RDF, RDFS & JSON-LD

Nikola Milikić
nikola.milikic@fon.bg.ac.rs

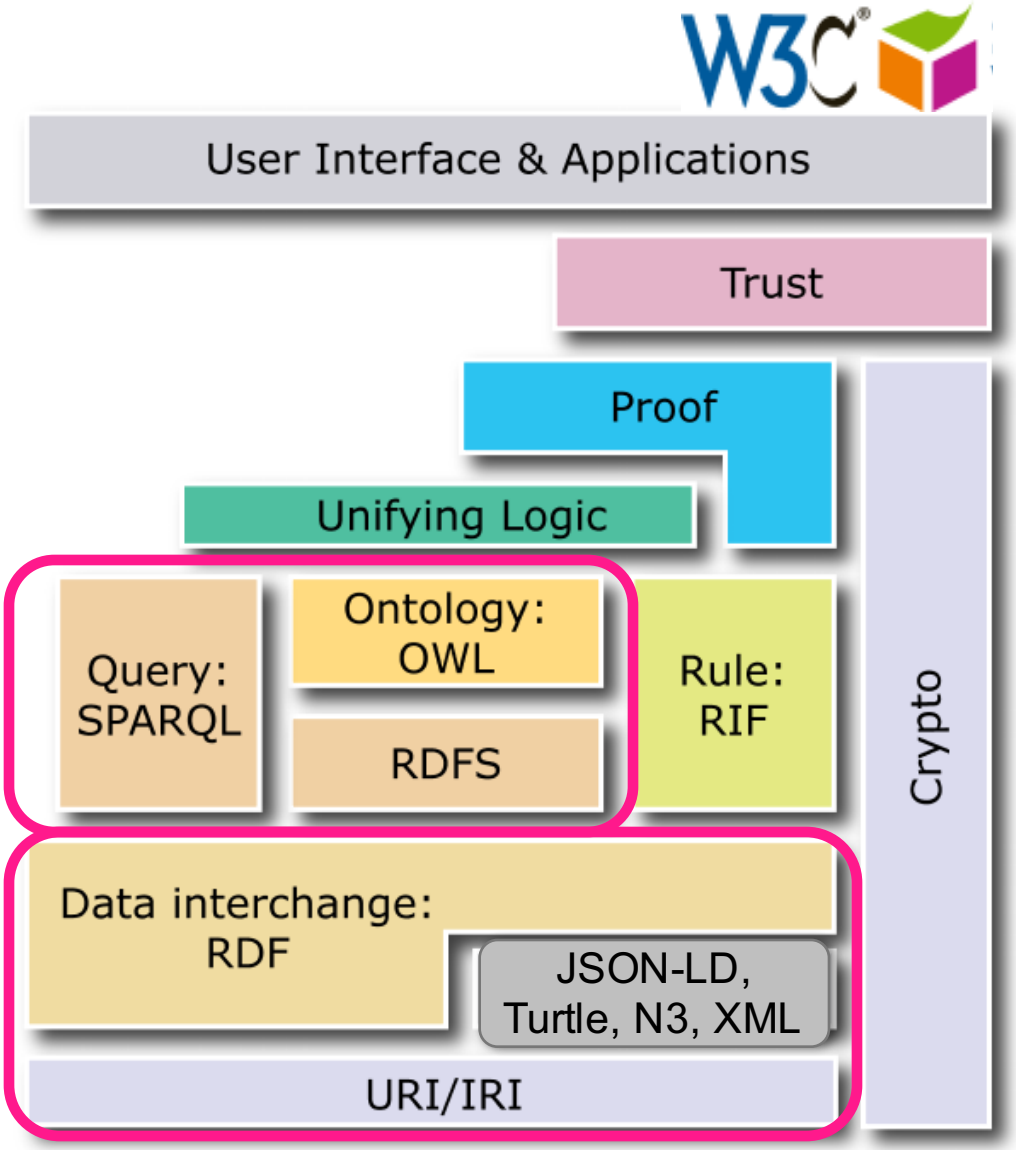
Jelena Jovanović
jeljov@fon.bg.ac.rs

What is RDF?

- Resource Description Framework
- W3C* standard for describing resources on the Web
- One of the key standards that Web of Data / Semantic Web is based upon

*W3C = World Wide Web Consortium (https://www.w3.org/)

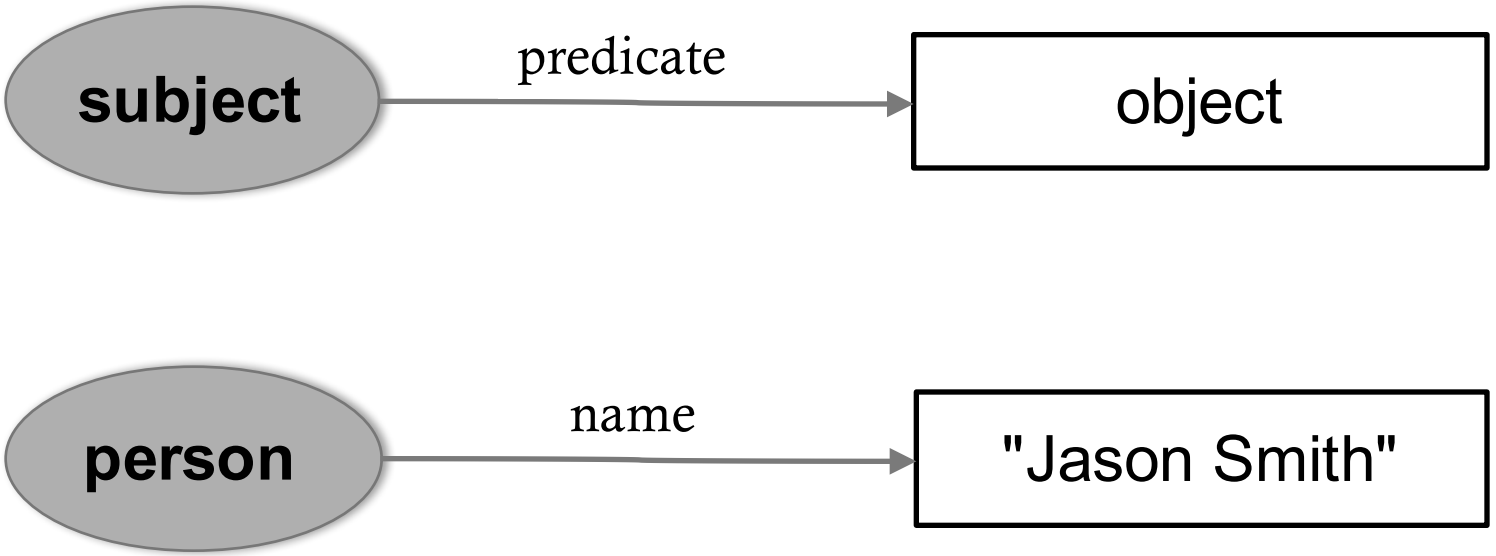
Open Technologies and standards for the Web of Data / Semantic Web



What is RDF?

- RDF is the data model for the Web of data / Semantic Web
- Simple model, based on a graph
- Describes resources and relations between resources

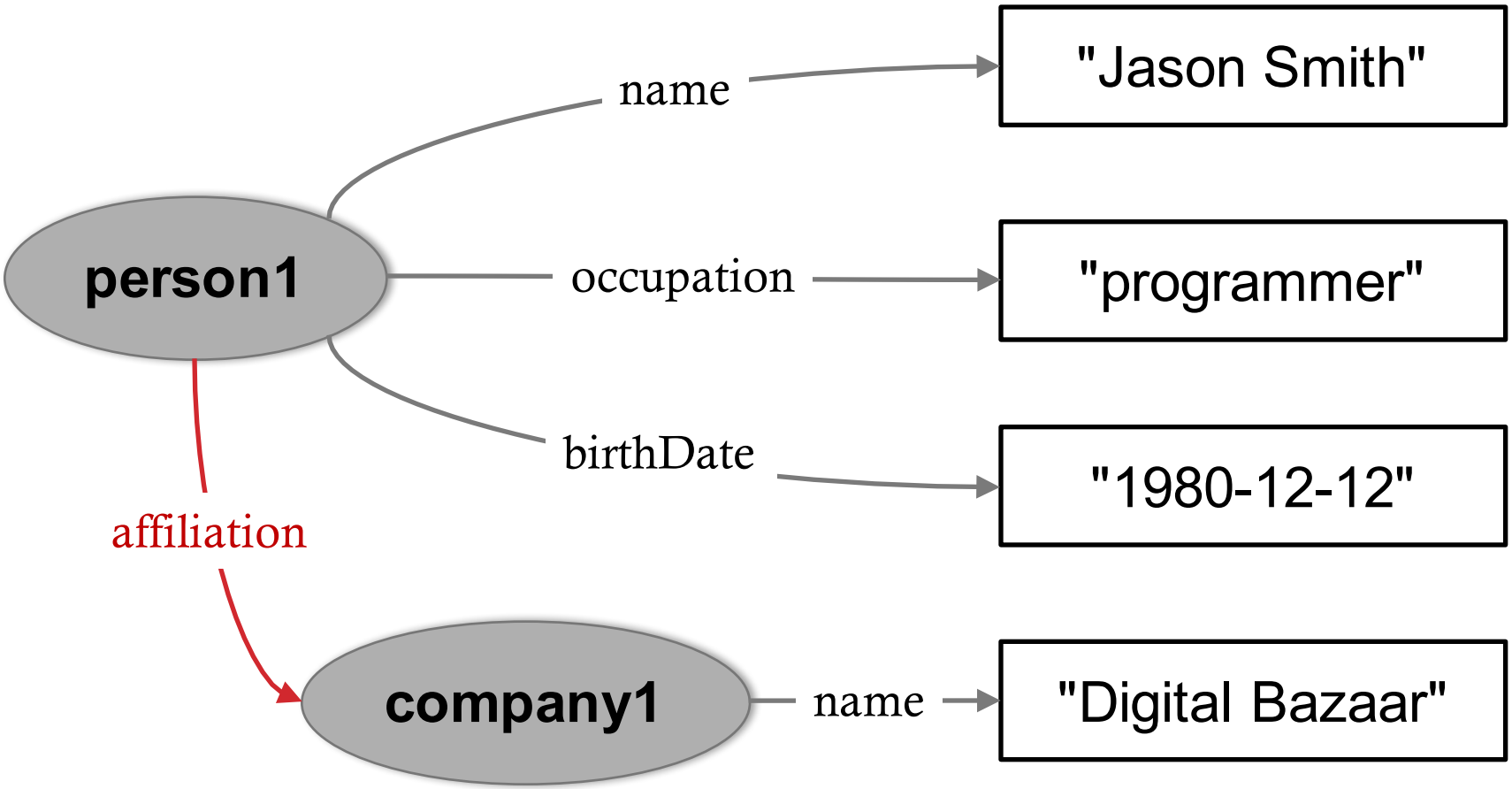
RDF is based on triples



RDF is a graph

- Nodes represent subjects and objects of triples
- Nodes are depicted as
  - ellipses when they represent resources
  - rectangles when they represent literals
- Directed edges represent predicates: properties and relations

Example of an RDF graph



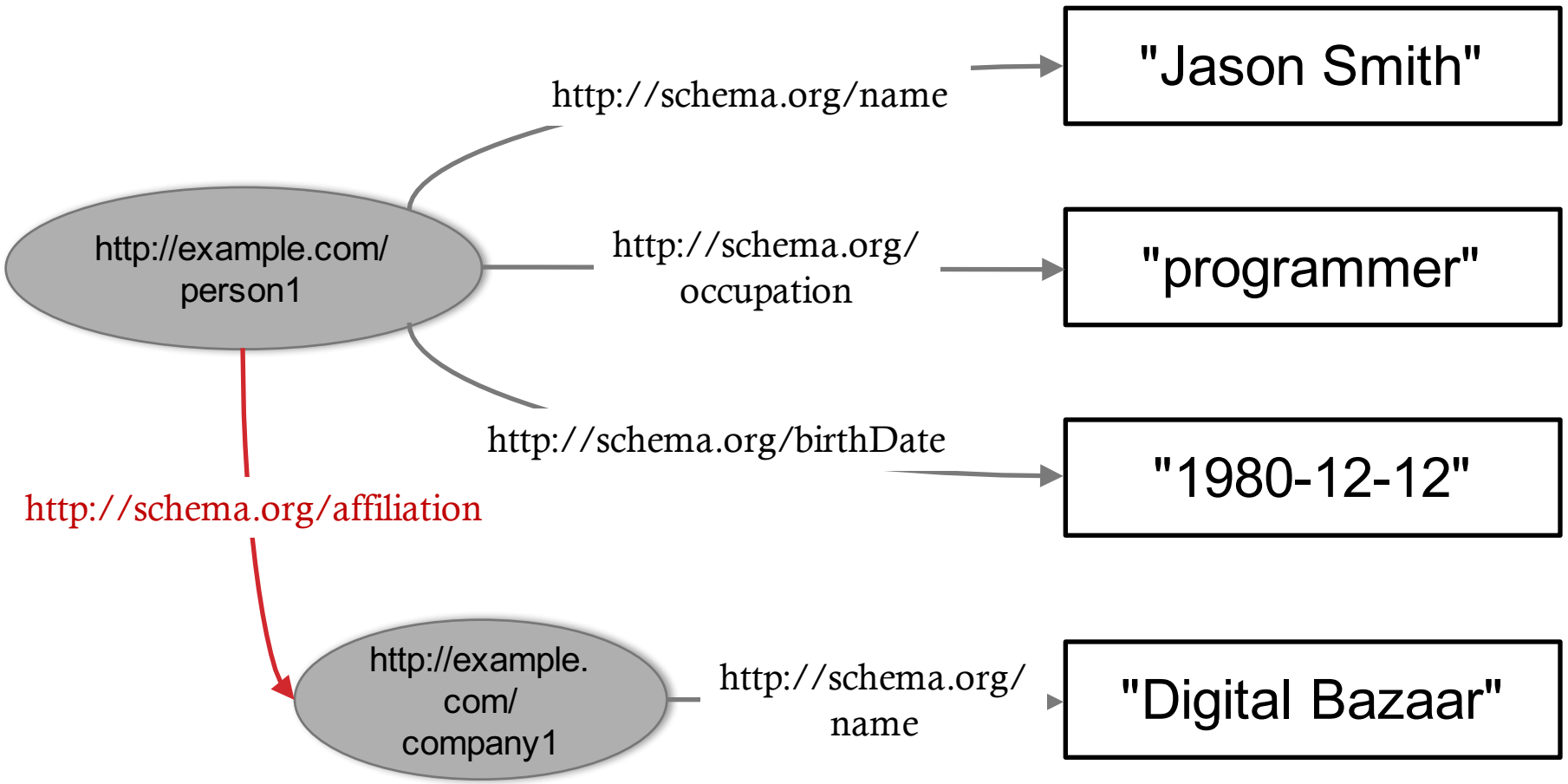
Uniquely identifying resources

- In the *global data repository* on the Web, we must identify resources globally and uniquely
- This is a key prerequisite for linking data on the Web, and realizing the Web of Data
- To uniquely identify resources on the Web, we use URIs (usually starting with "*http://*")

Note the difference:

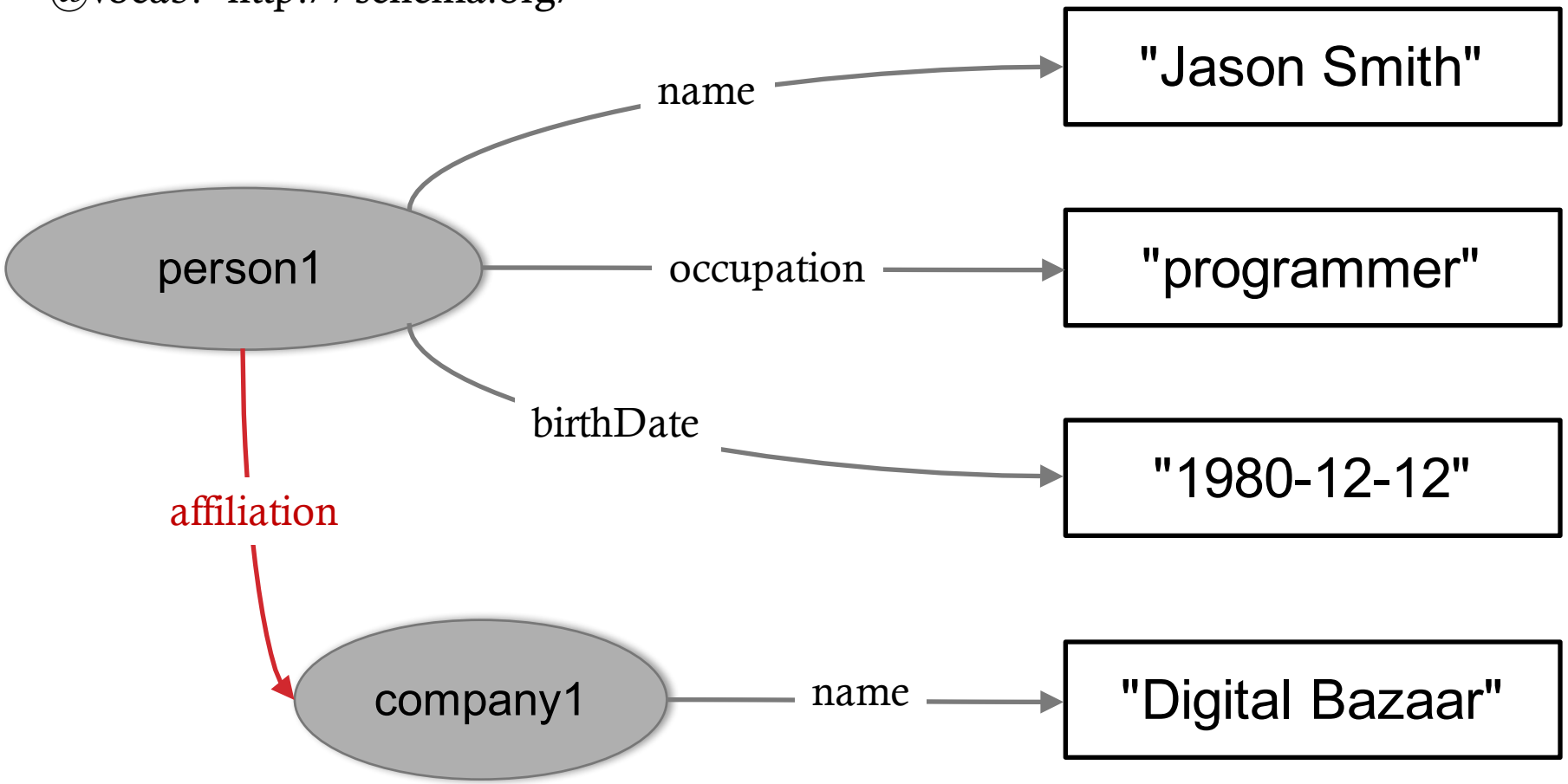
URL – Uniform Resource Locator	location
URI – Uniform Resource Identifier	identifier
IRI – International Resource Identifier	identifier

RDF graph uniquely identified resources and connections



Using vocabularies

@vocab: "http://schema.org/"



Triplet form of RDF graph

@vocab: "http://schema.org/"

person1	name	"Jason Smith" .
person1	occupation	"programmer" .
person1	birthDate	"1980-12-12" .
company1	name	"Digital Bazaar" .
person1	affiliation	company1 .

Simple Rules

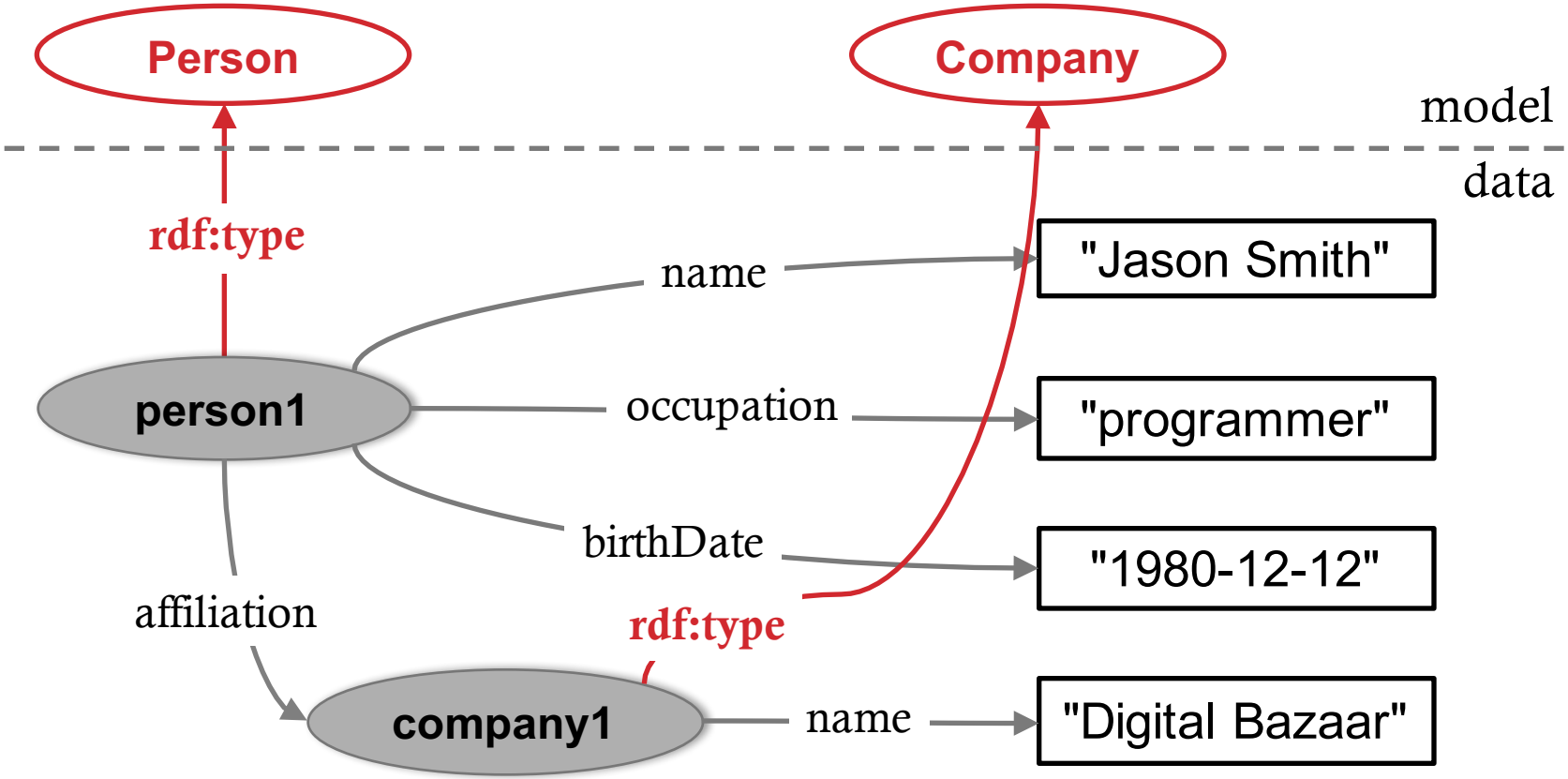
- Use URIs to identify resources and connections
- When the same URI is used on multiple places, all those occurrences (of the URI) refer to the same resource
- This enables for easy interlinking of dispersed data about a particular resource (i.e. data stored and maintained in different repositories)

RDFS

RDFS

- RDFS - RDF Schema
- Adding semantics to RDF
- Creating data schema – vocabulary
- Vocabulary is defined using the same data model

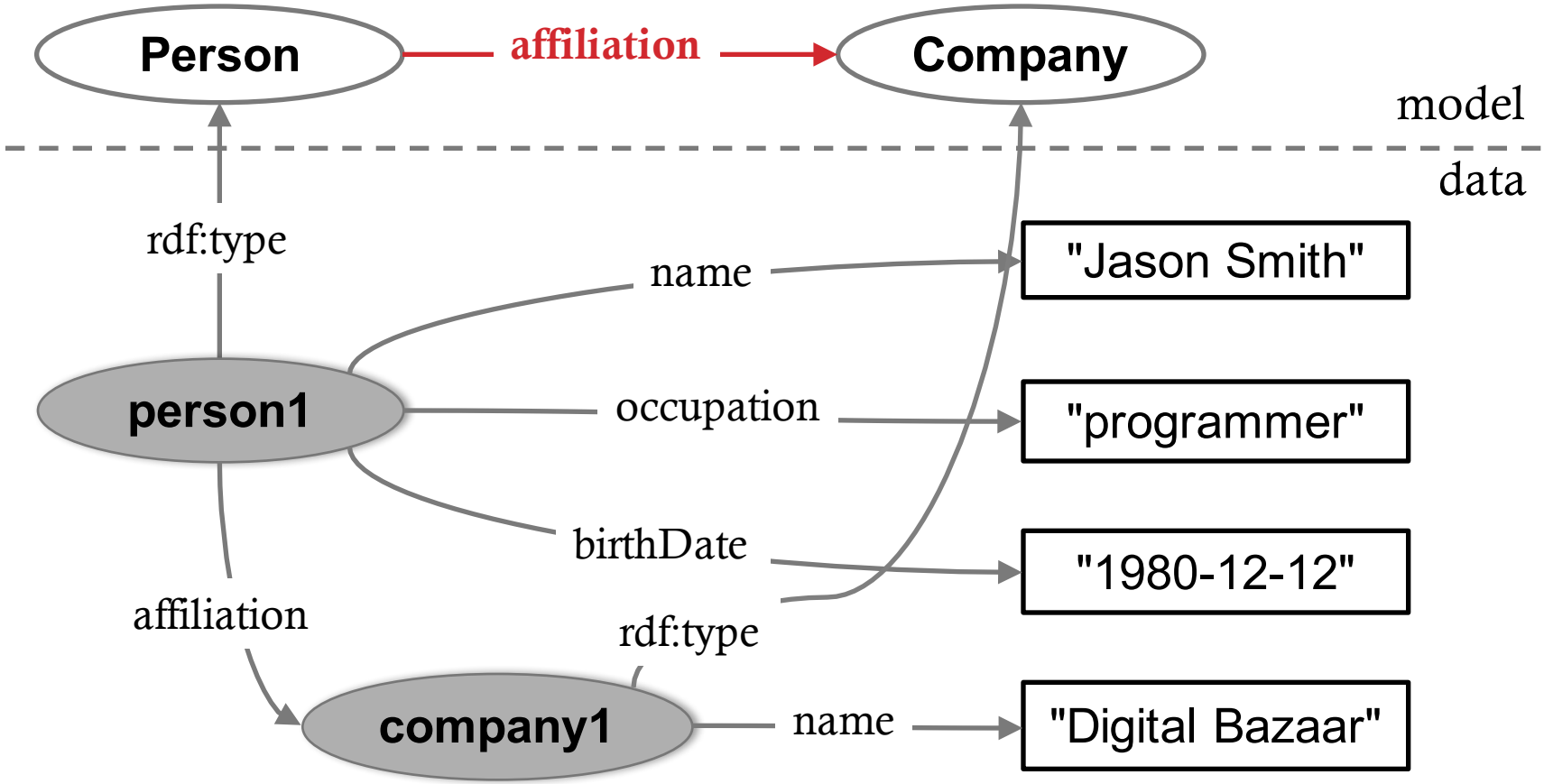
Defining Classes



@vocab: "http://schema.org/"

Person	rdf:type	rdfs:Class .
person1	rdf:type	Person .

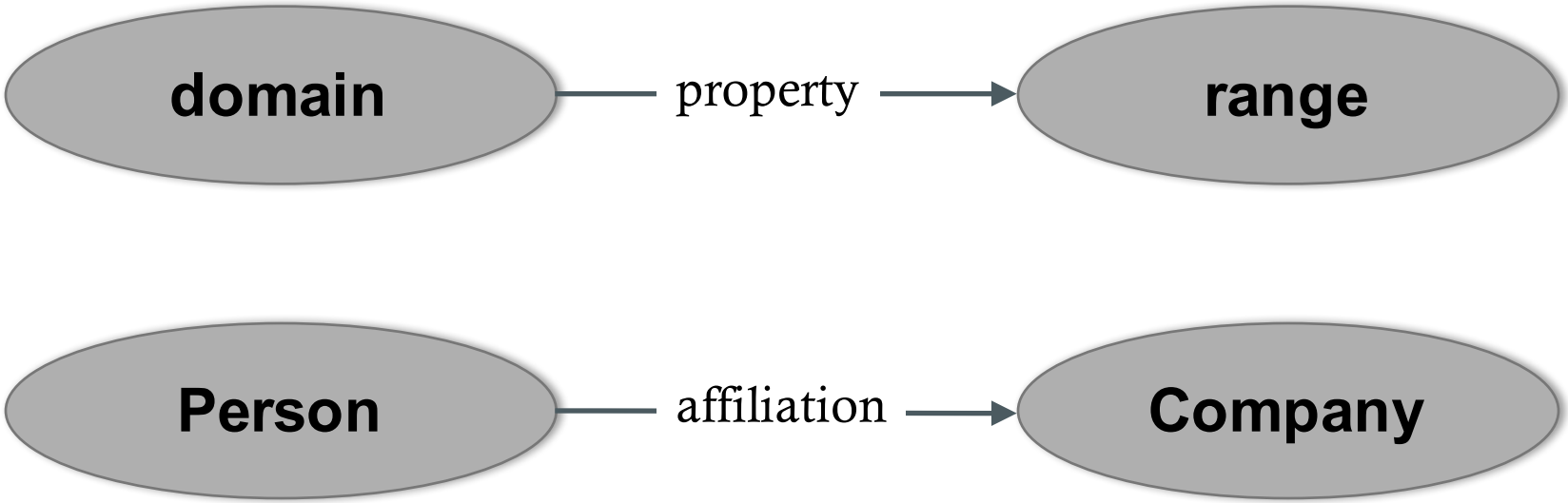
Defining properties



@vocab: "http://schema.org/"

Defining properties

@vocab: "http://schema.org/"




```
affiliation   rdf:type      rdf:Property .
affiliation   rdfs:domain   Person .
affiliation   rdfs:range    Company .
```

Defining properties

Domain represents the class (or multiple classes) a property can be used on

Range represents the class (or multiple classes) that defines the type of the property's value

Both domain and range are optional:

- If domain is not defined, property can be used on any class
- If range is not defined, value of the property can be of any class

Not the same as with OO languages

- Properties are not dependent on classes, they are first class citizens (not part of classes)
- Properties can be extended, they can have their own hierarchy of sub-properties
- Properties can not be overridden on a lower level of hierarchy (by sub-properties)

RDF(S) vocabulary

RDF and RDFS vocabularies

Prefixes: **rdf** i **rdfs**

Classes (some of them)

- rdfs:Class
- rdfs:Property
- rdfs:Literal

Properties (some of them)

- rdf:type (resurse is an instance of certain class)
- rdfs:subClassOf (class is a subclass of another class)
- rdfs:subPropertyOf (subproperty)
- rdfs:seeAlso (reference to a description)
- rdfs:domain (domain of a property)
- rdfs:range (range of a property)

Schema.org

Schema.org is a collaborative, community effort with a mission to create, maintain, and promote schemas for structured data in Web pages, email messages, and Internet in general.

It is initiated and supported by Google, Microsoft, Yahoo and Yandex

The vocabulary cover entities, relationships between entities and actions, and can be extended through a well-documented extension model

Schema.org

Some of the class defined in Schema.org:

- Different kinds of creative work: CreativeWork, Book, Movie, MusicRecording, Recipe, TVSeries …
- Different kinds of multimedia content: AudioObject, ImageObject, VideoObject
- Event
- Organization
- Person
- Place, LocalBusiness, Restaurant …
- Product, Offer, AggregateOffer
- Review, AggregateRating
- Action

For the entire list, check:
http://schema.org/docs/schemas.html

JSON-LD

JSON - JavaScript Object Notation

- Lightweight text-based format for data exchange
- Simple
  - for developers to use it
  - for machines to process it
- Independent of programming languages

JSON object

A set of name-value pairs

```
{
    "title" : "The Matrix",
    "producer" : "Joel Silver",
    "release_year" : 1999
}
```

JSON object

- A set of name-value pairs
- JSON object starts with an open brace ({), and ends with a closing brace (})
- Name and value are separated by colon (:), and name-value pairs are separated with comma (,)

JSON array

```
[
    {
        "title" : "The Matrix",
        "producer" : "Joel Silver",
        "release_year" : 1999
    },
    {
        "title" : "Equilibrium",
        "producer" : [
            {
                "name" : "Joel Silver"
            },
            {
                "name":"Lucas Foster"
            }
        ],
        "release_year" : 1999
    }
]
```

JSON array

- JSON array represents an ordered sequence of values.
- Starts with an opening square bracket [, and ends with a closing square bracket]
- Values are separated by comma

JSON-LD

- Syntax for serializing RDF data into JSON format
- The primary reason for its development was to facilitate:
  - use of Linked Data in Web-based programming environments,
  - development of interoperable Web services, and
  - storage of Linked Data in JSON-based repositories (e.g. MongoDB, ElasticSearch, etc.)
- It is compatible with other Web of Data / Semantic Web technologies (e.g. SPARQL)

JSON-LD

In addition to all the features JSON provides, JSON-LD introduces:

- a mechanism for universal identification of JSON objects based on IRIs
- a way to disambiguate keys shared among different JSON documents by mapping them to IRIs via a context
- a mechanism in which a value in one JSON object may refer to another JSON object on a different Web site
- the ability to annotate strings with the language tag

JSON-LD keywords

@id – for uniquely identifying things described in a document; identifiers are typically IRIs

@type – for setting the data type of a node

@context – for defining abbreviated names (*terms*) for vocabulary elements (classes, properties), which are used throughout a JSON-LD document

@language – for specifying the language of a particular string value

Example JSON snippet

```
{
        "name": "Jason Smith",
        "homepage": "http://jason.smith.org/",
        "image": "http://jason.smith.org/images/jason.png"
}
```

Example JSON-LD snippet

```
{
        "http://schema.org/name": "Jason Smith",
        "http://schema.org/url": {
                "@id": "http://jason.smith.org/"
        },
        "http://schema.org/image": {
                "@id": "http://jason.smith.org/images/jason.png"
        }
}
```

The '@id' keyword means 'This value is an identifier, that is, an IRI'

Every property is unambiguously identified by its IRI (e.g. http://schema.org/name).
Developers and programs can use the IRI to look up the property definition; this process is called **IRI dereferencing**.

Using @context element

@context is used to map terms to IRIs

```
{
    "@context": {
        "name": "http://schema.org/name",
        "image": {
            "@id": "http://schema.org/image",
            "@type": "@id"
        },
        "homepage": {
            "@id": "http://schema.org/url",
            "@type": "@id"
        }
    }
}
```

'image' is shorthand for 'http://schema.org/image'

value associated with 'image' should be interpreted as a unique identifier (IRI)

Defining @context inline

```
{
     "@context": {
          "name": "http://schema.org/name",
          "image": {
               "@id": "http://schema.org/image",
               "@type": "@id"
          },
          "homepage": {
               "@id": "http://schema.org/url",
               "@type": "@id"
          }
     },
     "name": "Jason Smith",
     "homepage": "http://jason.smith.org/",
     "image": "http://jason.smith.org/images/jason.png"
}
```

Defining @context externally

```
{
    "@context": "http://json-ld.org/contexts/person.jsonld",
      "name": "Jason Smith",
      "homepage": "http://jason.smith.org/",
      "image": "http://jason.smith.org/images/jason.png"
}
```

Defining the context in a separate document allows for reuse of the document definition and term-to-IRI mappings.

Defining resource type (class)

- The type of a particular node can be specified using the **@type** keyword
- Types are uniquely identified with IRIs; these originate from vocabularies, such as schema.org

```
{
        ...
        "@id": "http://example.org/places#BrewEats",
        "@type": "http://schema.org/Restaurant",
        ...
}
```

Defining resource type (class)

A node can be assigned more than one type by using a JSON array:

```
{
        ...
        "@id": "http://example.org/places#BrewEats",
        "@type": [
                "http://schema.org/Restaurant",
                "http://schema.org/Brewery"
        ],
        ...
}
```

Defining resource type (class)

The value of an @type key may also be a term defined in the active context:

```
{
    "@context": {
        ...
        "Restaurant": "http://schema.org/Restaurant",
        "Brewery": "http://schema.org/Brewery"
    },
    "@id": "http://example.org/places#BrewEats",
    "@type": [
        "Restaurant",
        "Brewery"
    ],
    ...
}
```

Defining vocabulary

If all properties and types come from the same vocabulary, keyword **@vocab** allows for defining the common namespace for all terms

```
{
        "@context": {
                "@vocab": "http://schema.org/"
        },
        "@id": "http://example.org/places#BrewEats",
        "@type": "Restaurant",
        "name": "Brew Eats"
        ...
}
```

Defining vocabulary

If we need to use more than one vocabulary, we can associate each vocabulary with a prefix (known as compact IRI)

```
{
    "@context": {
        "dbo": "http://dbpedia.org/ontology/"
         "schema": "http://schema.org/"
        ...
     },
     "@type": "dbo:Person",
     "schema:jobTitle": "Financial Manager",
     ...
}
```

`schema:jobTitle` expands into IRI http://schema.org/jobTitle

`dbo:Person` expands into IRI http://dbpedia.org/ontology/Person

Example 1

There is a class *Person*.

Person can have an attribute *name*.

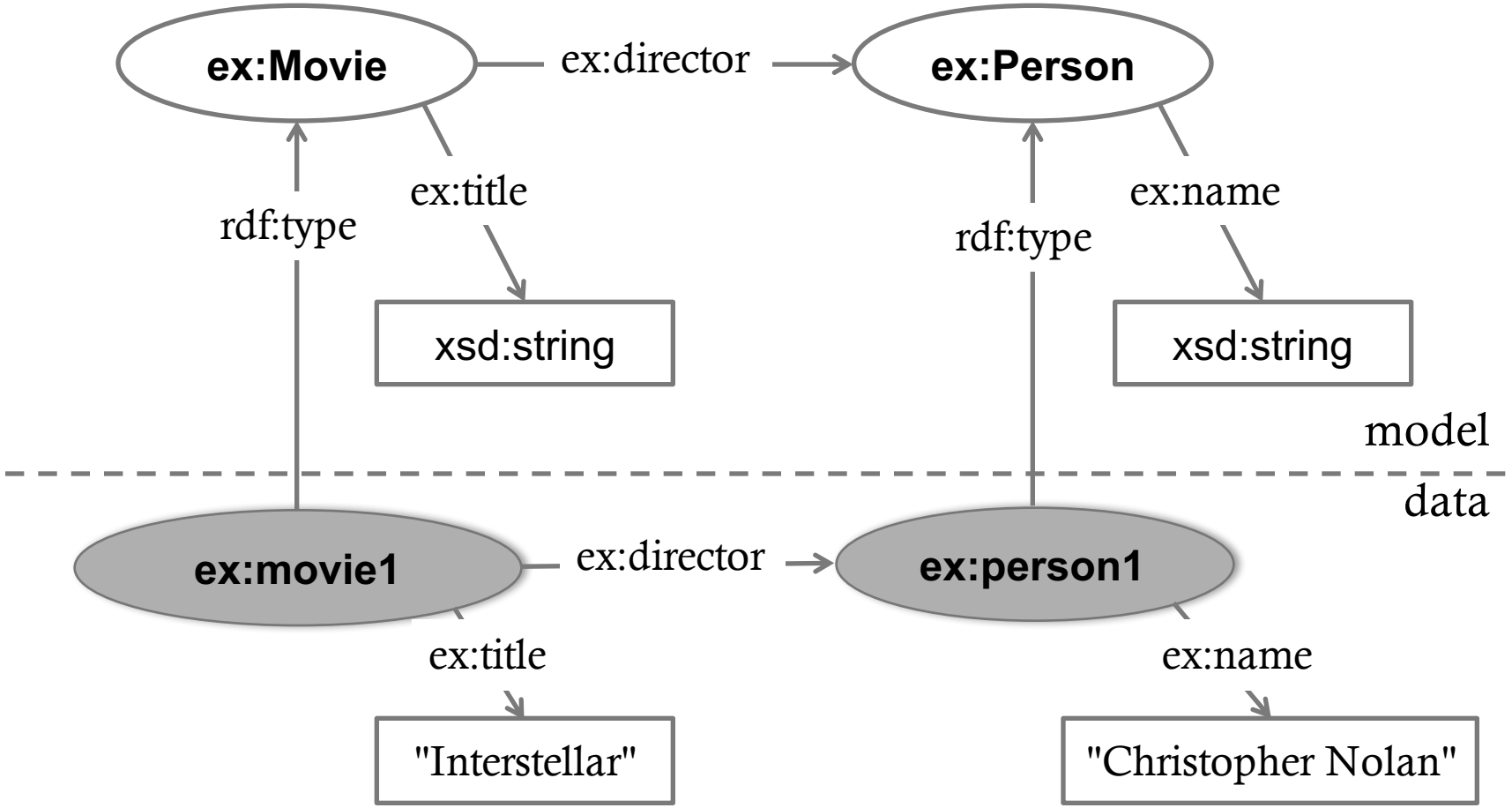
There is a class Movie.

Movie can have an attribute *title* that is a string, and an attribute *director* that is a person who directed the movie.

There is a movie titled "Interstellar". Name of the movie director is "Christopher Nolan".

Example 1 - Graph

```
@vocab: "http://example-vocab.com/"
```



Example 1 – JSON-LD

```
{
        "@context:":  {
                "@vocab":  "http://example-vocab.com/"
        },
        "@id":  "http//example-vocab.com/movie1",
        "@type":  "Movie",
        "title":  "Interstellar",
        "director":  {
                "@type":  "Person",
                "@id":  "http://example-vocab.com/person1",
                "name":  "Christopher  Nolan"
         }
}
```

Example 2

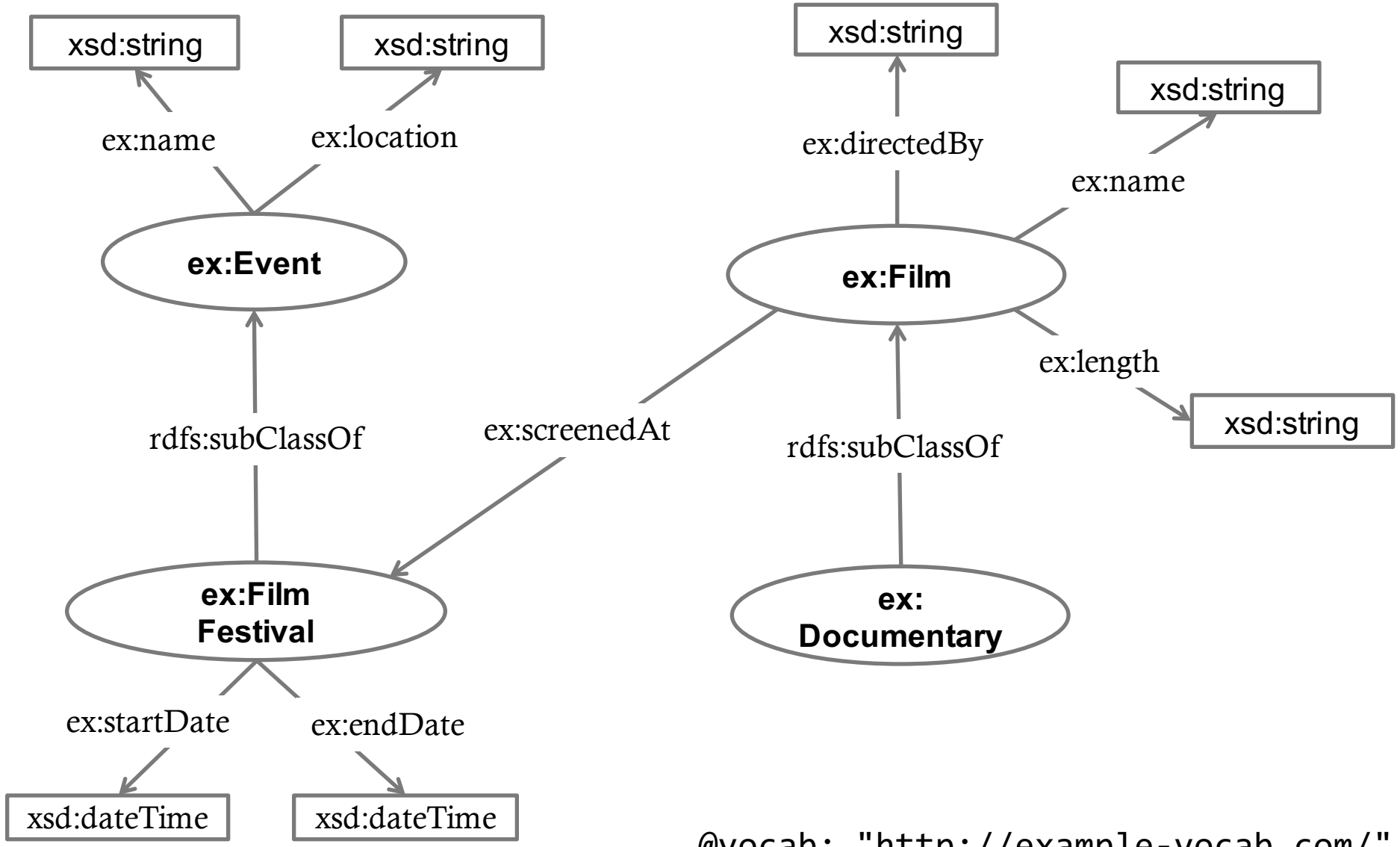
There are *Events* that can be described with *name* and *location*.

Film Festival is a kind of *Event*. *Film Festivals* have their *start date* and *end date*.

There are, also, *Films*. A *Film* can have a *name*, *a director*, and *length*. *Film* can, also, be *screened* on a Film Festival. *Documentary* is a kind of *Film*.

Free Zone is the name of the *Film Festival* that took place in *Belgrade*, from *10.11.2016* till *15.11.2016*. *Documentary* film *Tomorrow*, directed by *Malani Loren* and *Siril Dion*, was screened at this festival; the movie length is 118 minutes.

Example 2 – Graph of the data model




```
@vocab: "http://example-vocab.com/"
```

Example 2 – instance data in JSON-LD

```
{
  "@context": "http://example-vocab.com/" ,
  "@id": "http://www.demain-lefilm.com/",
  "@type": "Documentary",
  "name": "Tomorrow",
  "directedBy": ["Malani Loren" , "Siril Dion" ],
  "length": "118min",
  "screenedAt": {
       "@type": "FilmFestival",
       "@id": "http://freezonebelgrade.org/",
       "name": "Free Zone" ,
       "location": "Belgrade",
       "startDate": "2016-11-10T00:01",
       "endDate": "2016-11-15T23:59"
   }
}
```