name'.
- Top Panel:** Shows the property URI: `http://www.semanticweb.org/knut.hinkelmann/ontologies/2020/4/UniversityOntology#name`.
- Annotations:** An empty list with a '+' button to add annotations.
- Description:**
 - Functional**
 - Equivalent To: +
 - SubProperty Of: +
 - Domains (intersection): +
 - owl:topObjectProperty some lecturer** (with ? @ x o icons)
 - Ranges: +
 - xsd:string** (with ? @ x o icons)
 - Disjoint With: +

At the bottom, it indicates 'No Reasoner set. Select a reasoner from the Reasoner menu' and has a checked 'Show Inferences' option.

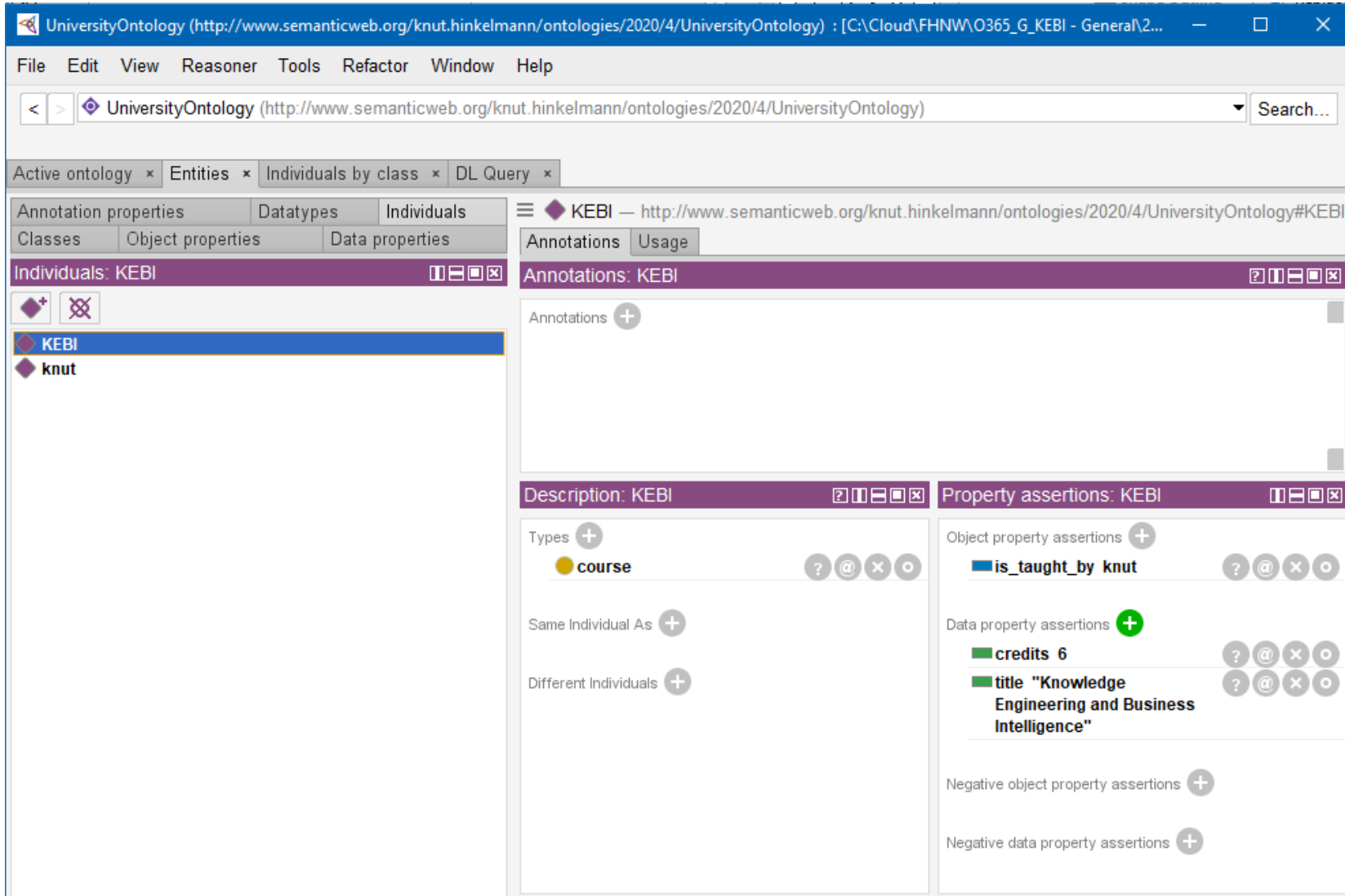
Data Property



The screenshot shows the UniversityOntology web interface. The browser address bar displays the URL: `http://www.semanticweb.org/knut.hinkelmann/ontologies/2020/4/UniversityOntology`. The interface includes a menu bar (File, Edit, View, Reasoner, Tools, Refactor, Window, Help) and a search bar. The main content area is divided into several panels:

- Data property hierarchy: credits:** A tree view showing the hierarchy of data properties. The root is `owl:topDataProperty`, which has three children: `credits` (highlighted in blue), `title`, and `name`.
- Annotations: credits:** A panel for managing annotations for the `credits` property. It currently shows no annotations.
- Description: credits:** A panel for defining the description of the `credits` property. It includes a **Functional** checkbox (unchecked) and several relationship sections:
 - Equivalent To:** Empty.
 - SubProperty Of:** Contains `owl:topDataProperty`.
 - Domains (intersection):** Contains `module`.
 - Ranges:** Contains `xsd:integer`.
 - Disjoint With:** Empty.

Instances



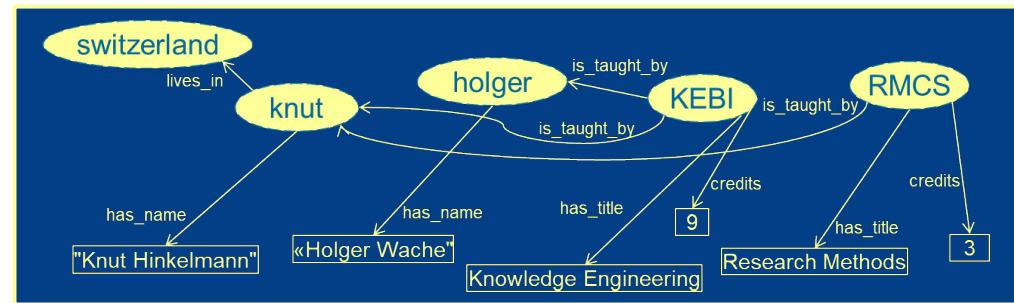
The screenshot shows a web browser window displaying the 'UniversityOntology' (http://www.semanticweb.org/knut.hinkelmann/ontologies/2020/4/UniversityOntology). The interface includes a menu bar (File, Edit, View, Reasoner, Tools, Refactor, Window, Help) and a search bar. The main content area is divided into several panels:

- Active ontology:** UniversityOntology
- Entities:** Individuals by class
- Annotation properties:** Classes, Object properties, Data properties
- Datatypes:** (empty)
- Individuals:** KEBI, knut
- Annotations: KEBI:** (empty)
- Description: KEBI:**
 - Types: course
 - Same Individual As: (empty)
 - Different Individuals: (empty)
- Property assertions: KEBI:**
 - Object property assertions: is_taught_by knut
 - Data property assertions: credits 6, title "Knowledge Engineering and Business Intelligence"
 - Negative object property assertions: (empty)
 - Negative data property assertions: (empty)

Exercise

- Add new class: country
- Add a property: A lecturer lives in a country
- Add new instance: Knut lives in Switzerland
- Add new classes and properties for the following knowledge
 - ◆ A project is a module
 - ◆ A Master Thesis is a project
 - ◆ Supervisor is a lecturer
 - ◆ A project has a supervisor
 - ◆ A project is performed by a student
- Add new instances
 - ◆ Giordano is a student who is performing a master thesis that is supervised by knut

Querying an Ontology



- Queries are mostly about navigating the graph in search of some patterns
- Sample types of queries
 - ◆ Navigating along a graph path, e.g. who are the lecturers of KEBI
`SELECT ?x WHERE { :KEBI :is_taught_by ?x }`
 - ◆ Navigating along a graph path with intermediate values, e.g. what are the names of the lecturers of KEBI
`SELECT ?x ?y WHERE { :KEBI :is_taught_by ?x.
?x :has_name ?y }`
 - ◆ Navigating a path in reverse, e.g. which modules is knut teaching
`SELECT ?x WHERE { ?x :is_taught_by :knut }`
 - ◆ Discover relationships, what is the relationship between KEBI and knut
`SELECT ?rel WHERE { :KEBI ?rel :knut }`
 - ◆ Chain of relationships, what chain exists between KEBI and switzerland
`SELECT ?rel1 ?y ?rel2 WHERE { :KEBI ?rel1 ?y.
?y ?rel2 :switzerland }`

Querying

- Query Language: SPARQL

- ◆ Variables: ?x

- Elements are denoted as URI

- ◆ Prefixes for Abbreviations

- Example: PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

- Sample query: Select all lecturers:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX uo: <http://www.semanticweb.org/knut.hinkelmann/ontologies/2020/4/UniversityOntology#>
SELECT ?subject
      WHERE { ?subject rdf:type uo:lecturer }
```

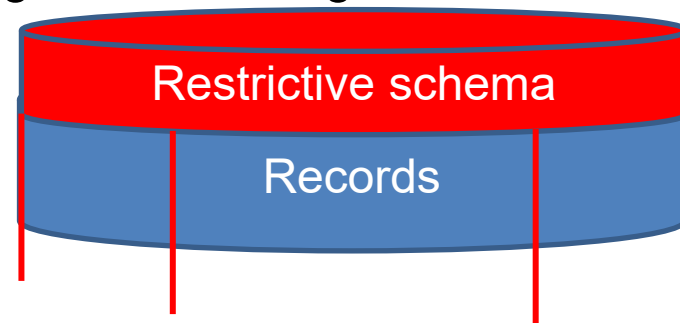
SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?instance
      WHERE { ?instance rdf:type lecturer }
```

RDF Graphs vs Databases

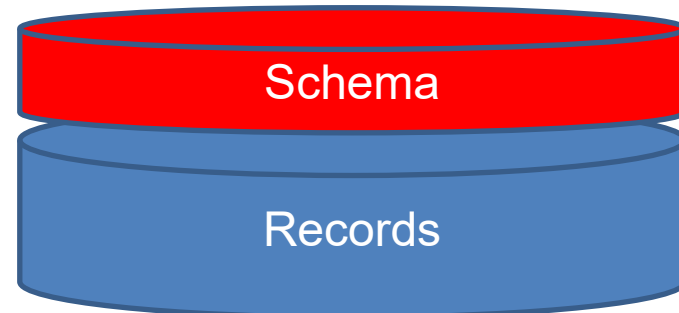
Schema is decoupled from data!

In SQL databases, you cannot do anything before having a schema (the "DB structure")



In RDF graphs, ***schema is decoupled from "records"***

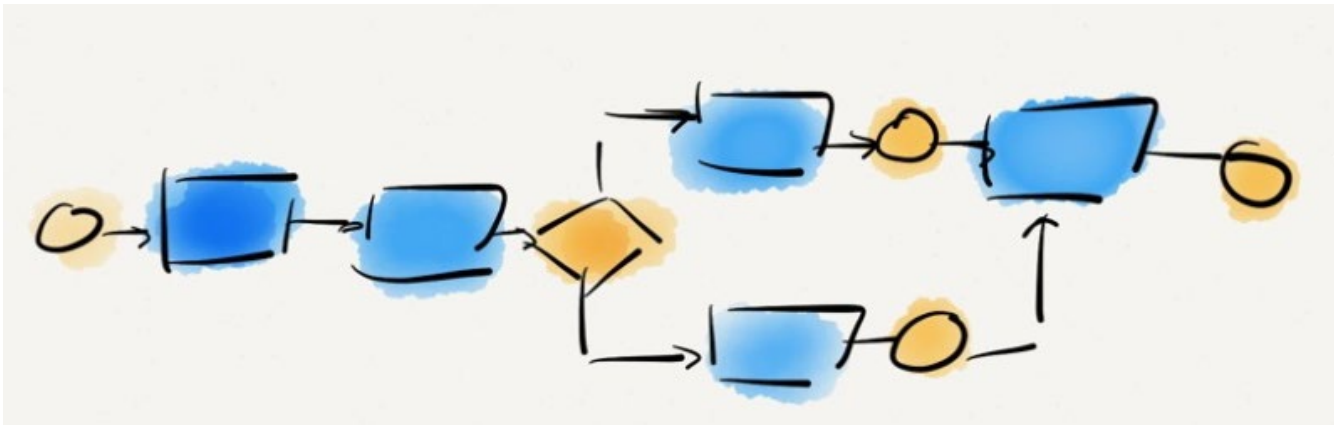
- Schema can be created after data
- Schema is optional (data can be queried in the absence of a schema)



Exercise: Modeling Process Knowledge in an Ontology

Exercise: Modeling Process Knowledge in an Ontology

- We create a knowledge base for process knowledge
 - ◆ Define the ontology
 - ◆ Represent knowledge of a process



Ontology Development 101

- 1 • Determine the domain and scope of the ontology
- 2 • Consider reusing existing ontologies
- 3 • Enumerate important terms
- 4 • Define classes and class hierarchy
- 5 • Define the data and object properties of classes
- 6 • Define the facets of properties
- 7 • Create instances

Determine the domain and scope of the ontology

- What is the domain that the ontology will cover?
- For what we are going to use the ontology?
- For what types of questions the information in the ontology should provide answers? → Competency questions
- Who will use and maintain the ontology?

Competency Questions

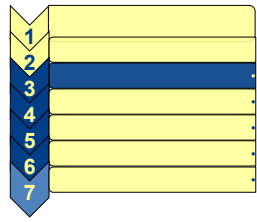
- One of the ways to determine the scope of the ontology is to sketch a list of questions that a knowledge base based on the ontology should be able to answer (Gruninger and Fox 1995)
 - ◆ Does the ontology contain enough information to answer these types of questions?
 - ◆ Do the answers require a particular level of detail or representation of a particular area?

- Exercise: We want to represent knowledge about
 - ◆ the process flow
 - ◆ Responsibilities for tasks
- Competency Questions:
 - Who executes task X?
 - Which task is executed after task X?
 - When can task X start?
- Sample process:

The waiter serves the beverages. Then the waiter serves the food. When the guests are finished, the waiter presents the bill.

Consider reusing existing ontologies

- It is always worth considering what others have done, and check if their work can be refined and extended for our particular domain and task
- Mandatory if the system needs to interact with other applications that have already committed to particular ontologies or controlled vocabularies



Enumerate important terms in the ontology

- What are the terms we would like to talk about?
- What are their properties?
- What would we like to say about those terms?

Define Classes and Class Hierarchy

- Several possible approaches in developing a class hierarchy:
 - ◆ Top-down: General to specific concepts
 - ◆ Bottom-up: Specific to general concepts
 - ◆ Combination: Salient to general and specific concepts
- Classes for
 - ◆ Modeling Objects
 - ◆ Relations

Define the properties of classes

- Describe the internal structure of concepts
 - ◆ Data Properties: Attributes
 - Range are data types like String, Integer, ...
 - ◆ Object Properties: Relations to other concepts
 - Range are Classes
- Inheritance to Subclasses

Create Instances

- Model a business process in an ontology

The waiter serves the beverages. Then the waiter serves the food. When the guests are finished, the waiter presents the bill.

Modeling Business Processes as graphical models is more adequate

