Introduction to MapVOWL

1 Goal

The goal of MapVOWL is to provide a visualization of mapping rules that state how Linked Data is generated based on raw data.

2 Running Example

To explain the different aspects of MapVOWL, we will use an example. We will create the mapping rules to generate RDF triples from a CSV file about books. The CSV file looks as follows:

ID	title	releasedAt
0	Harry Potter	1997
1	The Fellowship of the Ring	1954
2	A Game of Thrones	1996

3 Graph Visualization

MapVOWL uses a graph visualization to present the mapping rules. The visualization is based on the graphs that are constructed when multiple RDF triples are combined. MapVOWL aligns the rules and the final triples through the reuse of this graph structure. A graph consists of both nodes and edges. When details are provided in the graphs prefixes can be used instead of the full namespace. The list of prefixes and their corresponding names are defined outside the visualization/MapVOWL.

3.1 Nodes

A circle is used to represent IRIs and blank nodes. A rectangle is used to represent literals. Details about the IRIs and blank nodes can be found inside the circle:

- how the IRIs are constructed (only for IRIs)
- what the classes are of the entity

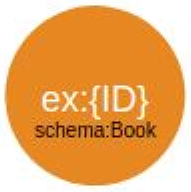

Figure 1a

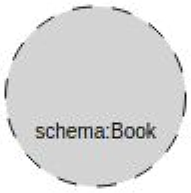

Figure 1b

In our example, we want to generate unique IRIs for books that look like this `http://example.com/0`, `http://example.com/1`, `http://example.com/2`, and so on. The common part between these IRIs is `http://example.com/`, and 0, 1, 2 is the id of each book. In the Figure 1a, you can see the circle for these IRIs. `http://example.com/{ID}` (called a template) will result in the generation of the desired IRIs. `{id}` is replaced with the id of every book. More complex IRIs can be created if needed. For example, the template `http://www.example.com/{ID}-book_{title}` relies on both the id and the title of a book. Additionally, we want to state that each book is of the class `schema:Book`. This information can be found underneath the IRI in the circle. Blank nodes are different from IRIs. They have a dashed black border, where an IRI does not have a border (see Figure 1b).

The generated triples based on Figure 1a are

```
ex:0 a schema:Book.
ex:1 a schema:Book.
ex:2 a schema:Book.
```

The generated triples based on Figure 1b are

```
_:b0 a schema:Book.
_:b1 a schema:Book.
_:b2 a schema:Book.
```

Details about the literals can be found inside the rectangle:

- reference to the data values that are used to create the literal value
- datatype of the literal value

- language of the literal value

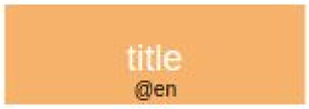

Figure 2

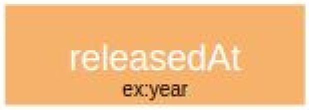

Figure 3

In our example, we want to generate literal values for every title of a book and we want to state that every title is given in English. In Figure 2, you can find the rectangle that contains this information. title refers to the column of the CSV file. The fact that the title is in English is reflected by `@en` underneath the column. In the case we want to generate literal values for the year a book is released, we use the information in the column releasedAt. Additionally, here we want to set the datatype to `ex:year`. In Figure 3, you can find the corresponding rectangle. We see `ex:year` underneath the column name, which specifies the datatype of every literal for the year a book was released. Note that neither a language or datatype are required, but are optional.

3.2 Edges

An edge is used to present a relationship between entities and between entities and their attributes. A directional edge is used. The source of an edge reflects the subject of a triple and the target the object. On the edge a rectangle is present that provide information about how the predicate, representing the relationship, is determined.

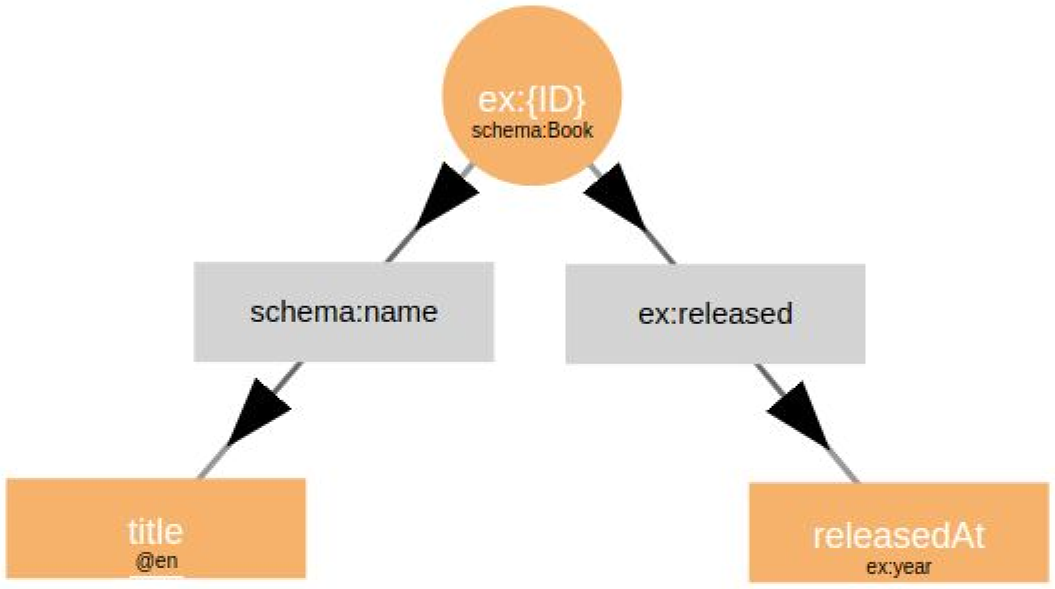

Figure 4

Earlier we created a circle to represent book entities, includes their IRIs and classes, and rectangles to represent two attributes of the books: title and year released. Now, we want to add the relationships between the entities and the attributes. Therefore, we will create a relationship with predicate `schema:name` between the book entity and the literal value for title, and a relationship with predicate `ex:released` between the entity and the literal value for releasedAt. In Figure 4, you can see the the resulting graph. Two edges are added. One for each relationship between the entity and the attribute. These mapping rules can now be used to generate RDF triples. They lead to the following triples:

```
ex:0 a schema:Book;
       schema:name “Harry Potter”@en;
       releasedAt “1997”^^ex:year.

ex:1 a schema:Book;
       schema:name “The Fellowship of the Ring”@en;
       releasedAt “1954”^^ex:year.

ex:2 a schema:Book;
       schema:name “A Game of Thrones”@en;
       releasedAt “1996”^^ex:year.
```

3.3 Multiple Datasets

In some cases during the generation of Linked Data, multiple datasets are considered. Data from different data sources is used to generate the different triples. MapVOWL uses colors for the nodes and edges to differ between the different data sources.

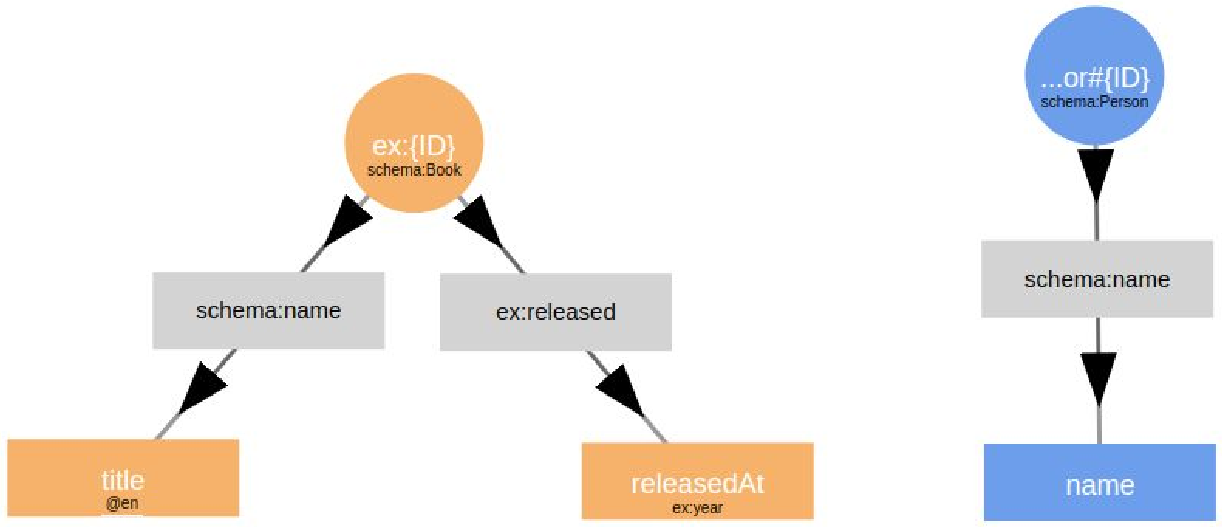

Figure 5

For example, let’s consider our data sources with books, and an additional data sources about authors. A visualization of potential mapping rules can be found in Figure 5. The nodes and edges that use data from the same data source have the same color. We appointed the book data source with yellow and the author data source with the color blue. Note that the edges are grey. Grey is the color used when a node or edge does not depend on data from a specific data sources. For the predicates in this example this is the case, because they are fixed to a single predicate. This leads to the following triples for the rules in Figure 5.

```
ex:0 a schema:Book;
       schema:name “Harry Potter”@en;
       releasedAt “1997”^^ex:year.

ex:1 a schema:Book;
       schema:name “The Fellowship of the Ring”@en;
       releasedAt “1954”^^ex:year.

ex:2 a schema:Book;
       schema:name “A Game of Thrones”@en;
       releasedAt “1996”^^ex:year.
```

```
ex:author#jkr a schema:Person;
            schema:name “J.K. Rowling”.

ex:author#jrrt a schema:Person;
            schema:name “J.R.R. Tolkien”.
```

3.4 Interlinking Datasets

In some cases you do not only need to consider multiple datasets, but you also need to model relationships between these datasets. The datasets need to be interlinked in the resulting Linked Data. In MapVOWL this is visualized by using an edge between two nodes that represent entities.

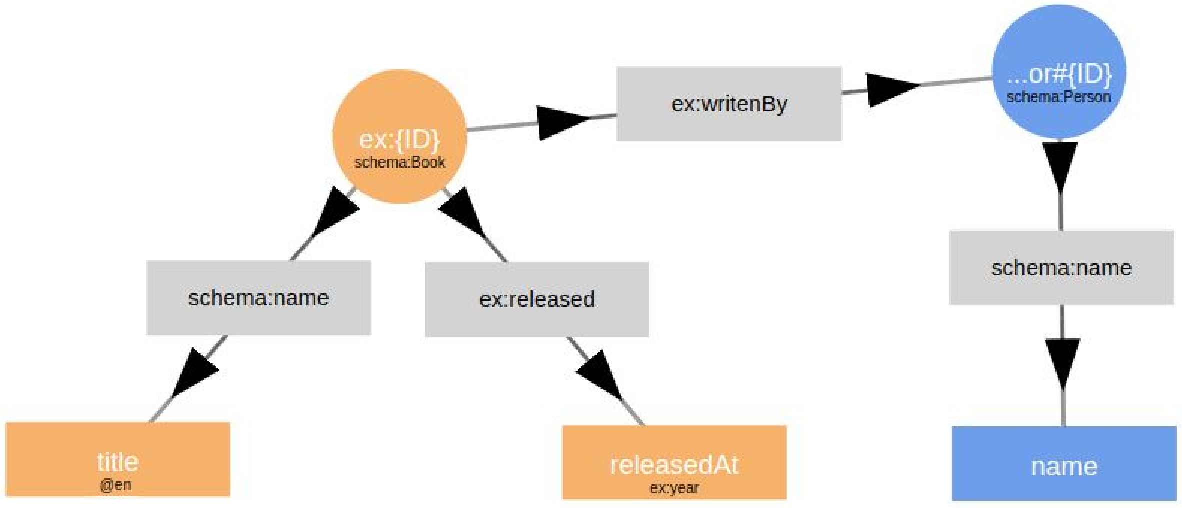

Figure 6

In Figure 6, we interlink books with authors. The edge between the nodes book and author entities states their relationship: a book is written by an author, using the predicate `ex:writenBy`. The use of the different colors makes clear that a link between different datasets is made. Note that details about the conditions when different entities should be linked is not visualized. This leads to the following triples for the rules in Figure 5.

```
ex:0 a schema:Book;
       schema:name “Harry Potter”@en;
       releasedAt “1997”^^ex:year;
       ex:writtenBy ex:author#jkr.

ex:1 a schema:Book;
```

```
       schema:name “The Fellowship of the Ring”@en;
       releasedAt “1954”^^ex:year;
       ex:writtenBy ex:author#jrrt.

ex:2 a schema:Book;
       schema:name “A Game of Thrones”@en;
       releasedAt “1996”^^ex:year.

ex:author#jkr a schema:Person;
            schema:name “J.K. Rowling”.

ex:author#jrrt a schema:Person;
            schema:name “J.R.R. Tolkien”.
```