Knowledge Graph Construction & SPARQL Anything

Enrico Daga

The Open University

Knowledge Graphs

Definition: a KG …

(i) mainly describes real world entities and their interrelations, organized in a graph,

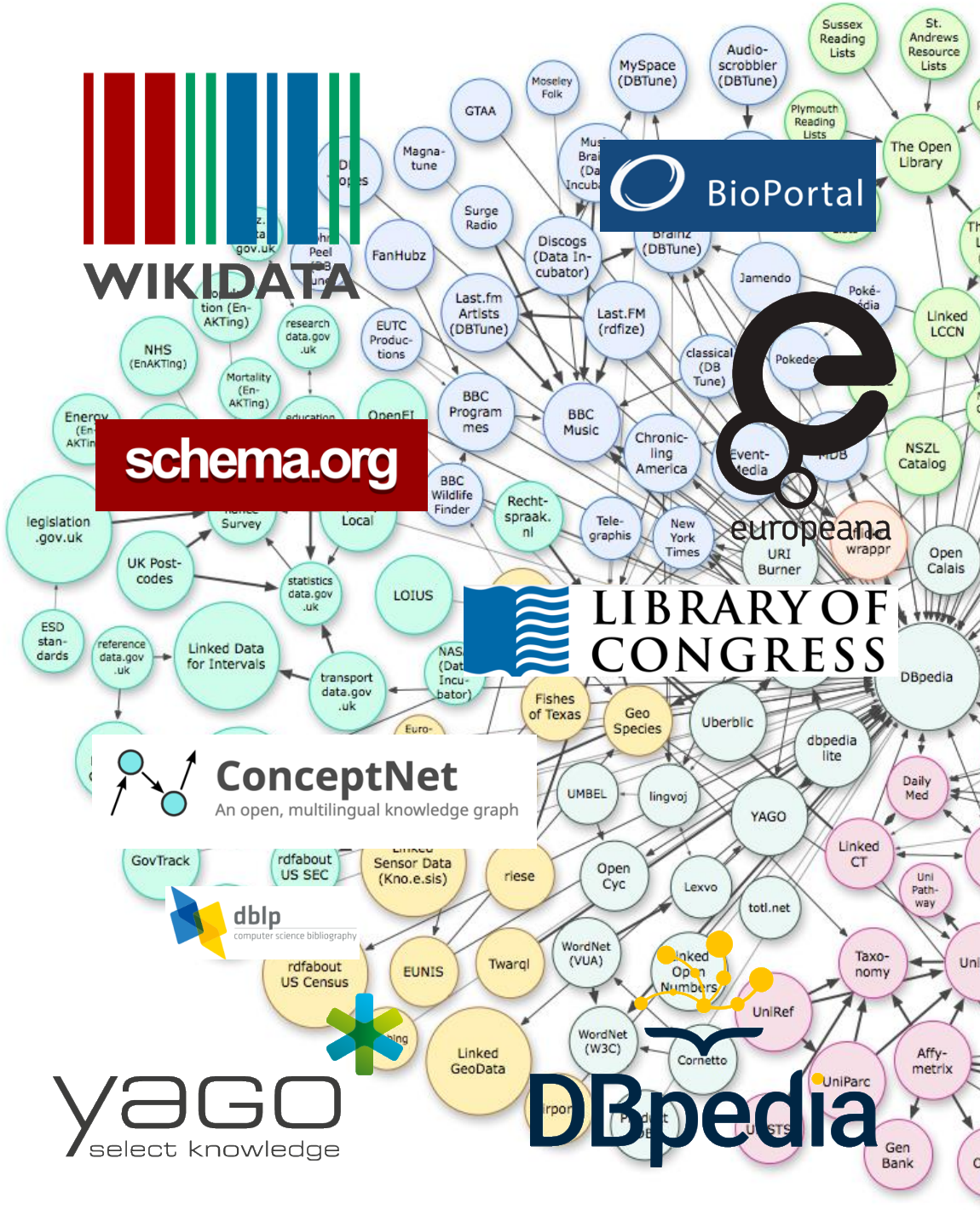
(ii) defines possible classes and relations of entities in a schema

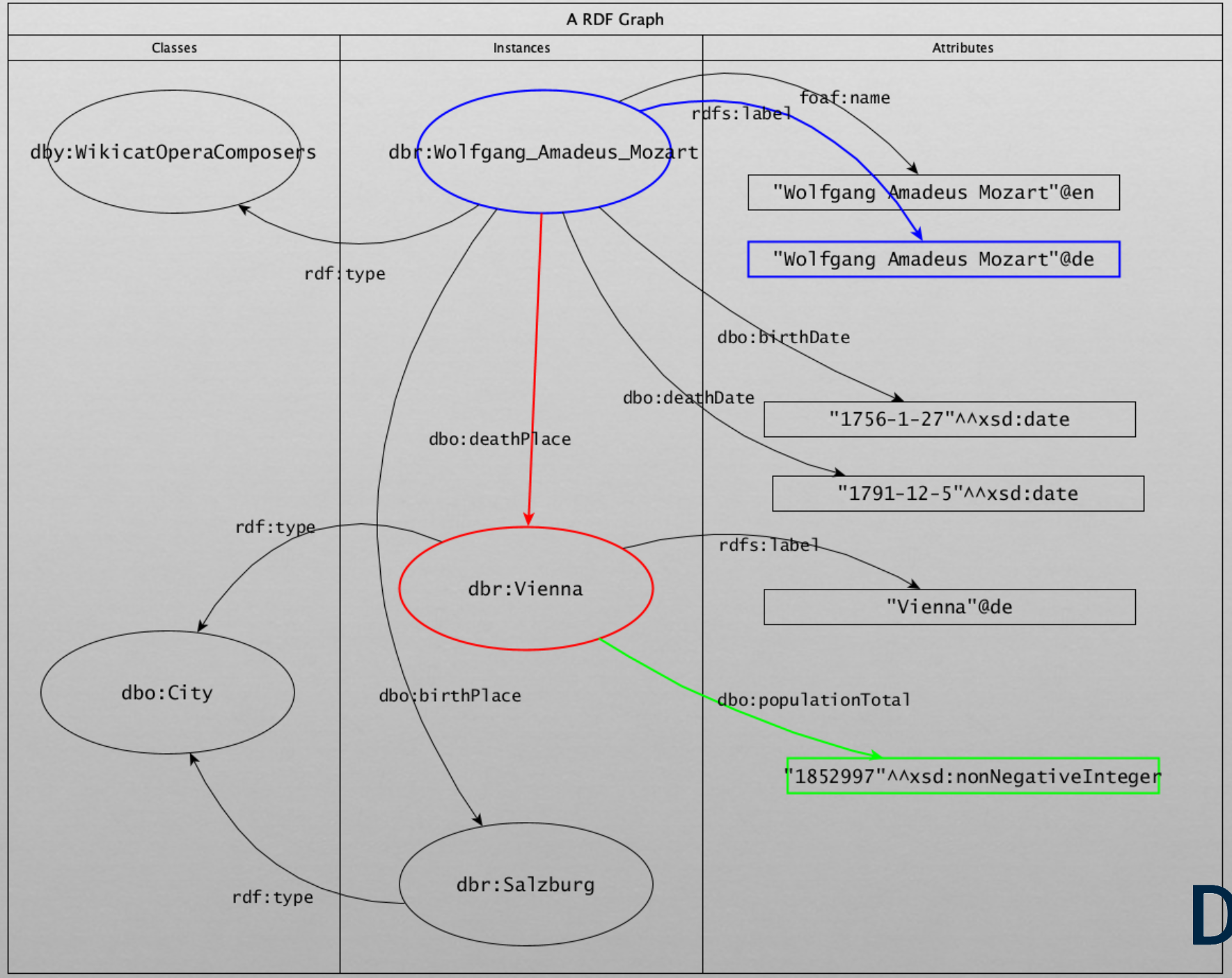
(iii) allows for potentially interrelating arbitrary entities with each other

(iv) covers various topical domains. [Paulheim, 2017]

R D F

SPARQL



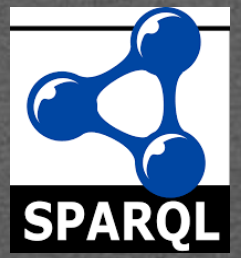

A RDF Graph
Classes
Instances
Attributes
dby:WikicatOperaComposers
dbr:Wolfgang_Amadeus_Mozart
foaf:name
rdfs:label
"Wolfgang Amadeus Mozart"@en
"Wolfgang Amadeus Mozart"@de
rdf:type
dbo:birthDate
dbo:deathDate
"1756-1-27"^^xsd:date
dbo:deathPlace
"1791-12-5"^^xsd:date
rdf:type
rdfs:label
dbr:Vienna
"Vienna"@de
dbo:City
dbo:birthPlace
dbo:populationTotal
"1852997"^^xsd:nonNegativeInteger
dbr:Salzburg
rdf:type


```
# RDF triples

dbr:Wolfang_Amadeus_Mozart rdfs:label “Wolfgang Amadeus
Mozart”@de .

dbr:Wolfang_Amadeus_Mozart dbo:deathPlace dbr:Vienna .

dbr:Vienna dbo:populationTotal
“1852997”^^xsd:nonNegativeInteger .
```


```
# SPARQL allows us to query the graph via pattern matching

?person rdfs:label ?label .

?person dbo:deathPlace ?place .

?place dbo:populationTotal ?population .
```

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

# Example of a SELECT query, retrieving 2 variables

SELECT ?person ?label
WHERE {
  #This is our graph pattern
  ?person rdfs:label ?label ;
          dbo:deathPlace dbr:Vienna
}
```

```
PREFIX schema: <http://schema.org/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

# Example of a CONSTRUCT query, which generates a new KG

CONSTRUCT {
   ?person rdf:type schema:Person ;
                   rdfs:label ?label ;
                   schema:deathPlace ?deathPlace

} WHERE {
  ?person rdfs:label ?label ;
          dbo:deathPlace ?deathPlace
}
```



schema.org

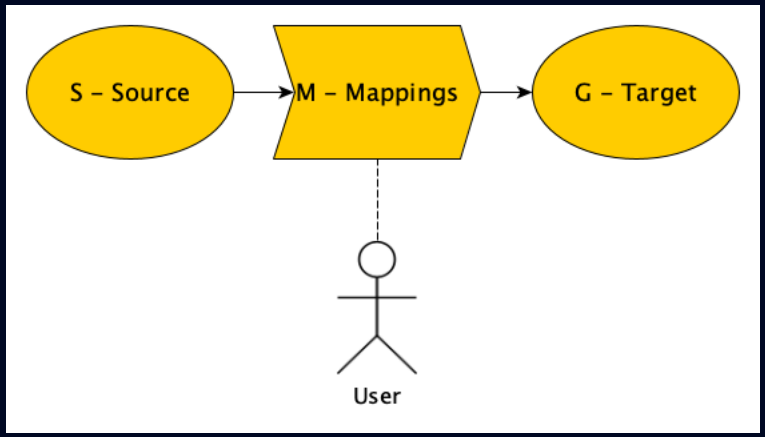
Data integration

"Data Integration is the dominant use case for Knowledge Graphs "- [Atkin, 2021, in Lassila et al, 2021]

Definition: *"Data integration involves combining data residing in different sources and providing users with a unified view of them"* [Lenzerini, 2002]

Data integration systems are defined as (G, S, M)

- G is the target common (global) schema
- S is the heterogeneous set of source schemas
- M are the mapping that maps queries between the source and the global schemas



Data integration theory assumes that data sources are *format-compatible*, i.e. it does not account for syntactic heterogeneity (it abstracts from the actual formats or assumes the same format, e.g. a relational database)

The Open University

SPICE

Social cohesion, Participation, and Inclusion through Cultural Engagement

SPICE Linked Data Hub

The SPICE Linked Data Hub forms part of the Linked Data infrastructure used within the wider SPICE project, an EU-H2020 funded project dedicated to citizen curation and cultural heritage.

The Linked Data Hub supports the acquisition and management of both static files and dynamic streaming data from a variety of sources.

Featured datasets

Irish Museum of Modern Art

Metadata of the Irish Museum of Modern Art collection.

Documents: 3698

FTM: Palazzo Madama

Open data from Fondazione Torino Musei: The Palazzo Madama Collection.

Documents: 5017

FTM: Museo di Arte Orientale

Open data from Fondazione Torino Musei: The Museo di Arte Orientale Collection.

Documents: 2044

FTM: Galleria di Arete Moderna

Open data from Fondazione Torino Musei: The Galleria di Arete Moderna Collection.

Documents: 24915

FTM: Fondo Gabinio

Open data from Fondazione Torino Musei: The Fondo Gabinio Collection.

Documents: 11404

The Tate Collection

2014 snapshot of metadata for around 70,000 artworks that Tate owns or jointly owns with the National Galleries of Scotland. Metadata for around 3,500 associated artists is also included.

Documents: 69186

https://spice-h2020.eu/

Polifonia

Playing the soundtrack of our history

Preserving musical heritage through knowledge graphs

Managing musical heritage collections through knowledge graphs

Studying musical heritage through (interlinked) knowledge graphs

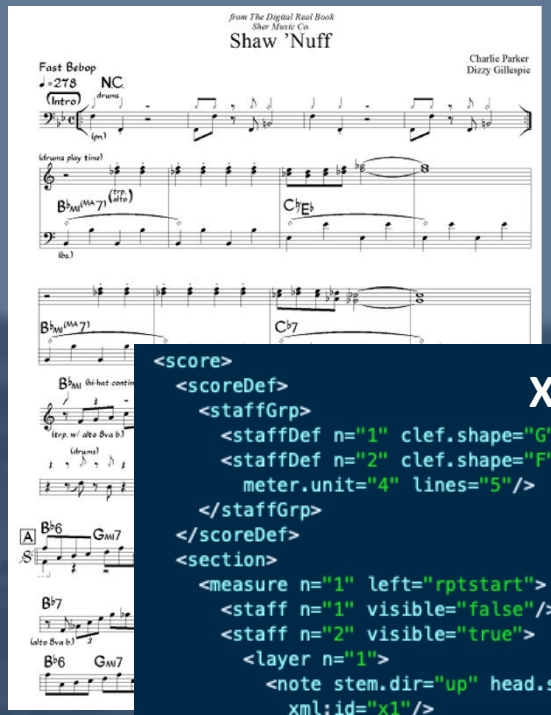
https://polifonia-project.eu/

The Open University

“Source” can be … anything!

```bibtex
@article{Knuth1984,
  title={Literate Programming},
  author={Donald E. Knuth},
  journal={The Computer Journal},
  volume={27},
  number={2},
  pages={97--111},
  year={1984},
  publisher={Oxford University Press}
}
```

BibTex



Goal

Andrea's goal is to discover and explore sacred music written from the period of th
unexpected links with his scholarly studies.

Scenario

Andrea is been very busy studying ancient texts and history books and he would lik
organs. However, he doesn't actually know what to look for. He is fond of sacred m
him if he could read curiosities about these topics or his favorite composers. And v
links to his scholarly studies without resorting to extensive study and research fron

Competency questions

CQ1: Can I find interesting materials without applying filters?

CQ2: Wha

CQ3: Is th

CQ4: Car

CQ5: Hov

CQ6: Hov

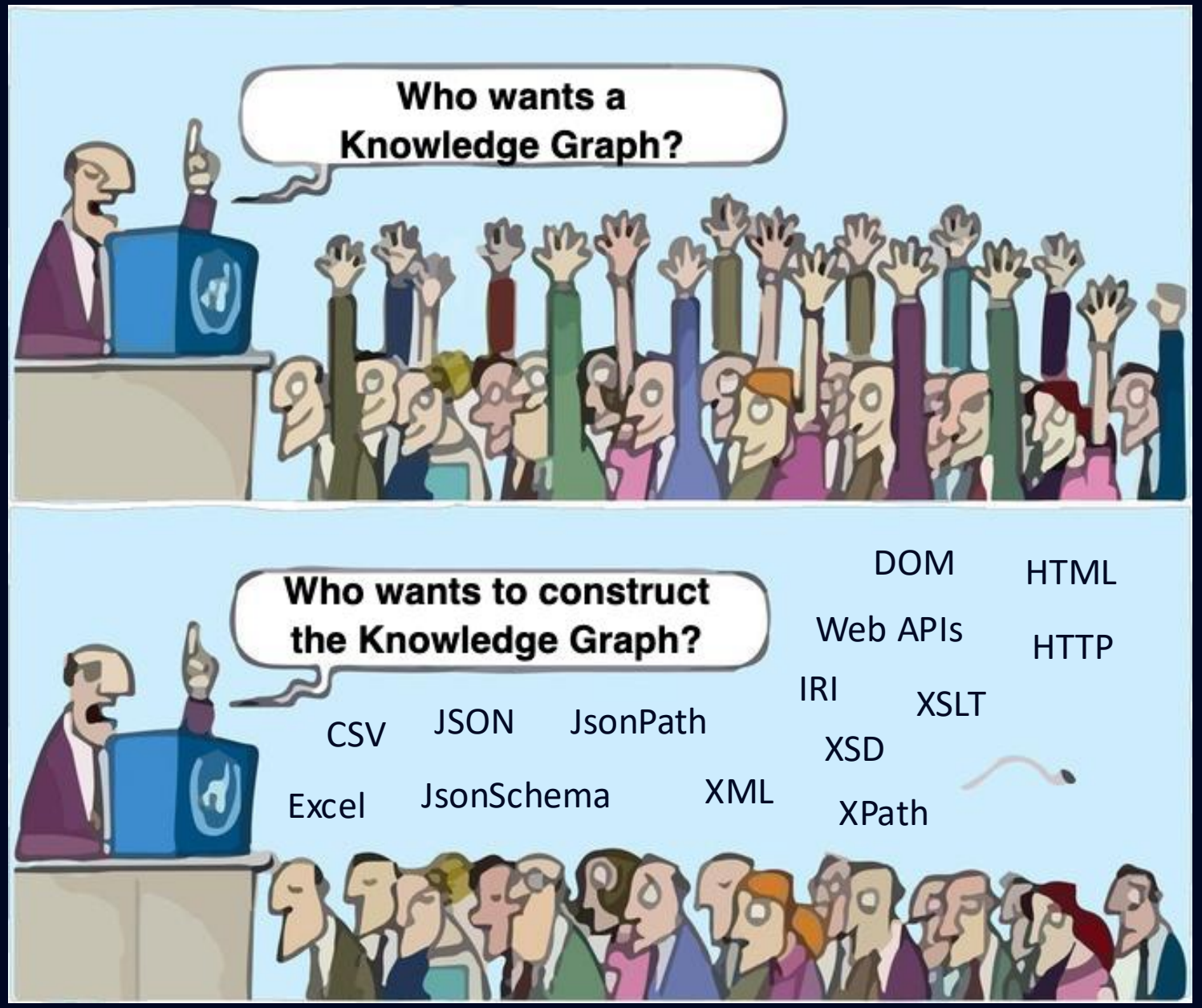
```
## Scenario
Andrea is been very busy studying ancient texts and history books and he would l
to his passion for music and organs. However, he doesn't actually know what to l
sacred music, classical music and organs and it would be cool for him if he coul
these topics or his favorite composers. And who knows, he might also find intere
to his scholarly studies without resorting to extensive study and research from
libraries.

## Competency questions

CQ1: Can I find interesting materials without applying filters?

CQ2: What types of resources can I find?

CQ3: Is there a way of visualizing all the materials connected to my interests?

CQ4: Can I keep getting suggestions in real time?

CQ5: How can I share what I find on the site?

CQ6: How I personalize my navigation experience without knowing the filtering cr
characteristics?
```

Markdown

HTML

```
<score>
  <scoreDef>
    <staffGrp>
      <staffDef n="1" clef.shape="G" clef.line="2" lines="5"
      <staffDef n="2" clef.shape="F" clef.line="4" key.sig="
        meter.unit="4" lines="5"/>
    </staffGrp>
  </scoreDef>
  <section>
    <measure n="1" left="rptstart">
      <staff n="1" visible="false"/>
      <staff n="2" visible="true">
        <layer n="1">
          <note stem.dir="up" head.shape="x" cue="true" dur=
            xml:id="x1"/>
          <note stem.dir="up" head.shape="x" cue="true" dur=
            xml:id="x2"/>
          <rest dur="2" cue="true"/>
        </layer>
        <layer n="2">
          <note pname="f" oct="3" dur="4" stem.dir="down" xm
          <note pname="f" oct="2" dur="4" stem.dir="up" xml:
          <rest dur="2"/>
        </layer>
      </staff>
      <tempo staff="1" tstamp="1" mm="278" place="above"/>
      <dir staff="1" place="above" tstamp="0">Fast Bebop</di
      <dir staff="1" place="above" tstamp="0">Intro</dir>
      <harm staff="1" place="above" tstamp="1">N.C.</harm>
      <dir staff="1" place="above" tstamp="1">drums</dir>
      <dir staff="2" place="below" tstamp="1">(pn.)</dir>
    </measure>
    <measure n="2">
      <staff n="1" visible="false"/>
      <staff n="2" visible="true">
        <layer n="1">
          <beam>
            <note pname.ges="b" oct="3" head.shape="x" xml:i
            <note pname.ges="b" oct="3" head.shape="x" xml:i
```

XML/MEI



Knowledge Graph Construction (KGC)

The Open University

Approaches

Targeting specific types/formats:

- Direct mapping https://www.w3.org/TR/rdb-direct-mapping/
- Tarql https://tarql.github.io/
- Apache Any23 https://any23.apache.org/
- JSON2RDF https://github.com/AtomGraph/JSON2RDF
- CSV2RDF https://github.com/AtomGraph/CSV2RDF
- CSVW, COW https://csvw.org/
- SPARQL Micro-services [Michel, 2019]

- Lack generality

Approaches

Specialised mapping languages, several types of

- R2RML https://www.w3.org/TR/r2rml/
- RML https://rml.io/specs/rml/ [Dimou, 2014]
- ShexML https://shexml.herminiogarcia.com/ [García-González, 2020]

- High learning demands
- Cold start problem
- Limited to few popular format
- Difficult to extend.

```
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ql: <http://semweb.mmlab.be/ns/ql#> .
@prefix : <http://example.org/rules/> .
@prefix ex: <http://example.org/> .

:TriplesMap a rr:TriplesMap;
  rml:logicalSource [
    rml:source "data/COLLEZIONI_PALAZZO_MADAMA_marzo2017.json";
    rml:referenceFormulation ql:JSONPath;
    rml:iterator "$.[*]"
  ].

:TriplesMap rr:subjectMap [
  rr:termType rr:BlankNode
].

:TriplesMap rr:predicateObjectMap [
  rr:predicate ex:Inventario ;
  rr:objectMap [
  rml:reference "Inventario"
 ]
].
```

RML

Approaches

Extending SPARQL with custom features: SPARQL Generate, high learning demands, difficult to extend to other formats. [Lefrançois, 2017]

```
PREFIX ite: <http://w3id.org/sparql-generate/iter/>
PREFIX ex: <http://exmaple.org/>


GENERATE {
[] ex:Autore ?autore ;
     ex:Datazione ?datazione ;
     ex:Titolo ?titolo ;
   ex:Tecnica ?tecnica ;
     ex:Immagine ?immagine ;
     ex:Dimensioni ?dimensioni .
}
ITERATOR ite:JSONPath(<1[illegible].json>,"$[*]","$.Autore","$.Datazione","$.Titolo","$.Tecnica","$.Immagine","$.Dimensioni")
   AS ?obj ?autore ?datazione ?titolo  ?tecnica ?immagine ?dimensioni
```

- **Cognitive complexity**: solutions transfer data source complexity to the user (e.g. need to know XPath for XML, JsonPath for JSON, …) or ask the user to learn a new language!

The Open University

Instead, our aim is

- Target an **open ended** set of formats (extendibility without changes to user-facing tool)

- **Reduce the learning curve** for Knowledge Graph practitioners (including non-developers from Healthcare, Cultural heritage, …)
  - Many SPARQL users fall into the category of end-user developer [Lieberman, 2006]
  - In a survey [Warren et al, 2018] **42% KG/SPARQL users are from non-IT areas**, including social sciences and the humanities, business and economics, and biomedical, engineering or physical sciences.

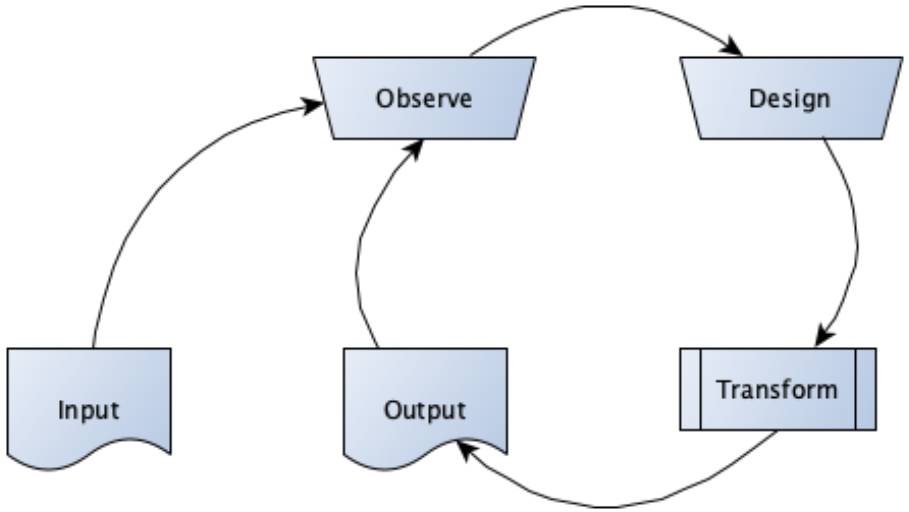
The Open University

KGC

Iterative process:

- **Observe**: the resource (e.g. a CSV file)
- **Design** mappings to a target ontology
- **Transform**: execute the mappings
- **Observe**: compare / evaluate

Trail and error approach, many iterations



KGC, revisited

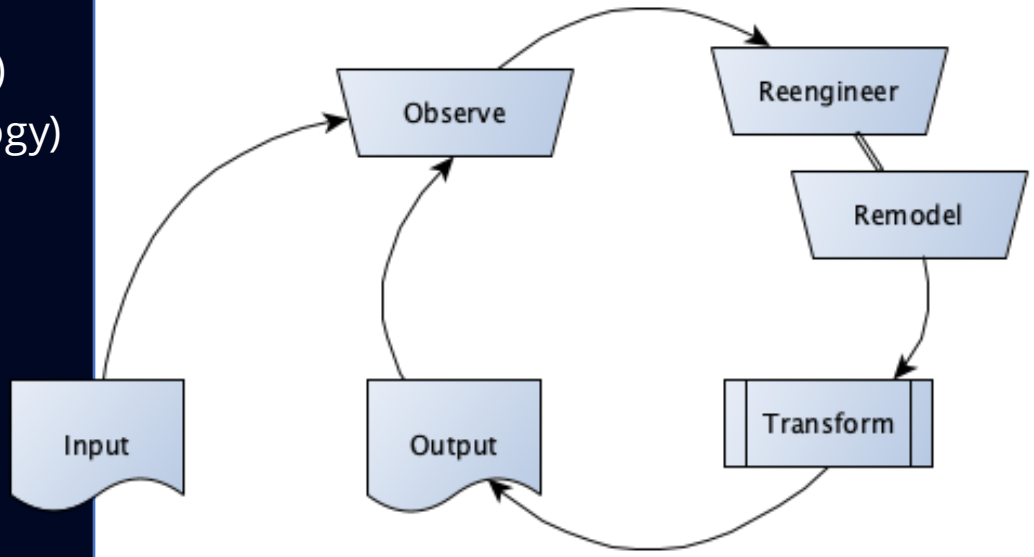
KG construction is a twofold job:

- perform a syntax/meta-model conversion (e.g. CSV to RDF)
- *project* semantics onto the data (applying a domain ontology)

A better model of the user process:

- **Observe**: the resource (e.g. a CSV file)
- **Reengineering Design**: what *syntax/meta-model* do I want?
- **Remodelling Design**: what *semantics* we want?
- **Transform**: execute the mappings
- **Observe**: compare / evaluate

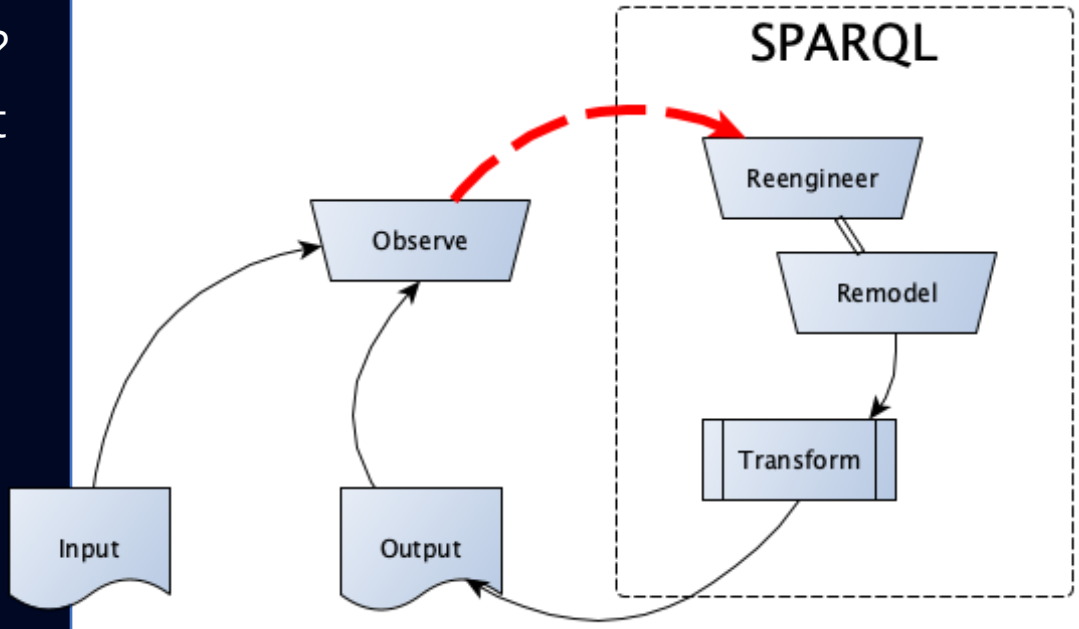
Trail and error approach, many iterations



KGC, opinionated

- **Reengineering**: what *syntax/meta-model* do we want?
  - We cannot know what structure our user wants but we know the meta-model: **RDF**
- **Remodelling**: what *semantics* do we project?
  - **SPARQL** is great for projecting semantics (change namespaces, create entities from literals, adding types, sophisticated relationships, composite structures, …)
  - **Data Integration** theory here!

How to develop a unified approach to *re-engineer* an open-ended set of formats into RDF?

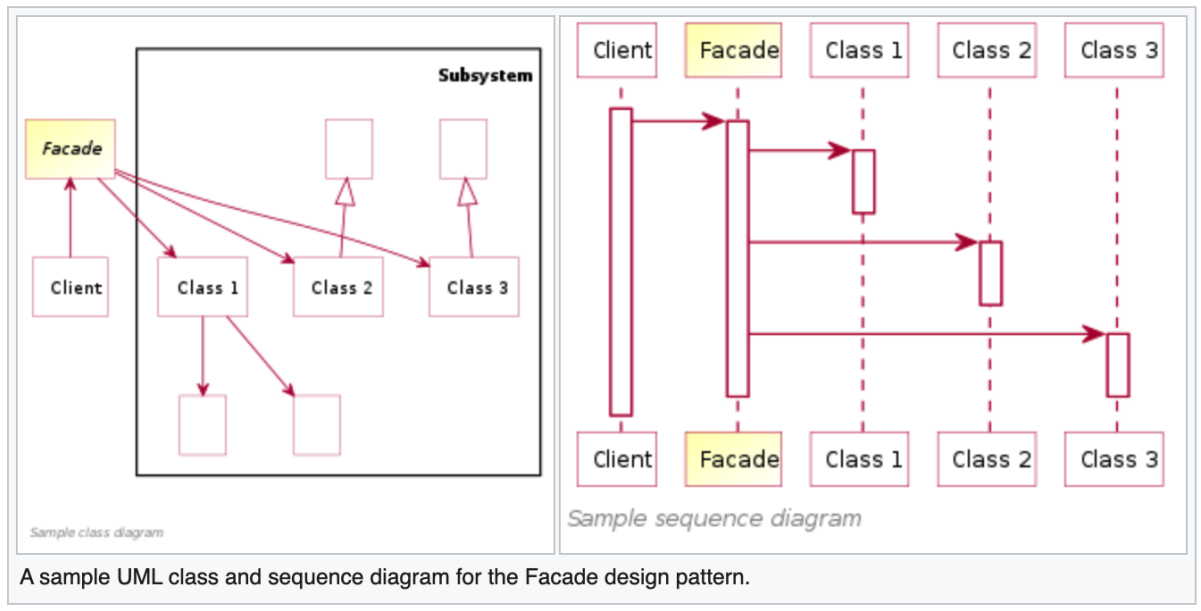


Concept

Façade Design Pattern

From Object Oriented Programming

A single abstraction on different, alternative interfaces



A sample UML class and sequence diagram for the Facade design pattern.

https://en.wikipedia.org/wiki/Facade_pattern

An RDF façade?

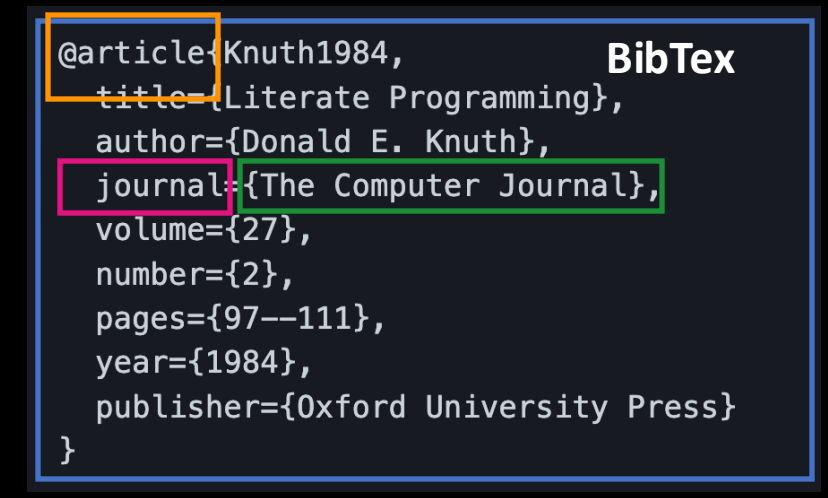
- A common RDF structure over diverse formats
- Focusing on the meta-model (data *structure*)
- Leaving domain semantics *as-it-is*!
  - apply *the least possible "ontological commitment"*

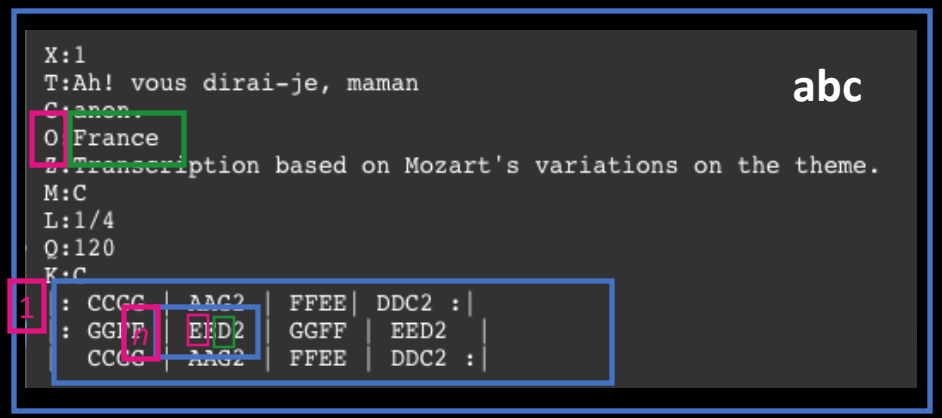
- **Problem Space**: CSV, JSON, HTML, XML, Binary (JPEG, PNG, …),Text
- **Solution space**: RDFS

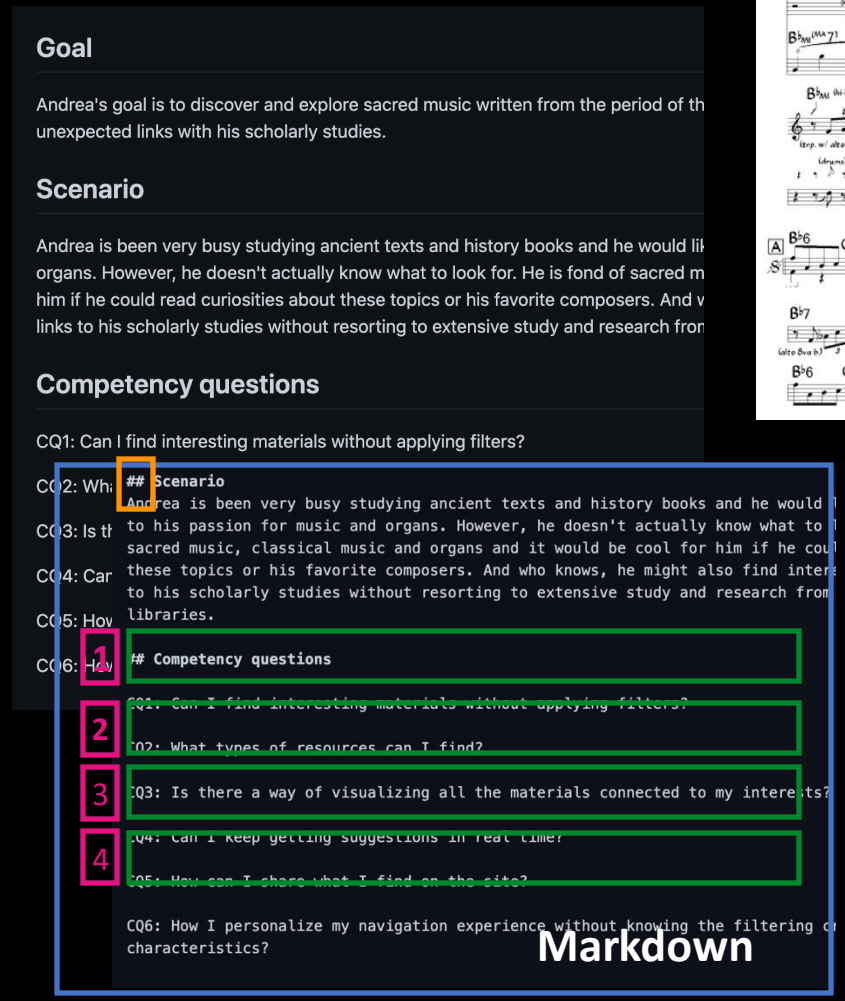
The Open University

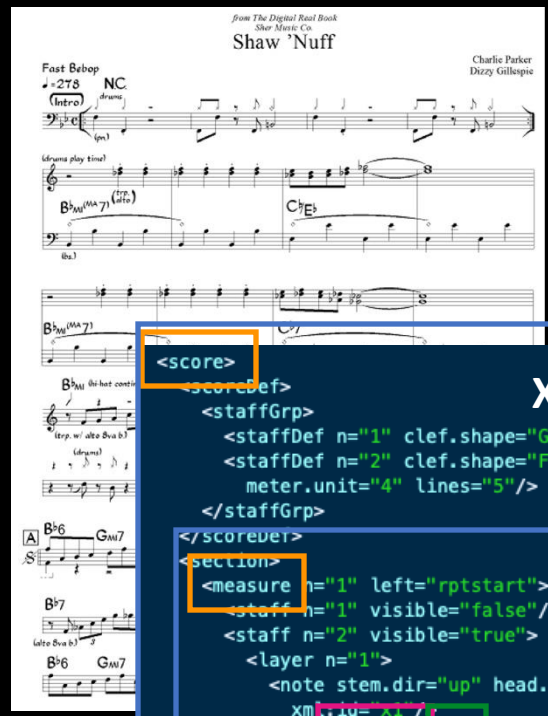
A simple intuition:

Data formats rely on a common set of primitives ...









Facade X

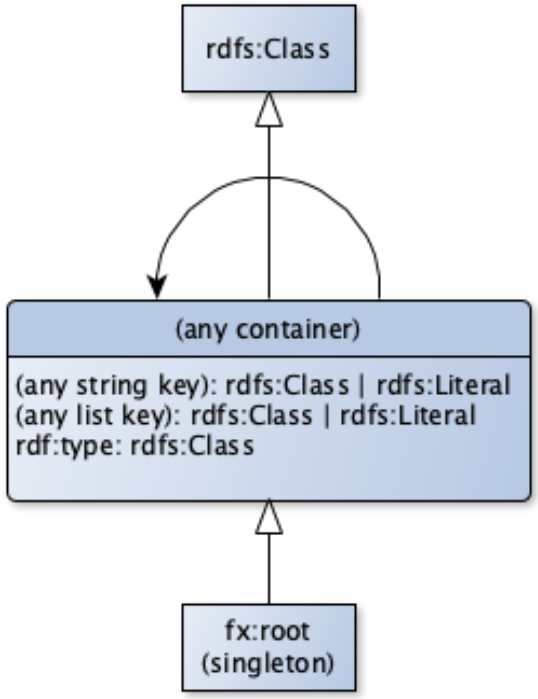
A simplified RDF meta-model, resembling a *list-of-lists*

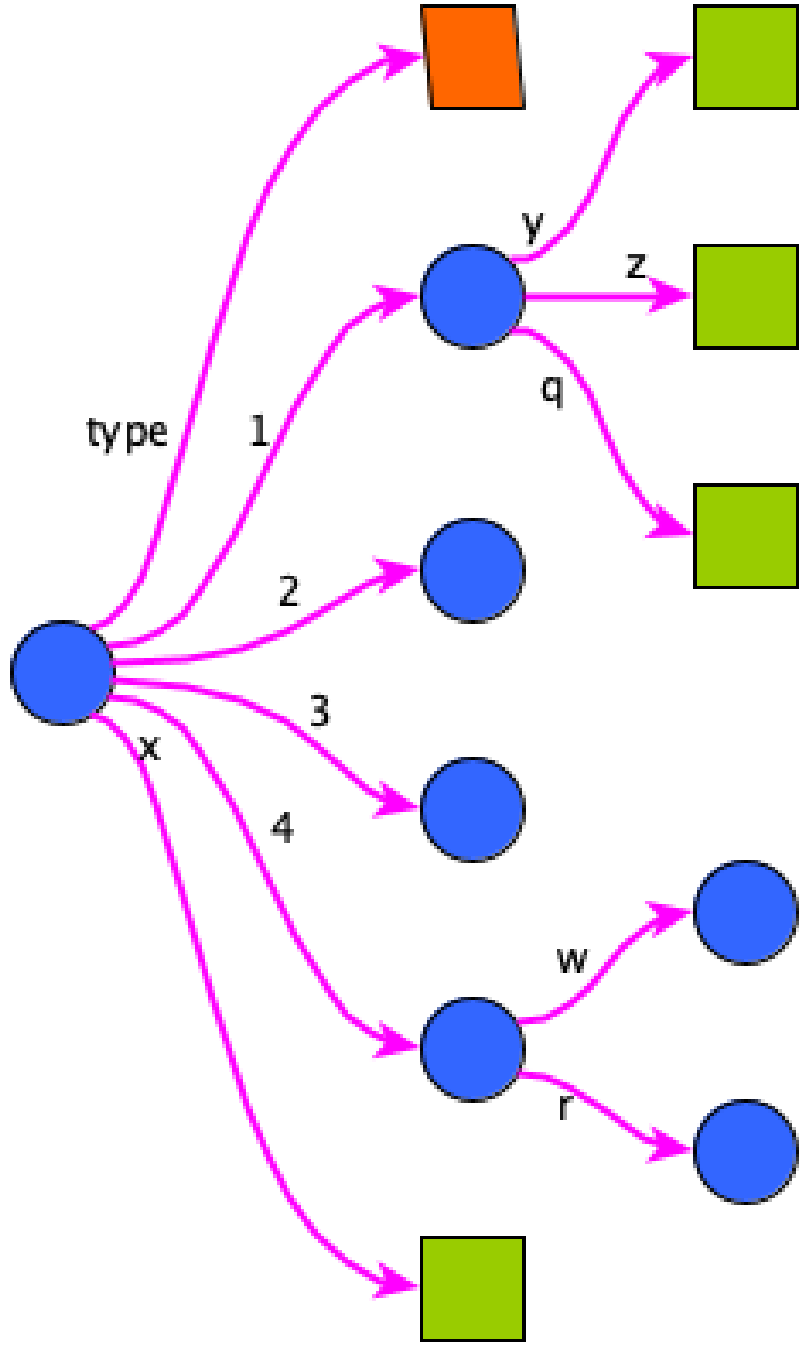
Components: Containers (typed), slots (string / int), values

Intuitive, abstract notions: key-value, sequence, type

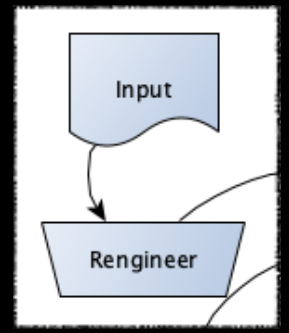
rdf:type	fx:root | xyz:*
xyz:*	“String”, 1, true,…
	xyz:row1, …
rdf:_*N*	“String”, 1, true
	xyz:row_*n*, xyz:…

```
PREFIX fx:  <http://sparql.xyz/facade-x/ns/>
PREFIX xyz: <http://sparql.xyz/facade-x/data/>
```





JSON

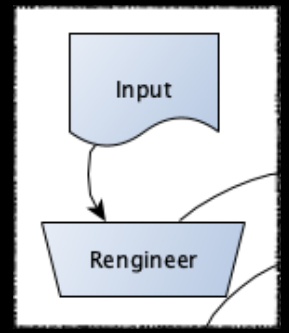


https://github.com/tategallery/collection/artworks/t/023/t02319-9205.json

```
{
  "acno": "T02319",
  "acquisitionYear": 1978,
  "all_artists": "Kazimir Malevich",
  "catalogueGroup": {},
  "classification": "painting",
  "contributorCount": 1,
  "contributors": [
    {
```

```
[ a fx:root ;
  xyz:acno                "T02319" ;
  xyz:acquisitionYear     “1978"^^xsd:int ;
  xyz:all_artists      "Kazimir Malevich" ;
  xyz:catalogueGroup      […]  ;
  xyz:classification      "painting" ;
  xyz:contributorCount    “1”^^xsd:int

…
```

DOM (HTML, XML, …)

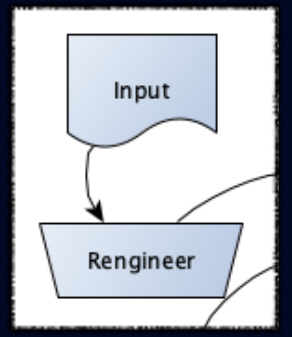



```
▼<div id="az-group">
  ▼<div class="letter-group" id="letter-group-A">
    <h4>A</h4>
    ▼<ul>
      ▶<li class="artist" data-image="https://imma.ie/wp-content/uploads/2018/11/48-676x1024.jpg" data-filter="collection  ">…
      </li>
      ▶<li class="artist" data-image="/wp-content/themes/imma/css/img/no-img-dark.png" data-filter="collection  usa ">…</li>
      ▼<li class="artist" data-image="https://imma.ie/wp-content/uploads/2018/11/845.jpg" data-filter="collection  ">
        ▼<a href="https://imma.ie/artists/marina-abramovic/"> == $0
            "Abramović, Marina "
            <span style> abramovic-marina</span>
          </a>
        </li>
      ▶<li class="artist" data-image="https://imma.ie/wp-content/uploads/2018/11/7493-1024x683.jpg" data-filter="collection  ">
      …</li>
```

```
[ a          fx:root , xhtml:div ;
  xhtml:id "az-group" ;
  rdf:_1     [ a              xhtml:div ;
               rdf:_1         [ a        xhtml:h4 ;
                                rdf:_1   "A" ;
                                <https://html.spec.whatwg.org/#innerHTML>
                                         "A" ;
                                <https://html.spec.whatwg.org/#innerText>
                                         "A"
                              ] ;
```

html.selector=#az-group
@prefix xhtml: <http://www.w3.org/1999/xhtml#> .

CSV




```
id,name,gender,dates,yearOfBirth,yearOfDeath,placeOfBirth,placeOfDeath,url
10093,"Abakanowicz, Magdalena",Female,born 1930,1930,,Polska,,http://www.tate.org.uk/art/artists/magdalena-abakanowicz-10093
…
```

csv.headers=true|false

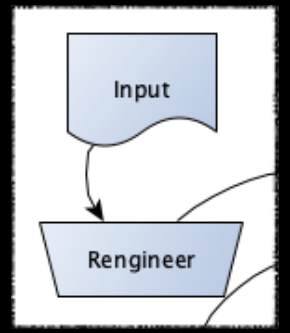
https://github.com/tategallery/collection/blob/master/artist_data.csv

```
[ a          fx:root ;
  rdf:_1     [ xyz:dates          "born 1930" ;
               xyz:gender         "Female" ;
               xyz:id             "10093" ;
               xyz:name           "Abakanowicz, Magdalena" ;
               xyz:placeOfBirth   "Polska" ;
               xyz:placeOfDeath   "" ;
               xyz:url            "http://www.tate.org.uk/art/artists/magdalena-
abakanowicz-10093" ;
               xyz:yearOfBirth    "1930" ;
               xyz:yearOfDeath    ""
             ] ;
```

```
[ a          fx:root ;
  rdf:_1     [ rdf:_1  "id" ;
               rdf:_2  "name" ;
               rdf:_3  "gender" ;
               rdf:_4  "dates" ;
               rdf:_5  "yearOfBirth" ;
               rdf:_6  "yearOfDeath" ;
               rdf:_7  "placeOfBirth" ;
               rdf:_8  "placeOfDeath" ;
               rdf:_9  "url"
             ] ;
```

The Open University

BibTex




```
@article{Knuth1984,
  title={Literate Programming},
  author={Donald E. Knuth},
  journal={The Computer Journal},
  volume={27},
  number={2},
  pages={97--111},
  year={1984},
  publisher={Oxford University Press}
}
```

```
[ a       <http://sparql.xyz/facade-x/ns/root> ;
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#_1>
          [ a       <http://sparql.xyz/facade-x/data/article> ;
            <http://sparql.xyz/facade-x/data/author>
                    "Donald E. Knuth" ;
            <http://sparql.xyz/facade-x/data/journal>
                    "The Computer Journal" ;
            <http://sparql.xyz/facade-x/data/number>
                    "2" ;
            <http://sparql.xyz/facade-x/data/pages>
                    "97--111" ;
            <http://sparql.xyz/facade-x/data/publisher>
                    "Oxford University Press" ;
            <http://sparql.xyz/facade-x/data/title>
                    "Literate Programming" ;
            <http://sparql.xyz/facade-x/data/volume>
                    "27" ;
            <http://sparql.xyz/facade-x/data/year>
                    "1984"
          ]
] .
```

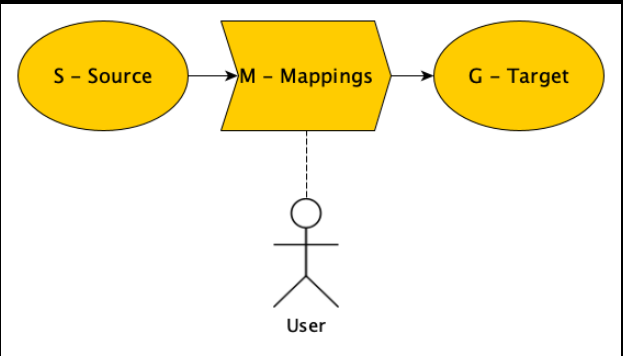
Mappings in SPARQL 1.1 !!!

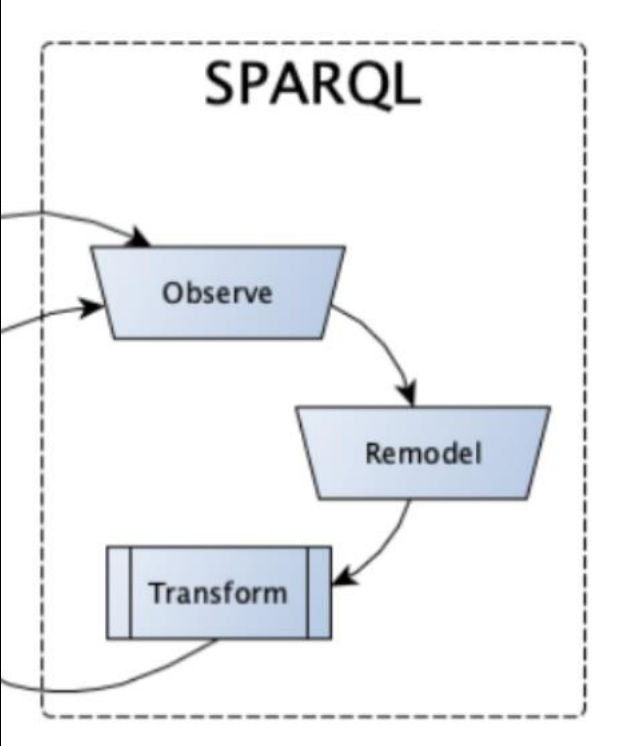
```
PREFIX fx: <http://sparql.xyz/facade-x/ns/>
PREFIX xyz: <http://sparql.xyz/facade-x/data/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX tate: <http://sparql.xyz/example/tate/>
PREFIX schema: <http://schema.org/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

CONSTRUCT {
  ?artist a schema:Person;
    rdfs:label ?name;
    schema:gender ?gender;
    schema:birthDate ?yearOfBirth;
    schema:deathDate ?yearOfBirth;
    schema:birthPlace ?placeOfBirth;
    schema:deathPlace ?placeOfDeath;
    schema:url ?url .
} WHERE {
  # Artists from the Tate Gallery open data!
  SERVICE <x-sparql-anything:csv.headers=true,location=./collection/artist_data.csv> {
    []  xyz:id ?id;
        xyz:name ?name;
        xyz:gender ?gender;
        xyz:yearOfBirth ?yearOfBirth;
        xyz:yearOfDeath ?yearOfDeath;
        xyz:placeOfBirth ?placeOfBirth;
        xyz:placeOfDeath ?placeOfDeath;
        xyz:url ?url
  }
  BIND (IRI(CONCAT(STR(tate:), "artist-", ?id )) AS ?artist) .
}
```

(1) Source schema

(2) Transform ...

(3) Map on target schema





Further with Knowledge Graphs. M. Alam et al. (Eds.)
AKA Verlag and IOS Press, 2021

doi:10.3233/SSW210035

Facade-X: An Opinionated Approach to SPARQL Anything

Enrico Daga[1][0000−0002−3184−5407], Luigi Asprino[2][0000−0003−1907−0677], Paul Mulholland[1][0000−0001−6598−0757], and Aldo Gangemi[3][0000−0001−5568−2684]

[1] The Open University (United Kingdom) {enrico.daga,paul.mulholland}@open.ac.uk
[2] University of Bologna (Italy) luigi.asprino@unibo.it
[3] Consiglio Nazionale delle Ricerche (CNR) aldo.gangemi@cnr.it

Abstract. The Semantic Web research community understood since its beginning how crucial it is to equip practitioners with methods to transform non-RDF resources into RDF. Proposals focus on either engineering content transformations or accessing non-RDF resources with SPARQL. Existing solutions require users to learn specific mapping languages (e.g. RML), to know how to query and manipulate a variety of source formats (e.g. XPATH, JSON-Path), or to combine multiple languages (e.g. SPARQL Generate). In this paper, we explore an alternative solution and contribute a general-purpose meta-model for converting non-RDF resources into RDF: *Facade-X*. Our approach can be implemented by overriding the SERVICE operator and does not require to extend the SPARQL syntax. We compare our approach with the state of art methods RML and SPARQL Generate and show how our solution has lower learning demands and cognitive complexity, and it is cheaper to implement and maintain, while having comparable extensibility and efficiency.

Keywords: SPARQL · Meta-model · Re-engineering

SPARQL.ANYTHING

Search docs

REFERENCE

- Introduction
  - Quickstart
  - Supported Formats
  - Configuration
  - Query templates and variable bindings (CLI only)
  - Functions and magic properties
  - Usage
  - Licence
  - How to cite our work
- Configuration
- Functions

Docs » Reference » Introduction

Edit on GitHub

DOI 10.5281/zenodo.13939383 License Apache 2.0 How to cite Java 11 passing Java 14 passing Java 17 passing

SPARQL Anything

SPARQL Anything is a system for Semantic Web re-engineering that allows users to ... query anything with SPARQL.

Main features:

- Provides a homogenous view over heterogeneous data sources, thanks to the Facade-X meta-model (see Facade-X specification)
- Query files in plain SPARQL 1.1, via the `SERVICE <x-sparql-anything:>` (see configuration) and build knowledge graphs with `CONSTRUCT` queries
- Supported formats: XML, JSON, CSV, HTML, Excel, Text, Binary, EXIF, File System, Zip/Tar, Markdown, YAML, Bibtex, DOCx, PPTX (see pages dedicated to single formats)
- Transforms files, inline content, or the output of an external command
- Generates RDF, RDF-Star, and tabular data (thanks to SPARQL)
- Full-fledged HTTP client to query Web APIs (headers, authentication, all methods supported)
- Functions library for RDF sequences, strings, hashes, easy entity building, ...
- Combine multiple SERVICE clauses into complex data integration queries (thanks to SPARQL)
- Query templates (using BASIL variables)
- Save and reuse SPARQL `Results Sets` as input for parametric queries

https://sparql-anything.cc

Quickstart: https://github.com/SPARQL-Anything/sparql.anything?tab=readme-ov-file#quickstart

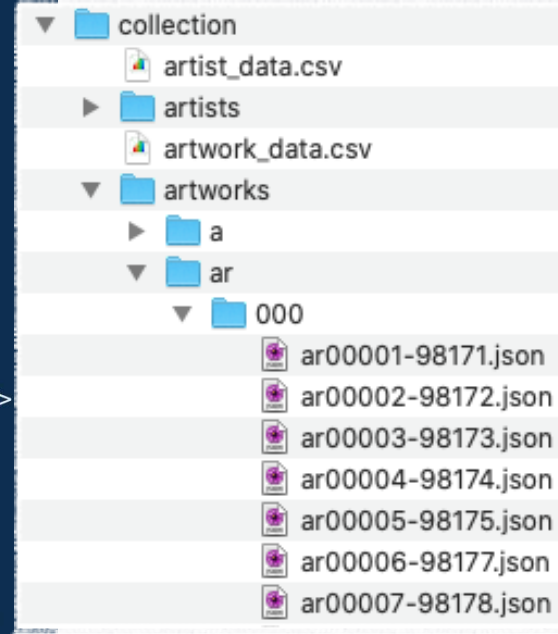
Tate Gallery Open Data

* CSV listing artworks
* JSON with details

Task: build a SKOS taxonomy of artwork subjects

https://github.com/SPARQL-Anything/showcase-tate

```
CONSTRUCT {
    ?scheme a skos:ConceptScheme .
    ?subject a skos:Concept ;
        rdfs:label ?subjectName ;
        skos:inScheme ?scheme ;
        skos:narrower ?child
        .
    ?child skos:broader ?subject .
} WHERE {
  # Browse the list of artworks, get info for locating JSON
  SERVICE <x-sparql-anything:csv.headers=true,location=./collection/artwork_data.csv>
    [] xyz:id ?id ; xyz:accession_number ?accId .
  }
  # Build location of JSON file with subjects, from previous SERVICE
  BIND ( IF ( STRSTARTS( ?accId, "AR" ) ,
    LCASE(CONCAT( "ar/", SUBSTR( ?accId ,3 ,3), "/", ?accId, "-", ?id , ".json" )),
    LCASE(CONCAT( SUBSTR( ?accId ,1 , 1), "/", SUBSTR( ?accId ,2 , 3), "/" , ?accId,
"-", ?id , ".json" ))
  ) AS ?filepath ) .
  # Build location of JSON file with subjects
  BIND (IRI(CONCAT("x-sparql-anything:location=./collection/artworks/", ?filepath ))
       AS ?artworkMetadata ) .
  SERVICE ?artworkMetadata {
      {
          [] xyz:subjects [
            xyz:id ?subjectId ; xyz:name ?subjectName ; # 1
                xyz:children [ ?li [ xyz:id ?childId ] ] ]
      } UNION {
          [] xyz:children [ ?li2 ?s ] .
          ?s xyz:id ?subjectId ; xyz:name ?subjectName .
          OPTIONAL {
            ?s xyz:children [ ?li [ xyz:id ?childId ] ]
          }
      }
  } .
  BIND (fx:entity( tsub:, ?subjectId ) AS ?subject) .
  BIND (fx:entity( tsub:, ?childId   ) AS ?child) .
  BIND (fx:entity( tsub:, "subjects" ) AS ?scheme) .
}
```



Goal

Andrea's goal is to discover and explore sacred music written from the p
unexpected links with his scholarly studies.

Scenario

Andrea is been very busy studying ancient texts and history books and he would li
organs. However, he doesn't actually know what to look for. He is fond of sacred m
him if he could read curiosities about these topics or his favorite composers. And w
links to his scholarly studies without resorting to extensive study and research fron

Competency questions

CQ1: Can I find interesting materials without applying filters?

CQ2: What types of resources can I find?

CQ3: Is there a way of visualizing all the materials connected to my interests?

CQ4: Can I keep getting suggestions in real time?

CQ5: How can I share what I find on the site?

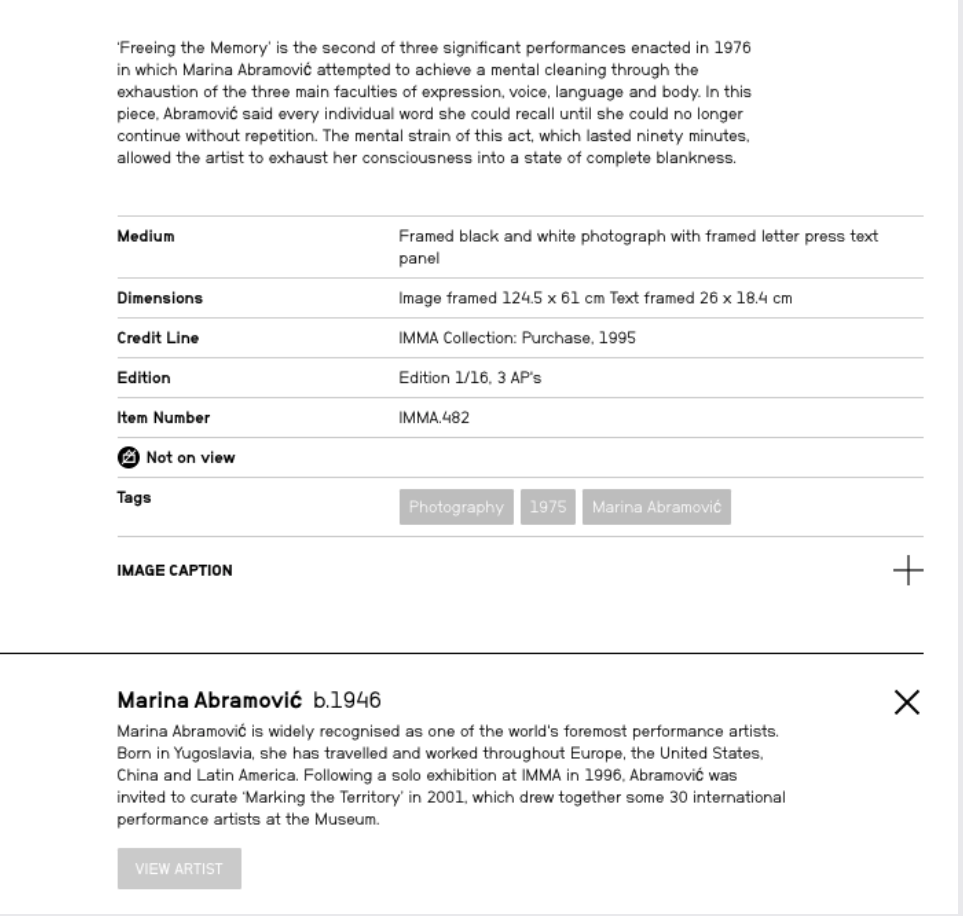
CQ6: How I personalize my navigation experience without knowing the filtering crit

```
SELECT ?item
WHERE {
    SERVICE <x-sparql-anything:> {
         fx:properties fx:location ?_file .
     [] fx:anySlot ?item .
     filter (fx:String.startsWith(?item, "CQ")) .
    }
}
```

Polifonia Ontology Network

* Scenarios collected on GitHub as Markdown files

Task: extract a list of competency questions from any scenario



'Freeing the Memory' is the second of three significant performances enacted in 1976 in which Marina Abramović attempted to achieve a mental cleaning through the exhaustion of the three main faculties of expression, voice, language and body. In this piece, Abramović said every individual word she could recall until she could no longer continue without repetition. The mental strain of this act, which lasted ninety minutes, allowed the artist to exhaust her consciousness into a state of complete blankness.

Medium	Framed black and white photograph with framed letter press text panel
Dimensions	Image framed 124.5 x 61 cm Text framed 26 x 18.4 cm
Credit Line	IMMA Collection: Purchase, 1995
Edition	Edition 1/16, 3 AP's
Item Number	IMMA.482
Not on view	
Tags	Photography 1975 Marina Abramović

IMAGE CAPTION +

Marina Abramović b.1946

Marina Abramović is widely recognised as one of the world's foremost performance artists. Born in Yugoslavia, she has travelled and worked throughout Europe, the United States, China and Latin America. Following a solo exhibition at IMMA in 1996, Abramović was invited to curate 'Marking the Territory' in 2001, which drew together some 30 international performance artists at the Museum.

VIEW ARTIST

https://imma.ie/collection/freeing-the-memory/

https://github.com/SPARQL-Anything/showcase-imma

Building a knowledge graph of artists and artworks scraping the IMMA museum website

In what follows, `fx` refers to the following command

```
java -jar sparql-anything-0.3.2-SNAPSHOT.jar
```

Process

Extract the list of artists from the Web page and build an XML result set with ?artistNickname and ?artistUrl.

```
fx -q imma-artists.sparql -o imma-artists.xml -f xml
```

Extract data from the artists' Web page and build one JSON-LD file each (create folder 'artists' first).

```
fx -q imma-artist.sparql -i imma-artists.xml -p "artists/?artistNickname.jsonld" -f json
```

Extract the list of artworks' Web pages from the JSON-LD files of the artists.

```
fx -q imma-artworks.sparql -l artists/ -o imma-artworks.xml -f xml
```

Extract data from the artworks' Web pages and build one JSON-LD file each (create folder 'artworks' first).

```
fx -q imma-artwork.sparql -i imma-artworks.xml -p "artworks/?artworkNickname.jsonld" -f json
```

Load into your favourite triple store.

MusicXML

```
</part-list>
<!--========================== Part 1 ==========================-->
<part id="P1"> rdf:_8
  <!--========================== Measure 0 ==========================-->
  <measure number="0" width="152.89">
    <print>
      <system-layout>
        <system-margins>
          <left-margin>67.9</left-margin>
          <right-margin>0</right-margin>
        </system-margins>
        <top-system-distance>199</top-system-distance>
      </system-layout>
    </print>
    <attributes>
      <divisions>10080</divisions>
      <key>
        <fifths>1</fifths>
        <mode>minor</mode>
      </key>
      <time symbol="common">
        <beats>4</beats>
        <beat-type>4</beat-type>
      </time>
      <clef>
        <sign>G</sign>
        <line>2</line>
      </clef>
```

Exploring Music Computing tasks:

- Melody extraction
- N-grams Extraction
- N-grams Analysis
- Music Note Ontology Population

https://github.com/SPARQL-Anything/showcase-musicxml

Ratta, Marco, and Enrico Daga. "Knowledge Graph Construction From MusicXML: An Empirical Investigation With SPARQL Anything."

System

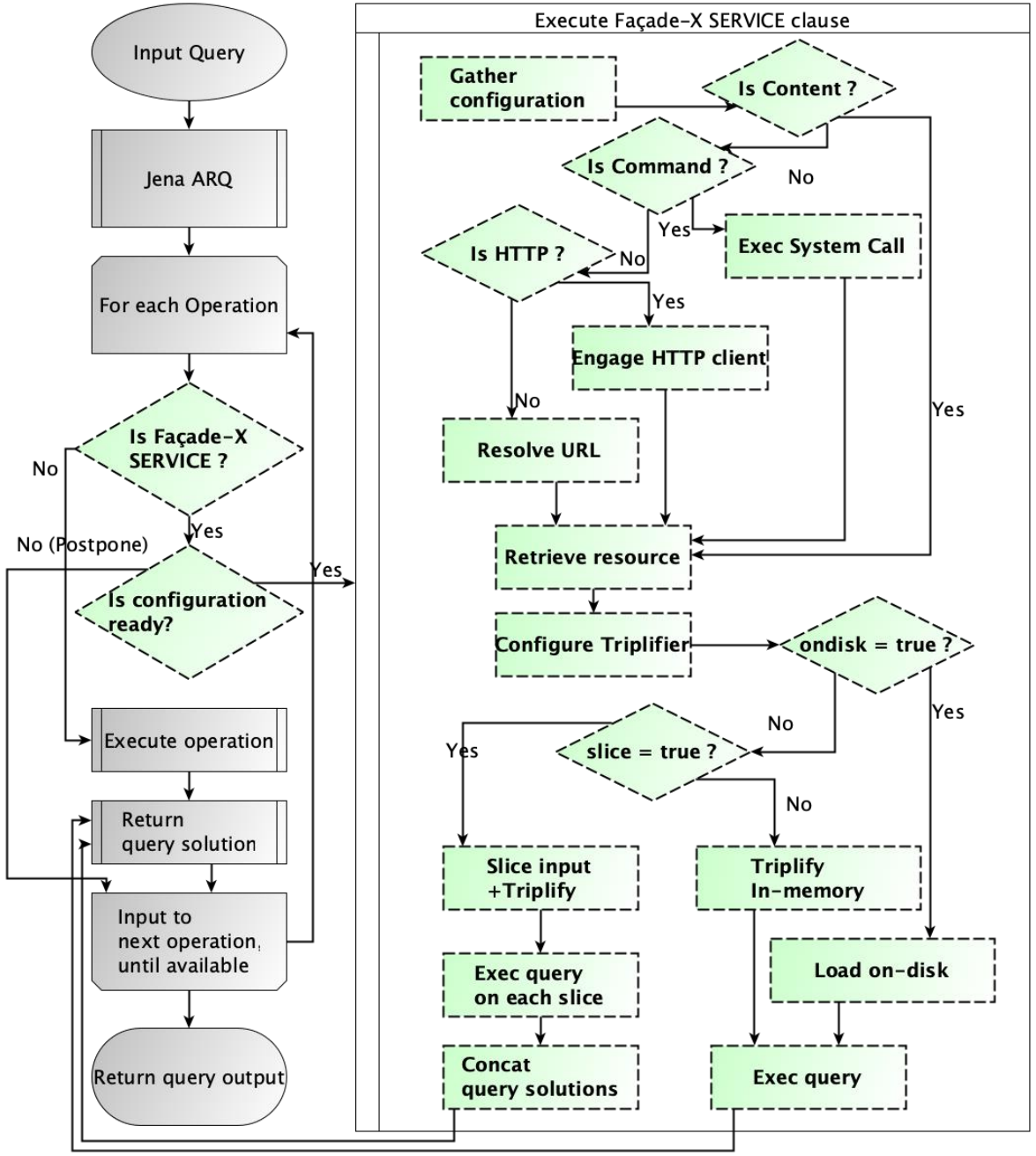
SPARQL Anything is implemented on top of a **standard SPARQL 1.1 query engine** (Apache Jena ARQ)

Query execution is always after **view materialisation**

In-memory / On-disk options available

Performance can be improved with **optimisations**: e.g. avoid materialising triples not needed by the query (**triple-filtering**)

Method can scale further by ***Slicing*** the data, instead of materialising a *Complete* view



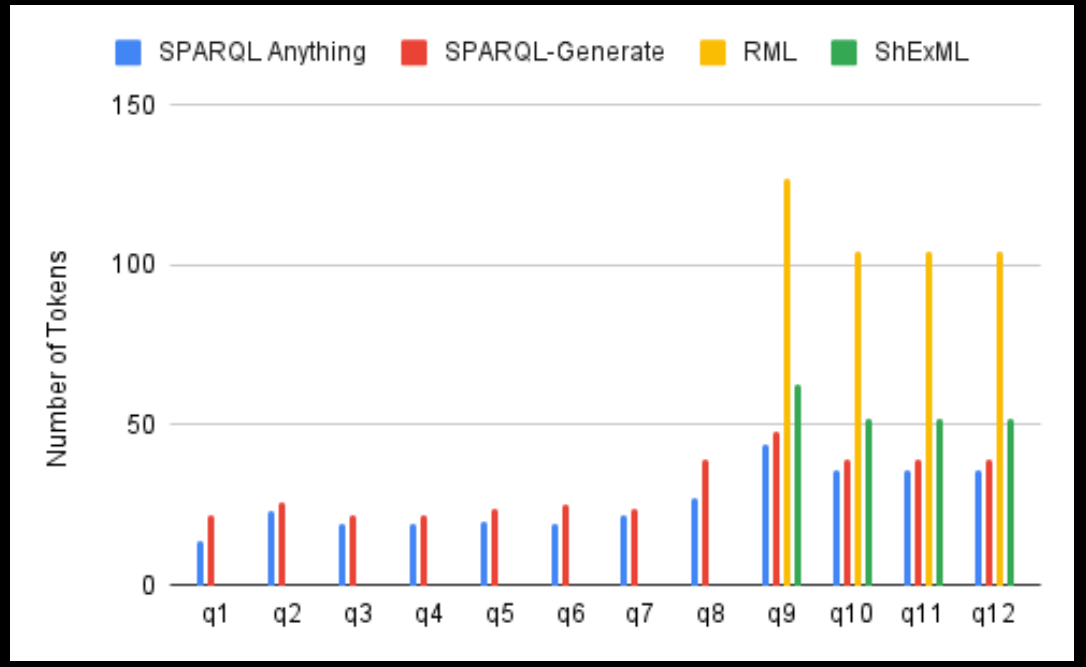
Comparisons

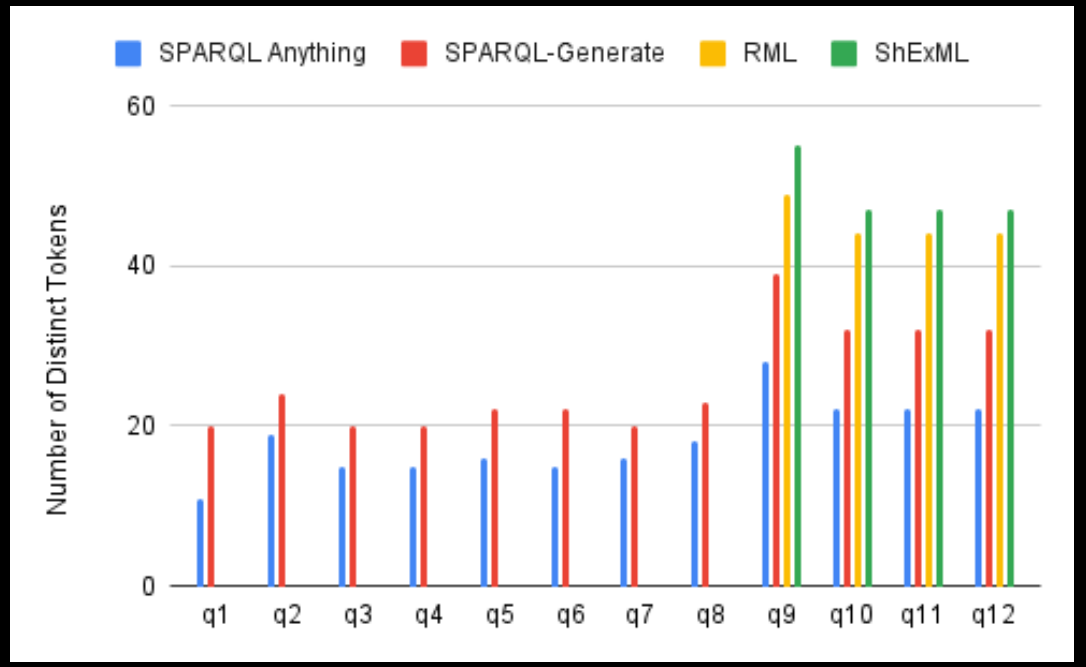
Lower (cognitive) complexity

- One measure of complexity is the number of (distinct) items or variables (Halford et al. 2004; Warren et al. 2015).

- vs SPARQL Generate: 8 queries from a museum collection use case
  - What are the titles of the artworks attributed to “ANONIMO”?
  - What are the titles of the artworks created in the 1935?
  - …
- vs RML / SPARQL Generate / ShexML
  - 4 mappings

- Avg distinct tokens:
  - SPARQL Anything: ~18
  - SPARQL Generate: ~25 (~39.72% more)
  - RML: ~45 (~150% more)

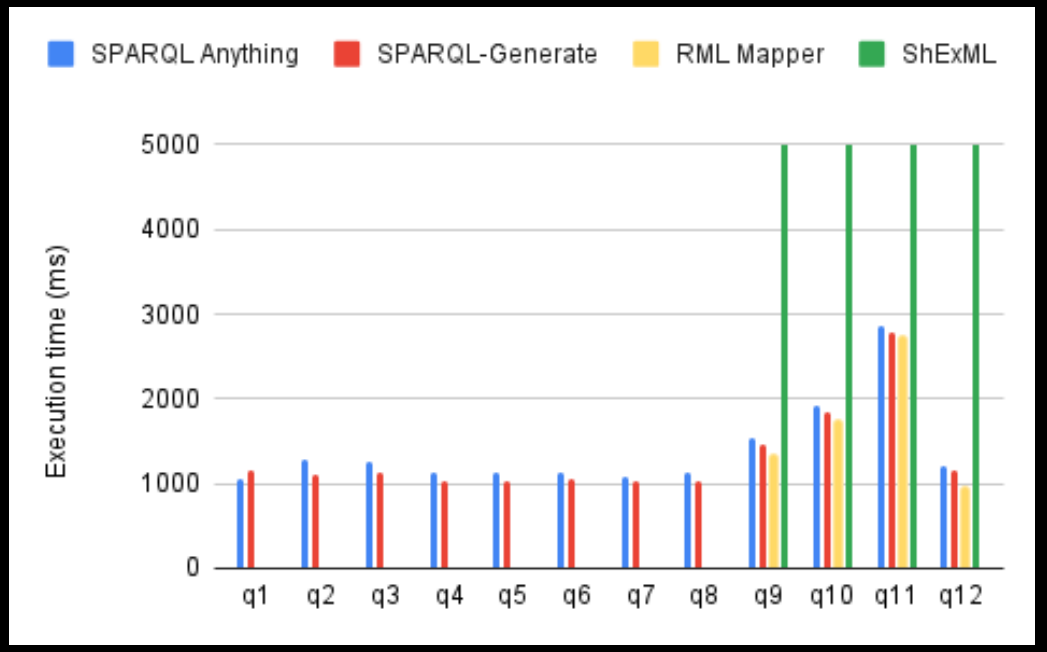


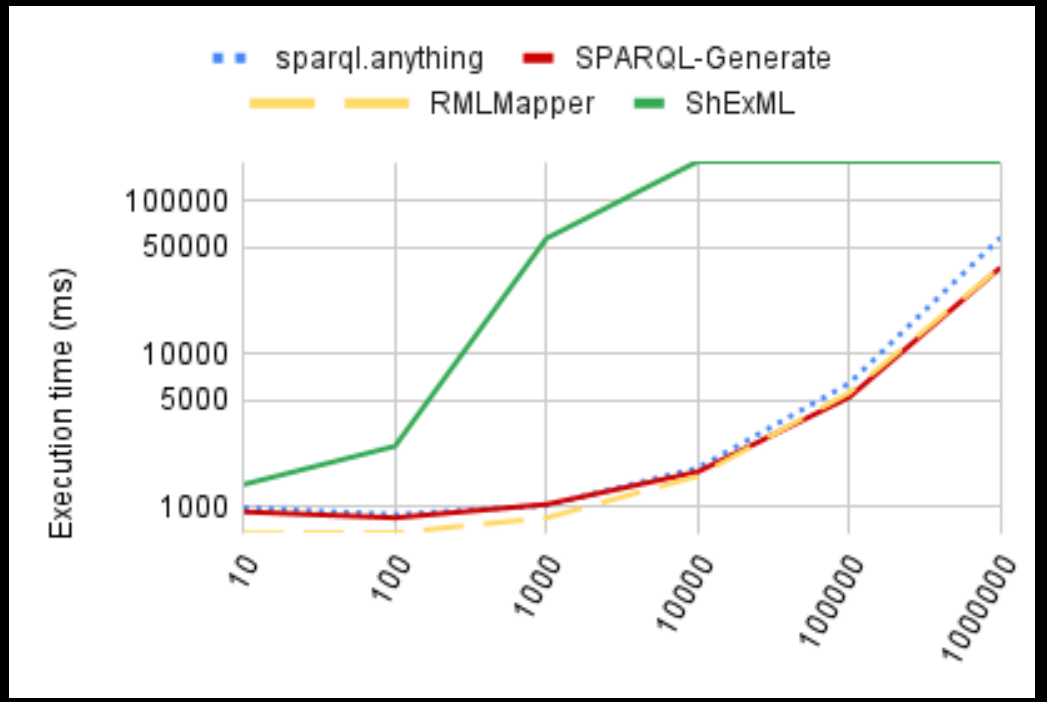


Comparisons

Practicable and sustainable

- Quantitative analysis of performance, to assess practicability
- In-Memory implementation (Naive)
- Execution time of q1-q12 (AVG on 10 executions)
- Comparable to RML Mapper and SPARQL Generate on files up to 1M JSON objects (~5M triples)
- In-Memory implementation **scales linearly**
- **The approach is sustainable**
- Lines of Java code to **maintain**: SPARQL Generate 12280 (core module); RML Mapper 7951; SPARQL Anything: 3842 (all transformers) — v0.2.0 (v0.7.0 has ~12k)





Evaluation

Is it truly generic?

Theoretical 1/2: any format expressible with a BNF grammar can be also represented as FX

Theoretical 2/2: FX can also represent the Entity-Relation model

Knowledge Graph Construction with a *façade*: a unified method to access heterogeneous data sources on the Web

LUIGI ASPRINO*, University of Bologna, Italy
ENRICO DAGA, The Open University, UK
ALDO GANGEMI, Consiglio Nazionale delle Ricerche (CNR) and University of Bologna, Italy
PAUL MULHOLLAND, The Open University, UK

Data integration is the dominant use case for RDF Knowledge Graphs. However, Web resources come in formats with weak semantics (for example CSV and JSON), or formats specific to a given application (for example BibTex, HTML, and Markdown). To solve this problem, Knowledge Graph Construction (KGC) is gaining momentum due to its focus on supporting users in transforming data into RDF. However, using existing KGC frameworks result in complex data processing pipelines, which mix structural and semantic mappings, whose development and maintenance constitute a significant bottleneck for KG engineers. Such frameworks force users to rely on different tools, sometimes based on heterogeneous languages, for inspecting sources, designing mappings, and generating triples, thus making the process unnecessarily complicated. We argue that it is possible and desirable to equip KG engineers with the ability of interacting with Web data formats by relying on their expertise in RDF and the well-established SPARQL query language [2].

In this article, we study a unified method for data access to heterogeneous data sources with Facade-X, a meta-model implemented in a new data integration system called SPARQL Anything. We demonstrate that our approach is theoretically sound, since it allows a single meta-model, based on RDF, to represent data from *(a)* any file format expressible in BNF syntax, as well as *(b)* any relational database. We compare our method to state-of-the-art approaches in terms of usability (cognitive complexity of the mappings) and general performance. Finally, we discuss the benefits and challenges of this novel approach by engaging with the reference user community.

CCS Concepts: • **Information systems** → **Resource Description Framework (RDF)**; **Information integration**.

Additional Key Words and Phrases: SPARQL, RDF, Meta-model, Re-engineering

Asprino, Luigi, Enrico Daga, Aldo Gangemi, and Paul Mulholland. "Knowledge graph construction with a façade: a unified method to access heterogeneous data sources on the web." ACM Transactions on Internet Technology 23, no. 1 (2023): 1-31.

Benefits

- Transform / Query resources having heterogeneous formats
- Low learning demands for KG practitioners — plain SPARQL 1.1
- A single+consistent abstraction for potentially **any data format**
- “Free lunch” data exploration and querying (no cold start problem)

- **Open-ended extendibility:** no changes to user-facing code required
  - FX can express ANY format representable as BNF (as well as relational data)

- Generate RDF (and RDF-Star) but also Tabular Data
  - Sometimes the KG is the intermediate object. Applications requiring other formats — e.g. tabular data is the usual format for data science, KG can be a feature engineering *medium*

The Open University

Challenges / Directions

Execute FX queries
• FX **view materialisation** only method supported so far.
• Study other execution strategies, e.g. query-rewriting, learn from Ontology Based Data Access (OBDA)

Support developers: how to streamline developing adapters to new formats?

Features:
• More inputs: relational DB, Mongodb, Image Annotations, …
• More connectors: Apache Parquet, Thrift, Hadoop/HDFS, RDBMS/JDBC, MongoDB, …
• Anything in … anything out! Equip SPARQL Anything with a full-fledged template engine

Support users with automating KGC
• PhD “AI Supported Knowledge Graph Design & Generation”

Analysis of KGC reporting in papers

- No two concrete pipelines found to be the same
- 65% of KG design processes found to involve custom code (Java or Python) for constructing the graphs
- 35% involved a data preparation + tool (reusable) + post-processing process.

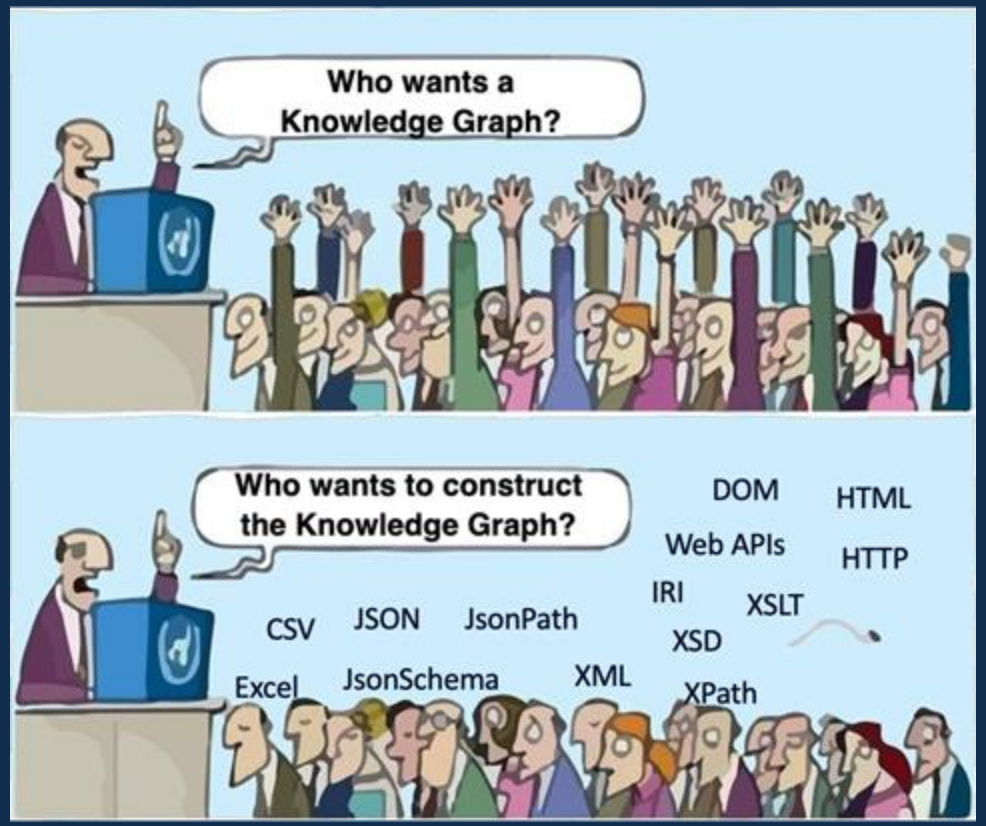
Table 1: KG building tasks

Task ID: Name	Task Description
T1: URI design	Constructing an expression pattern for a URI.
T2: URI mapping	Mapping an element's type to its specific URI pattern.
T3: Prefix mapping	Mapping a URI to its target prefix.
T4: Type mapping	Mapping a data type of the sources to the corresponding target type of the ontology.
T5: Entity linking	Mapping an entity/object in the source to the corresponding entity in an existing taxonomy/graph.
T6: Entity resolution	Establishing whether multiple objects refer to the same object.
T7: Domain ontology selection	Selecting ontologies that are relevant to the domain.

- Abstracted 55 different tasks & sub-tasks
  - Design tasks: URI, custom ontology terms + composition, linking, requirements
  - Structural tasks: RDF metatype mapping, explicitation, entity resolution
  - Understanding tasks: ontology coverage assessment, ontology understanding & suitability
  - Mapping tasks: Type mapping, URI mapping, prefix mapping
  - Quality tasks: versioning, provenance tracking, output validation

The Open University

PySPARQL-Anything Showcase



Did you know you can query (almost) anything with plain SPARQL?

Did you know you can do that now within your **Python** code?

SPARQL ANYTHING

https://sparql-anything.cc

Speak to us: we are building a community!

The Open University

- Domani siamo al terzo piano…

- Daga, E., Asprino, L., Mulholland, P., Gangemi, A.: Facade-x: an opinionated approach to sparql anything. In: SEMANTiCS 2021: 17th International Conference on Semantic Systems (2021)
- Atkin, M., Deely, T., Scharffe, F.: Knowledge Graph Benchmarking Report 2021 (version 2.0). Zenodo, http://doi.org/10.5281/zenodo.4950097 (June 2021)
- Lassila, O., Michael Schmidt, Brad Bebee, Dave Bechberger, Willem Broekema, Ankesh Khandelwal, Kelvin Lawrence, Ronak Sharda, and Bryan Thompson: Graph? Yes! Which one? Help!. 1st Squaring the circle on knowledge graphs workshop - Semantics (2021)
- Daga, E., Meroño-Peñuela, A., Motta, E.: Sequential linked data: the state of affairs. Semantic Web (2021)
- Warren, P., Mulholland, P.: Using sparql–the practitioners' viewpoint. In: European Knowledge Acquisition Workshop. pp. 485–500. Springer (2018)
- Corcho, O., Priyatna, F., Chaves-Fraga, D.: Towards a new generation of ontology based data access. Semantic Web 11(1), 153–160 (2020)
- Michel, F., Faron-Zucker, C., Corby, O., Gandon, F.: Enabling automatic discovery and querying of web apis at web scale using linked data standards. In: Companion Proceedings of The 2019 World Wide Web Conference. pp. 883–892 (2019)
- Dimou, A., Vander Sande, M., Colpaert, P., Verborgh, R., Mannens, E., Van de Walle, R.: Rml: a generic language for integrated rdf mappings of heterogeneous data. In: 7th Workshop on Linked Data on the Web (2014)
- García-González, H., Boneva, I., Staworko, S., Labra-Gayo, J.E., Lovelle, J.M.C.: Shexml: improving the usability of heterogeneous data mapping languages for firsttime users. PeerJ Computer Science 6, e318 (2020)
- Paulheim, Heiko. "Knowledge graph refinement: A survey of approaches and evaluation methods." Semantic web 8, no. 3 (2017): 489-508.
- Ko, A.J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., Scaffidi, C., Lawrance, J., Lieberman, H., Myers, B., et al.: The state of the art in enduser software engineering. ACM Computing Surveys (CSUR) 43(3), 1–44 (2011)
- Lefrançois, M., Zimmermann, A., Bakerally, N.: A sparql extension for generating rdf from heterogeneous formats. In: European Semantic Web Conference. pp. 35– 50. Springer (2017)
- Lieberman, H., Paternò, F., Klann, M., Wulf, V.: End-user development: An emerging paradigm. In: End user development, pp. 1–8. Springer (2006)
- Cyganiak, Richard. Tarql (sparql for tables): Turn csv into rdf using sparql syntax. Technical Report, 2015. http://tarql. github. io, 2015.
- Lenzerini, Maurizio. "Data integration: A theoretical perspective." In Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp. 233-246. 2002.