Semantic Web standards and a little bit of Python go a long way

~~Using R2RML/RML, SPARQL, and Python to transform CSV to RDF~~

Dr. Ora Lassila

Founder, So Many Aircraft

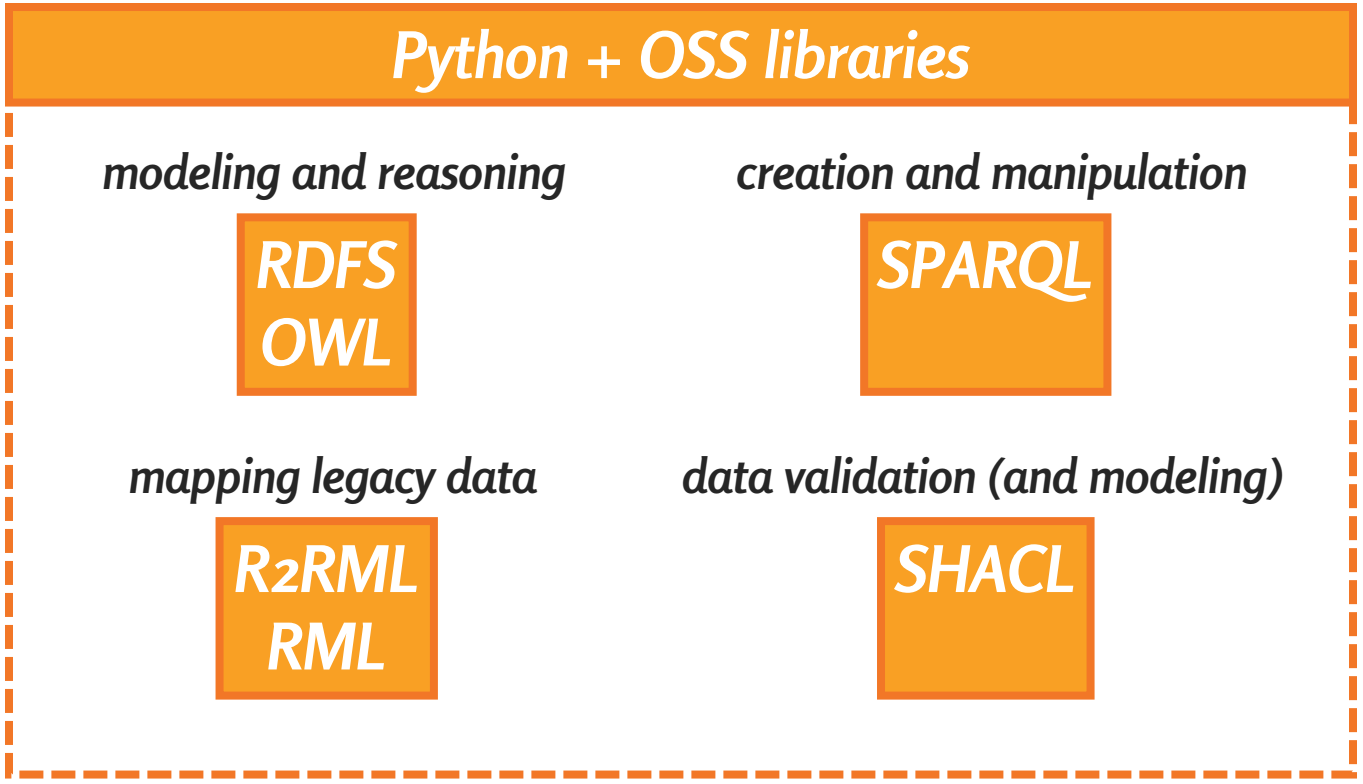
(also: Principal Technologist, Amazon Neptune)

July 2025

Premise: Tools for this exist (& have for many years)

The **Semantic Web stack** is a collection of easy-to-use standards and technologies.

Their application can be "orchestrated" with simple Python scripting and some open-source libraries.



All this ~~should be~~ is easy!

Some background

So Many Aircraft: a publisher focusing on aircraft and aviation history

- we have published two books (so far)
- we use a knowledge graph for our reference library (4,500+ pubs), photo collection (60k+ original photos), and an aviation related SKOS taxonomy (5,000+ concepts)

Over the years we have produced some open-source Python libraries for working with RDF, and built our internal software using those

So Many Aircraft

So Little Time

OMG Challenge solution overview

Simple flow that uses RML and SPARQL to create and manipulate RDF

- RML ≈ 70 triples, ontology ≈ 60 triples, code ≈ 40 lines of Python
- 79k lines of CSV → 1.4M triples, about 4 min on an old Mac (Intel)

Wrote an ontology for this domain that extends or otherwise uses FIBO and some other public ontologies

So Many Aircraft

So Little Time

What was difficult?

Dataset in general not particularly well documented

- had to make some guesses about the overall data model
- much of the model was really inferred from the types of values different columns of the data had, and not so much from the column names

Date format undocumented

- Excel representation seemed to work

In some cases we had multiple options how to map to existing ontologies

- e.g., for lat/long we chose OMG LCC, not schema.org or WGS84

Unclear how some of the IDs work

Implementation details

Built entirely on open-source software: RDFLib and our own libraries available on PyPI

tinyrml: a Python-friendly implementation of RML

- accepts `Iterable[dict]` (basically a list of objects) as source; allows Python expressions to be used in mappings

rdfhelpers: offers a way to easily compose operations

- "fluent" API: every operation produces a graph as a result of acting on the graph from the previous operation

Implementation details

```
Composable()\
    .mapIterable(global_bindings={"convert_excel_date": convert_excel_date},
                 mapping=MAPPING,
                 iterable=csv.DictReader(source)) \
    .bind("fdicib", FDICIB) \
    .bind("tmp", "https://somanyaircraft.com/data/experimental/fdicib/schema/tmp#") \
    .bind("Iptc4xmpCore", "http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/") \
    .bind("photoshop", PHOTOSHOP) \
    .bind("fibo-be-le-fbo", FIBO + "BE/LegalEntities/FormalBusinessOrganizations/") \
    .bind("fibo-fnd-dt-fd", FIBO + "FND/DatesAndTimes/FinancialDates/") \
    .bind("fibo-fnd-plc-adr", FIBO + "FND/Places/Addresses/") \
    .bind("lcc-cr", "https://www.omg.org/spec/LCC/Countries/CountryRepresentation/") \
    .bind("cmns-dt", "https://www.omg.org/spec/Commons/DatesAndTimes/") \
    .update("""
        DELETE { ?bank ?p ?o }
        WHERE {
            ?bank a fibo-be-le-fbo:Branch ; ?p ?o
            FILTER (isBlank(?bank))
        }
    """) \
    .update("""
        DELETE {
            ?bank tmp:city ?city ; tmp:address ?address ; tmp:zip ?zip ; tmp:state ?state
        }
        INSERT {
            ?bank fdicib:address [
                Iptc4xmpCore:Location ?address ;
                photoshop:City ?city ;
                photoshop:State ?state ;
                fibo-fnd-plc-adr:hasPostalCode ?zip
            ]
        }
        WHERE {
            ?bank a fibo-be-le-fbo:Branch ;
            tmp:city ?city ; tmp:address ?address ; tmp:zip ?zip ; tmp:state ?state
        }
    """) \
    .serialize(TARGET_RDF)
```

1. Create pipeline
2. Map CSV
3. Add namespaces
4. Delete anomalous data
5. Transform address data
6. Serialize

Possible enhancements

Multiple sources files (and source formats) ✓ *already supported*

- map multiple files, map repeatedly, use new formats (e.g., Excel)

More expressive mappings ✓ *already supported*

- **tinyrml** lets you use arbitrary Python expressions in mappings

Source data quality control ✓ *already supported*

- use SHACL to find errors or missing values, PROV-O for lineage

LLM integration? ✗ *under consideration*

Questions?

https://www.somanyaircraft.com

ora@somanyaircraft.com