VKG 2019

Generating and querying (Virtual) Knowledge Graphs from heterogeneous data sources

David Chaves Fraga, Ahmad Alobaid, Freddy Priyatna,
Andrea Cimmino, Oscar Corcho

Ontology Engineering Group
Universidad Politécnica de Madrid

dchaves@fi.upm.es
@oeg-upm

02/06/2019
ESWC2019 - Potoroz

VKG 2019

Introduction

David Chaves Fraga, Ahmad Alobaid, **Freddy Priyatna,**
Andrea Cimmino, Oscar Corcho

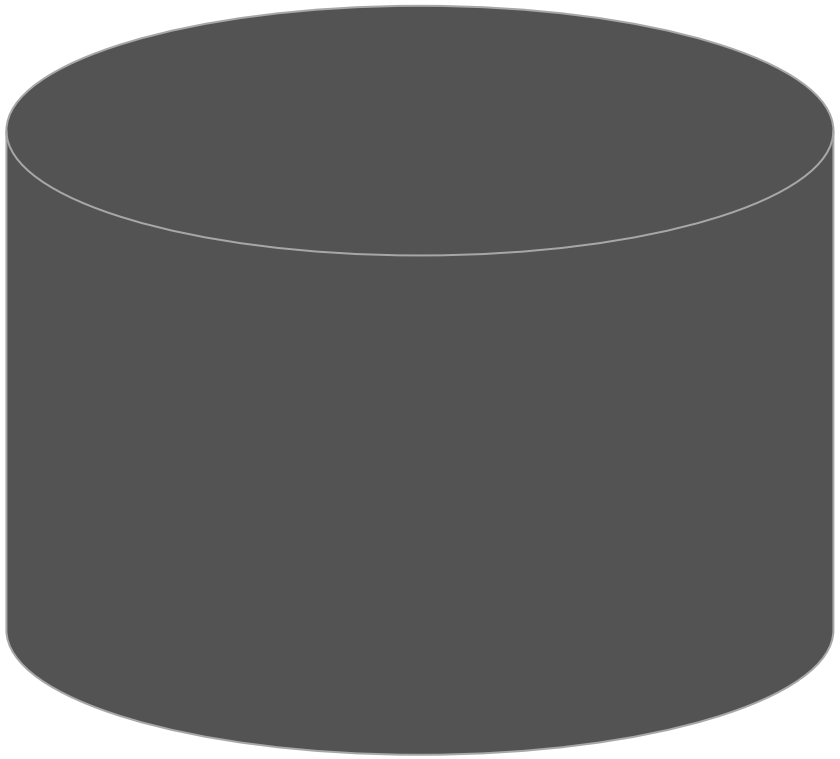
Ontology Engineering Group
Universidad Politécnica de Madrid

dchaves@fi.upm.es
@oeg-upm

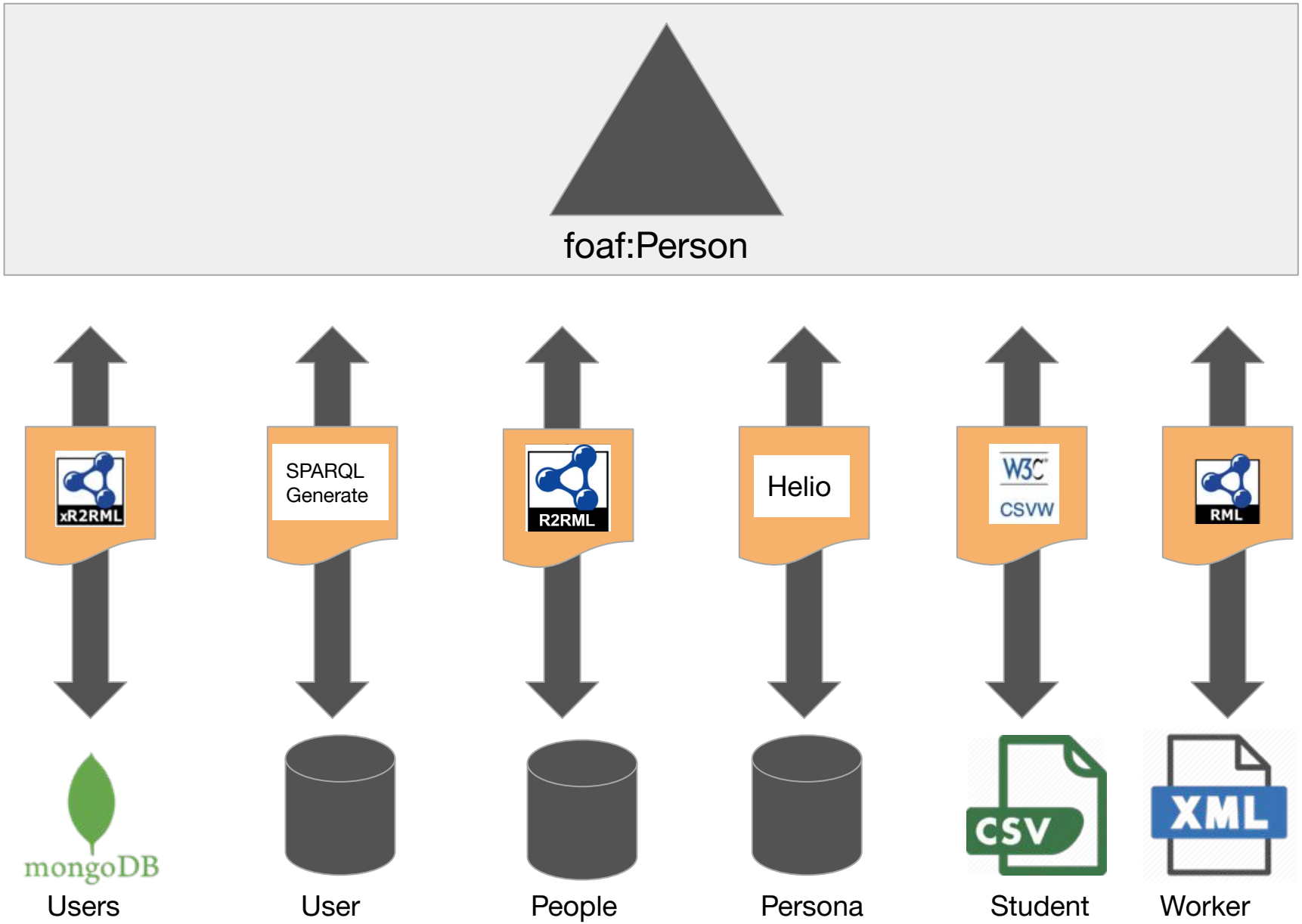
02/06/2019
ESWC2019 - Potoroz

Pre-requirements

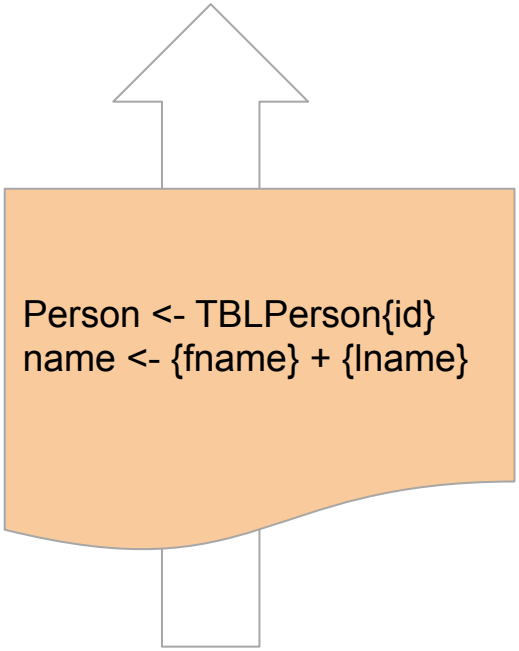
1. *git clone https://github.com/oeg-upm/vkg-tutorial-eswc2019.git*
2. *Install java, docker and docker-compose*
   a. *https://docs.docker.com/install/*
   b. *https://docs.docker.com/compose/install/*



Ontology Based Data Access



Person/1 a Person .
Person/1 name “David Chaves” .



id	fname	lname
1	David	Chaves

TBLPerson

Query Translation

Person/1 a Person .
Person/1 name “David Chaves” .

```
SELECT * WHERE {
   ?x a Person .
   ?x name ?y .
}
```

Person <- TBLPerson{id}
name <- {fname} + {lname}

id	fname	lname
1	David	Chaves

TBLPerson

```
SELECT id AS x, CONCAT(fname, lname) AS y
FROM TBLPerson
```

Outline

4 tools (explanation + hands on work):

1. morph-CSV
2. morph-GraphQL
3. TADA
4. Helio

Morph

David Chaves-Fraga, Freddy Priyatna

Ontology Engineering Group
Universidad Politécnica de Madrid

dchaves@fi.upm.es
@oeg-upm

02/06/2019
ESWC2019 - Potoroz

Suite of open source OBDA tools:

- **Morph-RDB (2015):** Query translation and optimizations using R2RML mappings to query RDBs
- **Morph-CSV (2018-2019):** Query translation using RML+FnO mappings and CSVW annotations to query real CSVs
- **Morph-GraphQL (2019):** Generation of GraphQL servers from R2RML to query RDBs

http://morph.oeg-upm.net

Morph-CSV

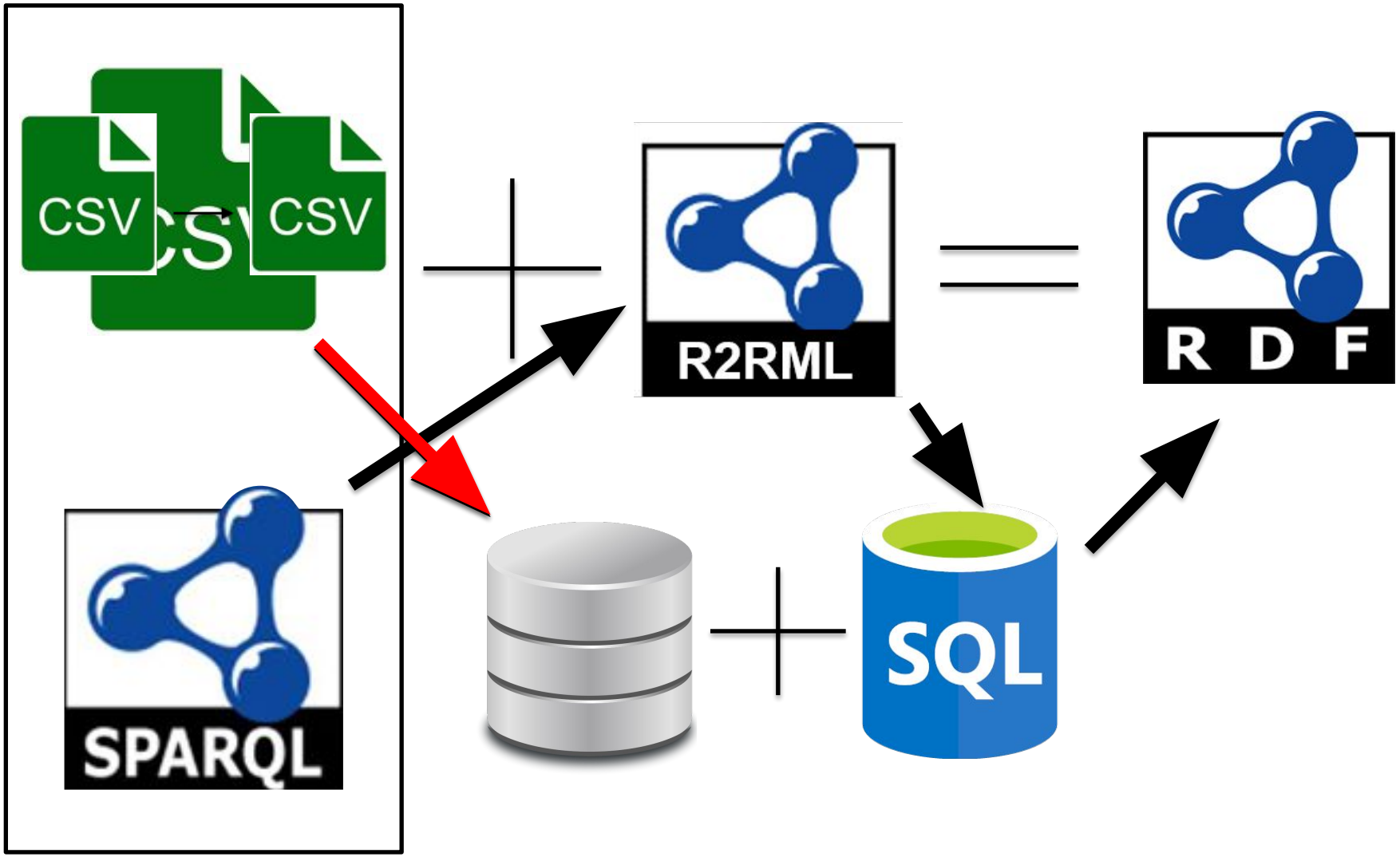
David Chaves-Fraga

Ontology Engineering Group
Universidad Politécnica de Madrid

dchaves@fi.upm.es
@dchavesf

02/06/2019
ESWC2019 - Potoroz

1. *cd vkg-tutorial-eswc2019/morph-csv*
2. *docker-compose up -d*


CSV
CSV
CSV
R2RML
R D F
SPARQL
SQL

Motivating Example and Problems

comments.csv

```
date, username, comment, modifiedDates,nOfLikes
"20181001","fpriyatna","Hallo Dunia","20181001-20181101",1
"20181002","dchaves","Hola Mundo","20181002-20181204",8
"20181130","fpriyatna","Hello World","20181130",10
"20181128","dchaves","Hello World","20181128-20190101",50
```

people.csv

```
"Freddy","Priyatna","","Indonesia","Republic of Indonesia",
"David","Chaves","Fraga","Spain","Kingdom of Spain",3
"Ahmad","Alobaid","", "Kuwait","State of Kuwait",
"Oscar","Corcho","Garcia","Spain","Kingdom of Spain",7
```

- No schema
- Not normalized
- Not uniform
- Missing data
- Missing meta-data
- No explicit joins

Can we reuse existing optimizations in SPARQL-to-SQL translation and R2RML mappings for querying heterogeneous CSVs?

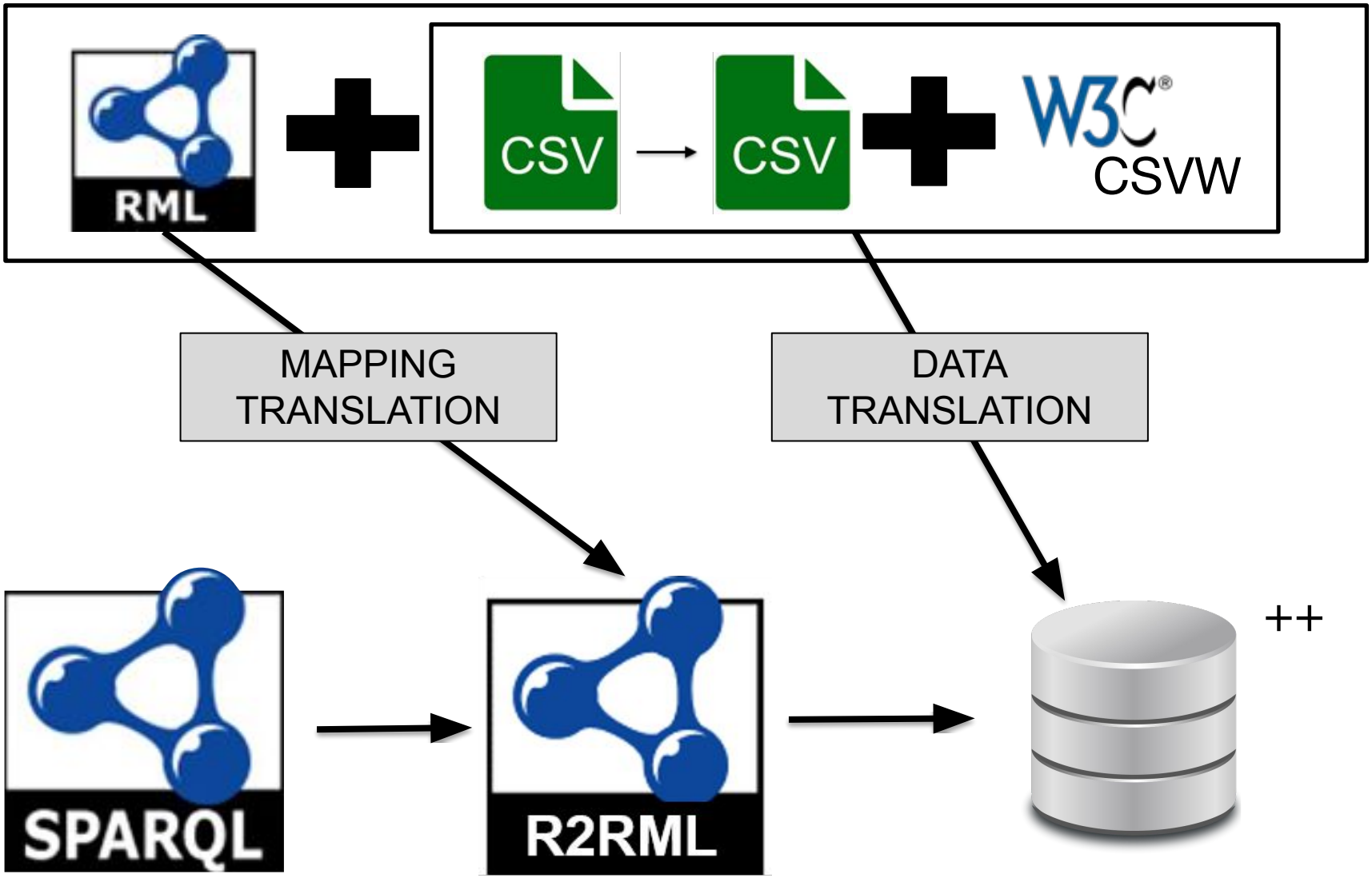
CSV-oriented mappings and annotations

- RML: RDF Mapping Language for heterogeneous data sources
  - Formats like XML, JSON, RDB, CSV
  - Purpose: Transformation to RDF

- CSVW: Metadata for CSVs (used by Google)
  - W3C recommendation
  - Purpose: Describe the content

- The Function Ontology: Defining transformation functions (e.g. fromCelsiusToFarenheit)
  - Integrated with RML

Enabling OBDA query-translation over CSV



For example… Data Translation

CSVW

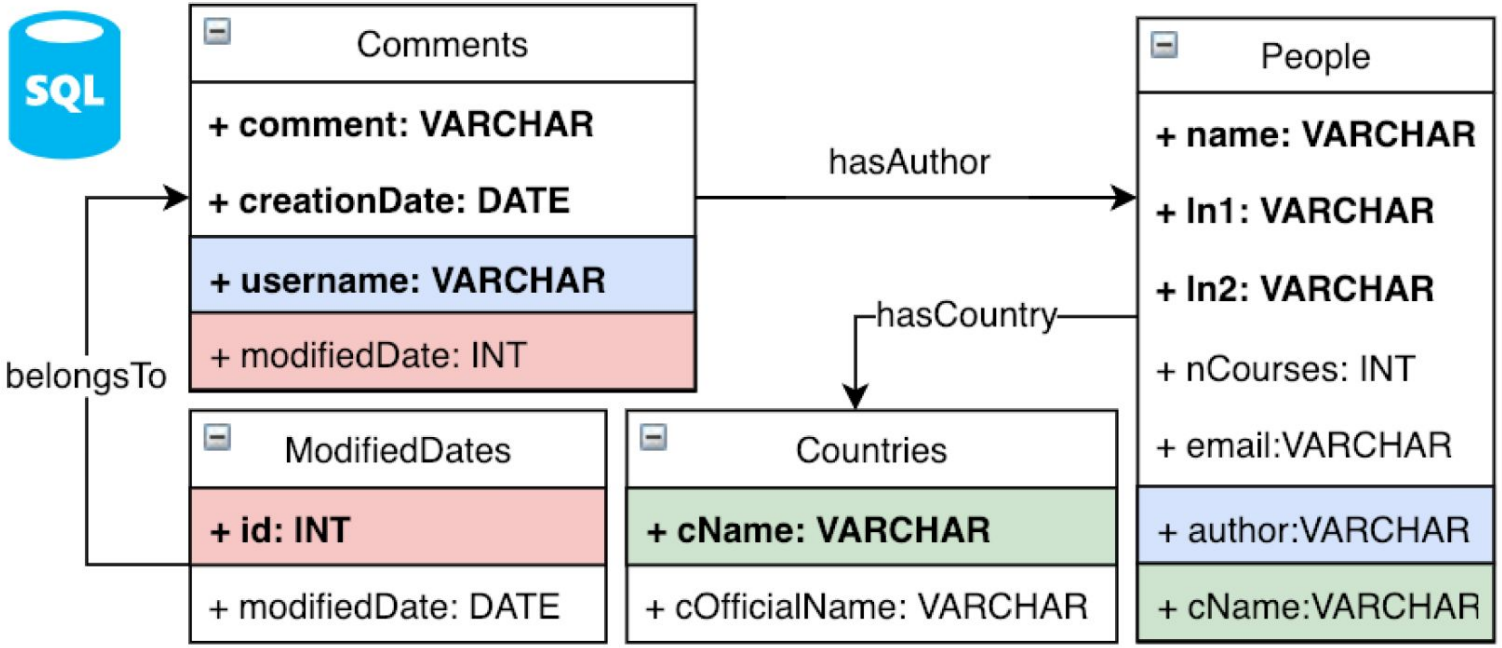
RML

comments.csv

```
date, username, comment, modifiedDates,nOfLikes
"20181001","fpriyatna","Hallo Dunia","20181001-20181101",1
"20181002","dchaves","Hola Mundo","20181002-20181204",8
"20181130","fpriyatna","Hello World","20181130",10
"20181128","dchaves","Hello World","20181128-20190101",50
```

people.csv

```
"Freddy","Priyatna","","Indonesia","Republic of Indonesia",
"David","Chaves","Fraga","Spain","Kingdom of Spain",3
"Ahmad","Alobaid","", "Kuwait","State of Kuwait",
"Oscar","Corcho","Garcia","Spain","Kingdom of Spain",7
```

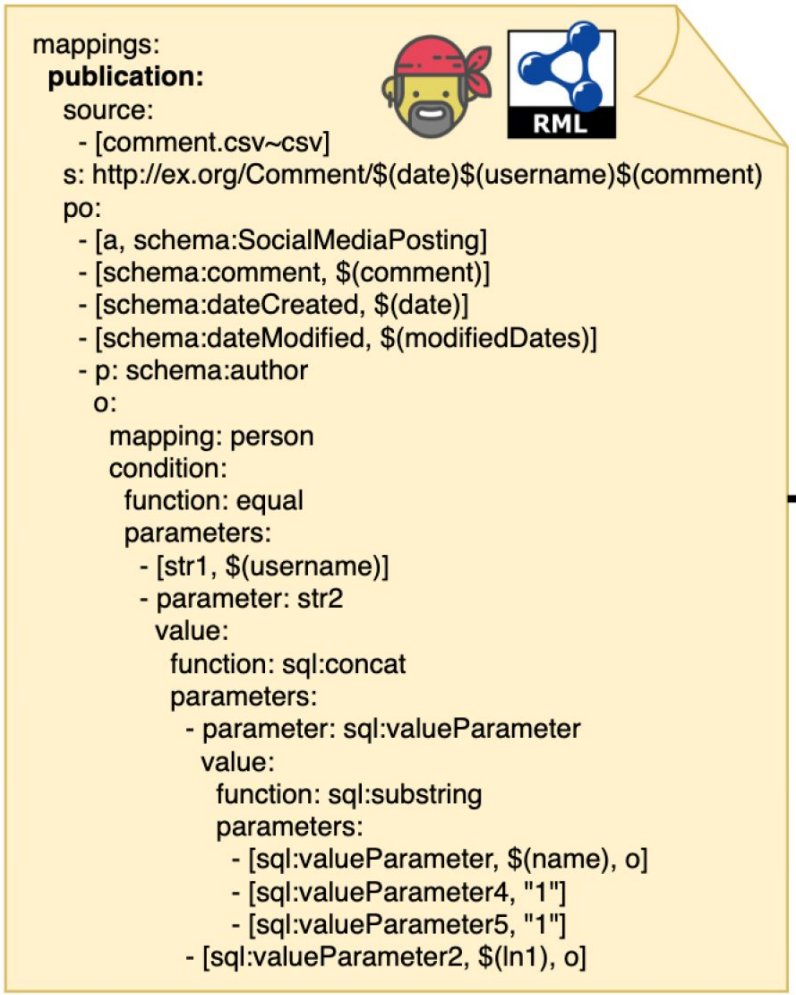


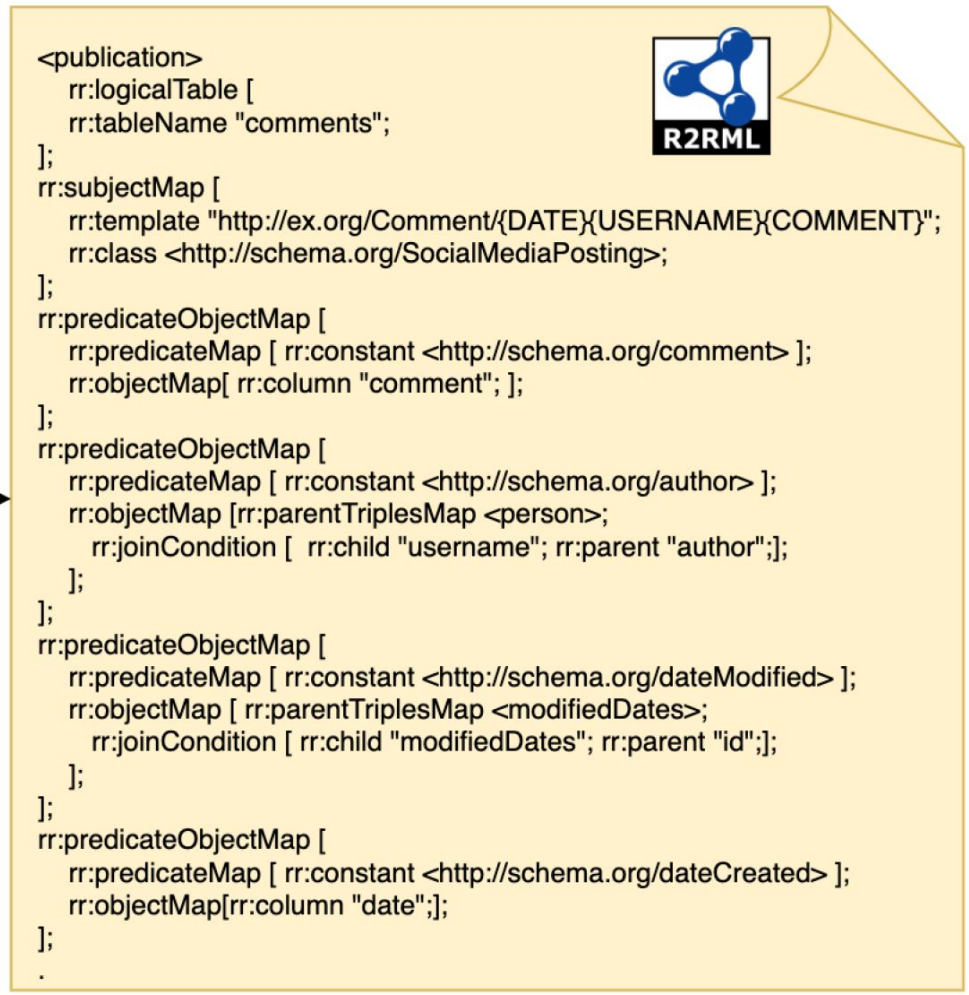
For example… CSVW spec

```
{
"@context": ["http://www.w3.org/ns/csvw"],
"url": "comments.csv",
"tableSchema": {
  "columns": [{
     "titles": "date",
     "datatype": {
        "base": "date",
        "format": "yyyyMMdd"
     }
  },{
     "titles": "modifyDate",
     "sepatator": "-",
     "datatype": {
        "base": "date",
        "format": "yyyyMMdd"
     }
  }]
}}
```

```
{
"@context": ["http://www.w3.org/ns/csvw"],
"url": "people.csv",
"tableSchema": {
      "rowTitles":
          ["name","ln1","ln2","cName,"cOfficialName","nCourses"],
      "columns":[{
          "titles":"ln2",
          "null": ""
          }, {
          "titles":"nCourses",
          "default": 0
      }]
}
```

For example… Mapping Translation





Time to play...

Three examples (configs folder):

- Simple: persons and comments
- Linking Open City Data
- Bio2RDF (without real RDF ;-))

How to run it?

cd run-scripts

./runTheScript.sh

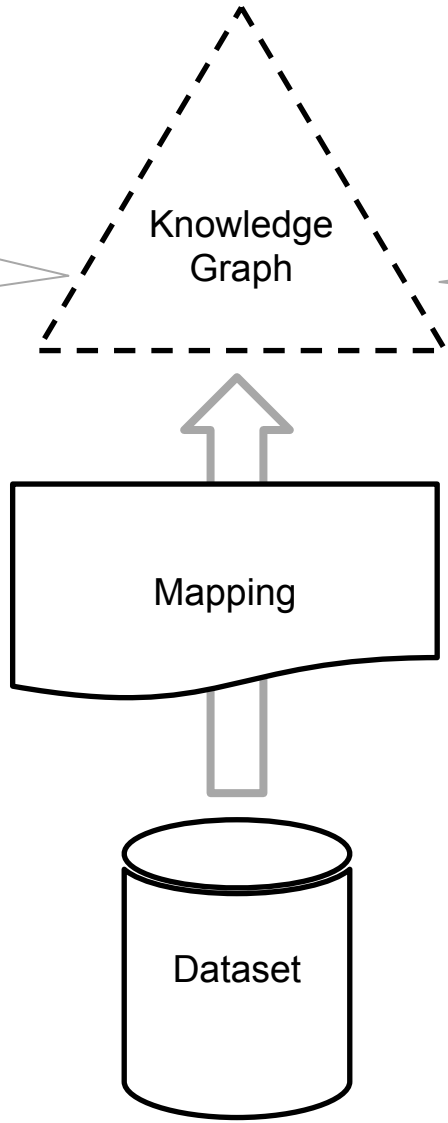
In continuation...

morph-CSV

```
SELECT ?identifier ?name
WHERE {
  ?identifier a Person
  ?identifier foaf:name ?name .
}
```

morph-GraphQL

```
query {
  listPerson {
    identifier
    name
  }
}
```



Morph-GraphQL

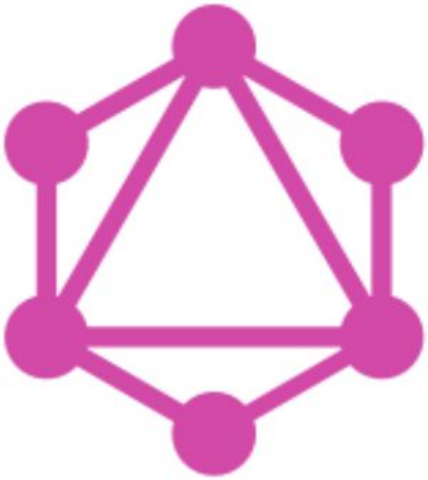
Freddy Priyatna

Ontology Engineering Group
Universidad Politécnica de Madrid

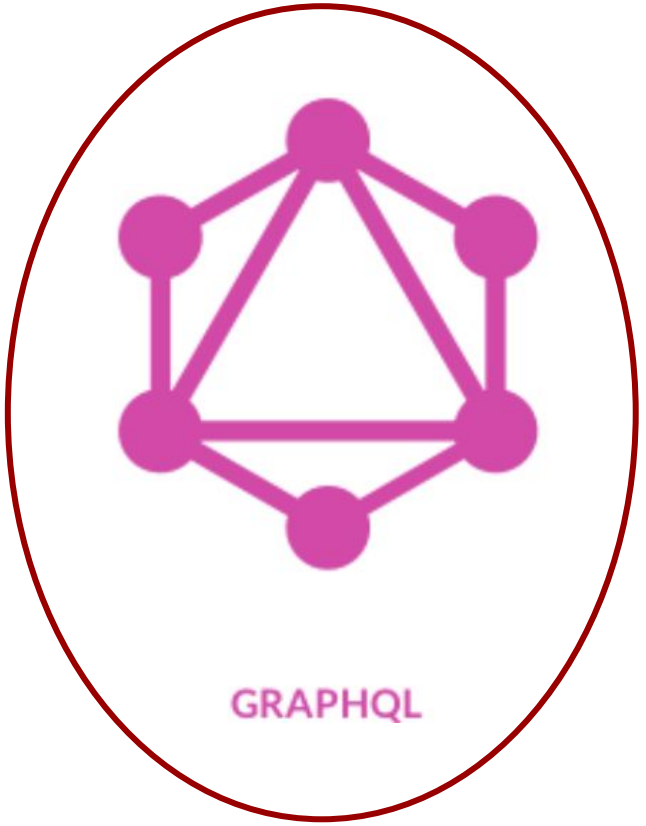
fpriyatna@fi.upm.es
@fpriyatna

02/06/2019
ESWC2019 - Potoroz


R2RML


GRAPHQL


R2RML


GRAPHQL

- Query Language for API
  - Alternative to REST API
- Components
  - Schema (What)
  - Resolvers (How)

"*Get the user's name + posts' titles + followers' names*" (REST-API)

1. /users/<id>

```
{
   "user": {
        "id": ...
        "name": ...
        "address": ...
        "birthday": ...
   }
}
```

2. /users/<id>/posts

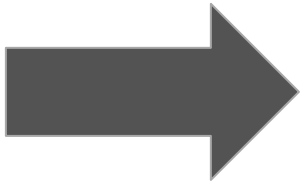
```
{
   "posts": [{
        "id": ...
        "title": ...
        "content": ...
        "comments": [ ...]
   }, ...
   ]
}
```

3. /users/<id>/followers

```
{
   "followers": [{
        "id": ...
        "name": ...
        "address": ...
        "birthday": ...
   }, ...
   ]
}
```

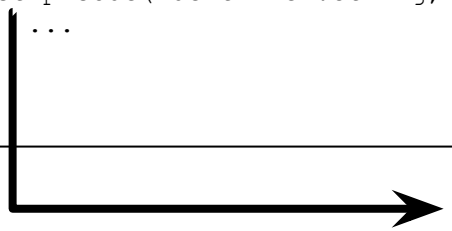
"*Get the user's name + posts' titles + followers' names*" (GraphQL)

```
query {
    User(id: "...") {
        name
        posts {
            title
        }
        followers {
            name
        }
    }
}
```



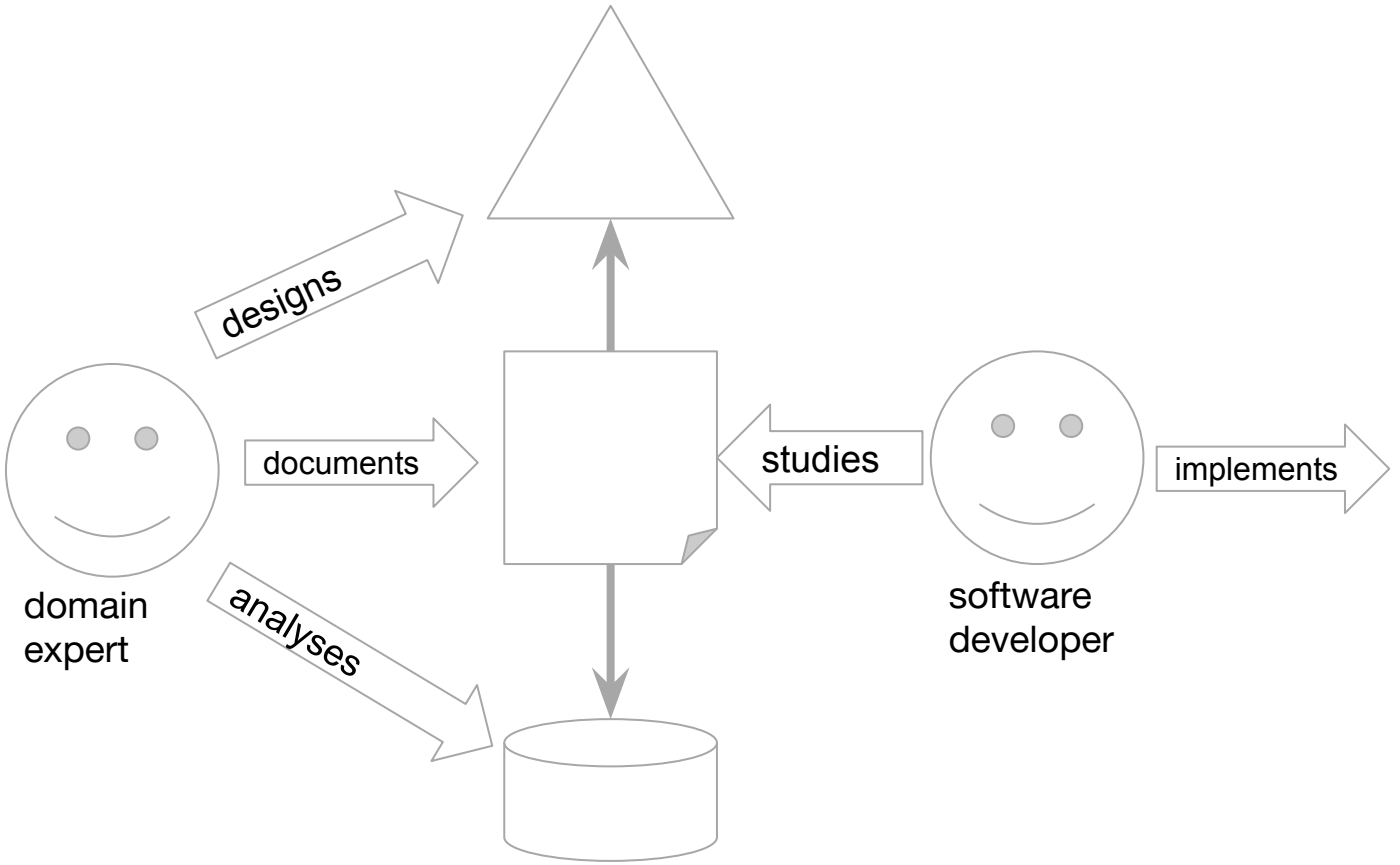
```
"data" {
    "User"{
        "name": ... ,
        "posts": [
            { "title": ...}
        ],
        followers: [
            {"name": ...},
            {"name": ...},
            ...
        ]
    }
}
```

```
type Episode {
      identifier:String
      code:String
}

type Query {
  listEpisode(identifier:String, code:String): [Episode]
      ...
}
```



```
listEpisode: function({identifier,code}) {
  let sql = `SELECT
      'ex.com/episode/' || eid AS c1
      , ecode AS c2
      FROM TblEpisodes
      WHERE
      c1 = ${identifier} AND c2 = ${code}
  `
  let data = db.all(sql);
  let allInstances = [];
  return data.then(rows => {
      rows.forEach((row) => {
      let instance = new Episode(
      row['c1'], row['c2']
      );
      allInstances.push(instance);
      })
      return allInstances;
  });
}
```

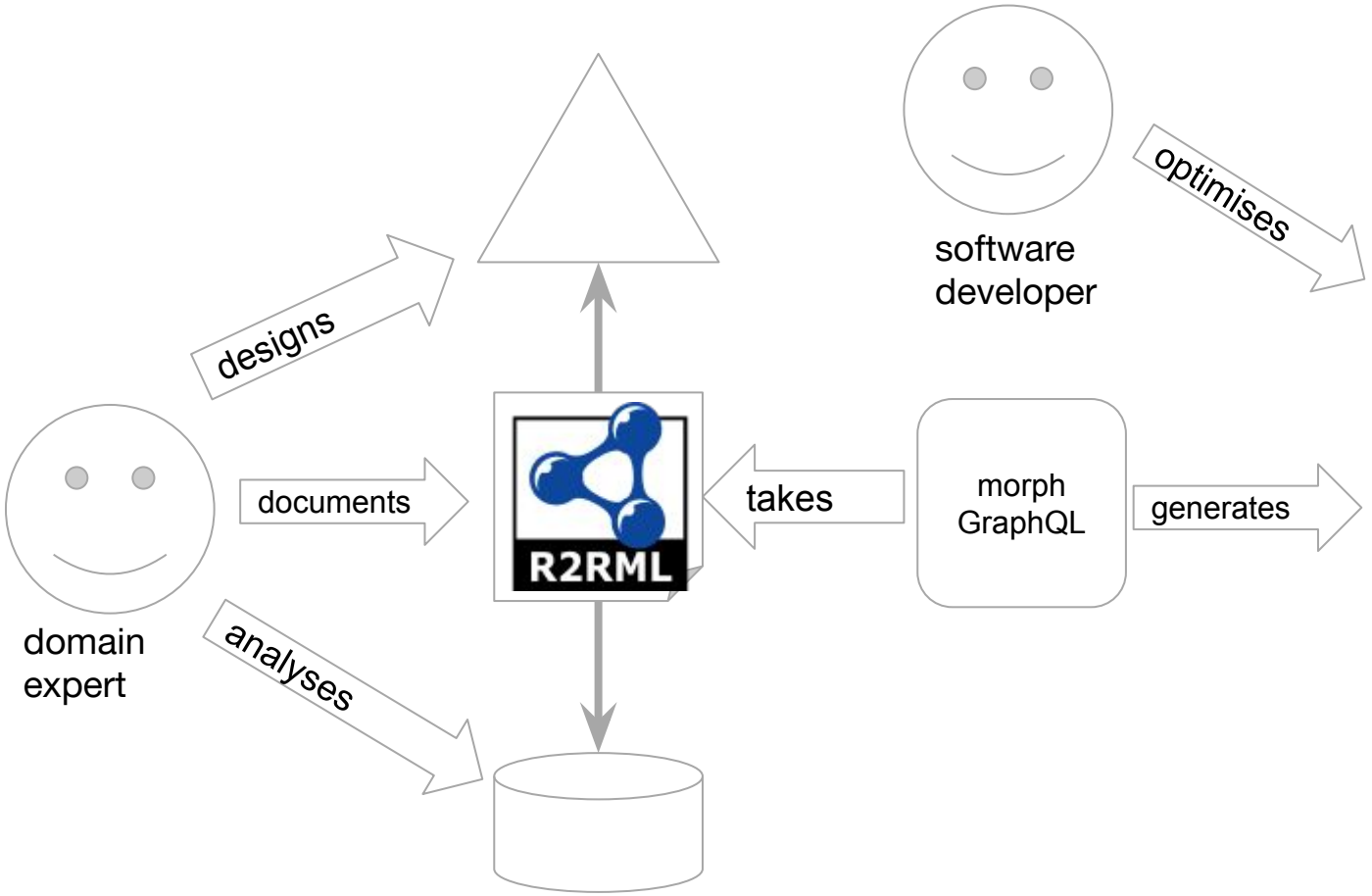
GraphQL Server Development Flow



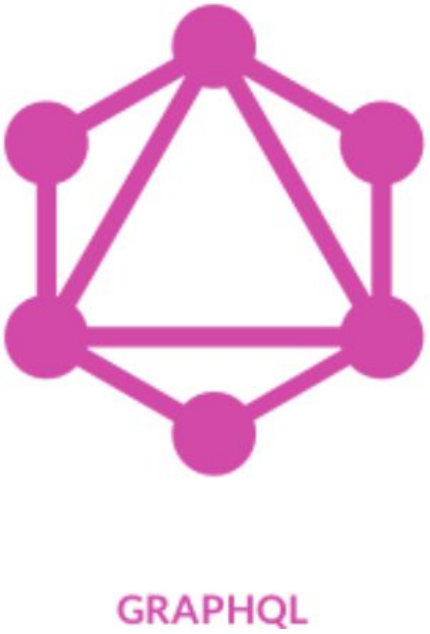

```
listEpisode: function({identifier,code}) {
  let sql = `SELECT
          'ex.com/episode/' || eid AS c1
          , ecode AS c2
          FROM TblEpisodes
          WHERE
          c1 = ${identifier} AND c2 = ${code}
  `
  let data = db.all(sql);
  let allInstances = [];
  return data.then(rows => {
          rows.forEach((row) => {
          let instance = new Episode(
          row['c1'], row['c2']
          );
          allInstances.push(instance);
          })
          return allInstances;
  });
}
```

GraphQL
Server

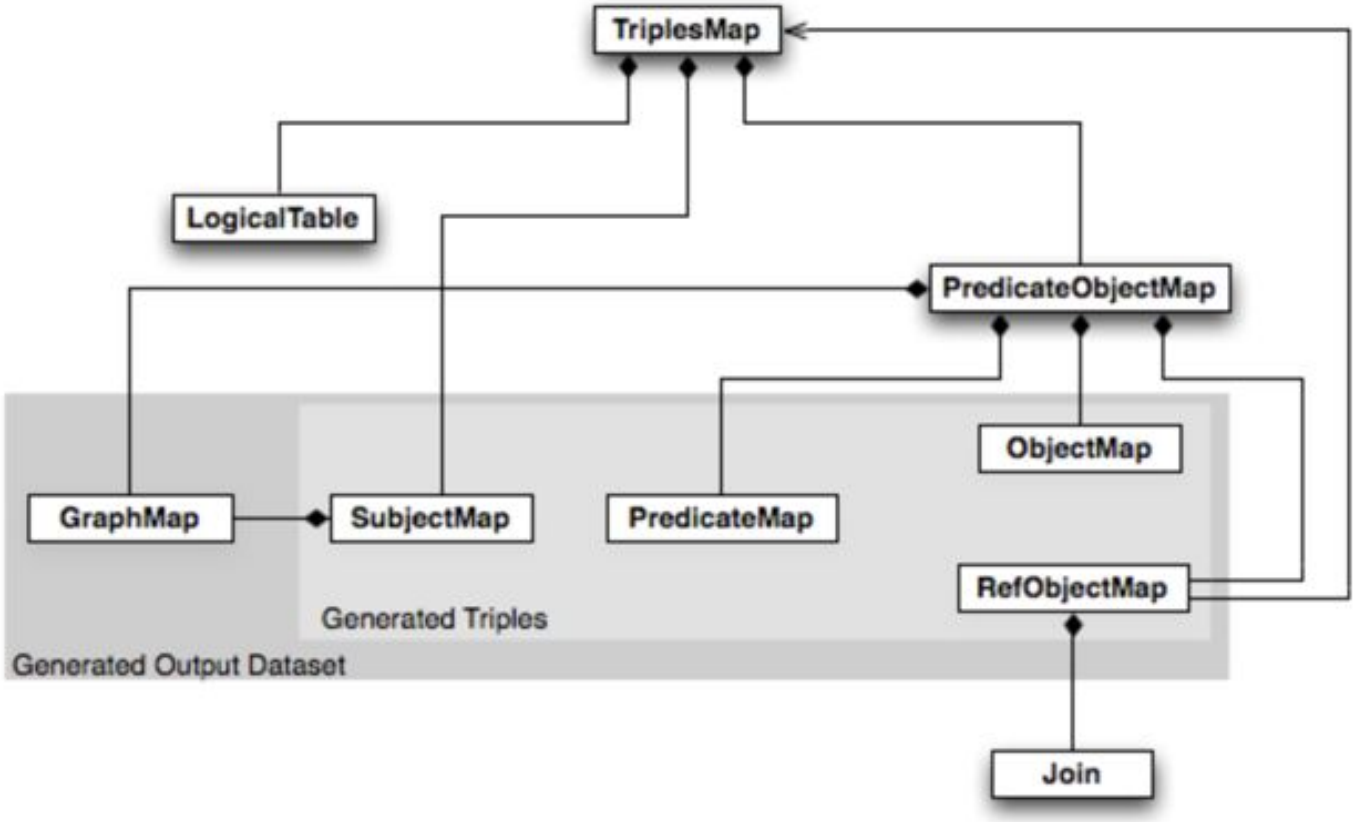
Our Proposal




R2RML


GRAPHQL

- W3C Recommendation
- RDB to RDF
- Declarative



R2RML Example

Example output data

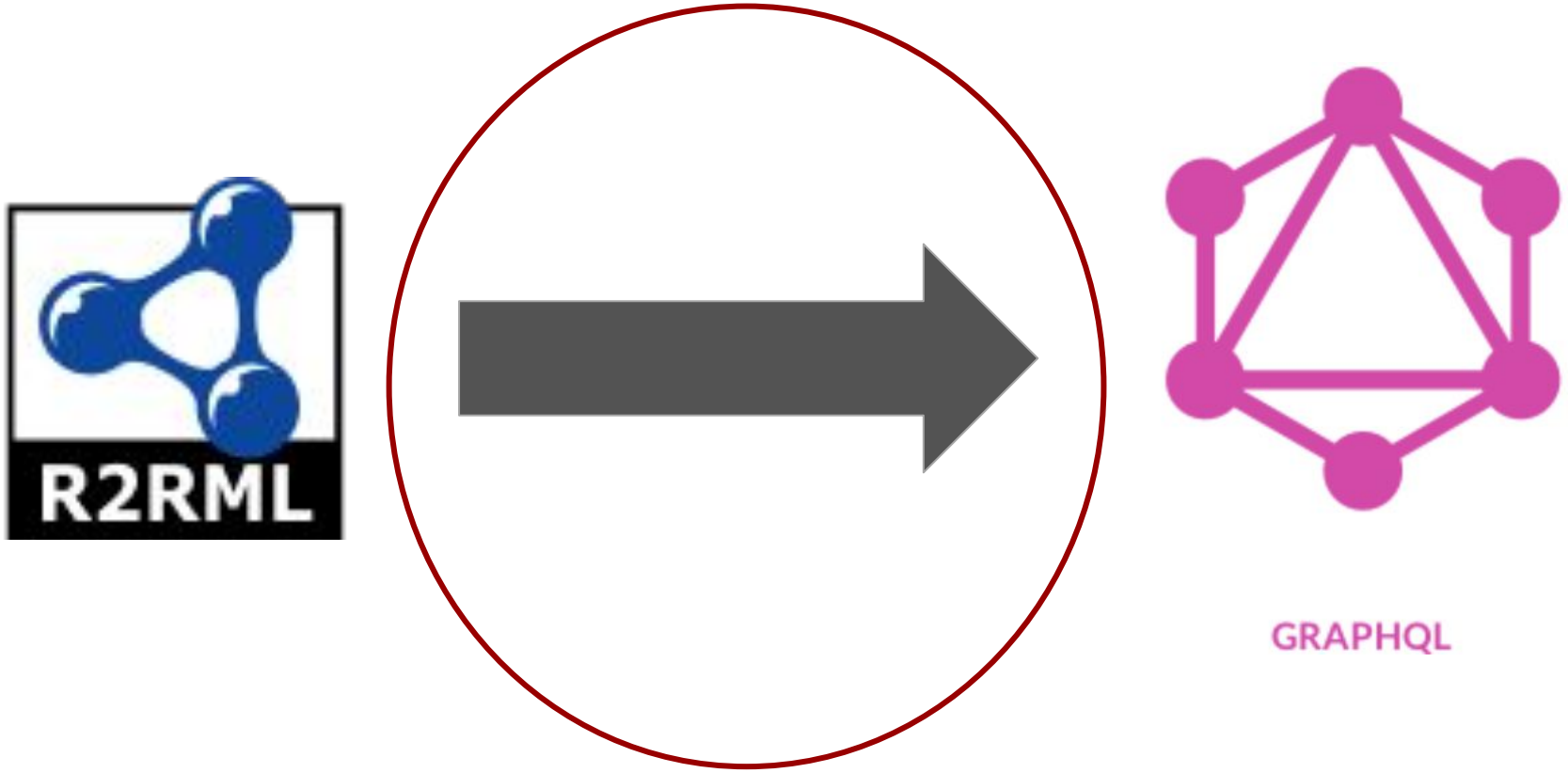
```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
```

Example R2RML mapping

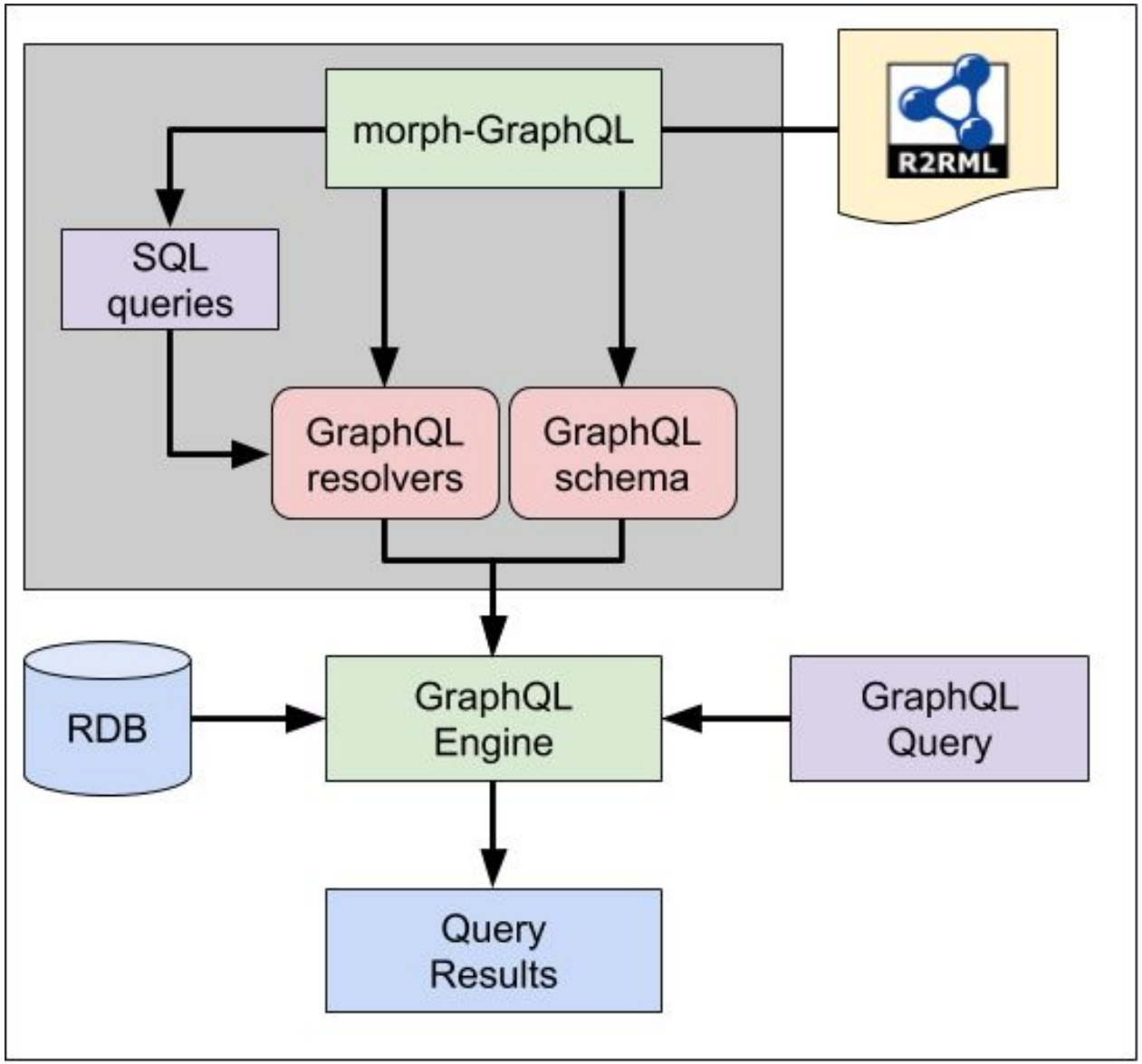
```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/ns#>.

<#TriplesMap1>
    rr:logicalTable [ rr:tableName "EMP" ];
    rr:subjectMap [
        rr:template "http://data.example.com/employee/{EMPNO}";
        rr:class ex:Employee;
    ];
    rr:predicateObjectMap [
        rr:predicate ex:name;
        rr:objectMap [ rr:column "ENAME" ];
    ].
```

EMP

EMPNO INTEGER PRIMARY KEY	ENAME VARCHAR(100)	JOB VARCHAR(20)	DEPTNO INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10


R2RML
GRAPHQL

morph-GraphQL Workflow



Availability

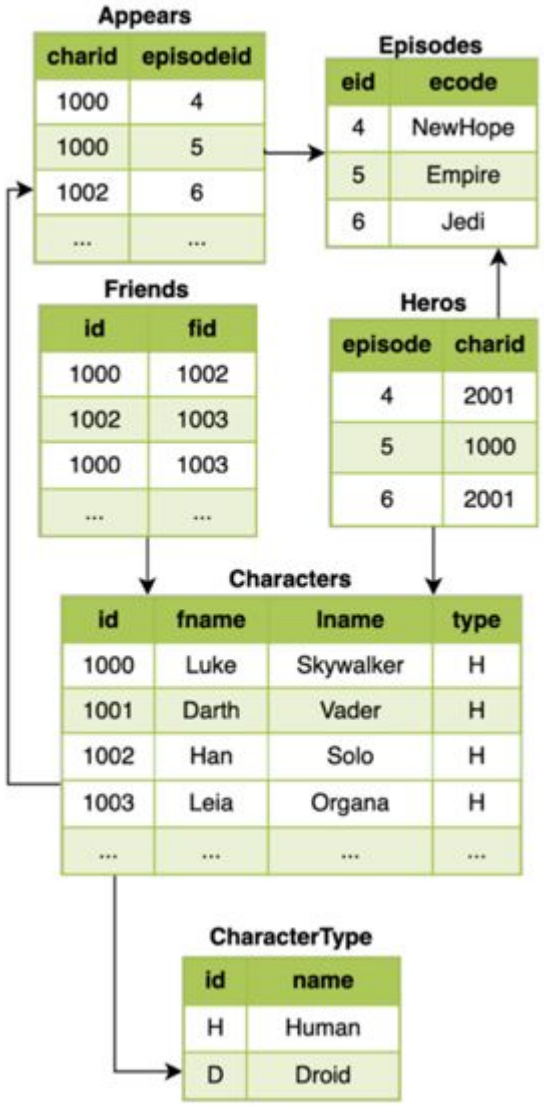
- Source code: https://github.com/oeg-upm/morph-graphql
- Deployment: http://graphql.morph.oeg-upm.net

Build your own morph-GraphQL

1. git clone https://github.com/oeg-upm/morph-graphql
2. cd morph-graphql/javascript/rdb
3. npm install
4. node app.js

morph-GraphQL is now ready at port 8082

Starwars dataset & queries



No	Desc	Tables
Q1	Get the hero of every episode	Heroes, Episodes, Characters
Q2	Get id and friends of R2-D2	Characters, Friends
Q3	Get info of Luke from his ID	Characters
Q4	Query both Luke and Leia	Characters
Q5	Verify that R2-D2 is a robot	Characters, Types
Q6	Verify that the hero of episode “Empire” is a human	Heroes, Characters, Episodes, Types

https://github.com/oeg-upm/morph-graphql/tree/master/examples/starwars

Build your own Starwars GraphQL server

https://github.com/oeg-upm/vkg-tutorial-eswc2019/tree/master/morph-graphql

1. mkdir output
2. cd output
3.
```
curl -X POST http://localhost:8082/transform -H 'Content-Type: application/json' -d '{ "prog_lang": "javascript", "dataset_type":"csv", "mapping_url":"https://raw.githubusercontent.com/oeg-upm/morph-graphql/master/examples/starwars/mappings.ttl", "db_name":"starwars.sqlite", "mapping_language":"rml", "queryplanner":"joinmonster" }' > output.zip
```
4. unzip output.zip
5. npm install
6. npm start
7. Go to http://localhost:4321/graphql from your browser

Note: replace http://localhost:8082 with http://graphql.morph.oeg-upm.net if you want to use the online server

http://localhost:4321/graphql

http://starwars.graphql.oeg-upm.net/graphql

Q1: To query the hero in every episode

GraphsiQL ▶ Prettify Docs

```
{
  listHeroes {
    episode {
      identifier
      code
    }
    hero {
      identifier
      name
    }
  }
}
```

```
{
  "episode": [
    {
      "identifier": "http://starwars.mappingpedia.linkeddata.es/episode/4",
      "code": "NewHope"
    }
  ],
  "hero": [
    {
      "identifier": "http://starwars.mappingpedia.linkeddata.es/character/2001",
      "name": "R2 D2"
    }
  ]
},
{
  "episode": [
    { ↔ }
  ],
  "hero": [
    {
      "identifier": "http://starwars.mappingpedia.linkeddata.es/character/1000",
      "name": "Luke Skywalker"
    }
  ]
```

```
SELECT
  "listHeroes"."episodeid" || "listHeroes"."charid" AS "epi#cha",
  "episode"."id" AS "episode__id",
  'http://starwars.mappingpedia.linkeddata.es/episode/' || "episode".id || '' AS "episode__identifier",
  "episode"."code" AS "episode__code",
  "hero"."id" AS "hero__id",
  'http://starwars.mappingpedia.linkeddata.es/character/' || "hero".id || '' AS "hero__identifier",
  '' || "hero".fname || ' ' || "hero".lname || '' AS "hero__name"
FROM heroes "listHeroes"
LEFT JOIN episodes "episode" ON "listHeroes".episodeid = "episode".id
LEFT JOIN characters "hero" ON "listHeroes".charid = "hero".id
```

Q2: to query the ID and friends of R2-D2

GraphsiQL ▶ Prettify Docs

```
{
  listCharacter(name: "R2 D2"){
    identifier
    name
    friends {
      identifier
      charid
      friendId
    }
  }
}
```


```
{
  "data": {
    "listCharacter": [
      {
        "identifier": "http://starwars.mappingpedia.linkeddata.es/character/2001",
        "name": "R2 D2",
        "friends": [
          {
            "identifier": "http://starwars.mappingpedia.linkeddata.es/friends/2001/1000",
            "charid": "2001",
            "friendId": "1000"
          },
          {
            "identifier": "http://starwars.mappingpedia.linkeddata.es/friends/2001/1002",
            "charid": "2001",
            "friendId": "1002"
          },
          {
            "identifier": "http://starwars.mappingpedia.linkeddata.es/friends/2001/1003",
            "charid": "2001",
            "friendId": "1003"
          }
```


```
SELECT
  "listCharac"."id" AS "id",
  'http://starwars.mappingpedia.linkeddata.es/character/' || "listCharac".id || '' AS "identifier
  '' || "listCharac".fname || ' ' || "listCharac".lname || '' AS "name",
  "friends"."id" || "friends"."fid" AS "friends__id#fid",
  'http://starwars.mappingpedia.linkeddata.es/friends/' || "friends".id || '/' || "friends".fid |
  "friends"."id" AS "friends__charid",
  "friends"."fid" AS "friends__friendId"
FROM characters "listCharac"
LEFT JOIN friends "friends" ON "listCharac".id = "friends".id
WHERE '' || "listCharac".fname || ' ' || "listCharac".lname || '' = 'R2 D2'
```

Q3: to query Luke Skywalker using his ID

GraphsiQL ▶ Prettify Docs

```
{
  listCharacter(
      identifier: "http://starwars.mappingpedia.linkeddata.es/character/1000")
  {
    name
  }
}
```

```
{
  "data": {
    "listCharacter": [
      {
        "name": "Luke Skywalker"
      }
    ]
  }
}
```

```
SELECT
  "listCharac"."id" AS "id",
  '' || "listCharac".fname || ' ' || "listCharac".lname || '' AS "name"
FROM characters "listCharac"
WHERE 'http://starwars.mappingpedia.linkeddata.es/character/' || "listChar
```

Q4: to query Luke & Leia

GraphsiQL ▶ Prettify Docs

```
query FetchLukeAndLeiaAliased {
  luke: listCharacter(
    identifier: "http://starwars.mappingpedia.linkeddata.es/character/1000")
  {
    name
  }
  leia: listCharacter(
    identifier: "http://starwars.mappingpedia.linkeddata.es/character/1003")
  {
    name
  }
}
```

QUERY VARIABLES

```
{
  "data": {
    "luke": [
      {
        "name": "Luke Skywalker"
      }
    ],
    "leia": [
      {
        "name": "Leia Organa"
      }
    ]
  }
}
```

```
SELECT
  "listCharac"."id" AS "id",
  '' || "listCharac".fname || ' ' || "listCharac".lname || '' AS "name"
FROM characters "listCharac"
WHERE 'http://starwars.mappingpedia.linkeddata.es/character/' || "listChar

SELECT
  "listCharac"."id" AS "id",
  '' || "listCharac".fname || ' ' || "listCharac".lname || '' AS "name"
FROM characters "listCharac"
WHERE 'http://starwars.mappingpedia.linkeddata.es/character/' || "listChar
```

Q5: to verify that D2-R2 is a droid

GraphsiQL ▶ Prettify Docs

```
{
  listCharacter(name: "R2 D2
    identifier
    name
    type(name: "Droid") {
      identifier
      name
    }
  }
}
```


```
{
  "data": {
    "listCharacter": [
      {
        "identifier": "http://starwars.mappingpedia.linkeddata.es/character/2001",
        "name": "R2 D2",
        "type": [
          {
            "identifier": "http://starwars.mappingpedia.linkeddata.es/type/D",
            "name": "Droid"
          }
        ]
      }
    ]
  }
}
```


```
SELECT
  "listCharac"."id" AS "id",
  'http://starwars.mappingpedia.linkeddata.es/character/' || "listCharac".id || '' AS "identifi
  '' || "listCharac".fname || ' ' || "listCharac".lname || '' AS "name",
  "type"."id" AS "type__id",
  'http://starwars.mappingpedia.linkeddata.es/type/' || "type".id || '' AS "type__identifier",
  "type"."name" AS "type__name"
FROM characters "listCharac"
LEFT JOIN types "type" ON "listCharac".typeid = "type".id
WHERE '' || "listCharac".fname || ' ' || "listCharac".lname || '' = 'R2 D2' AND "type".name = '
```

Q6: to verify that the hero of episode Empire is a human

GraphsiQL ▶ Prettify Docs

```
{
  listHeroes {
    identifier
    hero {
      identifier
      name
      type(name: "Human") {
        identifier
        name
      }
    }
    episode(code: "Empire") {
      identifier
      code
    }
  }
}
```

QUERY VARIABLES


```
{
  "data": {
    "listHeroes": [
      {
        "identifier": "http://starwars.mappingpedia.linkeddata.es/heroes/5/1000",
        "hero": [
          {
            "identifier": "http://starwars.mappingpedia.linkeddata.es/character/1000",
            "name": "Luke Skywalker",
            "type": [
              {
                "identifier": "http://starwars.mappingpedia.linkeddata.es/type/H",
                "name": "Human"
              }
            ]
          }
        ],
        "episode": [
          {
            "identifier": "http://starwars.mappingpedia.linkeddata.es/episode/5",
            "code": "Empire"
```


```
SELECT
  "listHeroes"."episodeid" || "listHeroes"."charid" AS "epi#cha",
  'http://starwars.mappingpedia.linkeddata.es/heroes/' || "listHeroes".episodeid || '/' || "
  "hero"."id" AS "hero__id",
  'http://starwars.mappingpedia.linkeddata.es/character/' || "hero".id || '' AS "hero__ident
  '' || "hero".fname || ' ' || "hero".lname || '' AS "hero__name",
  "type"."id" AS "hero__type__id",
  'http://starwars.mappingpedia.linkeddata.es/type/' || "type".id || '' AS "hero__type__iden
  "type"."name" AS "hero__type__name",
  "episode"."id" AS "episode__id",
  'http://starwars.mappingpedia.linkeddata.es/episode/' || "episode".id || '' AS "episode__i
  "episode"."code" AS "episode__code"
```

How to generate mappings?

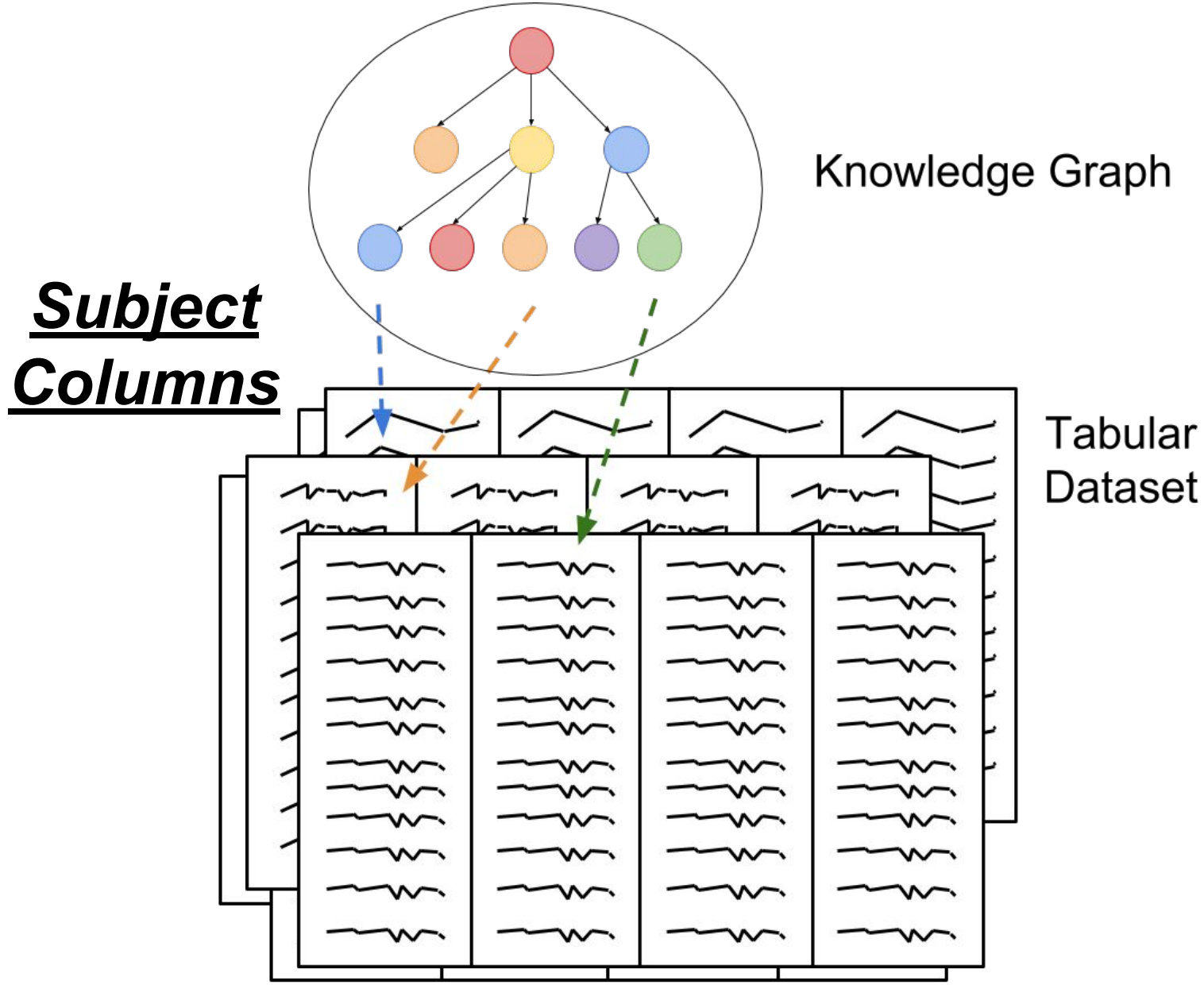
TADA

Ahmad Alobaid

Ontology Engineering Group
Universidad Politécnica de Madrid

aalobaid@fi.upm.es
@oeg-upm

02/06/2019
ESWC2019 - Potoroz


Knowledge Graph
Subject Columns
Tabular Dataset

Subject Column

creator	{|creation}	{|age (years)
louis braille	invented a system of printing and writir	15
blaise pascal	formulated pascal,Äôs theorem	16
galileo galilei	discovered the laws of pendulum motio	17
edwin land	patented his first polarizing light filter	19
george westinghouse	first patent for a rotary steam engineir	1922
guglielmo marconi	invented a system of radio telegraphy	21
joshua lederberg	discovered bacterial conjugation	21
thomas edison	invented automated relaying communi	22
carl gauss	proved the theorem of complex coeffici	22
john nash	published his theory of non-cooperative	22
brian josephson	predicted the josephson effect	22
james hillier	developed the electron microscope	22
isaac newton	calculus principles of optics elements of	23
louis parker	invented a low frequency receiver for ra	23
srinivasa ramanujan	published his first mathematical papers	24
satyendra bose	published is first statistical mechanics p	24
paul dirac	quantum mechanics for motion of atom	24
richard feynman	published his theory of electromagnetic	24

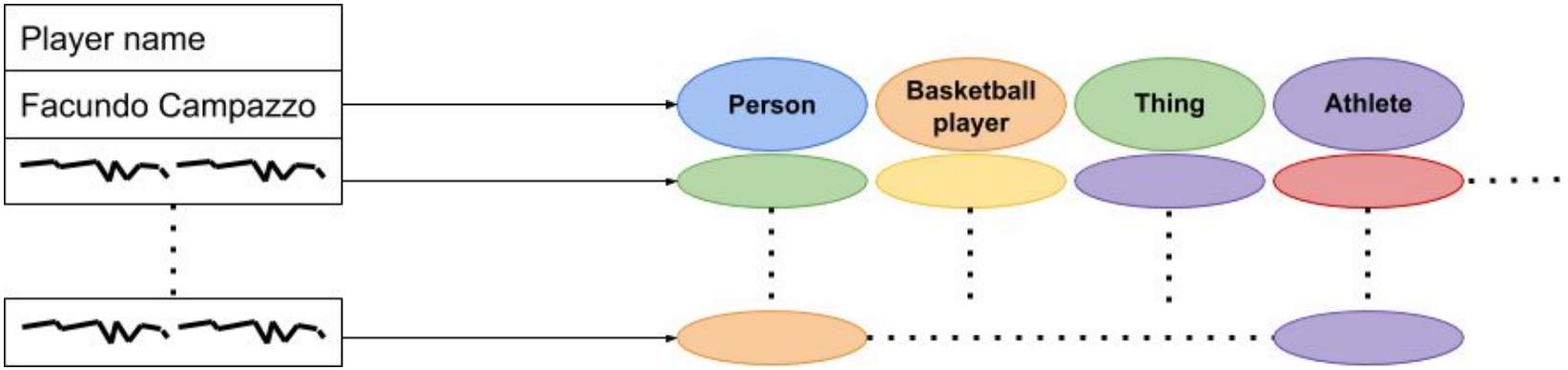
How? link cells to entities?

1. Get entities for each cell

```
select distinct ?subject
where{select distinct ?subject
where{
?subject ?property "Facundo Campazzo"@en}
```

2. Get types for each entity

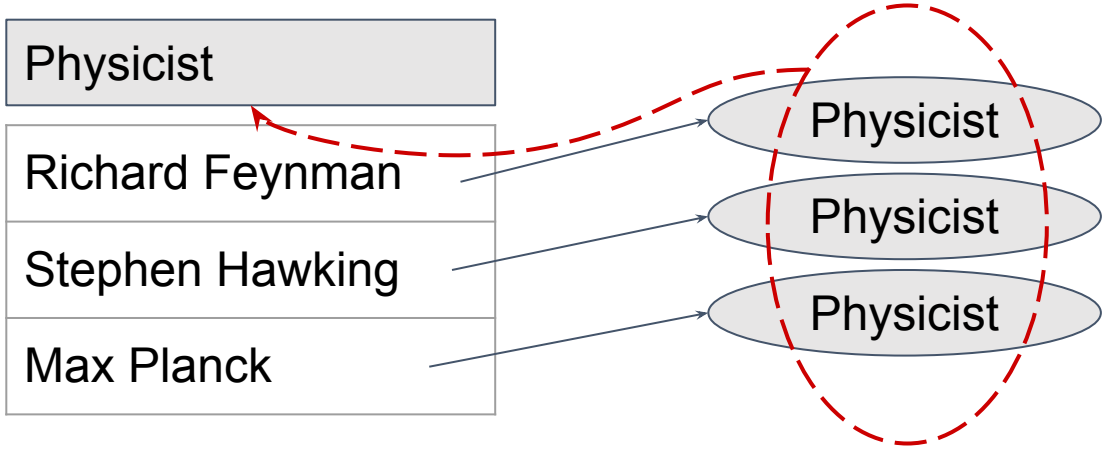
```
select distinct ?class where{
<http://dbpedia.org/resource/Facundo_Campazzo> a ?class}
```



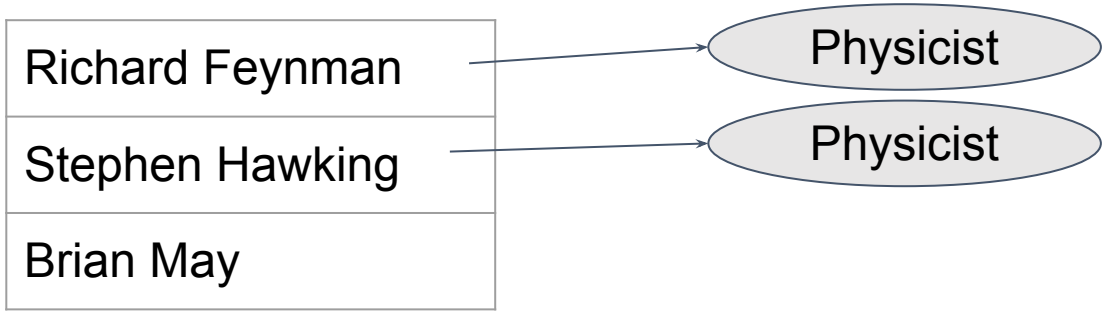
Now what?

https://giphy.com

1. Obvious case:

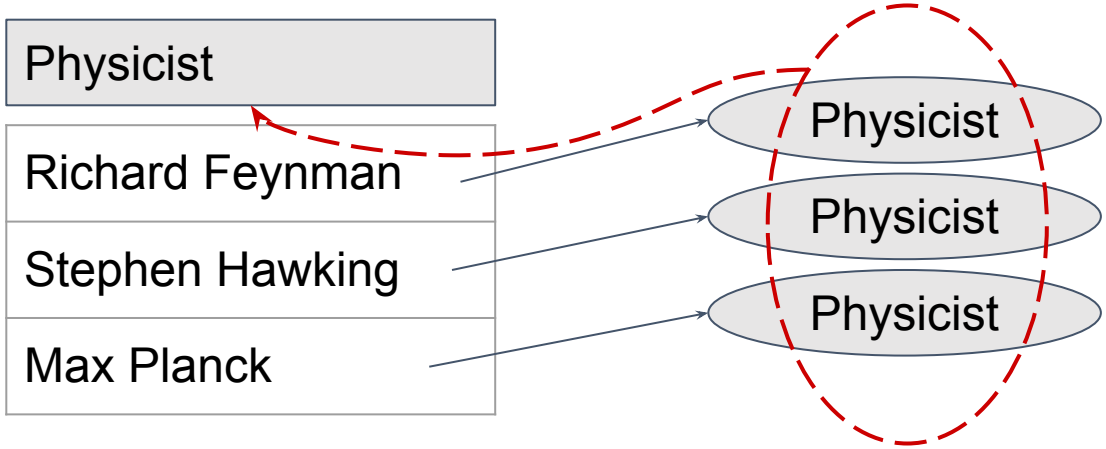


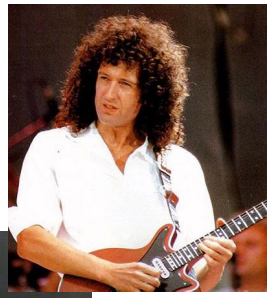
2. Not-so-obvious case:



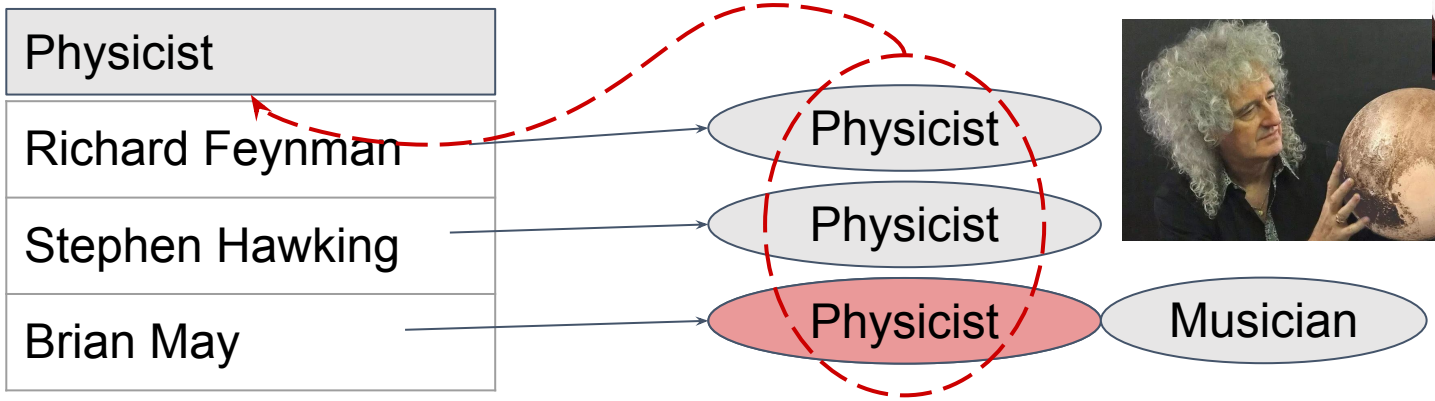
https://cdn.britannica.com/s:300x300/73/20973-004-F71E20CB.jpg
https://upload.wikimedia.org/wikipedia/en/thumb/4/42/Richard_Feynman_Nobel.jpg/220px-Richard_Feynman_Nobel.jpg
https://www.queenie.cz/storage/temp/8f2b24db44d09bd8e3d563d0bb099fc2-400x800x1.jpg
https://www.thetimes.co.uk/imageserver/image/methode%2Ftimes%2Fprod%2Fweb%2Fbin%2F3f44abc8-2774-11e8-acc5-262aff1ca7a6.jpg?crop=988%2C556%2C910%2C214&resize=685

1. Obvious case:



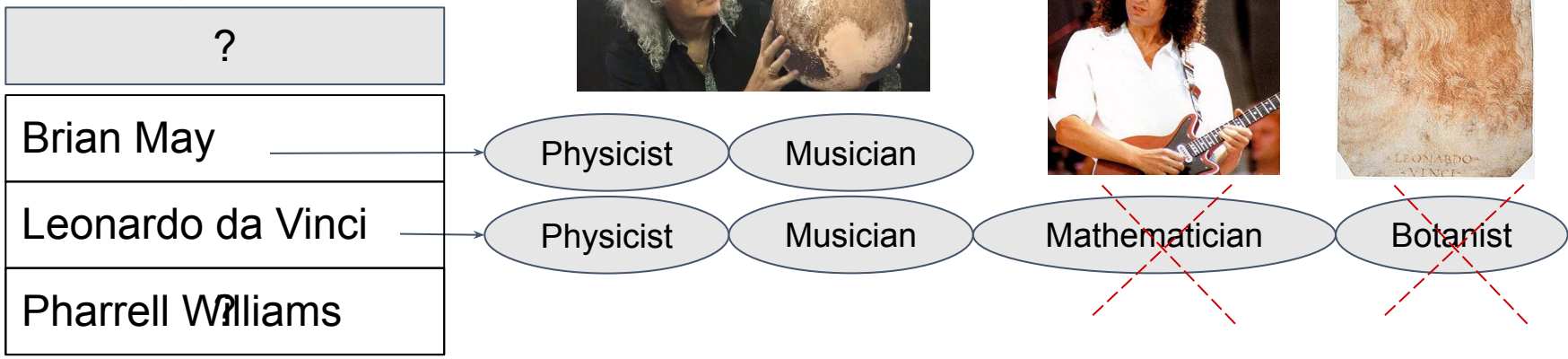


2. Not-so-obvious case:



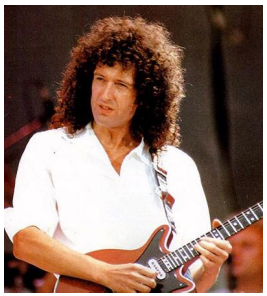
https://cdn.britannica.com/s:300x300/73/20973-004-F71E20CB.jpg
https://upload.wikimedia.org/wikipedia/en/thumb/4/42/Richard_Feynman_Nobel.jpg/220px-Richard_Feynman_Nobel.jpg
https://www.queenie.cz/storage/temp/8f2b24db44d09bd8e3d563d0bb099fc2-400x800x1.jpg
https://www.thetimes.co.uk/imageserver/image/methode%2Ftimes%2Fprod%2Fweb%2Fbin%2F3f44abc8-2774-11e8-acc5-262aff1ca7a6.jpg?crop=988%2C556%2C910%2C214&resize=685
https://i1.wp.com/blog.eil.com/wp-content/uploads/2018/12/brian-may-astrophysicist-07212016.jpg?fit=970%2C545&ssl=1

3. Tricky cases



https://cdn.britannica.com/s:300x300/73/20973-004-F71E20CB.jpg
https://upload.wikimedia.org/wikipedia/en/thumb/4/42/Richard_Feynman_Nobel.jpg/220px-Richard_Feynman_Nobel.jpg
https://www.queenie.cz/storage/temp/8f2b24db44d09bd8e3d563d0bb099fc2-400x800x1.jpg
https://www.thetimes.co.uk/imageserver/image/methode%2Ftimes%2Fprod%2Fweb%2Fbin%2F3f44abc8-2774-11e8-acc5-262aff1ca7a6.jpg?crop=988%2C556%2C910%2C214&resize=685
https://i1.wp.com/blog.eil.com/wp-content/uploads/2018/12/brian-may-astrophysicist-07212016.jpg?fit=970%2C545&ssl=1

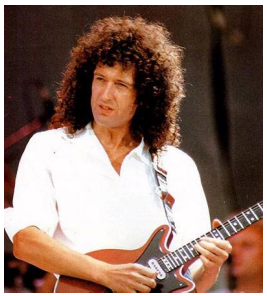
3. Tricky cases



Musician					
Brian May	→	Physicist	Musician		
Leonardo da Vinci	→	Physicist	Musician	Mathematician	Botanist
Pharrell Williams	→	Musician			

https://cdn.britannica.com/s:300x300/73/20973-004-F71E20CB.jpg
https://upload.wikimedia.org/wikipedia/en/thumb/4/42/Richard_Feynman_Nobel.jpg/220px-Richard_Feynman_Nobel.jpg
https://www.queenie.cz/storage/temp/8f2b24db44d09bd8e3d563d0bb099fc2-400x800x1.jpg
https://www.thetimes.co.uk/imageserver/image/methode%2Ftimes%2Fprod%2Fweb%2Fbin%2F3f44abc8-2774-11e8-acc5-262aff1ca7a6.jpg?crop=988%2C556%2C910%2C214&resize=685
https://i1.wp.com/blog.eil.com/wp-content/uploads/2018/12/brian-may-astrophysicist-07212016.jpg?fit=970%2C545&ssl=1

3. Tricky cases



https://cdn.britannica.com/s:300x300/73/20973-004-F71E20CB.jpg
https://upload.wikimedia.org/wikipedia/en/thumb/4/42/Richard_Feynman_Nobel.jpg/220px-Richard_Feynman_Nobel.jpg
https://www.queenie.cz/storage/temp/8f2b24db44d09bd8e3d563d0bb099fc2-400x800x1.jpg
https://www.thetimes.co.uk/imageserver/image/methode%2Ftimes%2Fprod%2Fweb%2Fbin%2F3f44abc8-2774-11e8-acc5-262aff1ca7a6.jpg?crop=988%2C556%2C910%2C214&resize=685
https://i1.wp.com/blog.eil.com/wp-content/uploads/2018/12/brian-may-astrophysicist-07212016.jpg?fit=970%2C545&ssl=1

3. Tricky cases

Person
Brian May
Leonardo da Vinci
Pharrell Williams
Bill Gates

Person

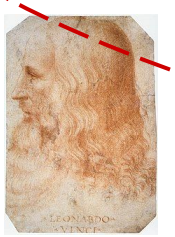
Musician

Person

philanthropist

Investor

Author



Person

Physicist

Botanist

Mathematician

Musician

Person

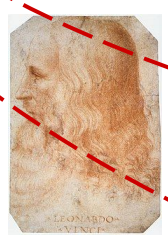
Physicist

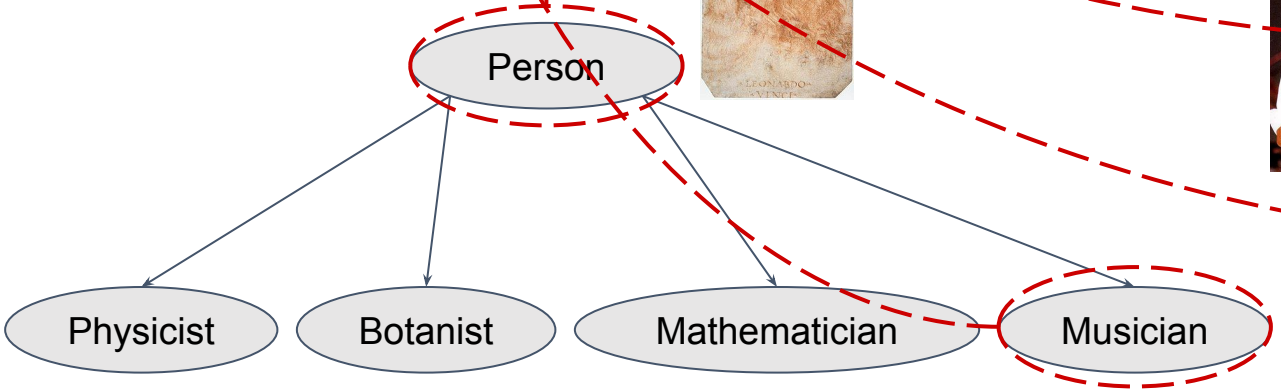
Musician

http://cdn01.cdn.justjared.com/wp-content/uploads/2014/03/williams-shorts/pharrell-williams-wear-shorts-on-oscars-2014-red-carpet-03.jpg
https://pbs.twimg.com/profile_images/988775660163252226/XpgonN0X.jpg
https://www.queenie.cz/storage/temp/8f2b24db44d09bd8e3d563d0bb099fc2-400x800x1.jpg
https://www.thetimes.co.uk/imageserver/image/methode%2Ftimes%2Fprod%2Fweb%2Fbin%2F3f44abc8-2774-11e8-acc5-262aff1ca7a6.jpg?crop=988%2C556%2C910%2C214&resize=685
https://i1.wp.com/blog.eil.com/wp-content/uploads/2018/12/brian-may-astrophysicist-07212016.jpg?fit=970%2C545&ssl=1

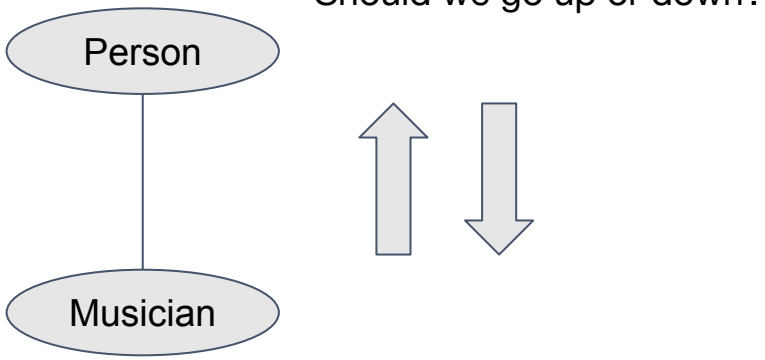
3. Tricky cases

Person
Brian May
Leonardo da Vinci
Pharrell Williams





http://cdn01.cdn.justjared.com/wp-content/uploads/2014/03/williams-shorts/pharrell-williams-wear-shorts-on-oscars-2014-red-carpet-03.jpg
https://pbs.twimg.com/profile_images/988775660163252226/XpgonN0X.jpg
https://www.queenie.cz/storage/temp/8f2b24db44d09bd8e3d563d0bb099fc2-400x800x1.jpg
https://www.thetimes.co.uk/imageserver/image/methode%2Ftimes%2Fprod%2Fweb%2Fbin%2F3f44abc8-2774-11e8-acc5-262aff1ca7a6.jpg?crop=988%2C556%2C910%2C214&resize=685
https://i1.wp.com/blog.eil.com/wp-content/uploads/2018/12/brian-may-astrophysicist-07212016.jpg?fit=970%2C545&ssl=1



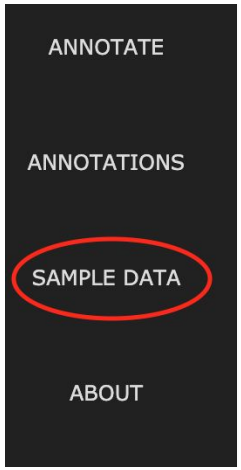
https://giphy.com

Questions?

https://giphy.com

1. Visit TADA's URL: http://tada-entity.linkeddata.es

2. Pick a sample CSV:



Sample CSV files can be found he

- Olympic Games
- Data web commons

3. Add the CSV file:

Name

Boxers

Prefix (optional)

http://dbpedia.org/ontology/

Upload CSV file

Choose file aaaboxers.csv

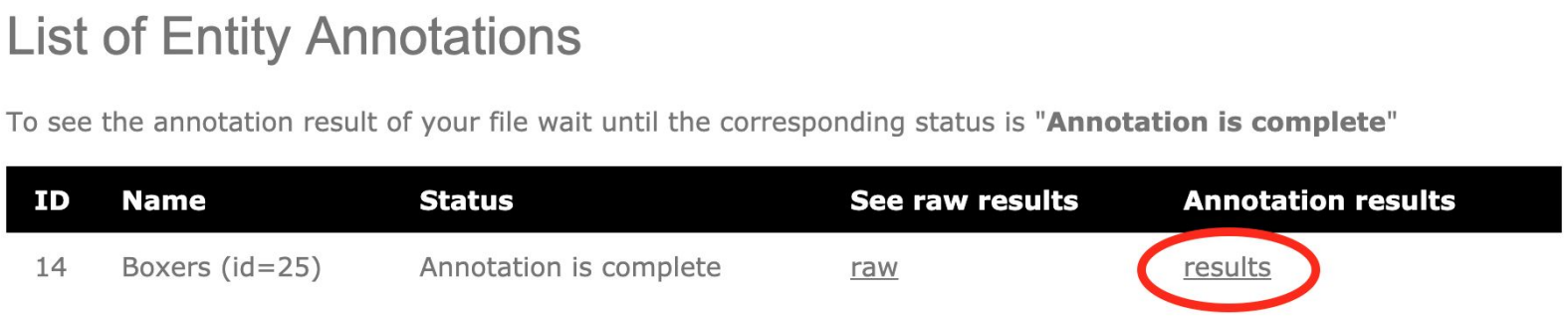
Subject column index (default is 0)

0

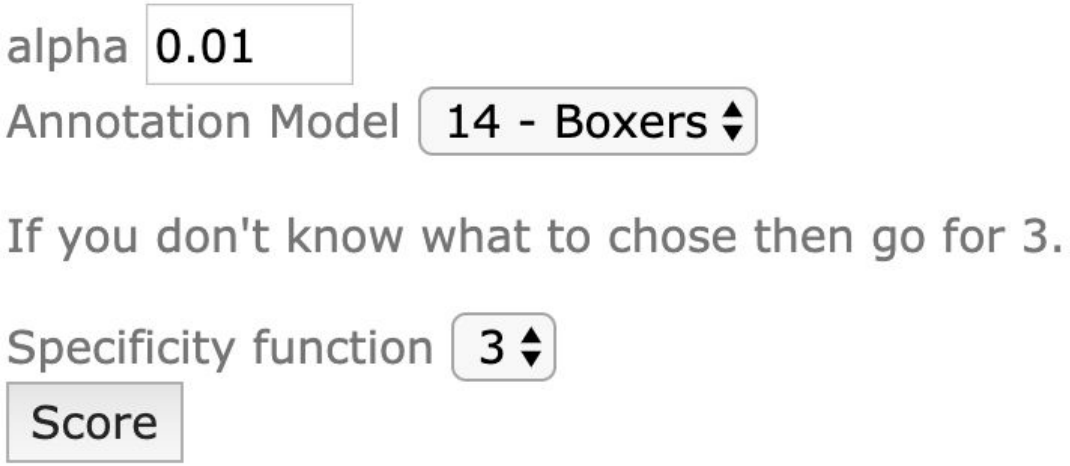
Force title case

Annotate

4. Go to the annotation page:



5. Balance the Coverage and Score:

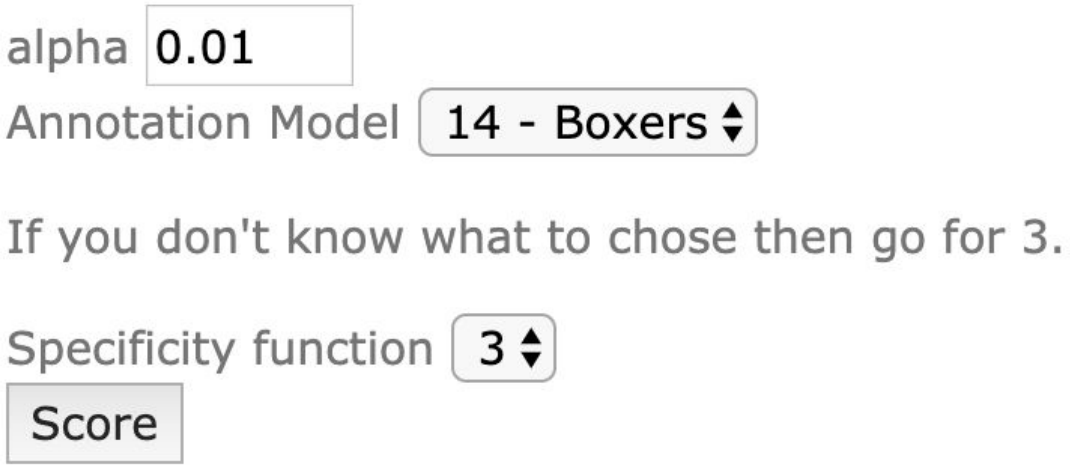


6.

Top K	Concept
1	http://dbpedia.org/ontology/Boxer
2	http://dbpedia.org/ontology/AmateurBoxer
3	http://dbpedia.org/ontology/Philosopher
4	http://dbpedia.org/ontology/SoapCharacter
5	http://dbpedia.org/ontology/VolleyballPlayer

(Optional)

7. Make an annotation more general (one step above)



(Challenge)

8. Can you fool the algorithm, so no alpha would yield a correct answer? (try to have 10 entries)

Helio

Publishing Link Data from heterogeneous data sources

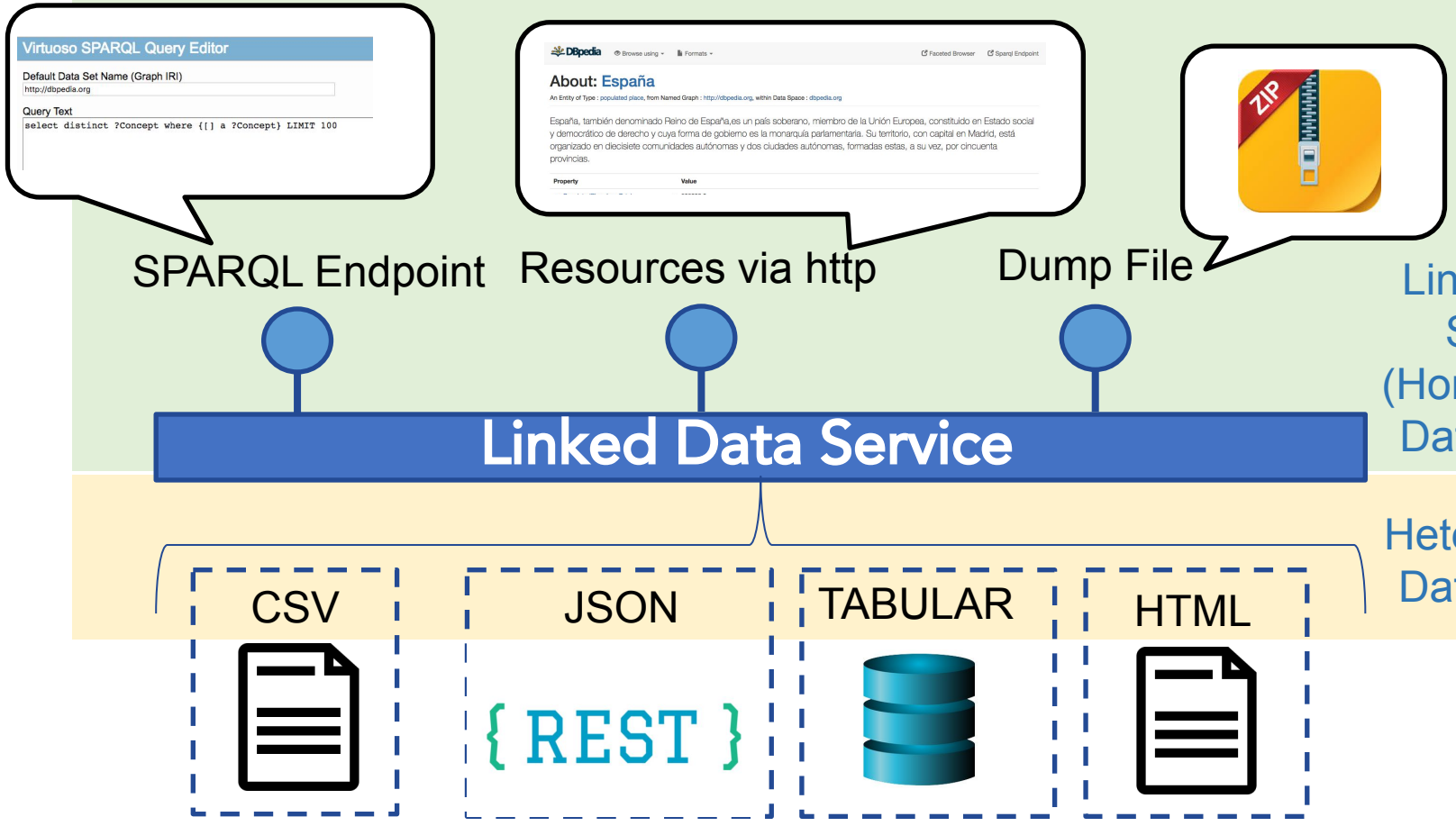
cimmino@fi.upm.es

Andrea Cimmino
Ontology Engineering Group
Universidad Politécnica de Madrid, Spain

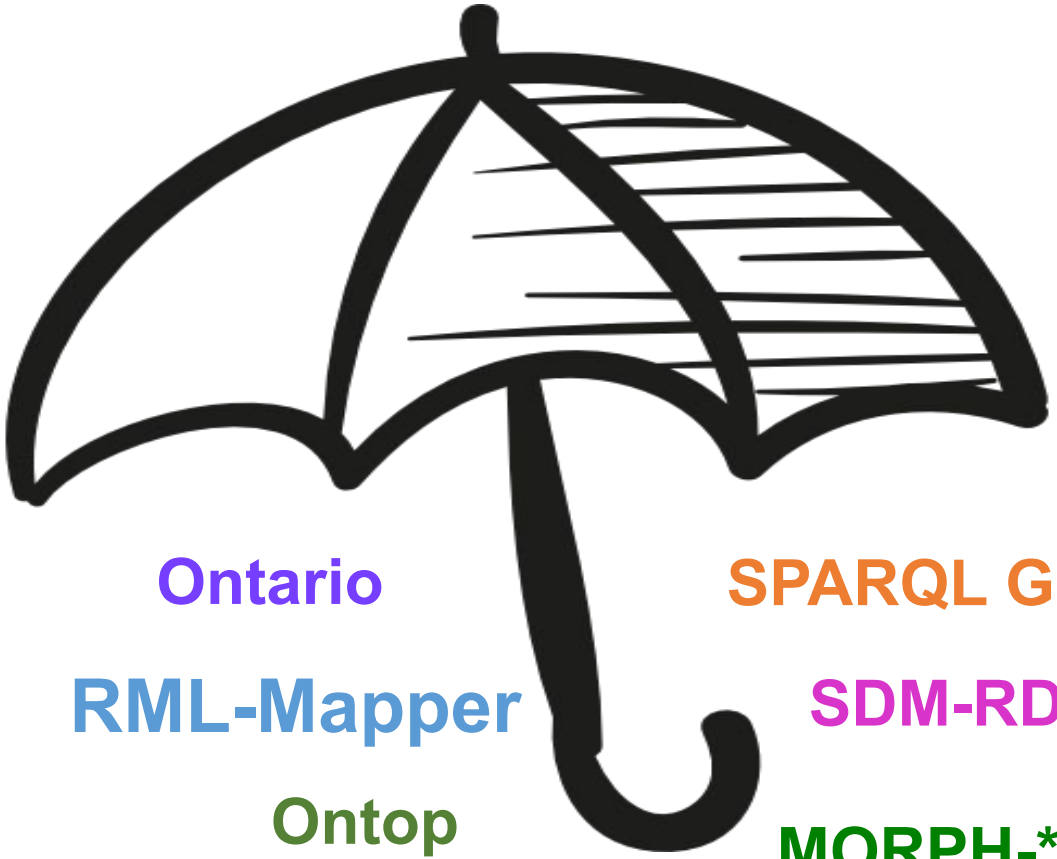
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 688467

European Commission | Horizon 2020 European Union funding for Research & Innovation

From Heterogeneous Data Sources to Linked Data Services



Integrating existing S.W. RDF generators

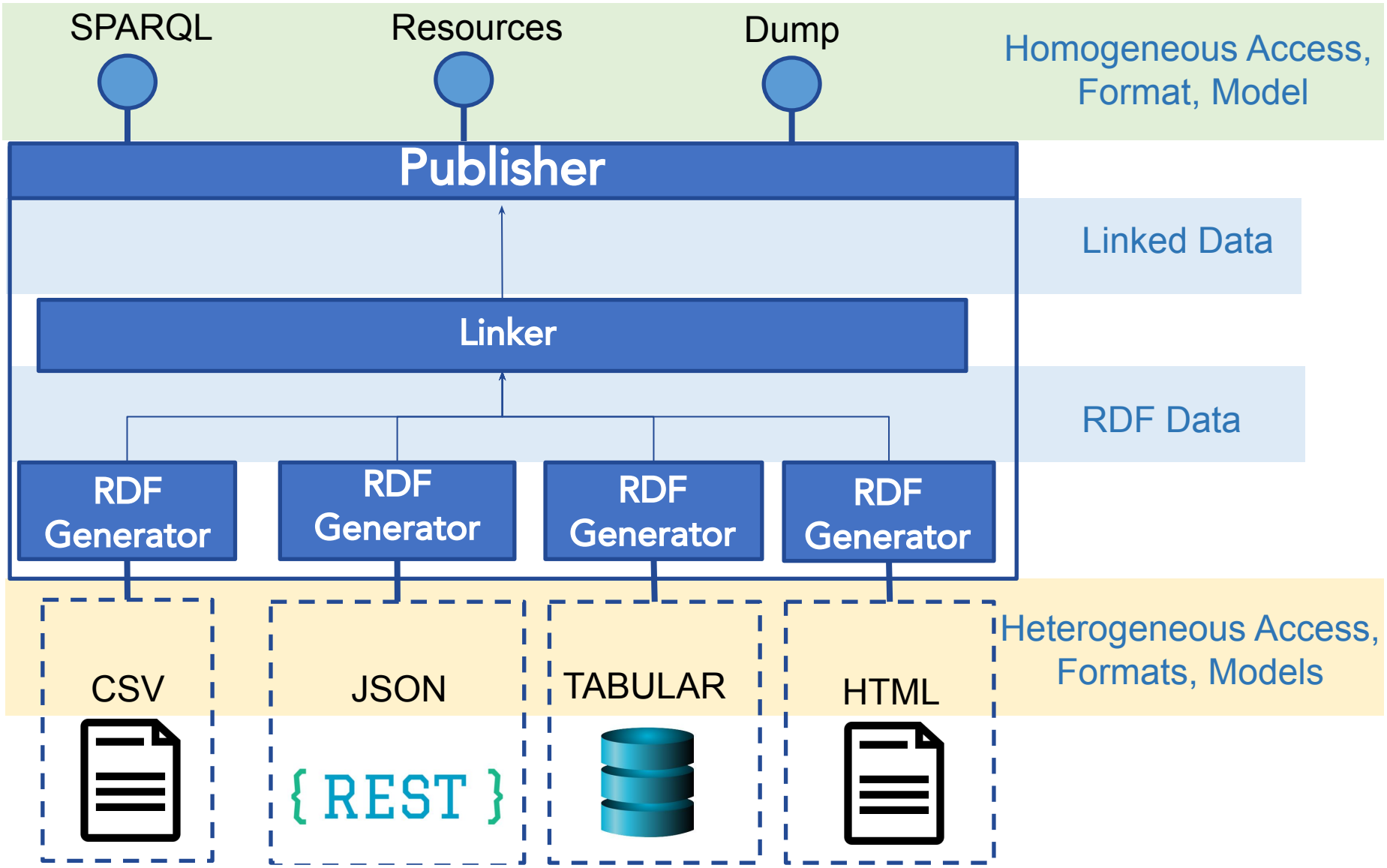


Challenges

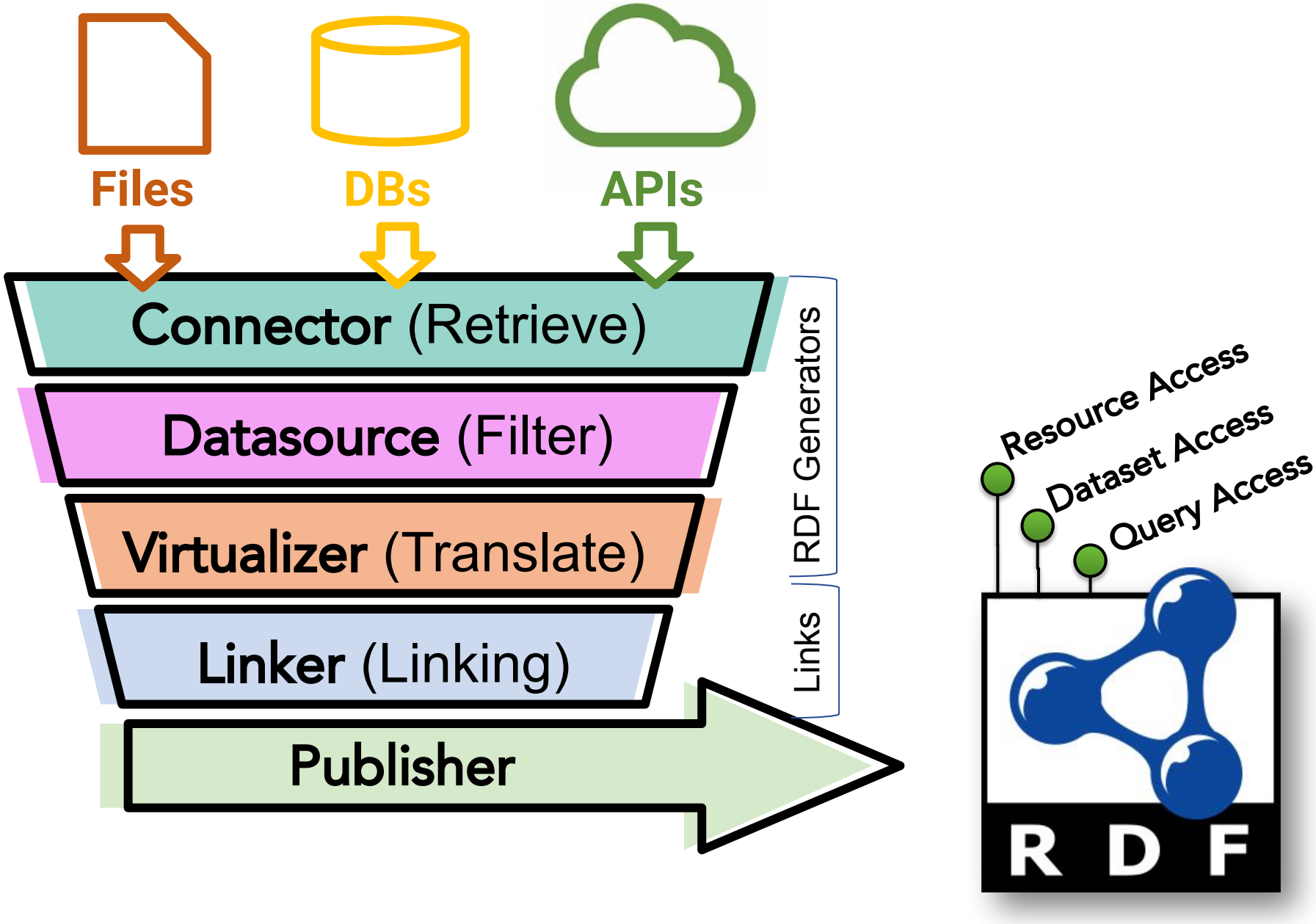
- Cope with the different data sources
  - Access methods, i.e., API, file, DB
  - Formats, i.e., JSON, CSV, Tabular, HTML
  - Security, e.g., APIs with Oath, files with passwords
- Clean data
  - Lowercase, missing values
- Relate data
  - Interlink data from different sources
- Publish as an RDF view the data
  - Enable a SPARQL endpoint
  - Allow resource access
  - Dump generation
- Others
  - Real-time data
  - API restrictions in the number of calls per day
  - Validation of published data
  - Integrate existing S.W. technologies to generate RDF

1. **Helio Solution**
2. Use Cases + Challenges
3. Helio deployment scenarios

Helio Approach



Helio Internal Pipeline



Datasource Example

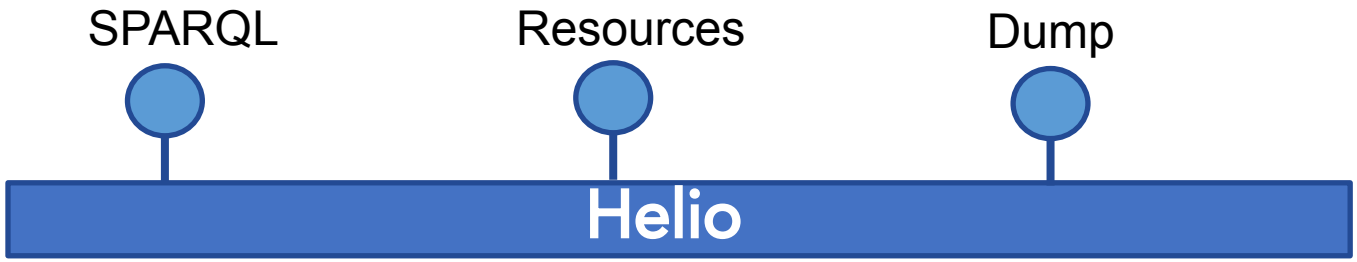
```
"datasources" : [
    {
      "id" : "Table PanTHERIA",
      "type" : "CsvDatasource",
      "arguments" : [";"],
      "refresh" : "22118400000",
      "connector"  : {
       "arguments" : ["./mappings/data/PanTHERIA.csv"],
       "type" : "LocalFileConnector",
      }
    }
 ],
```

(Datasource)

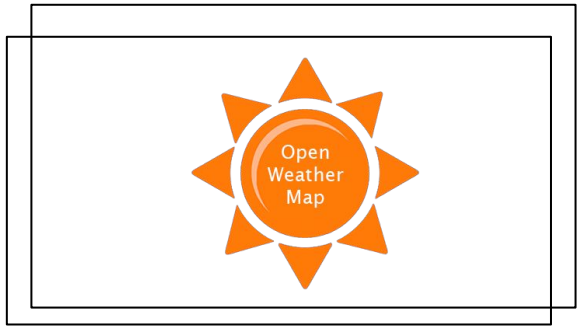
(Connector)

1. Helio Solution
2. **Use Cases + Challenges**
3. Helio deployment scenarios

	Connector	Datasource	Translation	Linking
Challenges	✓	✓	✗	✓



Json sensors data from distributed sources



Open weather map

Meta-data of sensors

Cities & countries

Connector challenge

- Gateway API from VICINTIY requires credentials
  - Our connector passes them as argument in the specification
  - **We do not specify “refresh” to have real-time values from sensors**
- The Database that stores the meta-data also requires auth

```
{
    "id" : ”Gateway API vicinity connector",
    "type" : "JsonDatasource",
   - "refresh" : "300000",
    "arguments" : [”$.*"],
    "connector"  : {
     "arguments" :[“0XSA9123-Asd0123”, “asd-123-dsa-124Xfas”],
     "type" : ”VICINITYConnector",
    }
   }
```

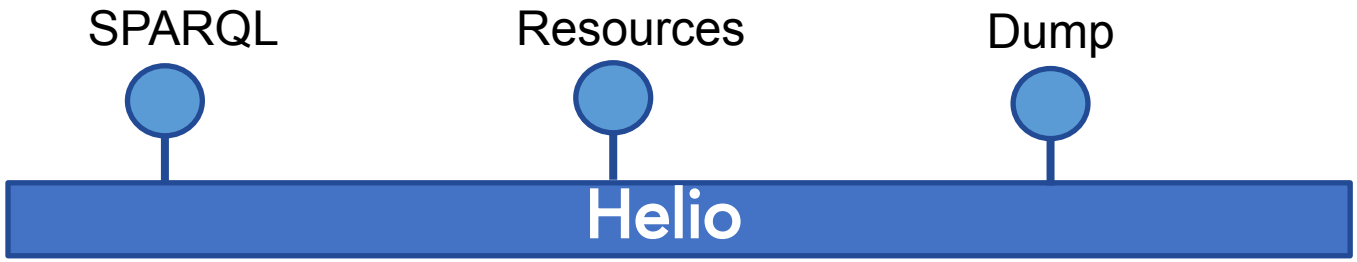
- Dbpedia does not change data often
  - To speed up the publishing process we include that data in Helio cache with refresh argument

```
{
    "id" : ”DBPedia Cities and countries",
    "type" : ”JSONDatasource",
    "refresh" : "36000000",
    "arguments" : [”…"],
    "connector"  : {
     "arguments" : [“…”],
     "type" : "GetConnector",
    }
  }
```

- Link countries and cities with the same name


```
"relationships": [
   {
    "condition" : "levenshtein(lower(regexp_replace(S({$.name}), '\\s+', ")),
                                lower(regexp_replace(escapeHtml4(stripAccents(T({$.city}))), '\\s+', '_'))) < 3",
    "predicates" : ["http://www.w3.org/2002/07/owl#sameAs"],
    "inverse_predicates" : ["http://www.w3.org/2002/07/owl#sameAs"],
    "source_resource_rule_id" : "STARS4ALL Alerts Cities metadata",
    "target_resource_rule_id" : "STARS4ALL Cities metadata"
   }
 ]
```

	Connector	Datasource	Translation	Linking
Challenges	✓	✓	✓	✓



Tweets of OEG Members

OEG Members

Static data of OEG Group

Scholar indexes

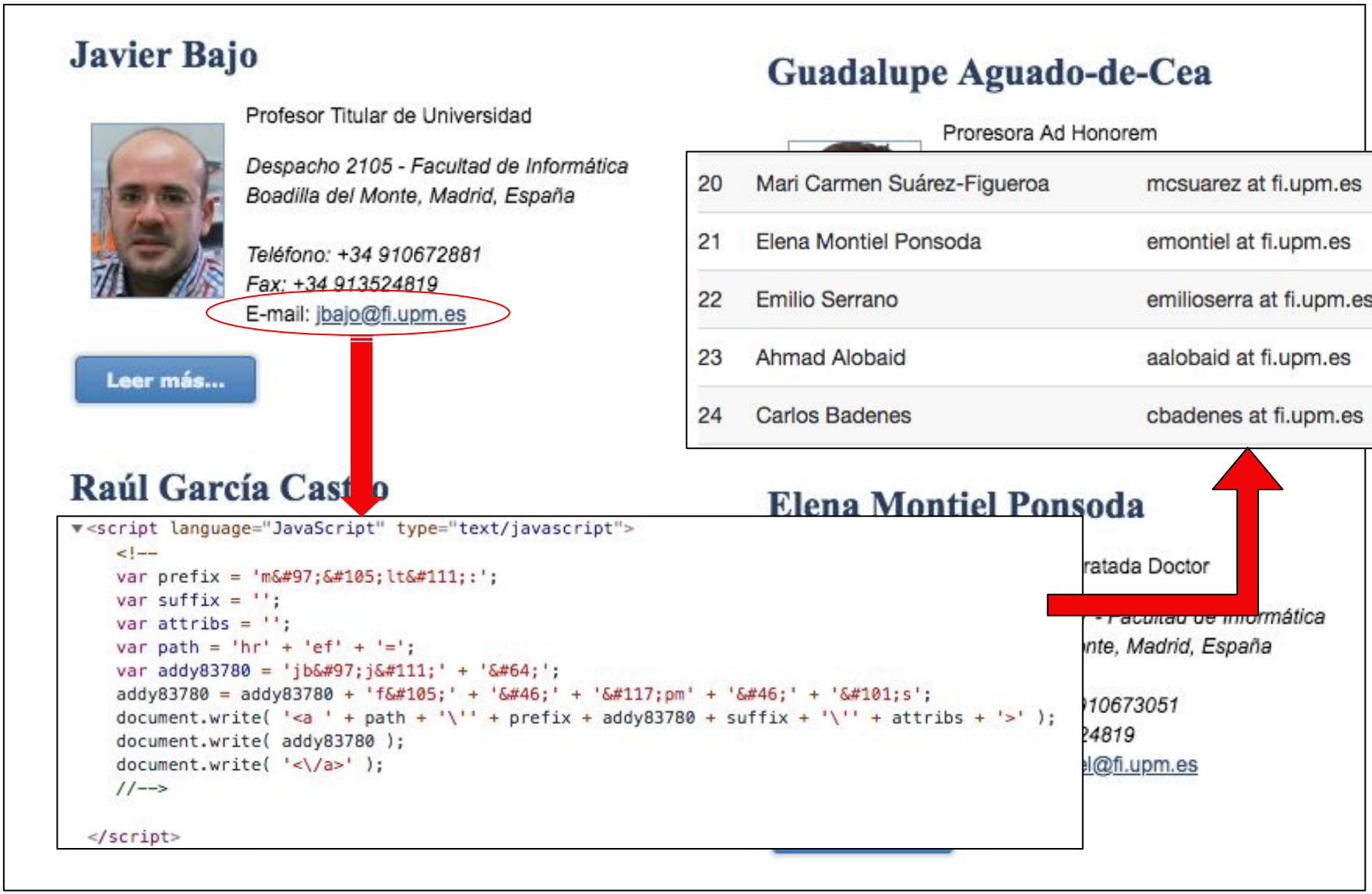
- Twitter API requires credentials
  - Our connector passes them as argument in the specification

```
{
    "id" : "Twitter asungomezperez",
    "type" : "JsonDatasource",
    "refresh" : "300000",
    "arguments" : ["$.tweets.[*]"],
    "connector"  : {
     "arguments" :["65UsI12RvUVH”,"uHUQcp9YXP”, "100880-QcvtT3”,
                          ”o4SZmiRfTh6","asungomezperez","100"],
     "type" : "TwitterConnector",
    }
  }
```

- Twitter API has a limitation of the number of calls
  - Our specification updates the data asyncronously from user requests

```
{
    "id" : "Twitter asungomezperez",
    "type" : "JsonDatasource",
    "refresh" : "300000",
    "arguments" : ["$.tweets.[*]"],
    "connector"  : {
     "arguments" :["65UsI12RvUVH”,"uHUQcp9YXP”, "100880-QcvtT3”,
                          ”o4SZmiRfTh6","asungomezperez","100"],
     "type" : "TwitterConnector",
    }
  }
```

- Data cleaning and transformation



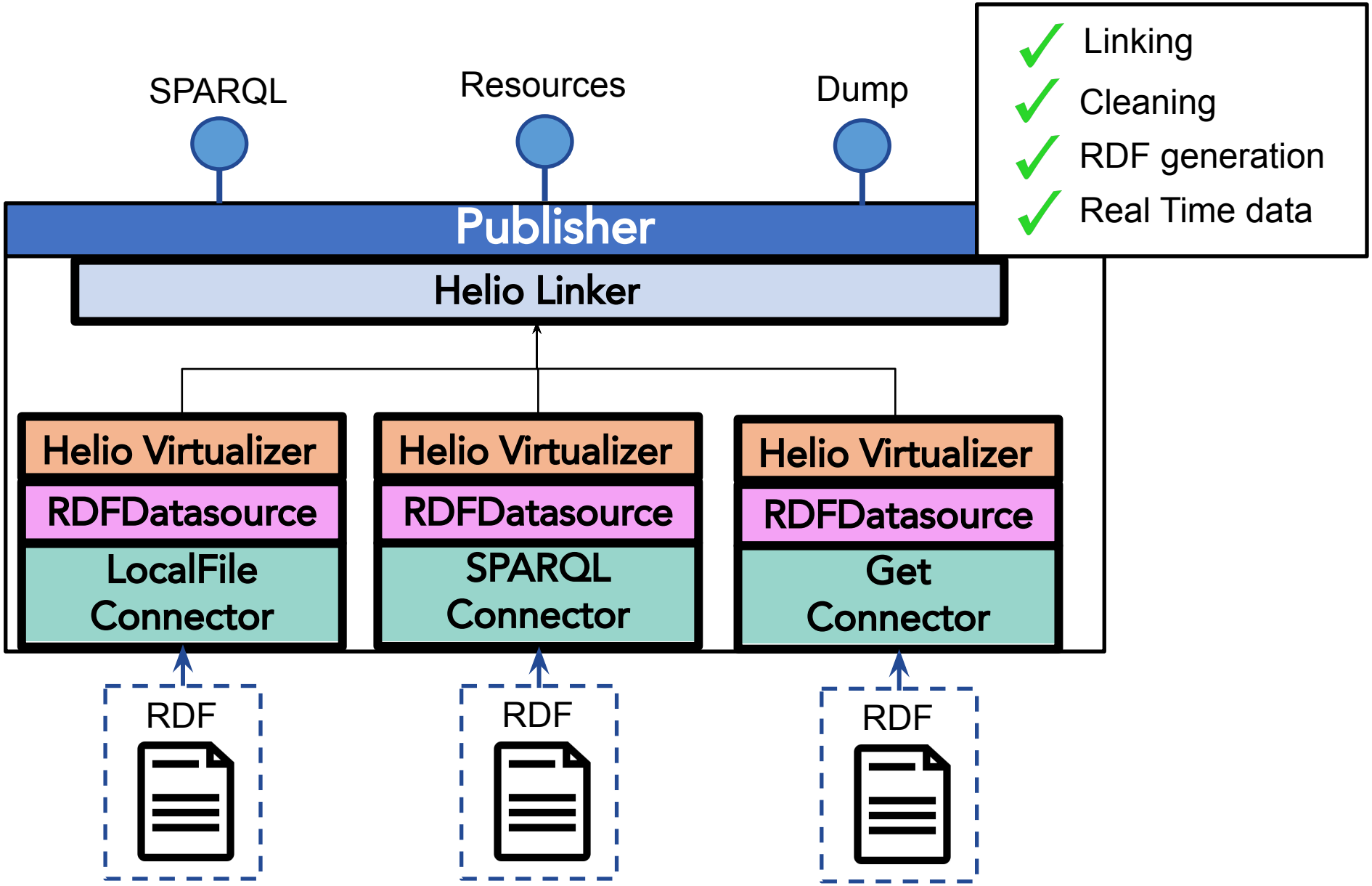
- Relate the author name in a tweet with his/her name in the OEG web
- **In addition we had to cope with Twitter API request limitations**

Equality of names does not solve this problem, Fuzzy rules required

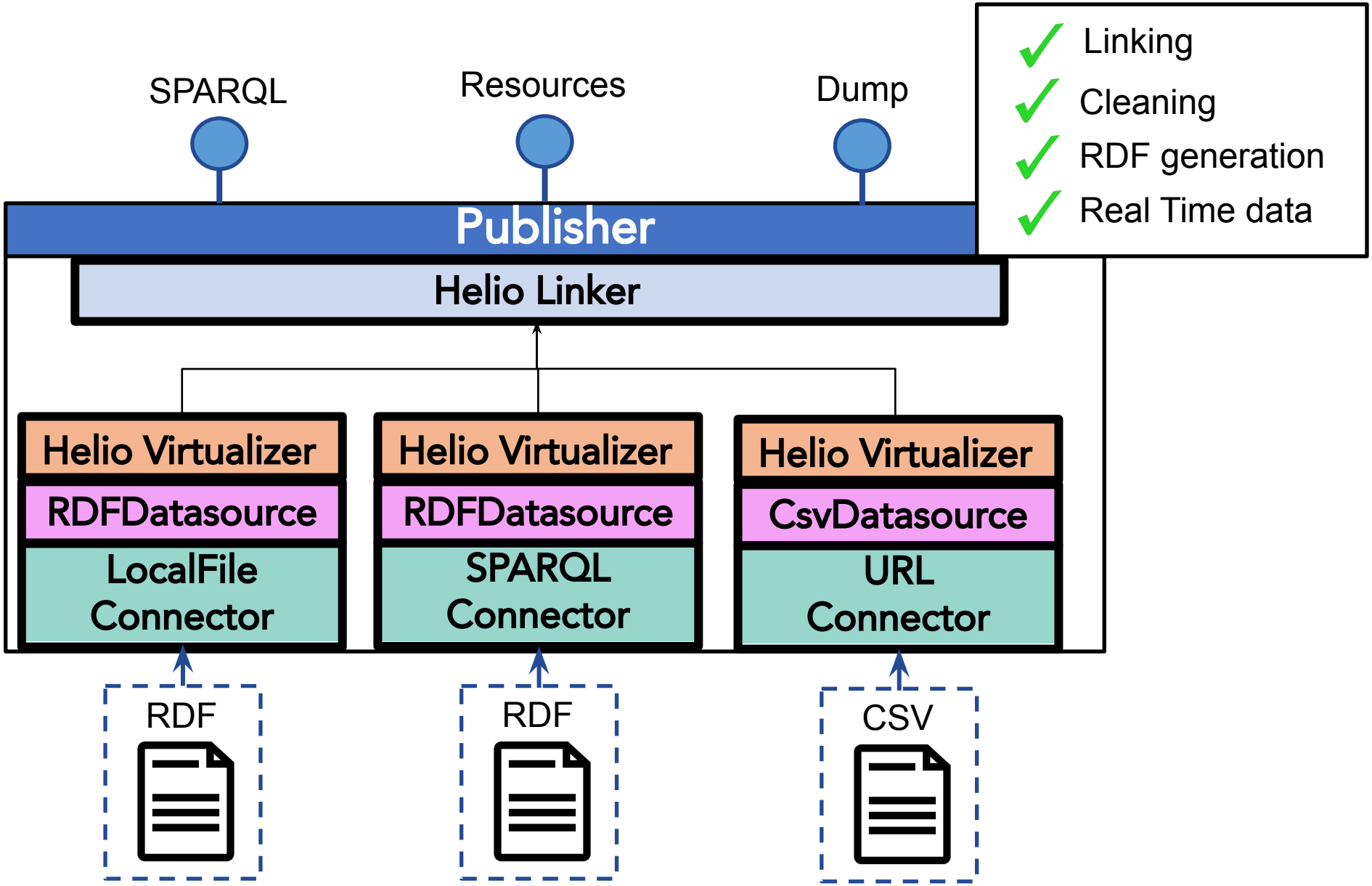
Showing 1 to 20 of 20 entries (in 1.584 seconds)

	tweetAuthor	tweetText	oegMember	oegWebPageName
1	Esteban González	La nebulosa Roseta en Ha OIII Luminancia y RGB (HaOIIILRGB). Esta imagen es el resultados de la unión de banda estrecha y L…	http://helio.linkeddata.es/oeg/people/Esteban%20Gonzalez%20Guardia	Esteban González Guardia
2	Esteban González	Small improvised @FossaSys stand at the @T3chFest conference. Great contacts and connections made! https://t.co/y3wQ74wY3M	http://helio.linkeddata.es/oeg/people/Esteban%20Gonzalez%20Guardia	Esteban González Guardia
3	Esteban González	Colabora, un proyecto de @Ayudame3D para mejorar la vida de las personas. Ilusión y ciencia. Buena mezcla. #AYUDAME3D #learnwith-t3chfest https://t.co/dKou42HQEO	http://helio.linkeddata.es/oeg/people/Esteban%20Gonzalez%20Guardia	Esteban González Guardia

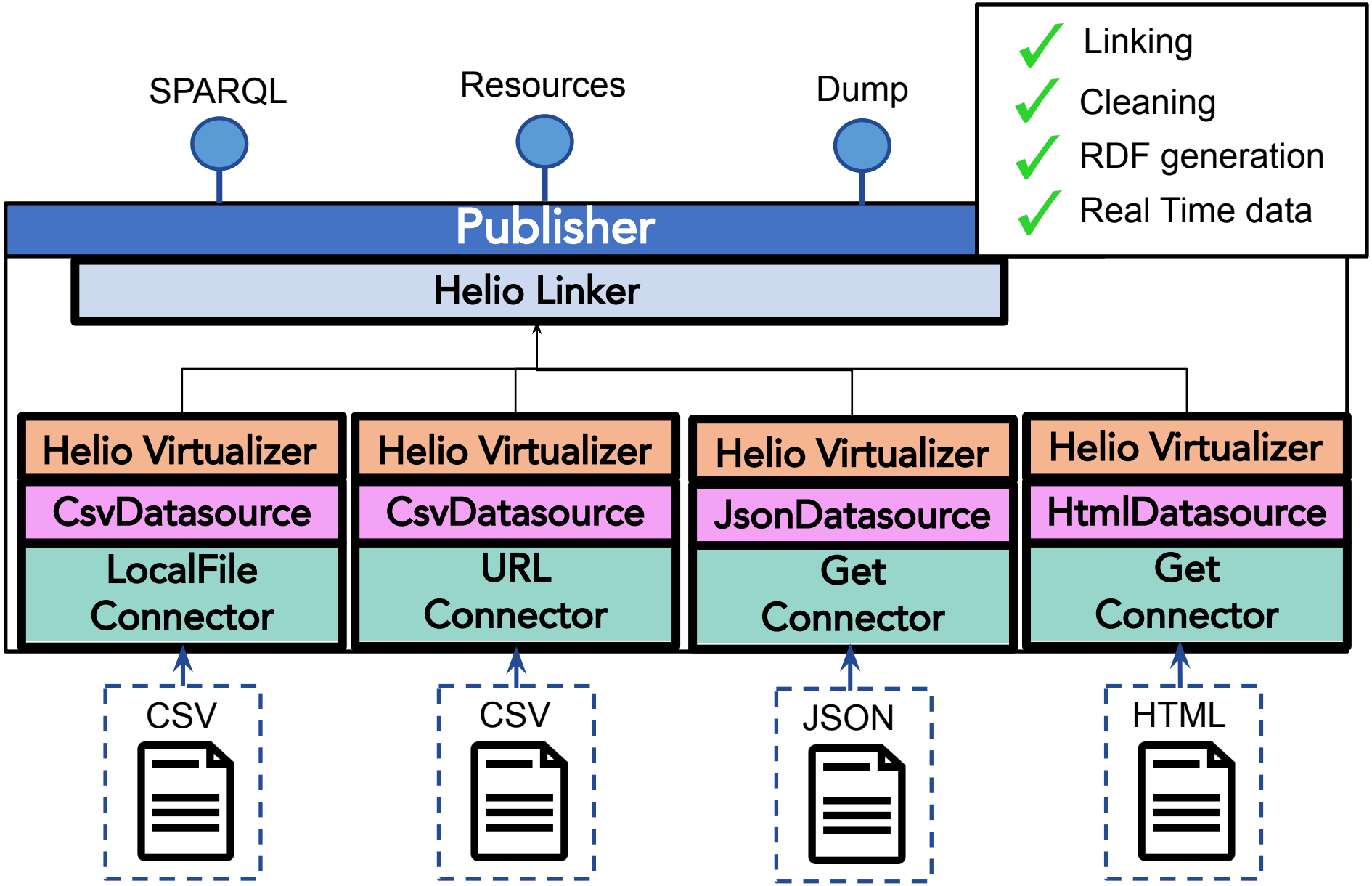
Helio Scenario 1: Refining RDF



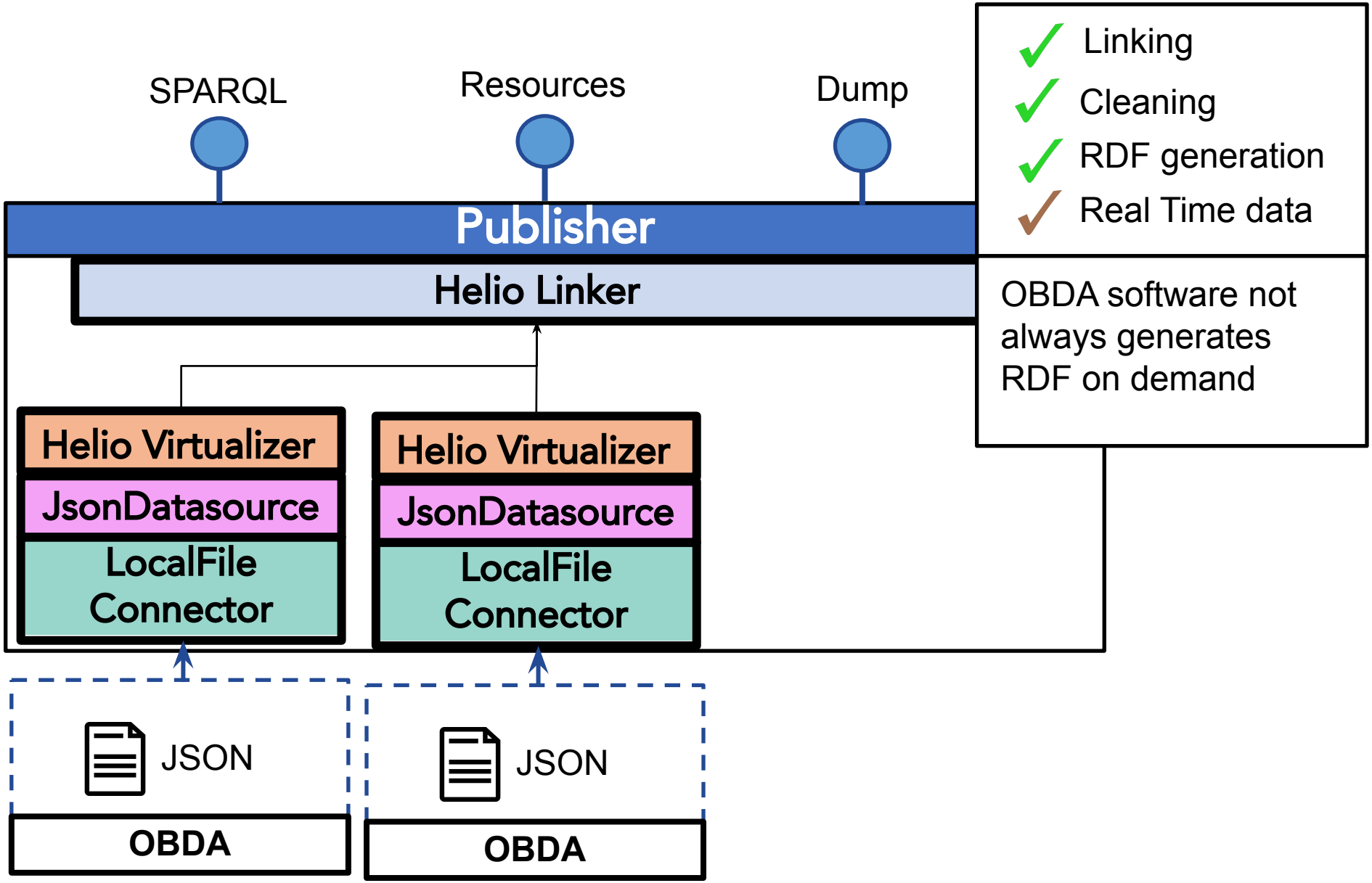
Helio Scenario 2: Enhancing RDF



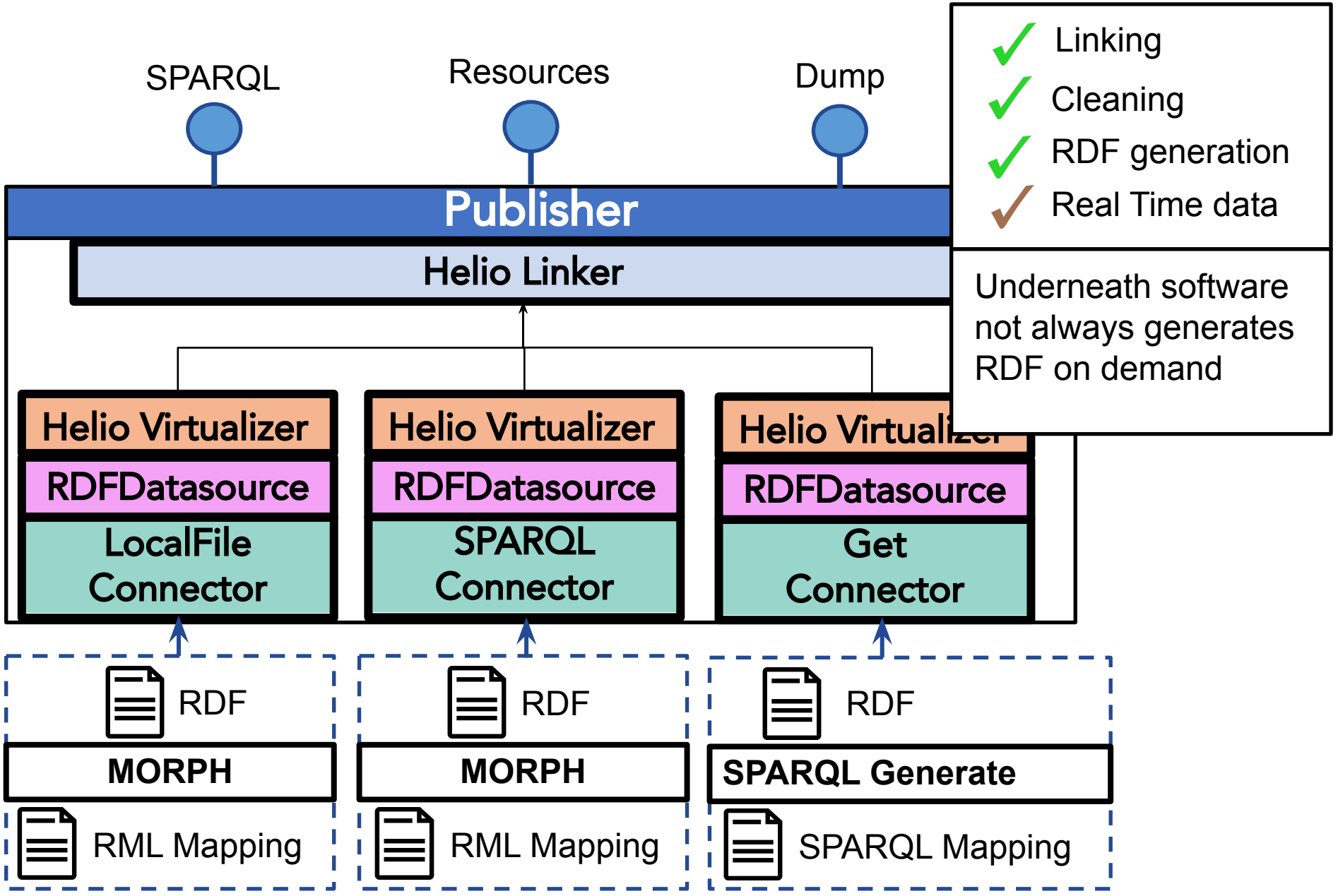
Helio Scenario 3: Data Integration



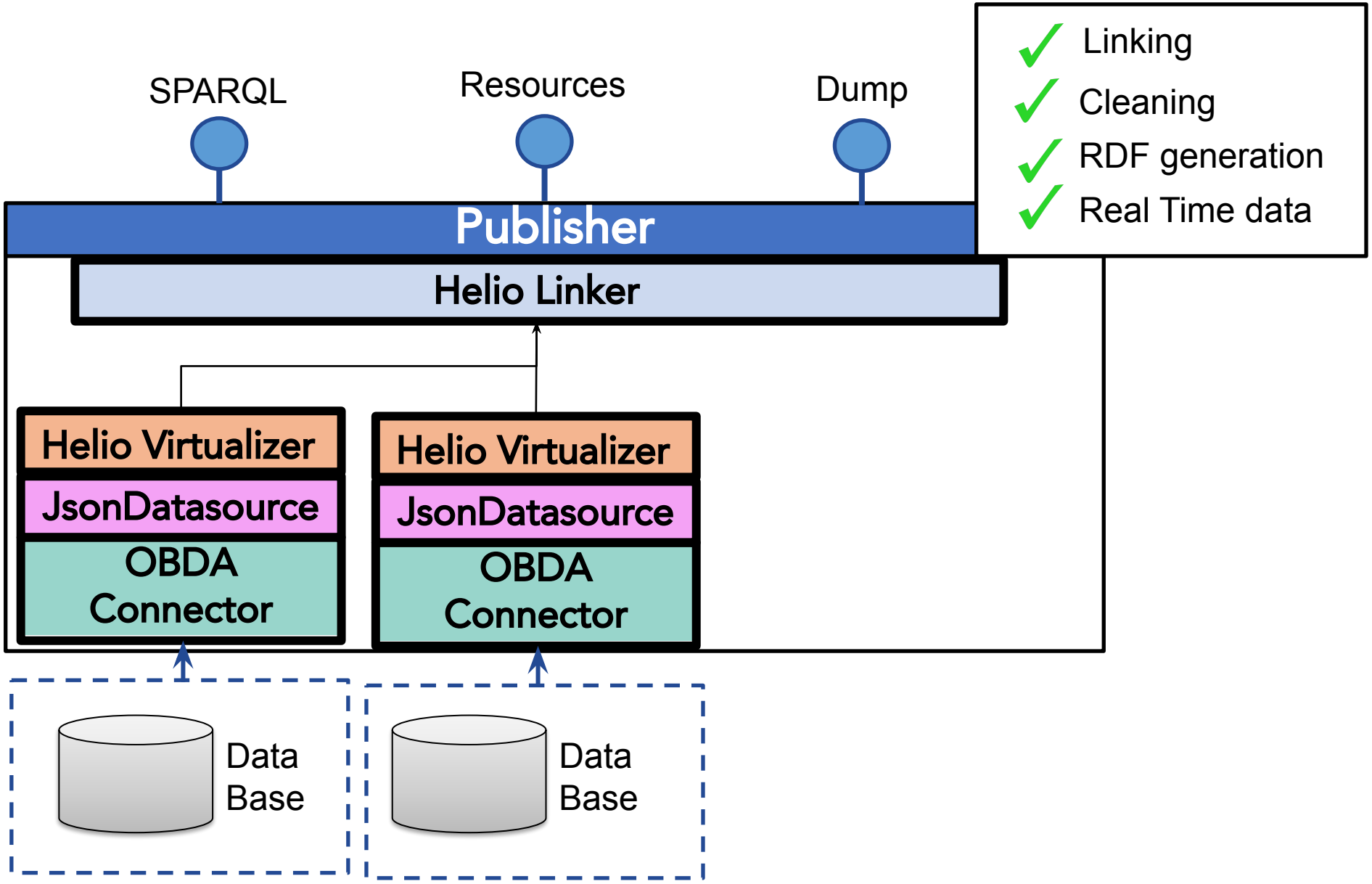
Helio Scenario 4: OBDA



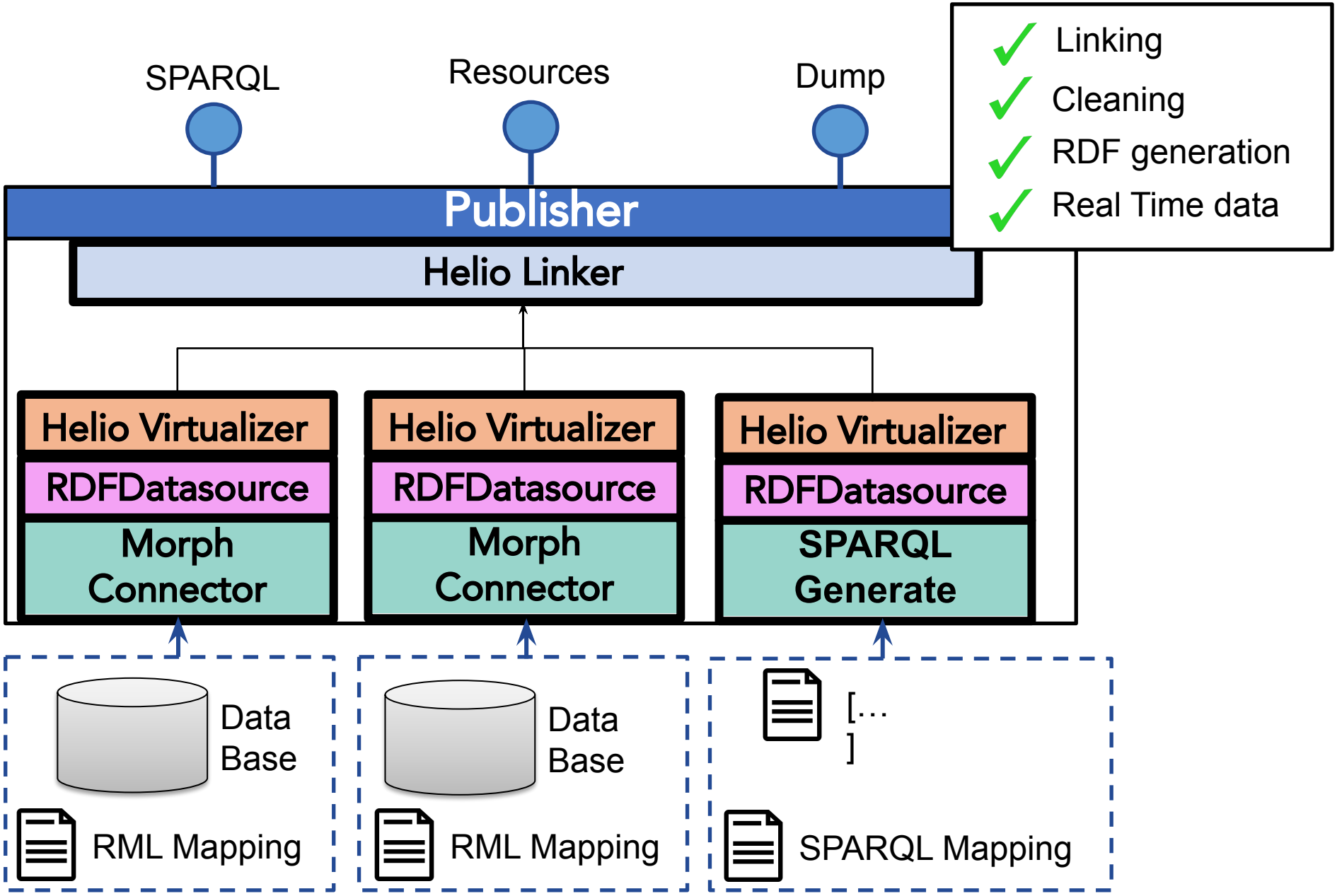
Helio scenario 5: Using third-part RDF generators



Helio Scenario 6: Integrating OBDA



Helio scenario 7: Integrating third-part RDF generators




QUESTIONS


2
He
Helio

VKG 2019

Generating and querying (Virtual) Knowledge Graphs from heterogeneous data sources

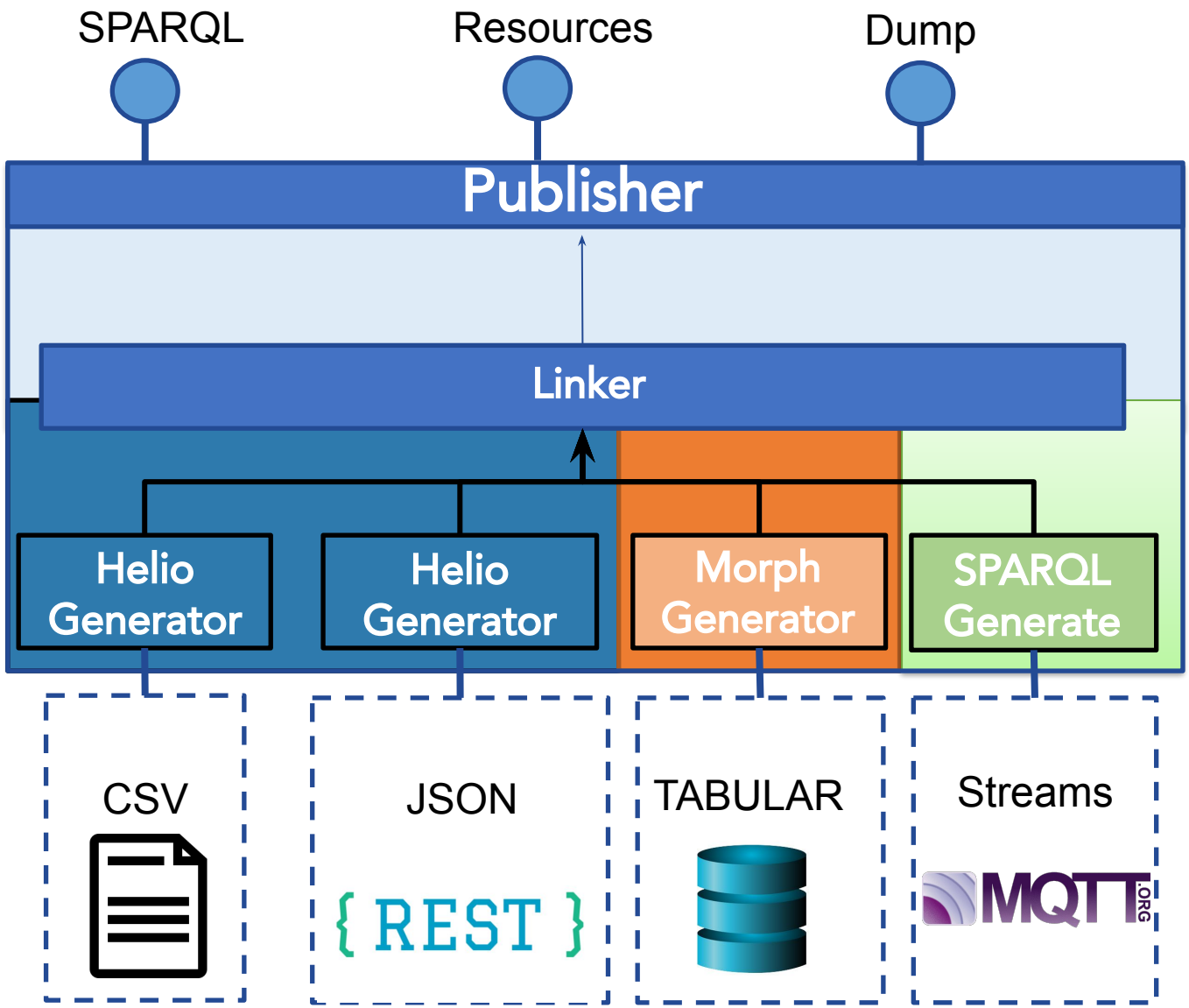
David Chaves Fraga, Ahmad Alobaid, Freddy Priyatna,
Andrea Cimmino, Oscar Corcho

Ontology Engineering Group
Universidad Politécnica de Madrid

dchaves@fi.upm.es
@oeg-upm

02/06/2019
ESWC2019 - Potoroz

One Helio to gather them all



Helio current status

Specification	Elements	Implemented
Helio Specification	Connector	FederatedSparqlConnector, GetConnector, LocalFileConnector, TwitterConnector, URLConnector, RMLConnector
	Datasource	CsvDatasource, HtmlDatasource, JsonDatasource, RDFDatasource, TextDatasource, XmlDatasource, JsonSpotlightDatasource
	Translator	Helio Virtualizer
	Linking	Helio Linker
RML* (proof of concept)	Connector	LocalFileConnector
	Datasource	JsonDatasource
	Translator	Helio Virtualizer
	Linking	Helio Linker

Acknowledgement

This work has been partially funded by:

DATOS 4.0:

"RETOS Y SOLUCIONES - UPM Spanish National Project (TIN2016-78011-C4-4-R)"
and FPI grant (BES-2017-082511)."

VICINITY project:

"VICINITY project (H2020-688467),funded by the European Commission Directorate-General for Research and Innovation, under the ICT-30 IoT action of its Horizon 2020 Research and Innovation Programme"

SPRINT project:

"Grant Agreement number: 826172 — SPRINT funded by H2020 program"