

Knowledge Graphs

ANNA QUERALT, OSCAR ROMERO

(FACULTAT D'INFORMÀTICA DE BARCELONA)

The Overall Idea

can be referred universally

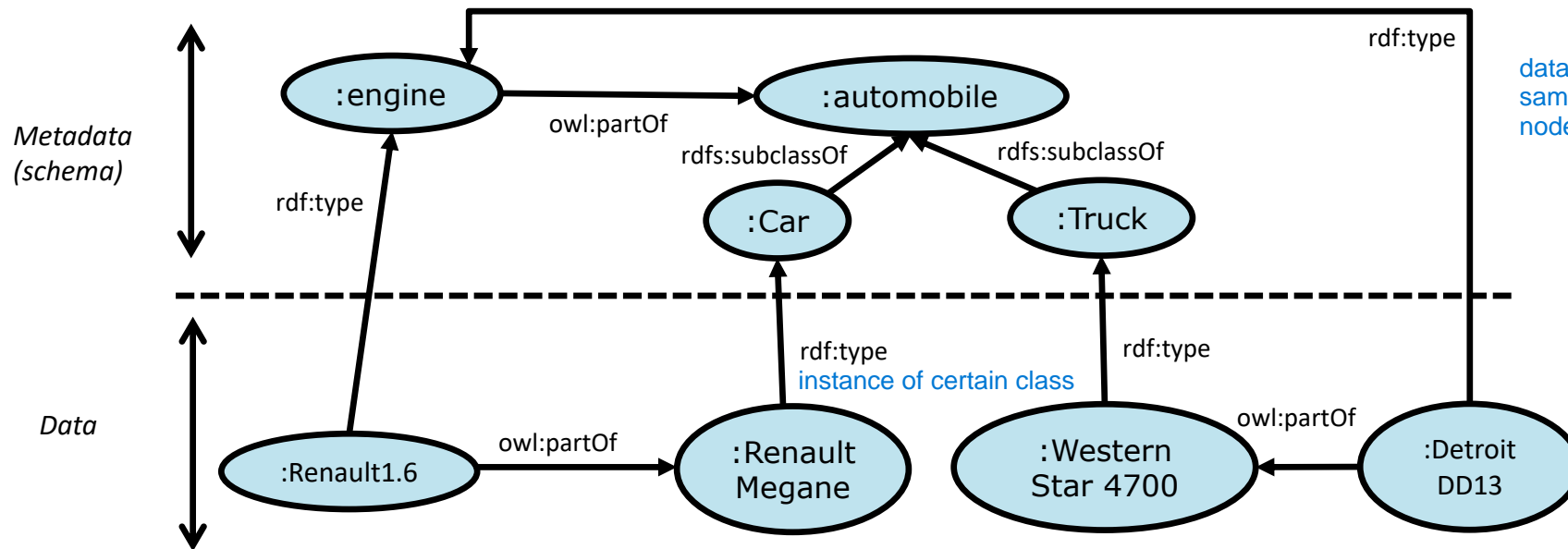
:x is universal identifier

Given we have KG in two different places, we can refer to one another

It is a graph! But...

- Every node is represented with a unique identifier and can be **universally** referred
- *Metadata* is represented as nodes and edges (no special construct such as labels or attributes)
 - Pre-defined identifiers (i.e., nodes and edges) set semantics: pre-defined vocabularies (e.g., to distinguish data and metadata)

To add metadata in property graph, we add it as label or property but in knowledge graph there is no label and property



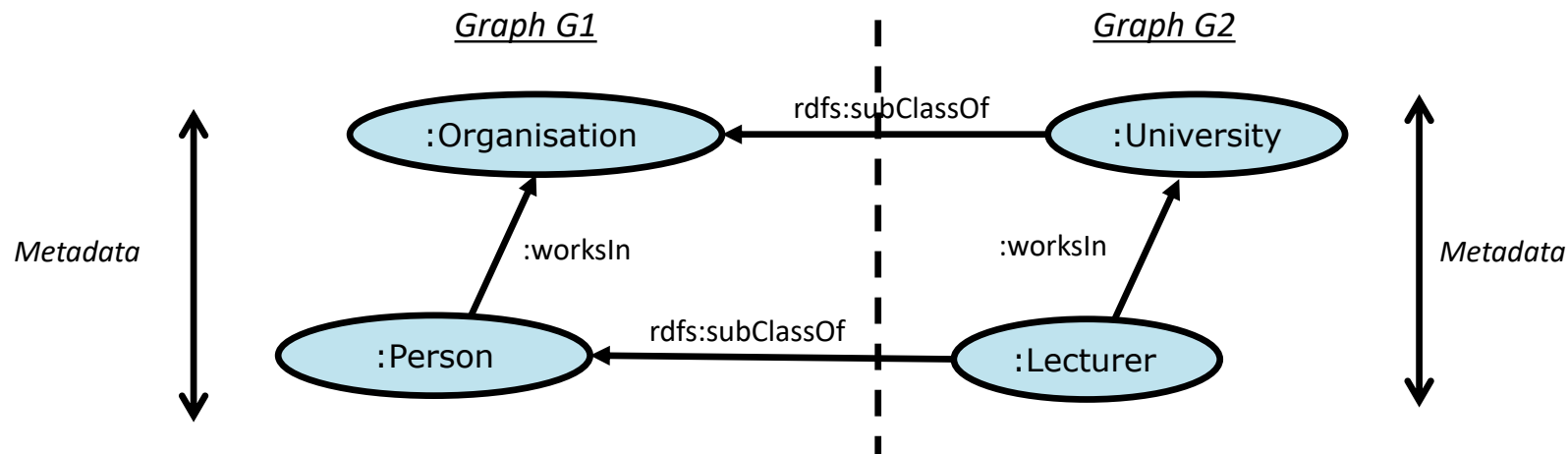
Can you think of any data model allowing you to model data and metadata (schema) with the same constructs?

we always distinguish schema from data

The Power of Knowledge Graphs

Knowledge graphs facilitate **relating** (*linking, crossing, integrating*) data

- Linking metadata is way more powerful than linking data: a unique feature of their own



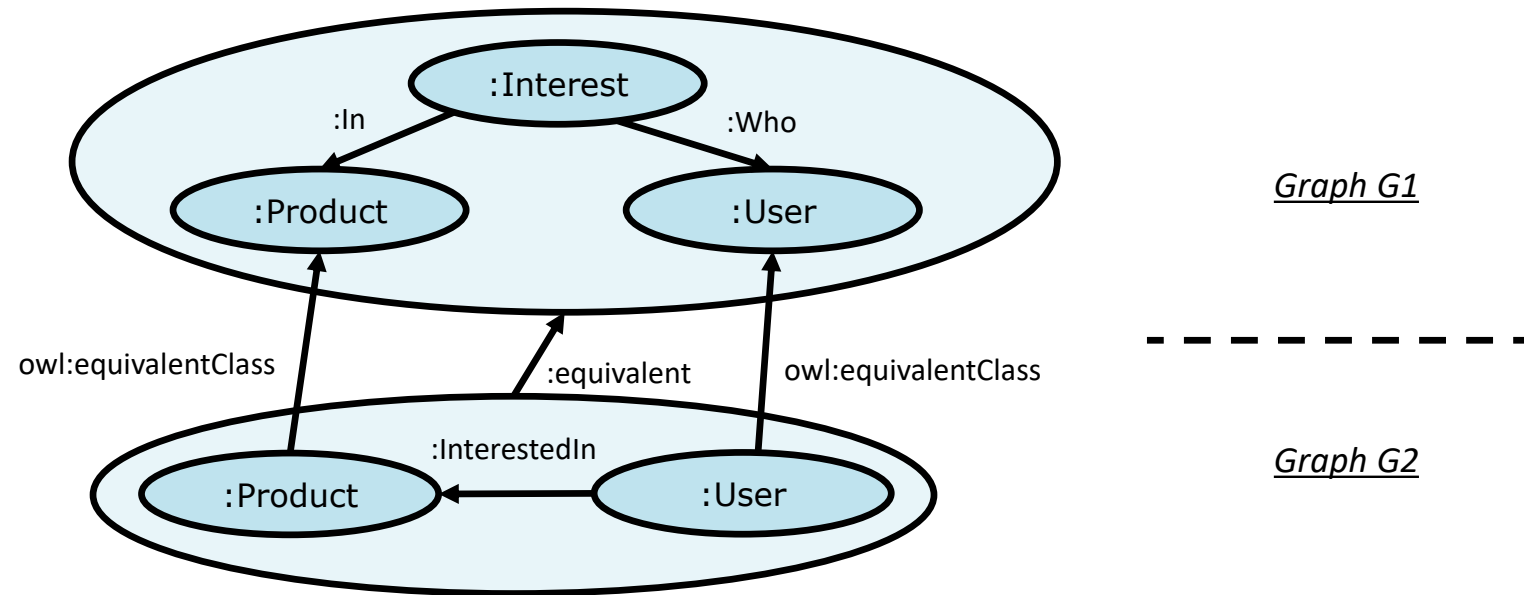
How to do this in a property graph?

The Power of Knowledge Graphs

Knowledge graphs facilitate **relating** (*linking, crossing, integrating*) data

- Linking metadata is way more powerful than linking data: a unique feature of their own

But KGs are even more flexible than that...



Activity: Using KGs to Integrate Data

Objective: Understand the knowledge graph data model [in notebook](#)

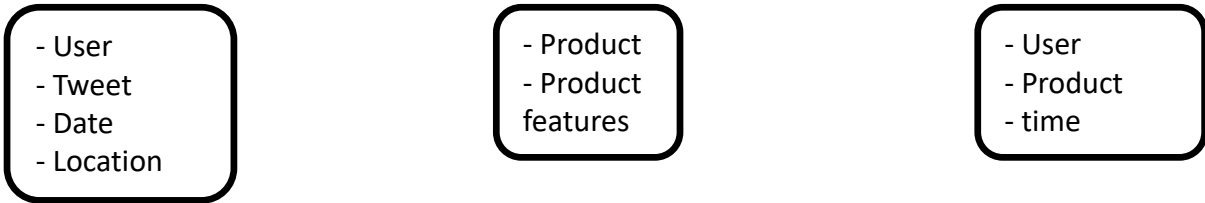
Tasks:

1. With a teammate think of the following:

I. Assume KGs as canonical data model

II. First, model as graphs each source (separately):

I. Model schema (metadata) and some instances for each source



- User
- Tweet
- Date
- Location

- Product
- Product
features

- User
- Product
- time

III. Now, relate the metadata from each graph with new edges generating a unique connected graph

I. Look for similar or identical concepts

II. Think of **interesting relationships** you could exploit later

Note: assume the following pre-defined edges: `owl:equivalentClass`, `rdf:type` and `rdfs:subClassOf`, which embed the semantics already discussed. Anyhow, feel free to create new ones if required (justify their need)

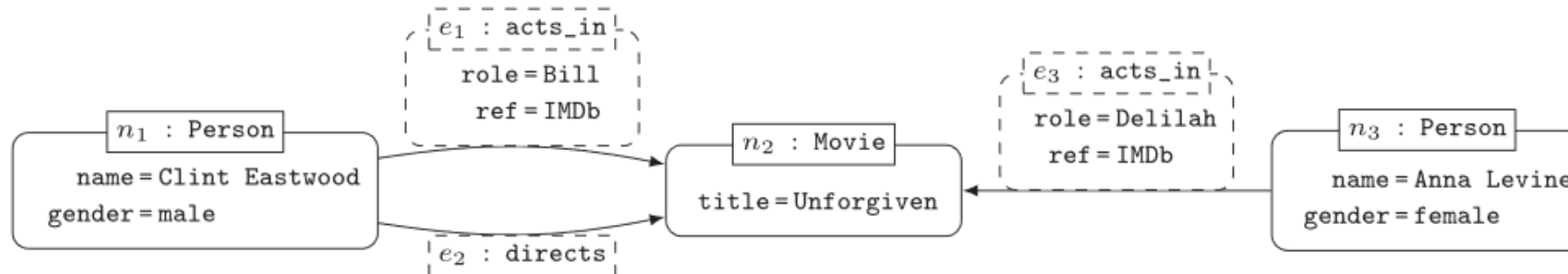
Activity: Property Vs. Knowledge Graphs

Objective: understand the differences between both graph data models

Tasks:

- I. Consider the property graph below*
- II. Represent it in the form of a knowledge graph*
 - I. What are the main differences between both formalisms?*
 - II. What are the similarities?*

An example of Property Graph:



Semantics as Cornerstone

Schema.org is a global initiative to mark up (i.e., attach semantics to) data [initiative to bring semantic to data - semantic.org](https://schema.org/)

- It provides a pre-defined vocabulary of terms (concepts: i.e., nodes and edges) and their relationships:
<https://schema.org/docs/full.html>

Example:

```
<div itemscope  
  itemtype="https://schema.org/Offer">
```

The element markup starts by
typing it

```
<span itemprop="name">Blend-O-Matic</span>
```

```
<span itemprop="price">$19.95</span>
```

State properties (attrs)
of the element

```
<link itemprop="availability"
```

```
  href="https://schema.org/InStock"/>Available  
today! </div>
```

Data values can be
constrained with
enumerations

Schema.org and Knowledge Graphs are the foundations on which the Google Knowledge Graph was built on: <https://blog.google/products/search/introducing-knowledge-graph-things-not/>

Knowledge Graph Families

W3C Standards for *sharing meaning (things, not strings)*

- RDF (Resource Description Framework)
 - Introduces the basic data structure: <subject predicate object>
- RDFS (RDF Schema)
 - Predefined vocabulary to assert basic schema information and distinguish it from instances
- OWL / Ontologies
 - Richer conceptualizations (i.e., richer schema-related information)

RDF - basic construct
put constraint on top of rdf

RDF

THE FOUNDATIONS OF KNOWLEDGE GRAPHS

Disclaimer

Even if knowledge graphs were born in the Semantic Web (c.f., to fulfill the concept of Linked Data) we are going to use them for a broader range of purposes

DO NOT UNDERSTAND RDF AS SYNONYM OF THE SEMANTIC WEB

RDF: Resource Description Format

The basic RDF block is the **RDF statement**:

It is a ***triple*** representing: a *binary relationship* between two *resources*
or between a *resource* and a *literal*

language to define RDF statement
RDF triple (subject, predicate and object)
in principle, all of them should be universal identifier
<:S,:P,:O>
Universal Identifier - URL +URN
: sth means URL:URN

object can also be constant (literals)

<subject predicate object>

- Subject S has value object O for predicate P
- Subject and predicate are resources and must be **URIs**
- Object can be a resource (URI) or a *literal* (i.e., a constant value)

Resources (or objects) are identified by URIs (**global identifiers**): URL+URN

- URN (Universal Resource Name) ~ id
- URL (Universal Resource Location) ~ *where* to locate it
- Many times, we omit the URL for simplicity and refer to the URI as **:URN**
- Blank nodes are resources without a defined URL (i.e., **_**)
- Literals are atomic values such as string, dates, numbers, etc.

Artificial node created
_:b 0 Express url without uri
blank node - empty URL

A set of triples form an RDF graph

- RDF graphs are also known as *semantic graphs*

basic knowledge graph family
because others are created on top of this

RDF: Resource Description Format

The basic RDF block is the **RDF statement**:

It is a **triple** representing: a *binary relationship* between two *resources*
or between a *resource* and a *literal*

It is a simple language for describing *annotations* (facts) about resources

- The most basic **ontology language** way to describe knowledge
- Triples map to First Order Logic as grounded atomic formulas (subject and object are constants)
- Blank nodes map to existential variables

Later in the course we will formally define ontologies and their relationship with logics

Querying:

- **SPARQL** is the de facto language to query RDF
- It is also the standard to query RDF variants / extensions
- Inspired in SQL but oriented to express graph operations (*hello pattern matching!*)

once rdf triple is created, there should be language to query it
SPARQL
SPARQL is way to query any knowledge graph family
has SQL like syntax + express pattern matching on top of it

RDF: The Big Picture

Semantics: <http://www.w3.org/TR/rdf11-primer/>

- **Triples** (i.e., RDF statements),
- To create **semantic graphs** (aka RDF graphs)

Syntax

- **XML-based serialization** (aka RDF syntax)
 - The way to express RDF triples in a machine-processable format

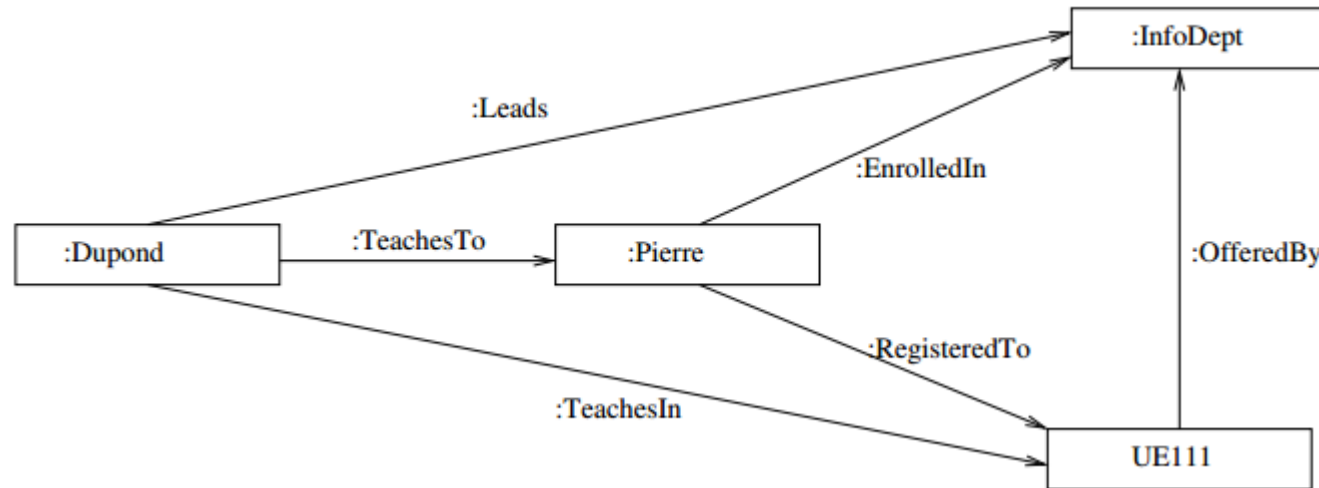
RDF Example: Semantics

RDF triplets or statements:

```
< :Dupond :Leads :CSDept >  
< :Dupond :TeachesIn :UE111 >  
< :Dupond :TeachesTo :Pierre >  
< :Pierre :EnrolledIn :CSDept >  
< :Pierre :RegisteredTo :UE111 >  
< :UE111 :OfferedBy :CSDept >
```

Subject	Predicate	Object
:Dupond	:Leads	:CSDept
:Dupond	:TeachesIn	:UE111
:Dupond	:TeachesTo	:Pierre
:Pierre	:EnrolledIn	:CSDept
:Pierre	:RegisteredTo	:UE111
:UE111	:OfferedBy	:CSDept

RDF graph:



RDF Example: Syntax

not generally used
Use library
RDFlib is for python

RDF was originally serialized as XML (syntax)

The rdf URL is the namespace for RDF (<http://www.w3.org/1999/02/22-rdf-syntax-ns#>)

Example:

```
<?xml version="1.0"?>

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:cd="http://www.recshop.fake/cd#">

  <rdf:Description
    rdf:about="http://www.recshop.fake/cd/Empire Burlesque"> Subject
    <cd:artist>Bob Dylan</cd:artist>
    <cd:country>USA</cd:country>
    <cd:company>Columbia</cd:company>
    <cd:price>10.90</cd:price>
    <cd:year>1985</cd:year>
  </rdf:Description>
```

Predicate + Object for same subject

ways to serialize triple

Other RDF Syntaxes:

- **Turtle** (human-readable)
- N-triples
- Notation 3
- ...

In turtle anything with URL is resource

Turtle:

```
(...)
<#empire-burlesque>
  cd:artist <#Bob-Dylan> ;
  cd:country <#USA> ;
  cd:price 10.90 ;
  cd:year 1985 .
(...) . says we need to add subject
in next line
```

<https://www.w3.org/TR/turtle/>

, - we only need to specify object

Activity: Modeling in RDF

Objective: Grasp the main idea behind RDF modeling

Model a RDF Graph about lecturers, courses and students:

A student enrolls several courses from her faculty per semester. He is forced to enroll, at least, one course per semester. Each course has one responsible lecturer but potentially, it might have several lecturers

Your graph must be a correct RDF graph and therefore defining the namespaces used

Afterwards, write the triples you created in Turtle syntax

Basics on RDF Modeling

RDF modeling is based on binary relationships artificial node added to represent more than 2 edges involved

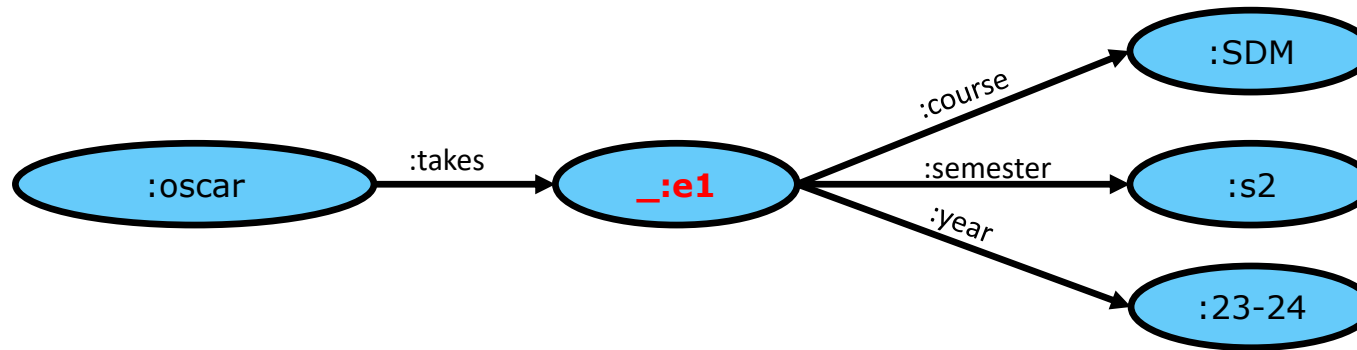
- Since n-ary relationships may be needed, blank nodes were presented as solution

We cannot express neither schema (not even the concept of label) nor additional constraints

- Example: *at least one, at most three, the domain of a property must be of type X*, a node is of type lecturer, etc.
- RDFS and OWL are languages that build on top of RDF to express schema and constraints

Blank Nodes: Reification

Example of use (“quoting”):



Statements involving blank nodes can be written as:

- *[property object]* (meaning the subject of this triple is a blank node)
- e.g., :oscar :takes [:course :SDM]

Blank Nodes

Blank nodes do **not** have a URI and cannot be referenced

- They can only be subjects or objects

some people say it is identifier without URI

Its semantics are yet not clear, though

- De facto use (i.e., most spread use, also in SPARQL): An **identifier** without a URI
- W3C position: Incomplete data (potentially two blank nodes might be the same resource)
 - An unknown value,
 - A value that does not apply / anonymized value

The de facto use is a pragmatic use

- Facilitates reasoning when translating KGs to logics (CWA)

Good practices discourage their use

- All resources should have a proper URI (!)

RDF-star

extension of RDF to deal with blank nodes

RDF-star is an RDF extension

- Systematizes the use of blank nodes
- Compact and precise syntax for reification

<https://w3c.github.io/rdf-star/cg-spec/2021-02-18.html>

Example:

```
@prefix : <http://www.example.org/> .
```

```
:employee38 :familyName "Smith" .
```

Embedded triple

```
:employee22 :claims << :employee38 :jobTitle "Assistant Designer" >> .
```

SPARQL-star is an extension of SPARQL to query RDF-star

Summary

Sharing data for integration or data exchange purposes require new modeling standards

RDF represents resources by means of identifiers (URIs) that can be universally located

RDF, however, does not provide means to **annotate the resources with types** and rich annotations (e.g., constraints)

- We need RDFS and OWL (other KG language families) for that purpose

Modeling in RDF

- Very similar to property graphs
 - IRIs are mandatory
 - Property graph attributes do not exist in RDF
 - We may use blank nodes (although not advisable!)

Bibliography

RDF. W3C Recommendation. Latest version at <http://www.w3.org/TR/rdf-concepts/>

S. Abiteboul et al. Web Data Management, 2012 (chapter 7, until section 7.3.3)

Ian Robinson et al. Graph Databases. O'Reilly. 2013 (<http://graphdatabases.com/>)