A Gentle Introduction to SPARQL Anything

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

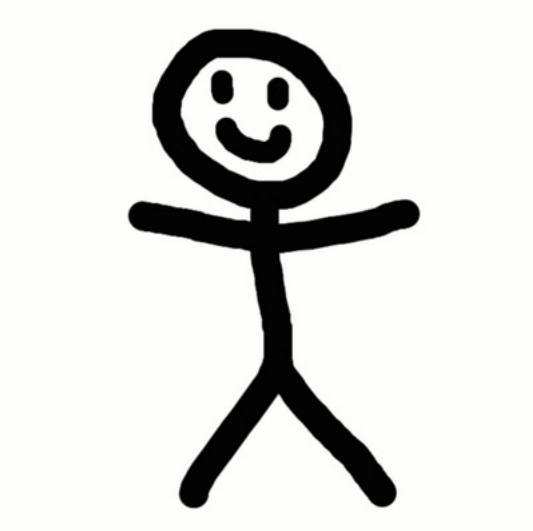
SPARQL.ANYTHING

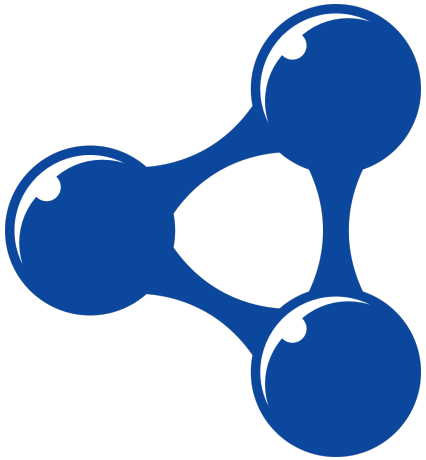
Luigi Asprino

Dipartimento di Lingue, Letterature e Culture Moderne (LILEC)
luigi.asprino@unibo.it
@LguSpree – http://luigiasprino.it

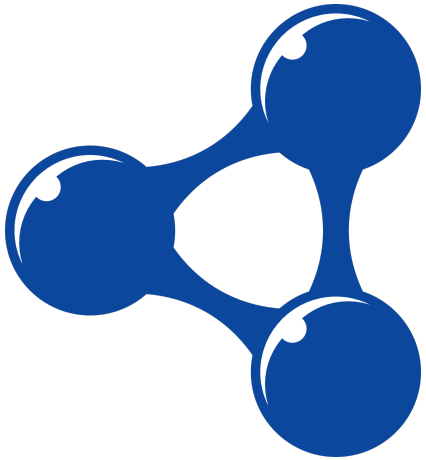
https://sparql-anything.cc/

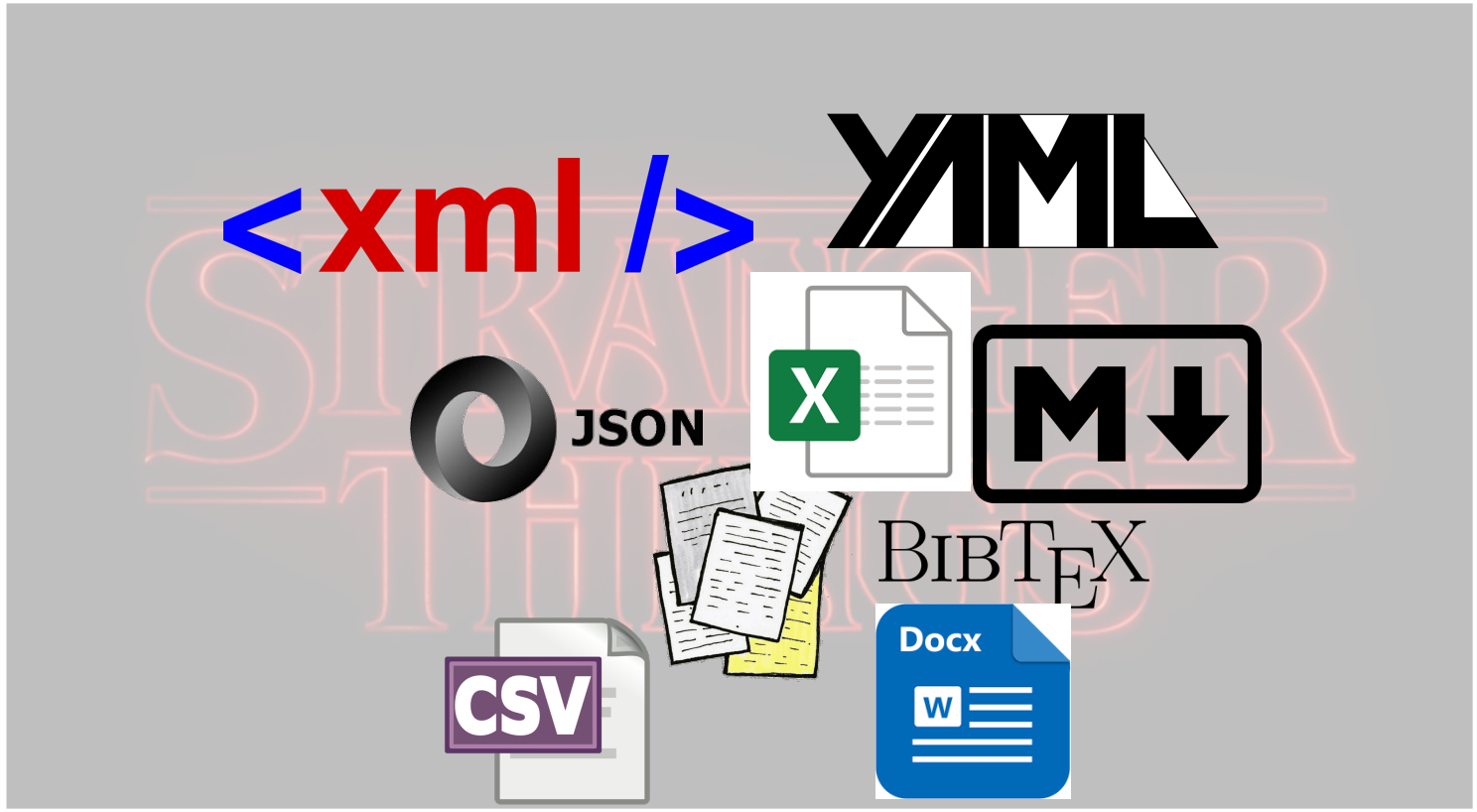
Introducing Bill



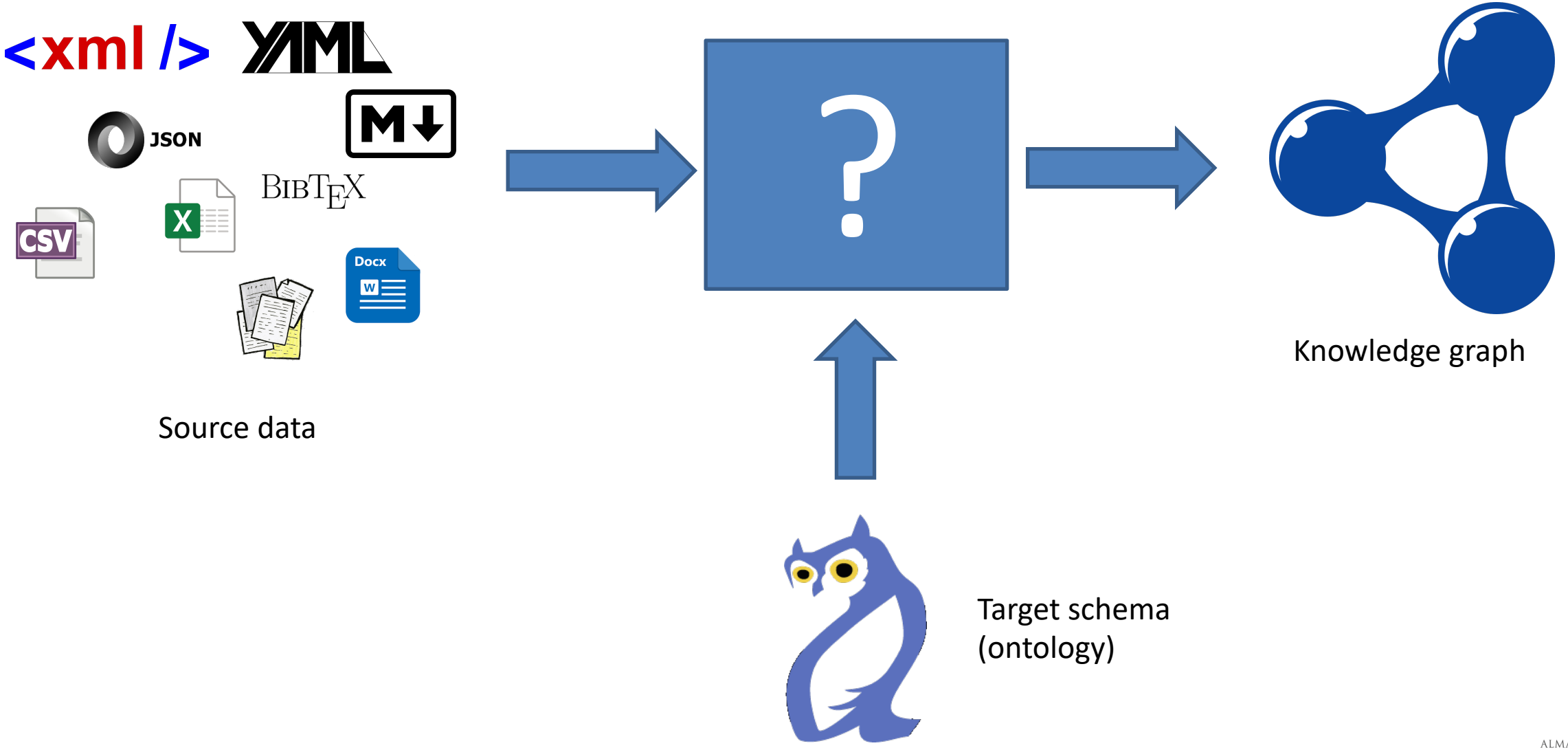


However...





Knowledge Graph Construction



Approaches to Knowledge Graph Construction

- **Ad-hoc transformers**, i.e. software tools able to transform specific sources to KGs according to specific ontologies (e.g. ArCo RDFizer [1])
    - Pros: Performance
    - Cons: one tool for each source
- **Format-specific transformers**, i.e. software tools targeting specific formats (e.g. CSV2RDF [2], JSON2RDF [3])
    - Pros: a single tool is able to transform any source of a given format
    - Cons: one tool for each format, fixed target schema
- **Mappers**, i.e. software tools transforming the sources according to a collection of rules, called mapping (e.g. RML [4], Morph [5], Ontop [6])
    - Pros: flexible transformation
    - Cons: mapping languages require high learning demands

Cons: Data needs to be transformed into RDF before it can be accessed/explored

SPARQL Anything

SPARQL Anything allows you to access any heterogenous data sources as if it was RDF.

It uses a single generic abstraction, called ***Facade-X***, to transform any data format into RDF and make it queriable from SPARQL.

- ✓ A single tool for any source/format
- ✓ A single meta-model for any data source
- ✓ No mapping is required
- ✓ Data can be accessed/explored via SPARQL

SPARQL.ANYTHING

Luigi Asprino, Enrico Daga, Aldo Gangemi, and Paul Mulholland. 2022. Knowledge Graph Construction with a façade: a unified method to access heterogeneous data sources on the Web. ACM Transactions on Internet Technologies (2022).

Facade-X

Lists

Dustin, Will, Mike, Lucas

Container Membership

[rdf:_1 “Dustin”; rdf:_2 “Will”; rdf:_3 “Mike”; rdf:_4 “Lucas”]

Maps

name -> Dustin
surname -> Henderson

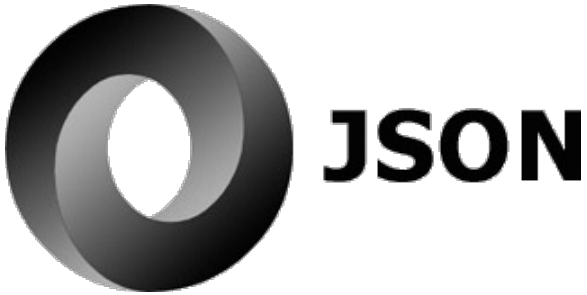
RDF resources

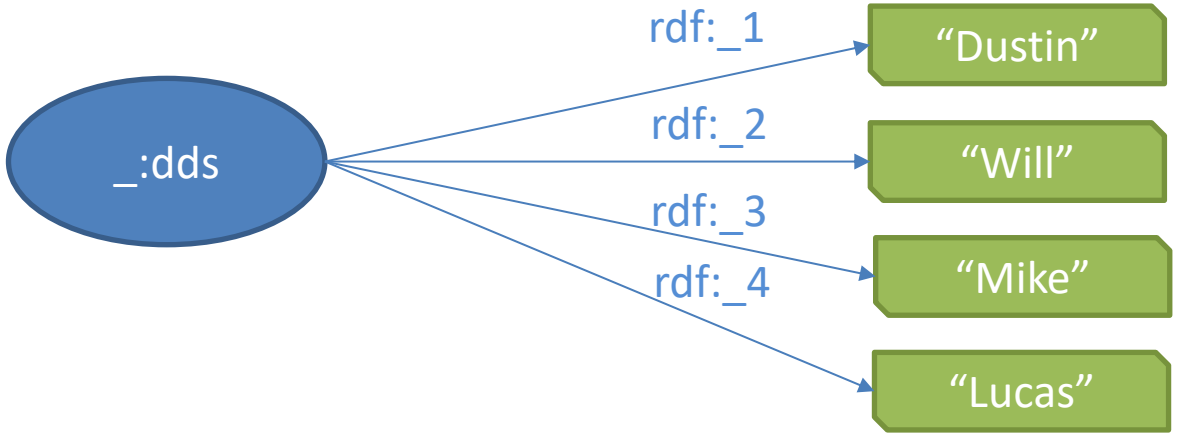
[ns:name “Dustin” ;
ns:surname “Henderson”]

Triplifying Lists

List

[“Dustin”, “Will”, “Mike”, “Lucas”]



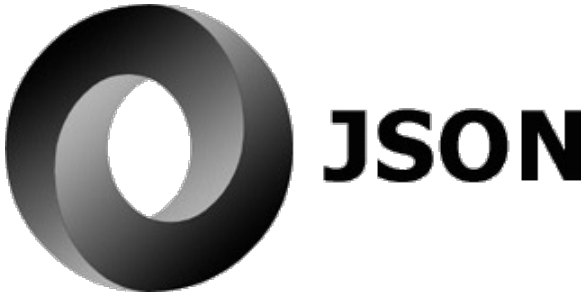


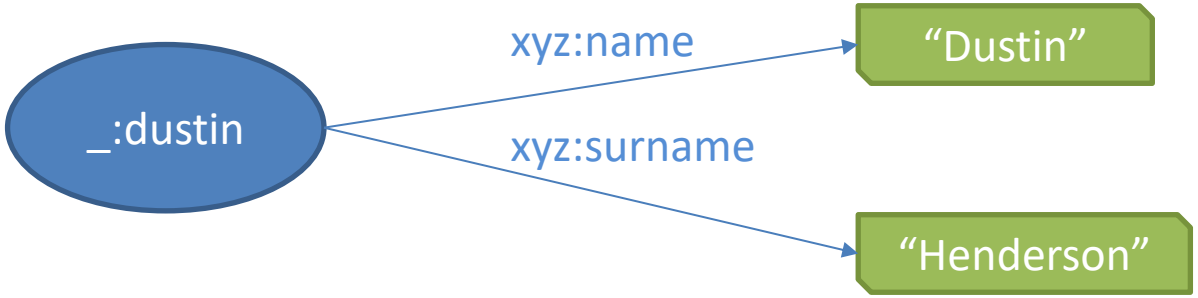
RDF Container membership properties https://www.w3.org/TR/rdf-schema/#ch_containervocab

Triplifying Maps

Maps

{“name”: “Dustin”,
“surname”: “Henderson”}

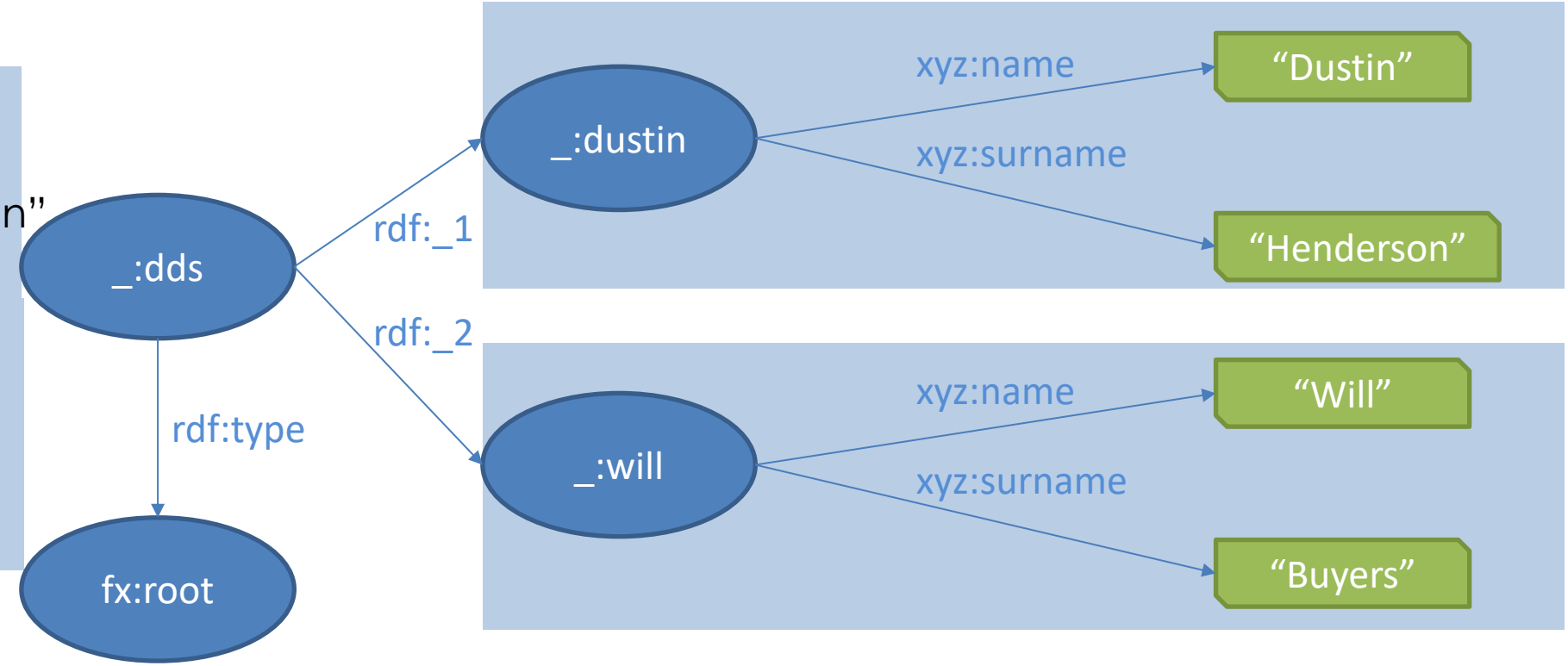




Default prefix xyz: <http://sparql.xyz/facade-x/data/> .

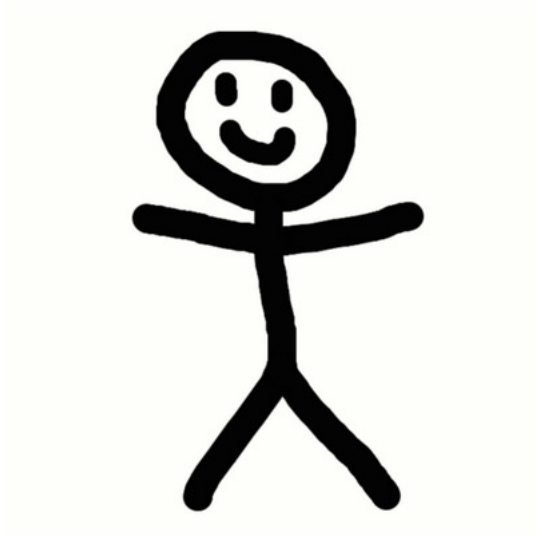
Triplifying nested collections

```
[
 {
  "name": "Dustin",
  "surname": "Henderson"
 },
 {
  "name":"Will",
  "surname": "Buyers"
 }
]
```



Default prefix fx: <http://sparql.xyz/facade-x/ns/> .

Triplifying any file format



SPARQL. ANYTHING

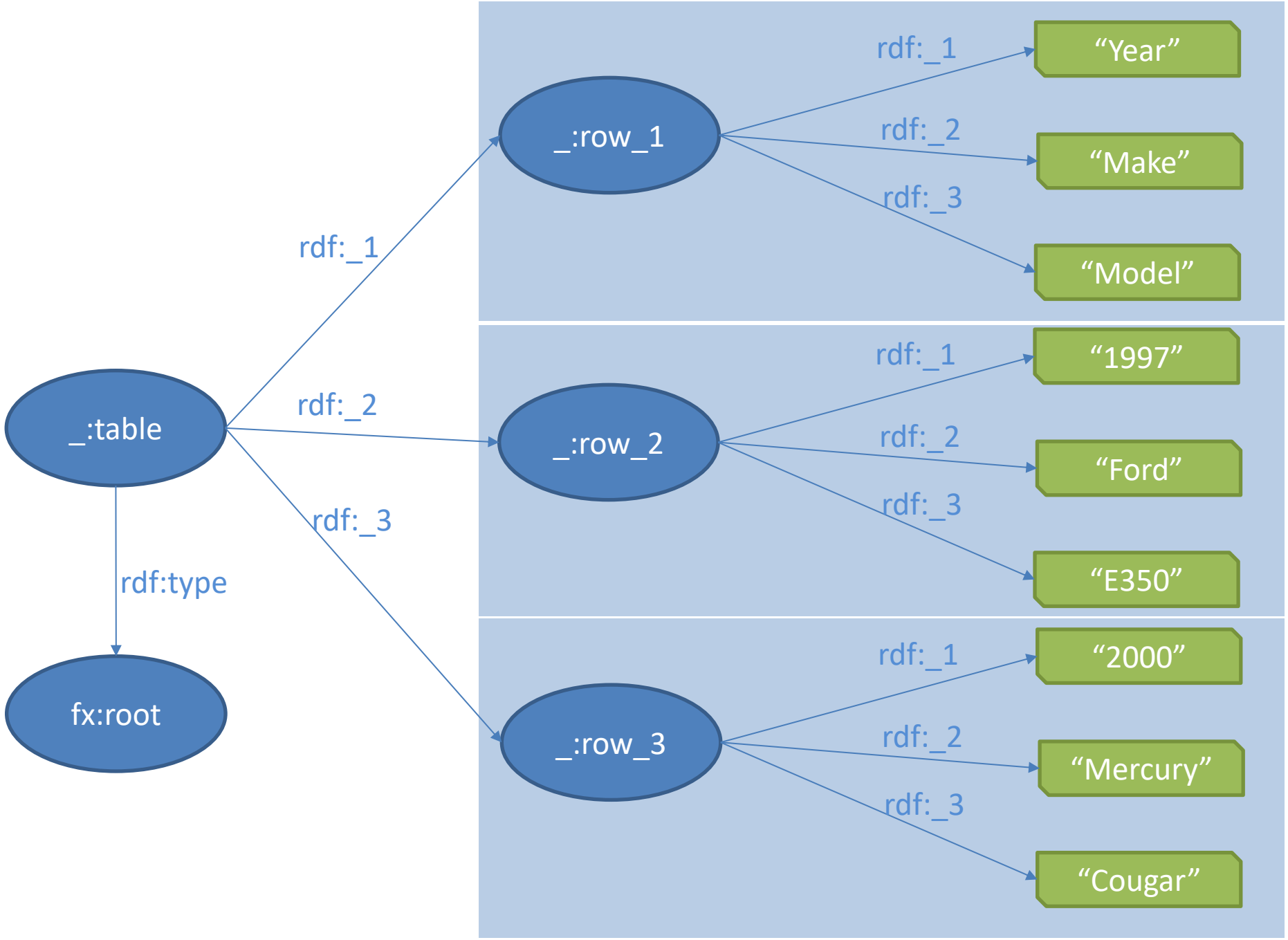
JSON

<xml />

Docx

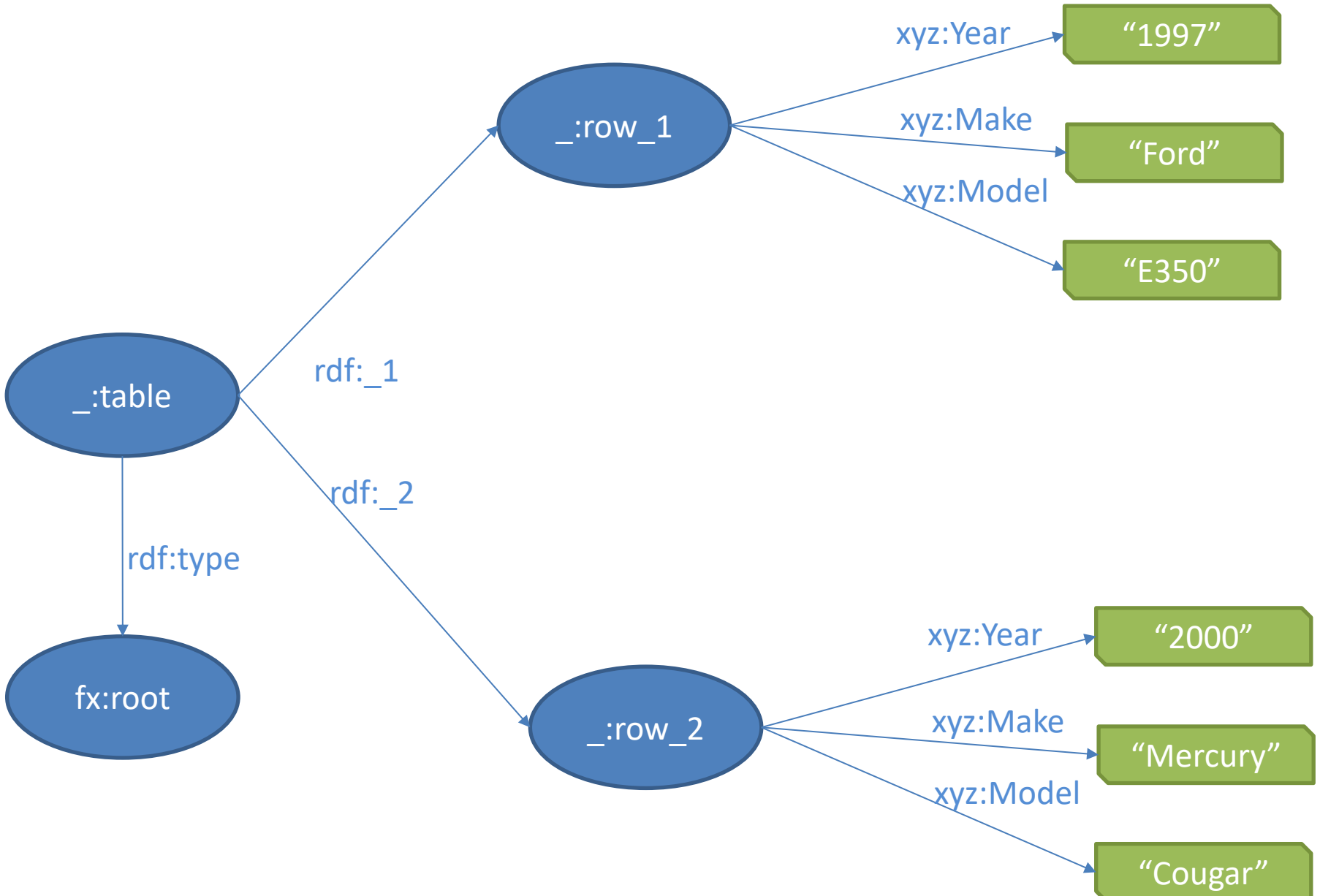
BIBTEX

Year,Make,Model
1997,Ford,E350
2000,Mercury,Cougar



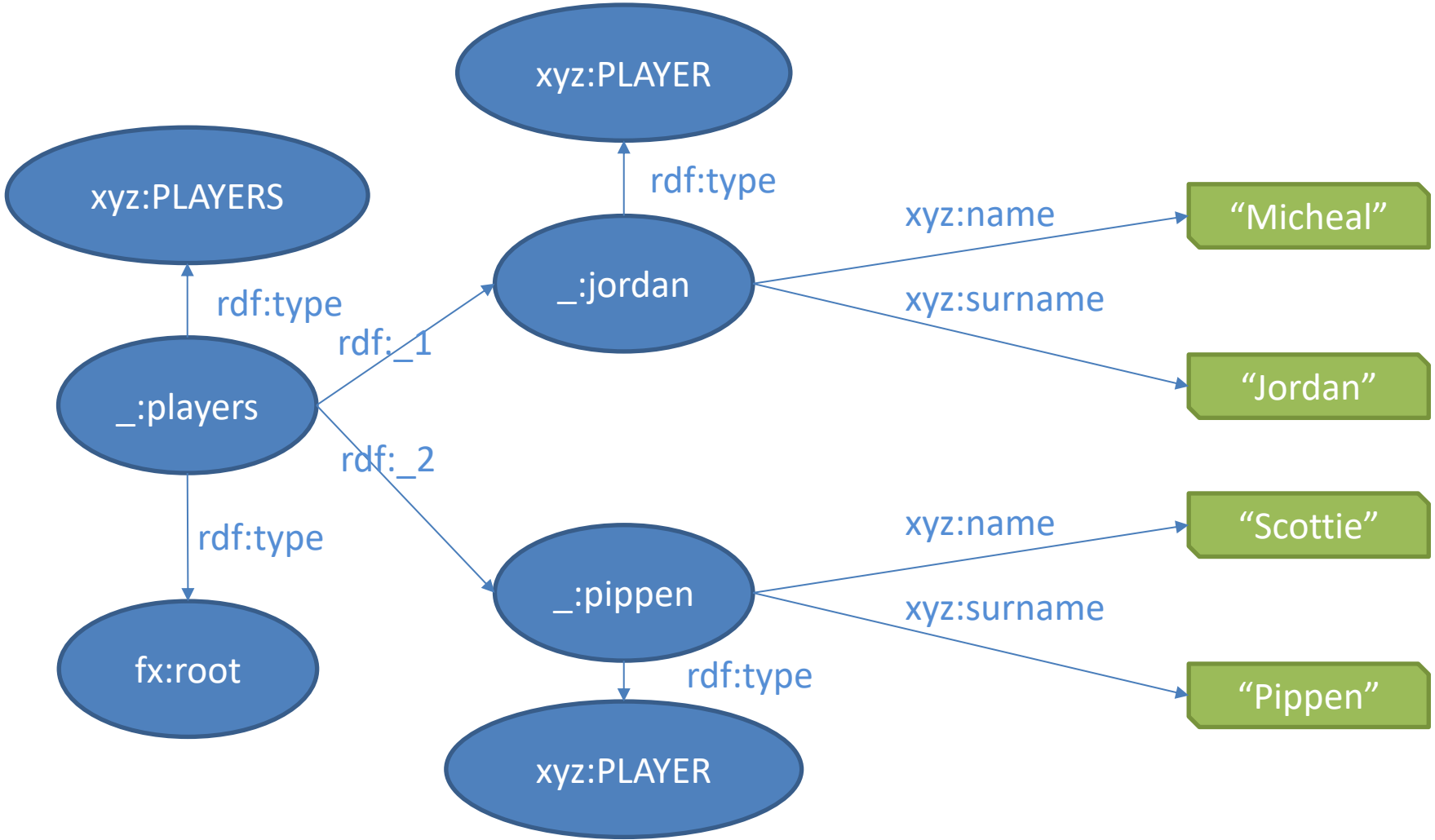
csv.headers=true

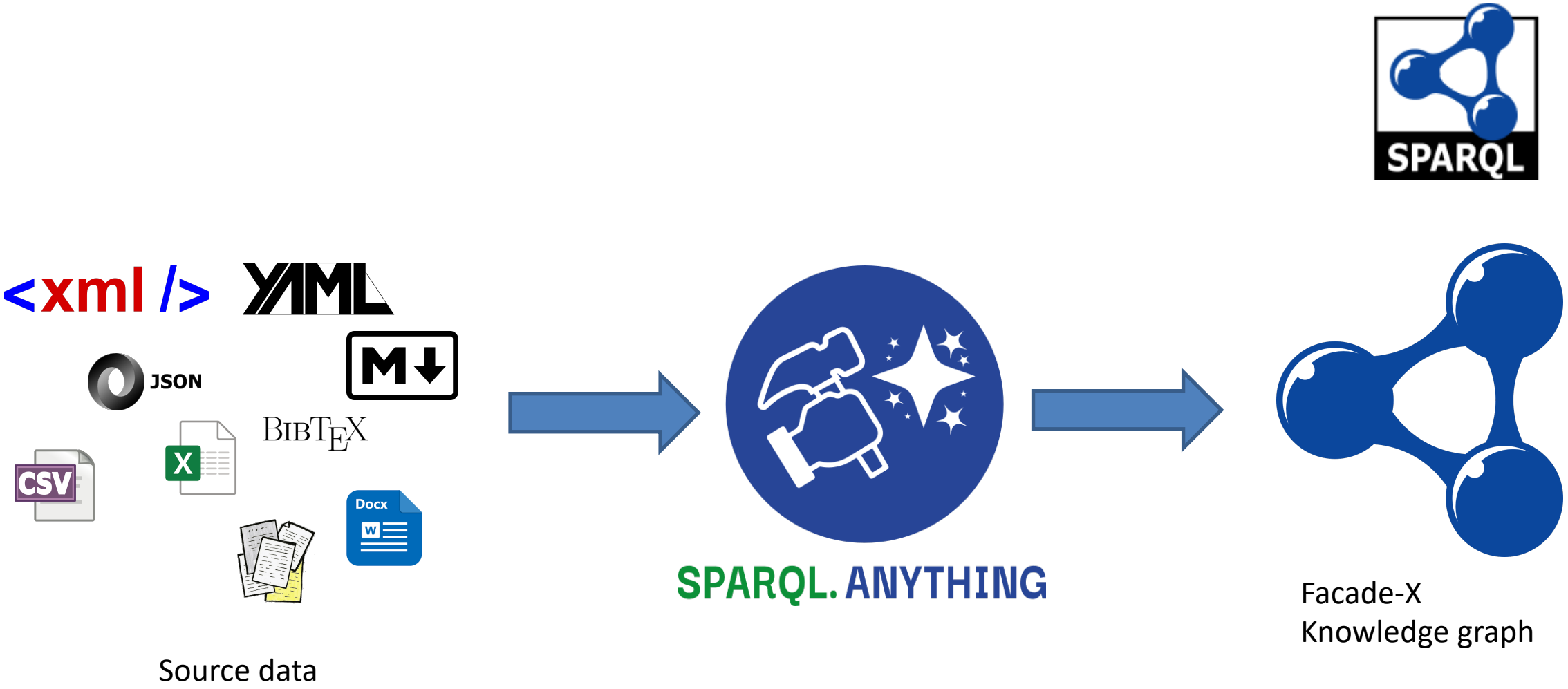
```
Year,Make,Model
1997,Ford,E350
2000,Mercury,Cougar
```



<xml />

```
<PLAYERS>
   <PLAYER name="Micheal" surname="Jordan"/>
   <PLAYER name="Scottie" surname="Pippen"/>
</PLAYERS>
```




SPARQL
<xml />
YAML
JSON
M↓
BIBTEX
CSV
X
Docx
W
SPARQL. ANYTHING
Facade-X
Knowledge graph
Source data

http://example.org/file.json

```
[
  {
    "name":"Gordon Gekko",
    "address":"123 Wall St."
  },
  {
    "name":"Daniel Ocean",
    "address":"7 Las Vegas bvd."
  },
  {
    "name":"Rocky Balboa",
    "address":"42 Market St."
  }
]
```

SPARQL Anything endpoint

Transform http://example.org/file.json into RDF according to the FacadeX metamodel

```
:1 xyz:name "Gordon Gekko" ;
    xyz:address "123 Wall St." .
:2 xyz:name "Daniel Ocean" ;
    xyz:address "7 Las Vegas bvd." .
:3 xyz:name "Rocky Balboa" ;
    xyz:address  "42 Market St." .
....
```

Evaluate query on the RDF version of "file.json"

```
:1 foaf:name "Gordon Gekko" .
:2 foaf:name "Daniel Ocean" .
:3 foaf:name "Rocky Balboa" .
....
```

```
CONSTRUCT {
        ?p foaf:name ?n
}
WHERE {
 SERVICE<x-sparql:anything:http://example.org/file.json>{
      ?p xyz:name ?n
  }
}
```

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

http://example.org/file.json

```
[
  {
    "name":"Gordon Gekko",
    "address":"123 Wall St."
  },
  {
    "name":"Daniel Ocean",
    "address":"7 Las Vegas bvd."
  },
  {
    "name":"Rocky Balboa",
    "address":"42 Market St."
  }
]
```

SPARQL Anything endpoint

Transform http://example.org/file.json into RDF according to the FacadeX metamodel

```
:1 xyz:name "Gordon Gekko" ;
    xyz:address "123 Wall St." .
:2 xyz:name "Daniel Ocean" ;
    xyz:address "7 Las Vegas bvd." .
:3 xyz:name "Rocky Balboa" ;
    xyz:address  "42 Market St." .
....
```

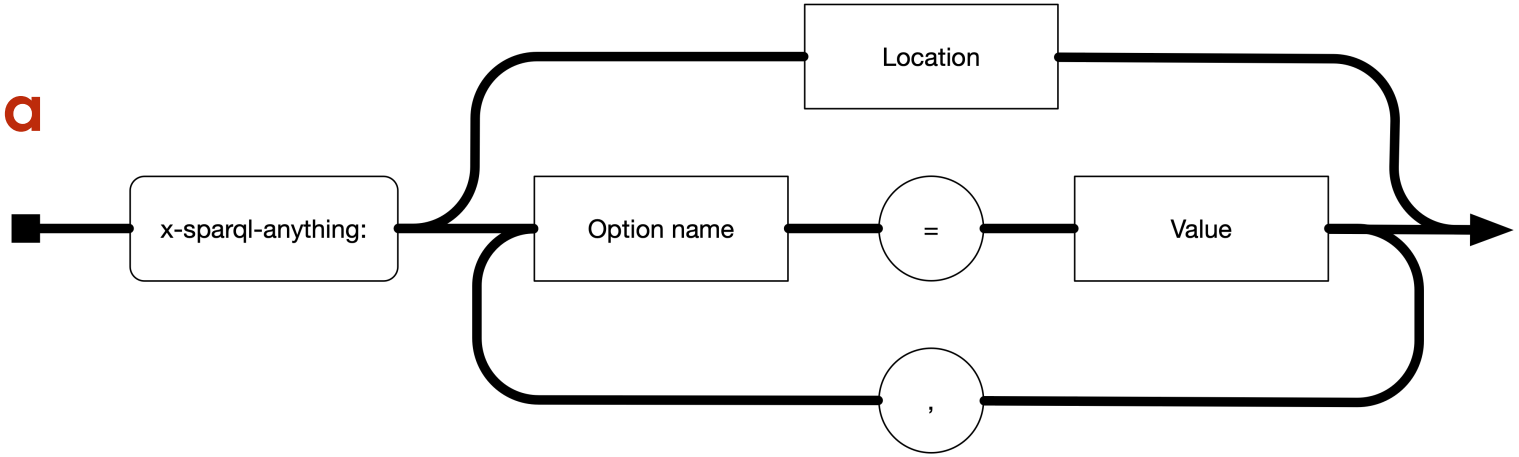
Evaluate query on the RDF version of "file.json"

```
---------------------
|     ?address     |
---------------------
| "123 Wall St." |
---------------------
```

```
SELECT ?address

WHERE {
  SERVICE<x-sparql:anything:http://example.org/file.json> {
      ?p xyz:name "Gordon Gekko"
  }
}
```

IRI schema



x-sparql-anything:location=https://sparql-anything.cc/example1.json,namespace=http://example.org/

option name

option value

A minimal URI that uses only the resource location is also possible.

x-sparql-anything:https://sparql-anything.cc/example1.json

The list of options available can be found here

https://sparql-anything.readthedocs.io/en/latest/#general-purpose-options

Installation

Requirements: Java 11+

Download the JAR executable of the SPARQL Anything server

https://github.com/SPARQL-Anything/sparql.anything/releases/download/v0.8.1/sparql-anything-server-0.8.1.jar

Run the server

```
java -jar sparql-anything-server-0.8.1.jar
```

SPARQL Anything GUI will be available at http://localhost:3000/sparql

Hands-on session

Let use SPARQL Anything to transform Propbank into a knowledge graph according to the Framester's ontology.

```
<!DOCTYPE frameset SYSTEM "frameset.dtd">
<frameset>
  <predicate lemma="eat">
    <roleset id="eat.01" name="consume, comsuming">
      <aliases>
        <alias framenet="Ingestion" pos="v" verbnet="">eat</alias>
        <alias framenet="" pos="n" verbnet="">eating</alias>
      </aliases>
      <note>EAT-V NOTES: Member of Vncls eat-39.1-1. (from eat.01-v)</note>
      <note>EATING-N NOTES: eat.01 (from eating.01-n)</note>
      <roles>
        <role descr="consumer, eater" f="pag" n="0">
          <vnrole vncls="39.1-1" vntheta="agent"/>
        </role>
        <role descr="meal" f="ppt" n="1">
          <vnrole vncls="39.1-1" vntheta="patient"/>
        </role>
      </roles>
```

Expected result

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<http://example.org/frame/eat.01>
  rdfs:label "eat.01" ;
  a <https://w3id.org/framester/schema/Frame> .

<http://example.org/frame/eat_up.02>
  rdfs:label "eat_up.02" ;
  a <https://w3id.org/framester/schema/Frame> .

<http://example.org/frame/eat_away.03>
  rdfs:label "eat_away.03" ;
  a <https://w3id.org/framester/schema/Frame> .
```

A complete example is available at https://github.com/SPARQL-Anything/showcase-propbank

References

[1] Carriero, Valentina Anita, et al. "Pattern-based design applied to cultural heritage knowledge graphs." Semantic Web 12.2 (2021): 313-357.

[2] https://github.com/AtomGraph/CSV2RDF

[3] https://github.com/AtomGraph/JSON2RDF

[4] Dimou, Anastasia, et al. "RML: A generic language for integrated RDF mappings of heterogeneous data." Ldow 1184 (2014).

[5] Chaves-Fraga, David, et al. "Enhancing virtual ontology based access over tabular data with Morph-CSV." Semantic Web 12.6 (2021): 869-902.

[6] Calvanese, Diego, et al. "Ontop: Answering SPARQL queries over relational databases." Semantic Web 8.3 (2017): 471-487.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Luigi Asprino

Dipartimento di Filologia Classica e Italianistica (FICLIT)

luigi.asprino@unibo.it