Semantic Data Cubes

In this context we define semantic data cubes as data cubes encoded and approached with 'semantic web' technologies.

The different pieces of the semantic web approach to data cubes are the following:

- The **Resource Description Framework** (RDF) as a data model for **statements** to express anything about anything in a way that is exchangeable and machine processable.
- The **RDF Schema Language** (RDFS) as a common way to attach semantics to parts of an RDF statement. Specifically, the **RDF Data Cube Vocabulary** as a way of expressing the specific properties and characteristics of data cubes as well as specific observations in RDF. We'll also briefly touch on the Simple Knowledge Organization System (SKOS) in this context.

In the next sections, we will elaborate on each of the above, and continue to build on the Olympic Games example that we also used to explain the fundamentals of data cubes. We will also introduce **Turtle** as the syntax to encode RDF statements.

RDF: a domain independent data model

The Resource Description Framework (RDF) is a domain-independent framework for expressing information about **resources**, intended for machine processing and exchange on the web. Resources can be anything, including documents, people, physical objects, and abstract concepts.

In our example, countries, editions of Olympics, medals, gender, athletes, disciplines, and so on, are all resources that we can refer to and make **statements** about.

RDF statements always have the structure of a simple sentence:

```
<subject> <predicate> <object>.
```

The **subject** is the thing, the resource the statement is about. This resource has a **property** or **relationship** (called **predicate**), of which the value is the **object**.

Because RDF statements consist of three resources they are called **triples**.

For example:

```
<USA> <is_a> <country> .
<USA> <is_situated_in> <Northern_America> .
<USA> <has_area> 9,833,520 km2 .
<Northern_America> <is_a> <geographic_region> .
```

It looks quite artificial for now, but just take this as a starting point. There are already several things that this example illustrates:

1. A particular resource may be the subject of one triple and the object of another. This makes it possible to find connections between triples, which is an important part of RDF's power.
2. Properties are also resources. They are a bit special in that they express some kind of relationship between two other resources, but other than that they are resources just like the subject and object.
3. We need a way of identifying resources unambiguously for machines to be able to process this and establish correct (sensible) links between the data they are given, and also to re-use resources that have already been

See also:

video course on RDF Data Cubes

identified (by ourselves or by others).

To identify the subjects, properties, and objects in RDF statements unambiguously, the following techniques are available: URIs and literals.

- The **URI** (Uniform Resource Identifier) can be a URL (Uniform Resource Locator) as is used for web addresses, but any kind of unique identifier will do. The URI just identifies a resource, without implying the location of that resource or how to access it, however it is common and recommended practice in the Linked Data world to use http(s), and also to use wording that is to a reasonable extent meaningful for human interpretation.
- **Literals** are basic values such as strings, dates, booleans, and numbers. Literals can only be used in the object position of an RDF triple. To allow correct parsing and interpretation, literals are best associated with a datatype, and strings can also optionally be associated with a language tag.

Applying this to the first triple in our example, we could arrive at the following statement:

```
<http://dbpedia.org/resource/United_States>
     <http://www.w3.org/1999/02/22-rdf-syntax-ns/type>
          <http://dbpedia.org/ontology/Country> .
```

Step by step:

- We use `<http://dbpedia.org/resource/United_States>` as the URI to refer to the United States. This is the id as used by DBpedia, the linked data version of Wikipedia. Try it out – load the URL into a web browser and see what you get: a document that describes the United States. In this case we used an existing URL, and by doing so we establish a link between our statement and the DBpedia statement and everything that has already been said there and in other places the reference that same URI. We could also assign our own if we had good reasons to do so, but if a suitable URI already exists, re-using it creates extra added value.
- We use a special RDF property to say that 'United States" is a resource of a certain 'type": `<http://www.w3.org/1999/02/22-rdf-syntax-ns/type>`. This property has been defined in the RDF specification, precisely to be able to indicate that a resource belongs to a class (we'll explain classes later).
- As value of the type property, we use another class identifier from DBpedia: `<http://dbpedia.org/ontology/Country>`. Load the URL into a web browser if you want to see what information is available on that class.

To illustrate the use of literals, let's also add this statement:

```
<http://dbpedia.org/resource/United_States>
   <http://www.w3.org/2000/01/rdf-schema/label>
      "Verenigde Staten"@nl .
```

Again, step by step:

- We use a common vocabulary called RDFS (again, more on that later) to say that the resource with id `<http://dbpedia.org/resource/United_States>` has a 'label" property <http://www.w3.org/2000/01/rdf-schema/label>.
- The value of that label in Dutch (`@nl`) is declared to be the string `Verenigde Staten`.

As may already be apparent from the examples above, when we start making multiple statements related to the same subject, things can quickly get quite verbose and lengthy, and we also see the same URIs being repeated entirely or partially quite often. To reduce this verbosity and make the statements more readable, a more compact syntax is available in the form of **Turtle**.

Turtle, or *Terse RDF Triple Language*, uses abbreviations and other shorthand notations:

- Prefixes: to avoid repetition of the common parts of URIs, a prefix can be specified once and then used instead.
- If several triples have the same subject, they can be grouped and the subject can be stated just once, followed by each of the predicates and objects, separated by ';' (semicolon).
- If several triples have the same subject and predicate, all of the different objects can be stated as one series, separated by ',' (comma).

If we apply this to our sample triples, we can reduce our statements to the following more condensed form (keeping in mind that prefixes only need to be defined once):

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dbo:       <http://dbpedia.org/ontology/> .
@prefix dbr:       <http://dbpedia.org/resource/> .

dbr:United_States rdf:type dbo:Country;
    rfds:label "Verenigde Staten"@nl;
   .
```

We will use Turtle as the syntax for our examples from here on forward.

RDFS: adding domain semantics

While RDF provides a way to make statements in triple format about resources, it does not however make any assumptions about what those resources represent. RDF in itself is domain-free.

To apply RDF in your domain, you need to specify 'the things" you want to talk about and the properties and relationships of relevance. In RDF-speak this translates into defining the **classes** and **properties** to be used in your RDF statements to describe your resources. Such a group of domain-relevant classes and properties are referred to as a **vocabulary**. The RDF Schema Language (RDFS) is what is used to define such vocabularies.

RDF Data Cube Vocabulary

Fortunately there is already a vocabulary for describing statistical information and data cubes, and what's more, it is a W3C standard: the **RDF Data Cube vocabulary**.

RDF Data Cube Classes

Let's have a quick reminder of the concepts we discovered while exploring the Olympics data:

- Data cubes
- Dimensions
- Observations
- Measurements
- Units
- Slices
- ...

These are our **classes** of interest, and nearly all of these have an RDFS translation/definition in the RDF Data Cube vocabulary (in what follows, `qb` is the prefix for `http://purl.org/linked-data/cube#`):

Data cube	`qb:DataSet`
Dimension	`qb:DimensionProperty`
Observation	`qb:Observation`
Measurement	`qb:MeasureProperty`
Unit	Not in RDF Data Cubes but can be found in other vocabularia, e.g. QUDT.
Slice	`qb:Slice`

Using that vocabulary and adding a few new 'words", a data cube has been defined as follows:

A qb:DataSet has a qb:DataStructureDefinition that defines the structure of the cube. The structure is specified by means of a qb:ComponentSpecification containing a number of qb:ComponentProperty sets, detailing qb:DimensionProperty to define the dimensions of the cube, qb:MeasureProperty to define the measured variables, and qb:AttributeProperty to define structural metadata such as the unit of measurement. The observations themselves are included as a qb:Observation for each cell of the cube.

For those who are interested to know more of the technical details, each of these classes has a formal semantic definition in the RDF Data Cube vocabulary specification. We'll limit ourselves to just one example here: the definition of the Observation Class.

```
  qb:Observation
   rdf:type rdfs:Class ;
   rdfs:comment "A single observation in the cube, may have one or more
    associated measured values"@en ;
   rdfs:isDefinedBy <http://purl.org/linked-data/cube> ;
   rdfs:label "Observation"@en ;
   rdfs:subClassOf qb:Attachable ;
   owl:equivalentClass scovo:Item ;
.
```

Some of this will already be sufficiently familiar and readable:

- The subject of our statements here is the resource identified as qb:Observation.
- We use rdf:type to say that qb:Observation is a resource of type rdfs:Class.
- With rdfs:comment, we add a human-readable description of qb:Observation.
- We use rdfs:isDefinedBy to indicate that qb:Observation is a resource described by the vocabulary known as <http://purl.org/linked-data/cube>.
- We use rdfs:label to assign the literal string 'Observation" as the label in English.

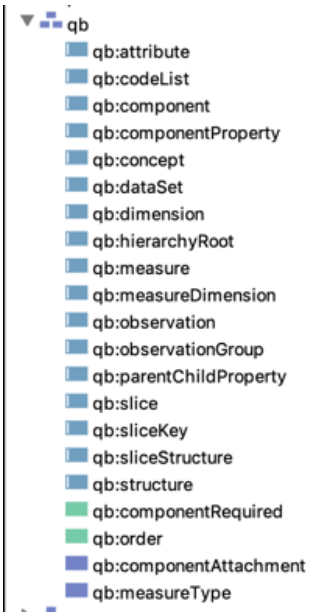
The formal semantics are expressed in the last two statements:

- qb:Observation is an rdfs:subClassOf the resource known as qb:Attachable.
- qb:Observation is an owl:equivalentClass of the resource known as scovo:Item.

Omitting the prefixes for a moment, the use of subClassOf means that every instance of an Observation is also an instance of the class Attachable (an abstract superclass for everything that can have attributes and dimensions.). The use of equivalentClass means that every instance of an Observation is also an instance of the class Item defined in another vocabulary called The Statistical Core Vocabulary (SCOVO, meanwhile deprecated), and conversely every such Item is also an instance of Observation.

RDF Data Cube Properties

The RDF data cube vocabulary defines also a list of properties:relations:



Again, more technical details are available in the RDF Data Cube vocabulary specification.

Let's have a more detailed look at one example:

```
:X qb:dataSet :Y .
```

Note a small but important detail here: the lowercase letter d in dataSet. Earlier we mentioned `qb:DataSet` with a capital D, which is the class for data cubes. Both `qb:DataSet` and `qb:dataSet` come from the same vocabulary, but the class `qb:DataSet` does not equal the property `qb:dataSet` – they are different things!

The triple states that the resource we identify by means of the URI :X has a `qb:dataSet` property of which the value is the resource identified by URI :Y. To understand what that means, we need to look at the formal definition of `qb:dataSet` in the RDF Data Cube vocabulary specification:

```
qb:dataSet
  rdf:type rdf:Property ;
  rdfs:comment "indicates the data set of which this observation
     is a part"@en ;
  rdfs:isDefinedBy <http://purl.org/linked-data/cube> ;
  rdfs:label "data set"@en ;
  rdfs:domain qb:Observation ;
  rdfs:range qb:DataSet ;
  owl:equivalentProperty scovo:dataset ;
.
```

Step by step:

- The subject of our statements here is the resource identified as `qb:dataSet`.
- We use `rdf:type` to say that `qb:dataSet` is a resource of type `rdfs:Property`.
- With `rdfs:comment`, we add a human-readable description of `qb:dataSet`.
- We use `rdfs:isDefinedBy` to indicate that `qb:dataSet` is a resource described by the vocabulary known as `<http://purl.org/linked-data/cube>`.
- We use `rdfs:label` to assign the literal string '`data set`" as the label in English.

The formal semantics are in the last 3 lines, which say the following things about `qb:dataSet`:

- The `rdfs:domain` is `qb:Observation`.
- The `rdfs:range` is qb:DataSet. (As said earlier, the difference in case is important!)
- There is an `owl:equivalentProperty` which is `scovo:dataset`.

So, given a triple that states `:X qb:dataSet :Y`, this means that:

- The subject :X of the triple becomes an instance of class `qb:Observation` (as specified by `rdfs:domain`).
- The object :Y of the triple becomes an instance of `qb:DataSet` (as specified by `rdfs:range`).
- If `:X qb:dataSet :Y` then also `:X scovo:dataset :Y`, and the reverse is true also (as specified by `owl:equivalentProperty`).

Supporting vocabularies

The Data Cube vocabulary in turn builds upon the following existing RDF vocabularies:

- SKOS for describing thesauri, controlled lists, concept schemes
- SCOVO for describing core statistical structures
- Dublin Core Terms for metadata
- VoiD for data access
- FOAF for describing agents
- ORG for describing organizations.

We'll very briefly elaborate on SKOS, and we'll also touch on SDMX (Statistical Data and Metadata eXchange).

SKOS

Values for dimensions within a data cube must be unambiguously defined. This can be achieved either by assigning data types to values to make sure they are correctly specified and interpreted, or by defining codes in code-lists in order to create controlled sets of values. Sometimes such code lists will already have been defined and may be suitable for re-use. If however you need to define your own lists, the Simple Knowledge Organization System (SKOS) is the recommended way to define both the codes and the code lists, where codes can be defined as `skos:Concept`, and the code lists as `skos:ConceptScheme` or `skos:Collection`. SKOS can also be used to define hierarchical code lists, thereby enabling aggregation of data in cubes.

Those who are keen to know more of the details of SKOS should add the SKOS primer to their reading lists.

An example:

```
<https://example.org/id/conceptscheme/competitions>   rdf:type skos:ConceptScheme ;
       rdfs:label "competitions"@en ;
       skos:hasTopConcept <https://example.org/id/concept/olympics> ;
       skos:hasTopConcept <https://example.org/id/concept/paralympics> ;
.
<https://example.org/id/concept/olympics>   rdf:type skos:Concept ;
       rdfs:label "olympics"@en ;
       skos:inScheme <https://example.org/id/conceptscheme/competitions> ;
.
<https://example.org/id/concept/paralympics>   rdf:type skos:Concept ;
       rdfs:label "paralympics"@en ;
       skos:inScheme <https://example.org/id/conceptscheme/competitions> ;
.
```

We have a controlled list (`skos:ConceptScheme`) here with label 'competitions' with 2 concepts:

- a concept labeled 'olympics'
- a concept labeled 'paralympics'

Notice the relationship (`skos:hasTopConcept`) for relating the controlled list with the concepts and `skos:inScheme` for the other way around.

SDMX

The model underpinning the RDF Data Cube vocabulary is compatible with the cube model that underlies SDMX (Statistical Data and Metadata eXchange), an

ISO standard for exchanging and sharing statistical data and metadata among organizations.

The SDMX standard includes a set of content oriented guidelines (COG) which define cross-domain concepts, code lists, and categories that support interoperability and comparability between datasets by providing a shared terminology between SDMX implementers. A community group has developed RDF encodings of these guidelines. While these encodings do not form part of the RDF Data Cube specification, they are however used by a number of existing Data Cube publications.

More information on SDMX and specifically how it relates to the RDF Data Cube vocabulary is available in the RDF Data Cube vocabulary specification.