



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**



VKG 2019

Introduction

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

Ontology Engineering Group
Universidad Politécnica de Madrid

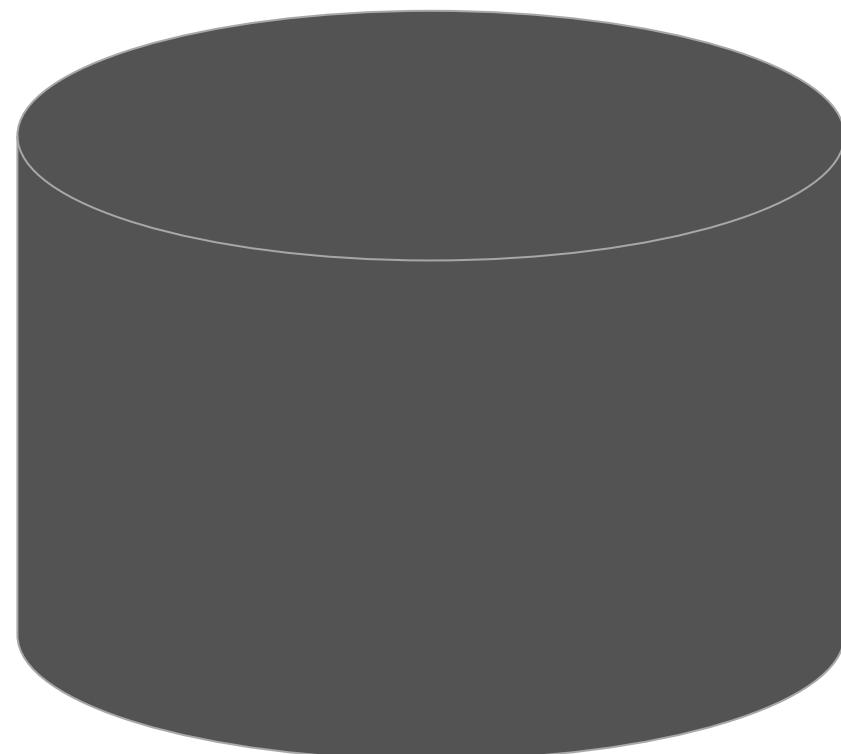
 dchaves@fi.upm.es

 @oeg-upm

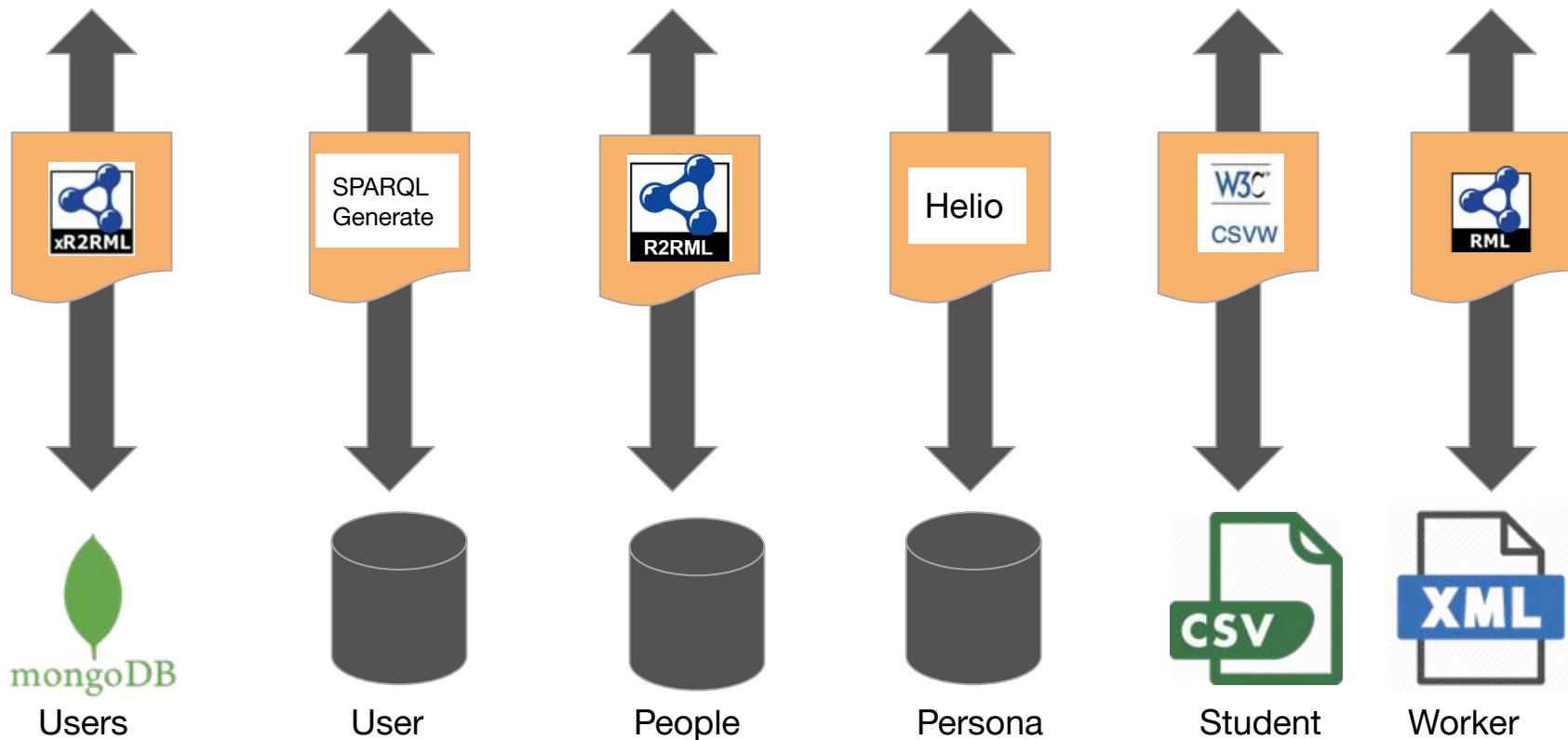
 02/06/2019

 ESWC2019 - Potoroz

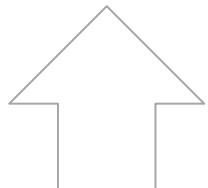
1. *git clone*
<https://github.com/oeg-upm/vkg-tutorial-eswc2019.git>
2. *Install java, docker and docker-compose*
 - a. <https://docs.docker.com/install/>
 - b. <https://docs.docker.com/compose/install/>



Ontology Based Data Access



Person/1 a Person .
Person/1 name "David Chaves" .



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

| 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 |

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

TBLPerson

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>

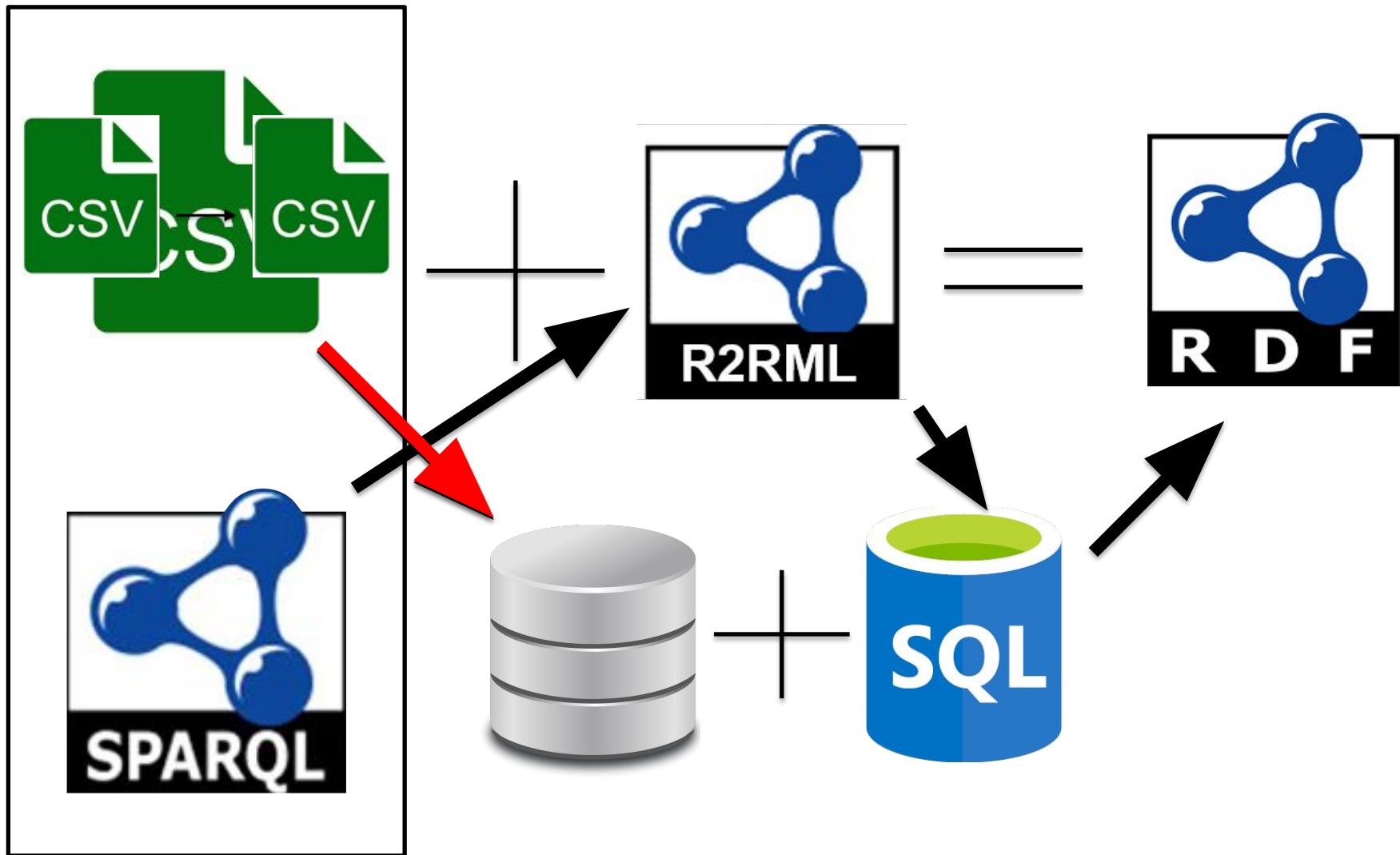


Morph-CSV

David Chaves-Fraga

Ontology Engineering Group
Universidad Politécnica de Madrid

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





Motivating Example and Problems

date, username, comment, modifiedDates,nOfLikes

comments.csv

"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

"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



people.csv

- No schema
- Not normalized
- Not uniform

- Missing data
- Missing meta-data
- No explicit joins

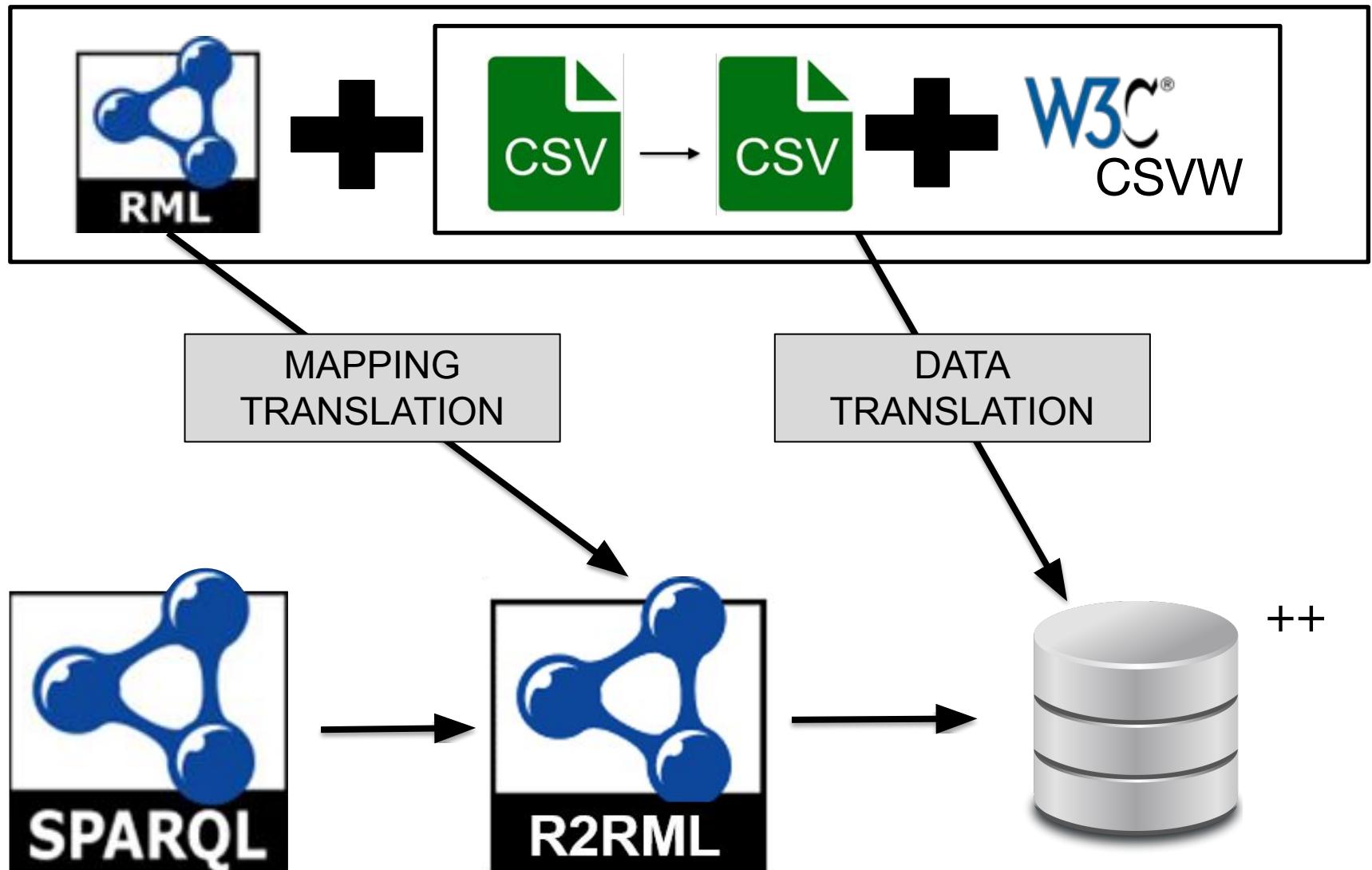


Can we reuse existing optimizations in SPARQL-to-SQL translation and R2RML mappings for querying heterogeneous CSVs?

- 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



W3C®
CSVW

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

comments.csv

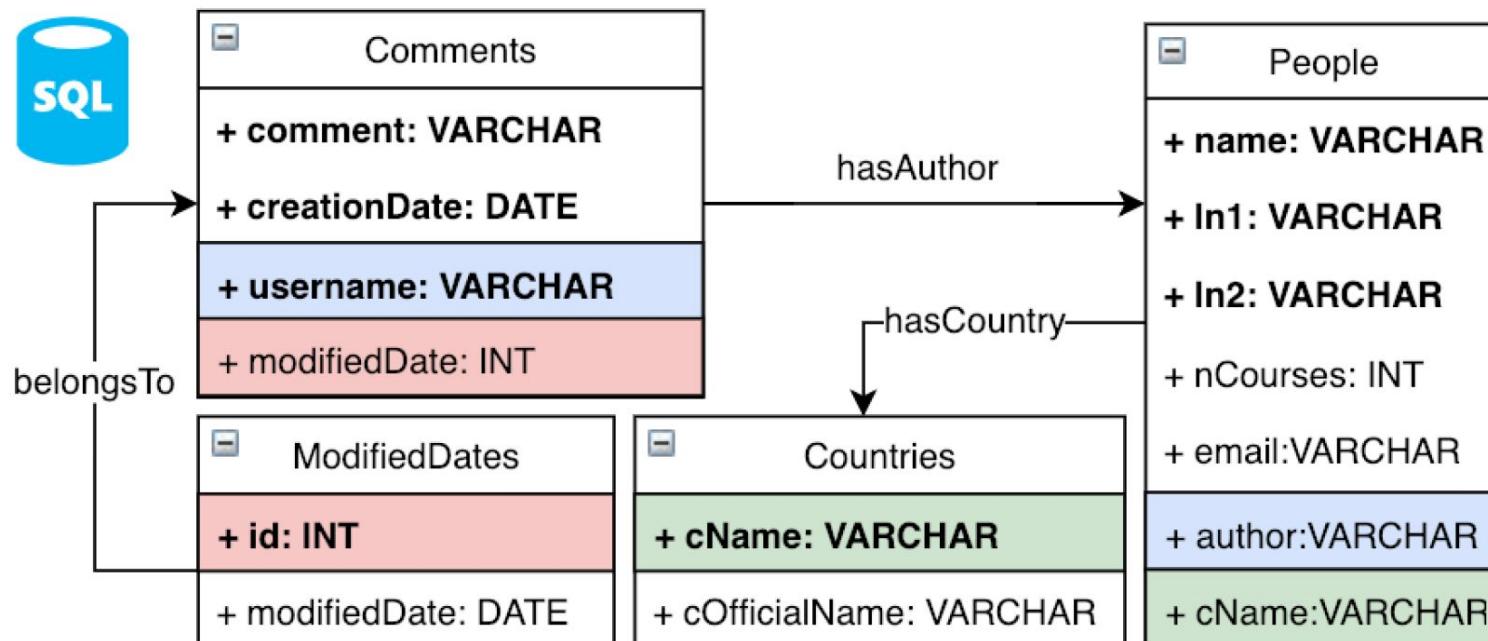
CSV



"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

people.csv

CSV



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",  
       "separatator": "-",  
       "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

```
mappings:  
publication:  
source:  
- [comment.csv~csv]  
s: http://ex.org/Comment/${(date)}${(username)}${(comment)}  
po:  
- [a, schema:SocialMediaPosting]  
- [schema:comment, ${(comment)}]  
- [schema:dateCreated, ${(date)}]  
- [schema:dateModified, ${(modifiedDates)}]  
- p: schema:author  
o:  
mapping: person  
condition:  
function: equal  
parameters:  
- [str1, ${(username)}]  
- parameter: str2  
value:  
function: sql:concat  
parameters:  
- parameter: sql:valueParameter  
value:  
function: sql:substring  
parameters:  
- [sql:valueParameter, ${(name)}, o]  
- [sql:valueParameter4, "1"]  
- [sql:valueParameter5, "1"]  
- [sql:valueParameter2, ${(ln1)}, o]
```



```
<publication>  
rr:logicalTable [  
rr:tableName "comments";  
];  
rr:subjectMap [  
rr:template "http://ex.org/Comment/{DATE}{USERNAME}{COMMENT}";  
rr:class <http://schema.org/SocialMediaPosting>;  
];  
rr:predicateObjectMap [  
rr:predicateMap [ rr:constant <http://schema.org/comment> ];  
rr:objectMap[ rr:column "comment"; ];  
];  
rr:predicateObjectMap [  
rr:predicateMap [ rr:constant <http://schema.org/author> ];  
rr:objectMap [rr:parentTriplesMap <person>;  
rr:joinCondition [ rr:child "username"; rr:parent "author"; ];  
];  
rr:predicateObjectMap [  
rr:predicateMap [ rr:constant <http://schema.org/dateModified> ];  
rr:objectMap [ rr:parentTriplesMap <modifiedDates>;  
rr:joinCondition [ rr:child "modifiedDates"; rr:parent "id"; ];  
];  
rr:predicateObjectMap [  
rr:predicateMap [ rr:constant <http://schema.org/dateCreated> ];  
rr:objectMap[rr:column "date";];  
];  
.
```



Three examples (configs folder):

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

How to run it?

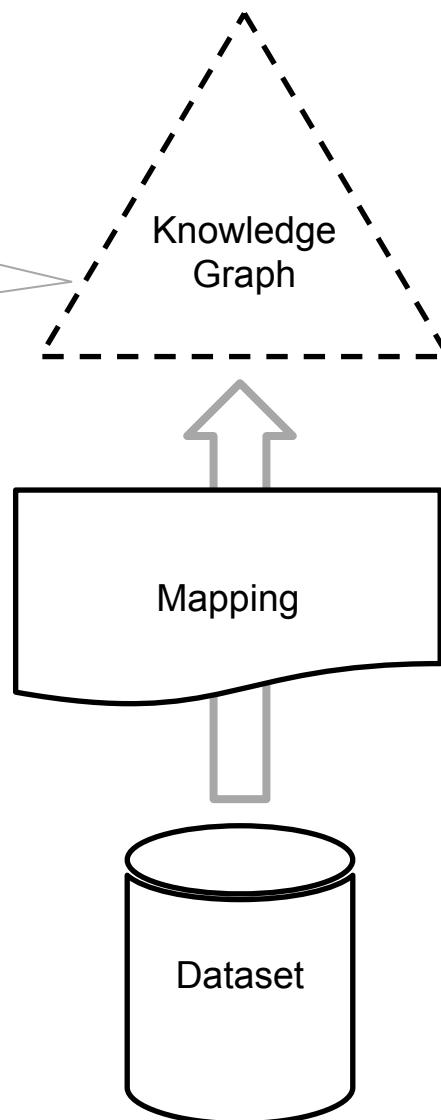
```
cd run-scripts  
./runTheScript.sh
```

morph-CSV

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

morph-GraphQL

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





Morph-GraphQL

Freddy Priyatna

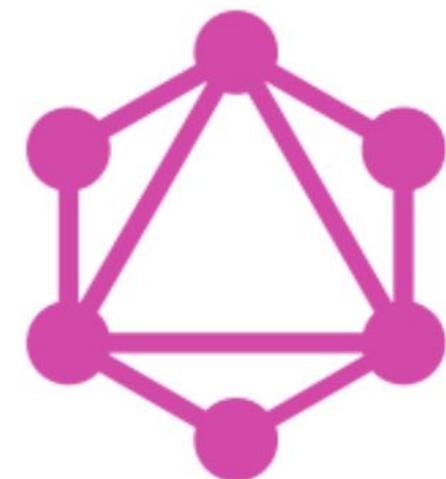
Ontology Engineering Group
Universidad Politécnica de Madrid

 fpriyatna@fi.upm.es

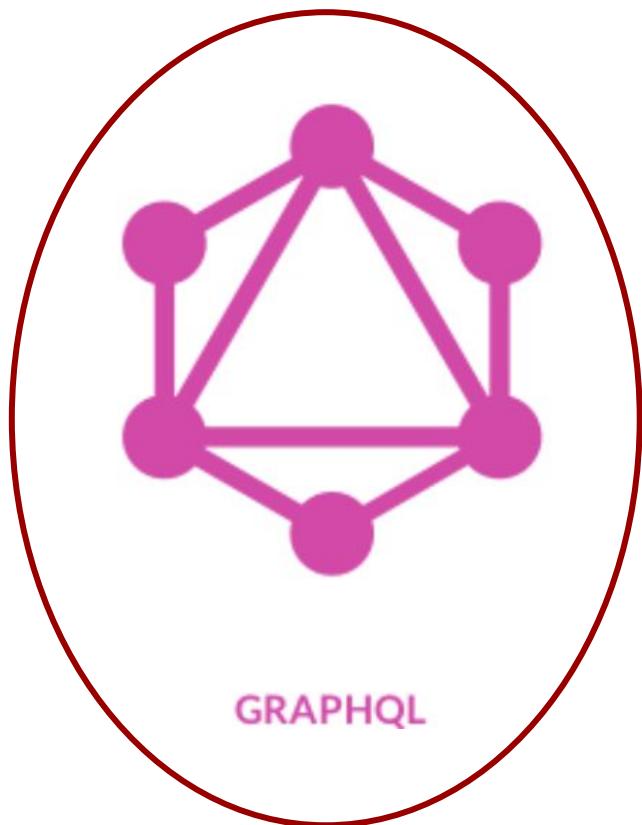
 [@fpriyatna](https://twitter.com/fpriyatna)

 02/06/2019

 ESWC2019 - Potoroz



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)

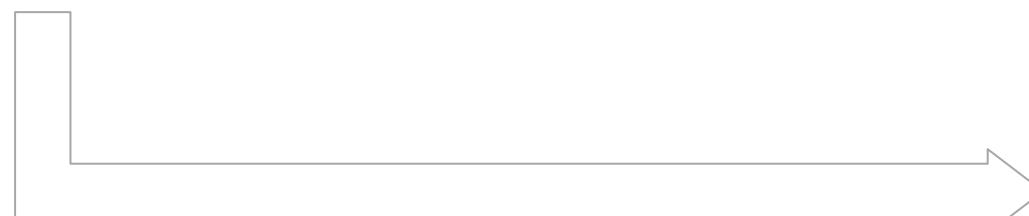
1. /users/<id>



2. /users/<id>/posts

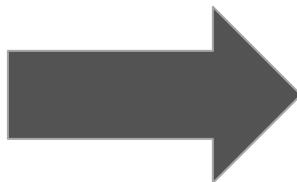


3. /users/<id>/followers



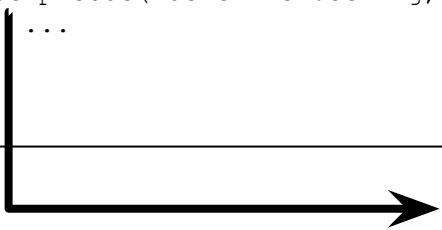
"Get the user's name + posts' titles + followers' names" (GraphQL)

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



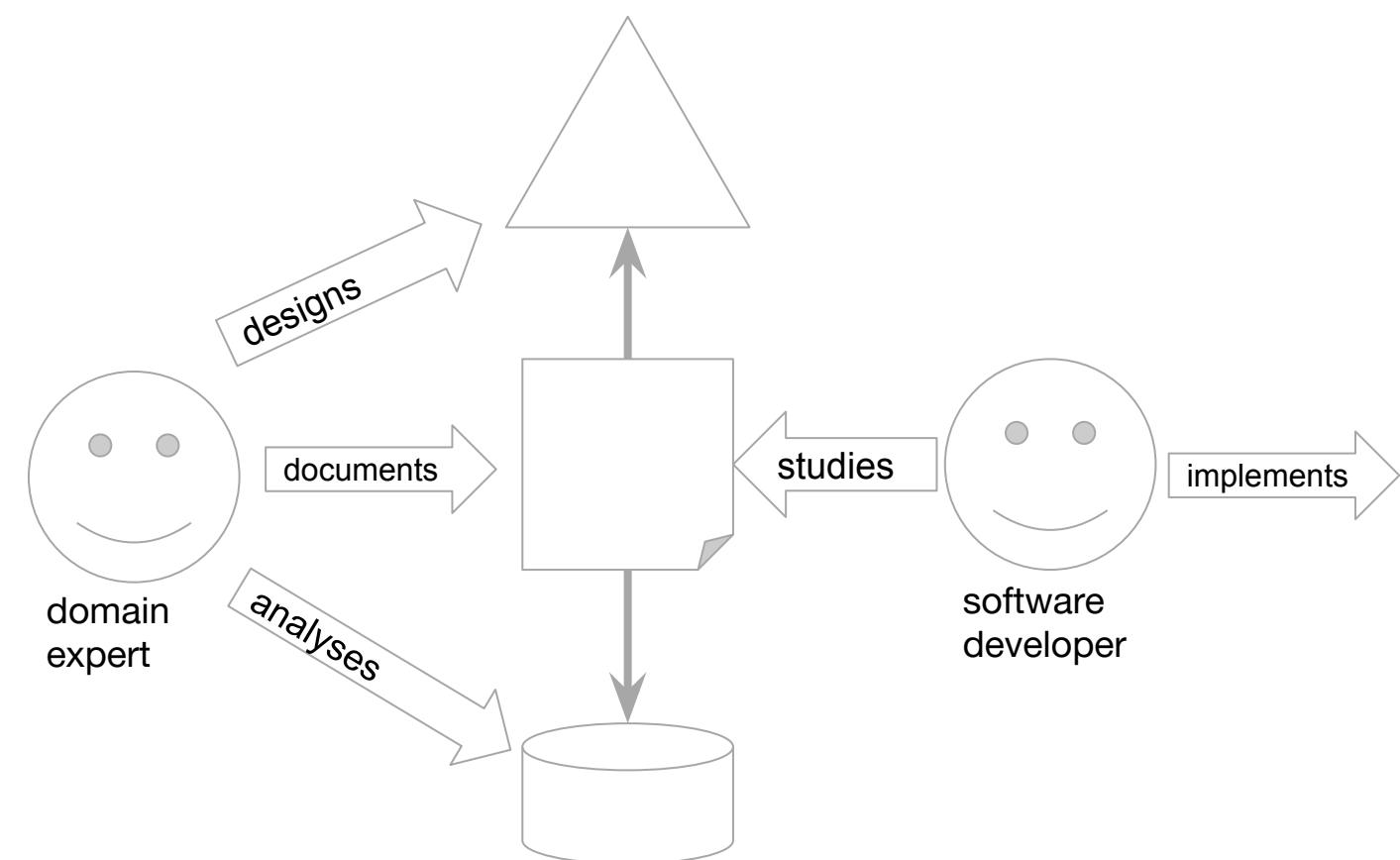
```
"data" {  
  "User": {  
    "name": "...",  
    "posts": [  
      { "title": ... }  
    ],  
    "followers": [  
      {"name": ...},  
      {"name": ...},  
      ...  
    ]  
  }  
}
```

```
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

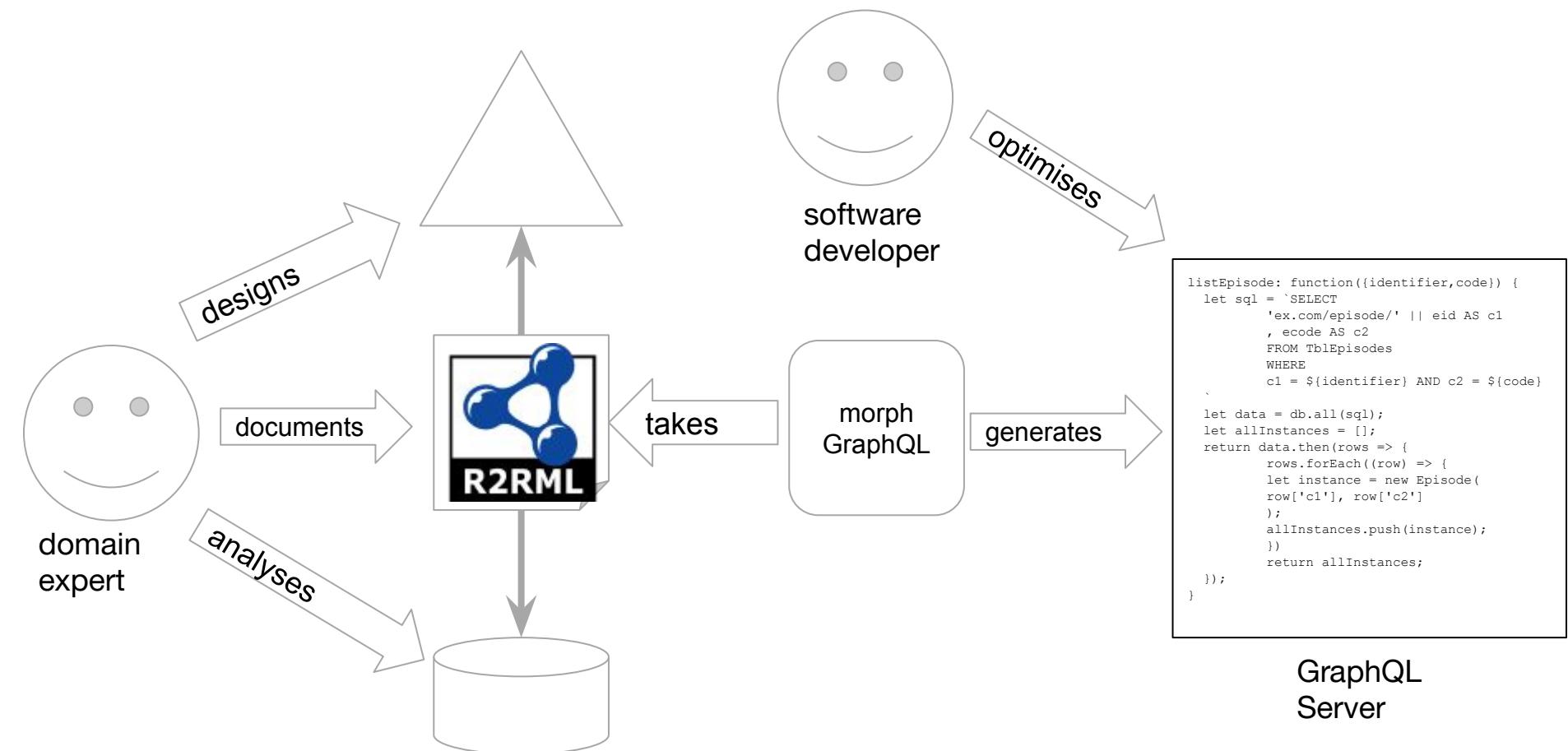


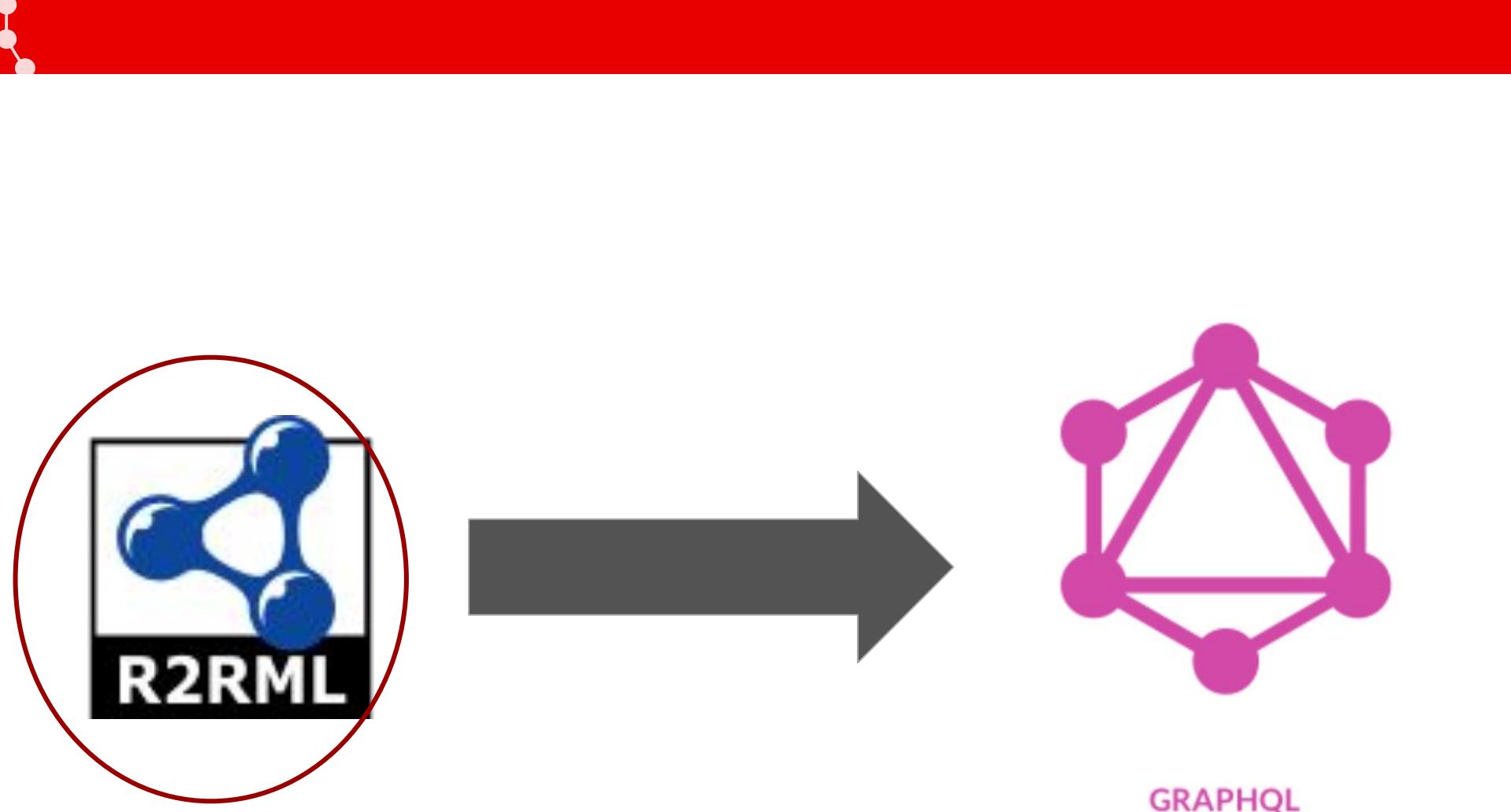
```
listEpisode: function({identifier,code}) {
  let sql = `SELECT
    'ex.com/episode/' || eid AS c1
    , ecode AS c2
  FROM TblEpisodes
  WHERE
  c1 = ${identifier} AND c2 = ${code}
  `

  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;
  });
}
```

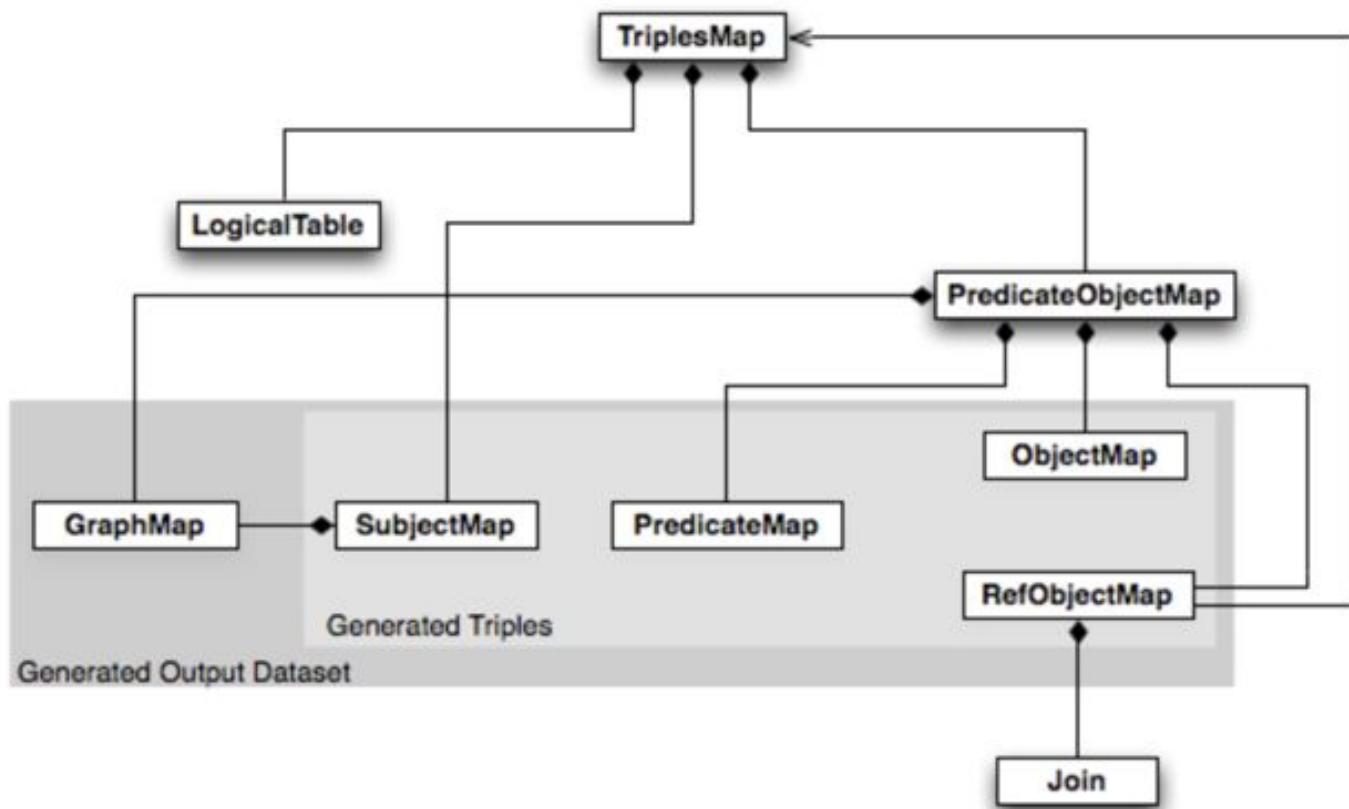
GraphQL
Server

Our Proposal





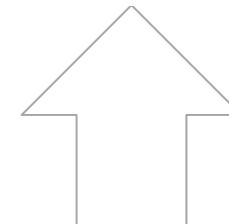
- W3C Recommendation
- RDB to RDF
- Declarative





Example output data

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

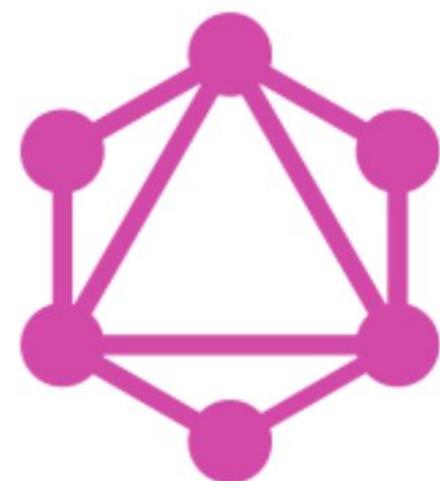
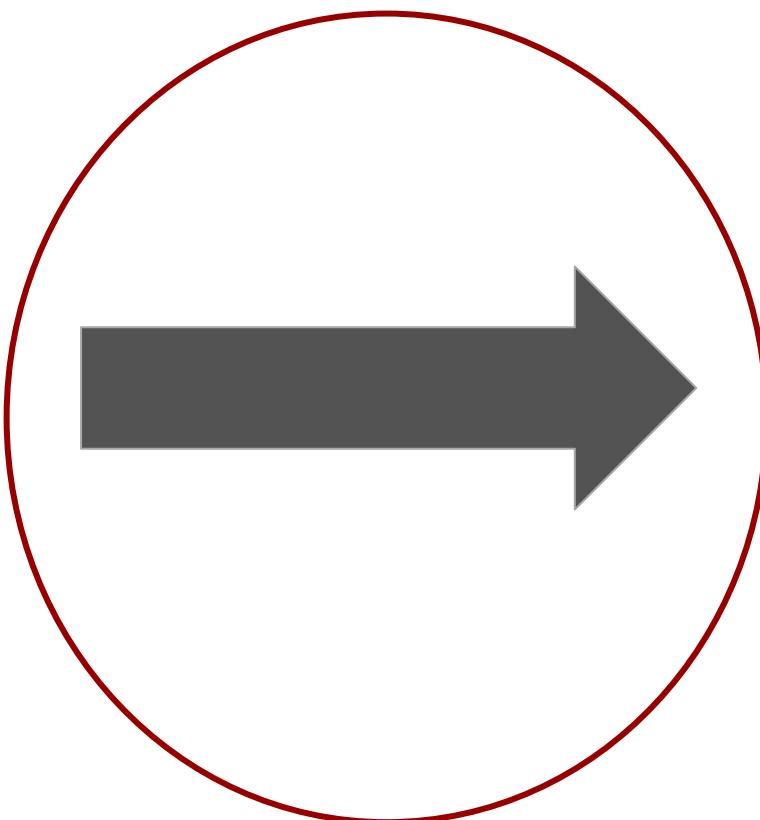


Example R2RML mapping

```
@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" ];  
    ].
```

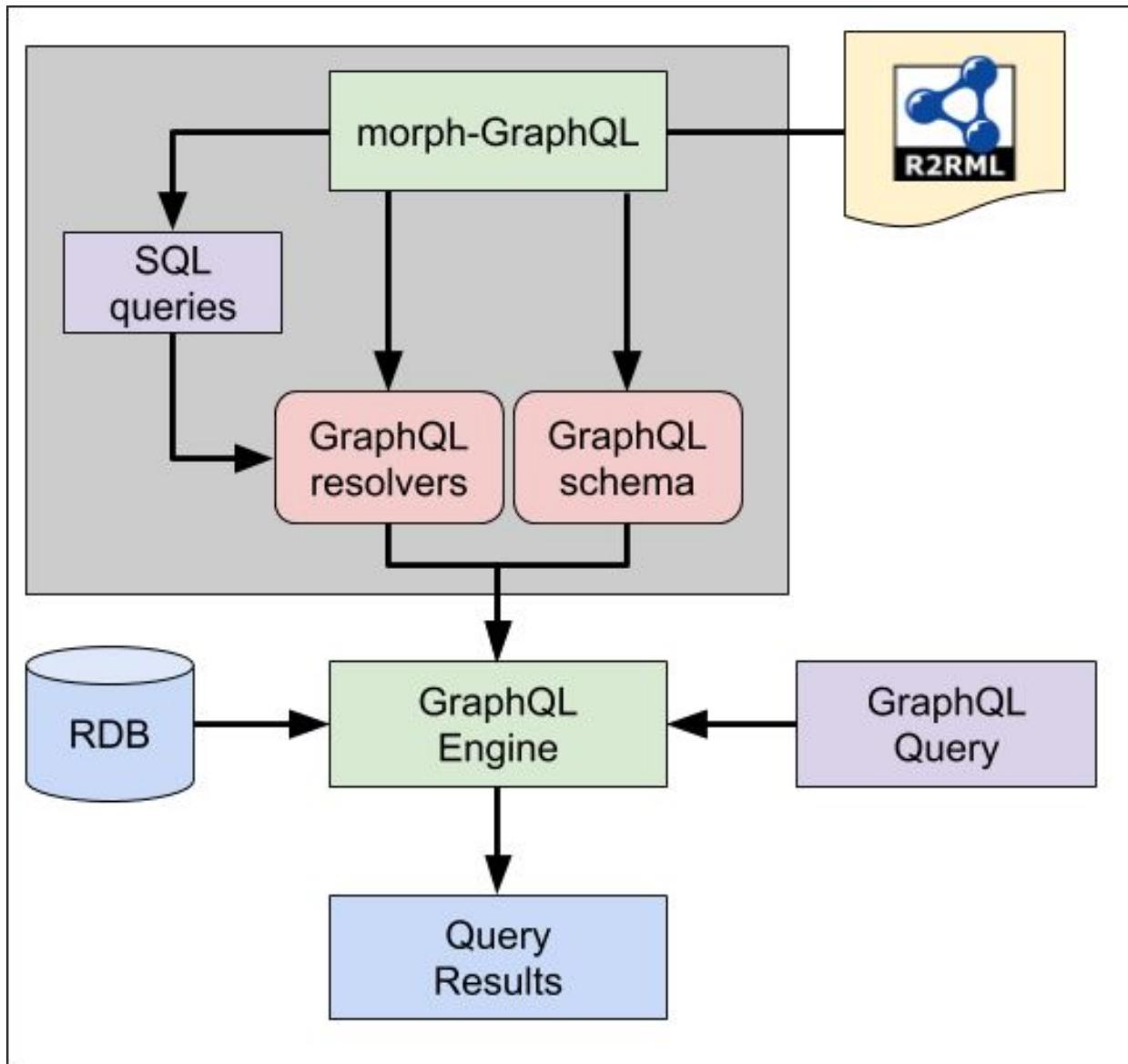
EMP

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



GRAPHQL

morph-GraphQL Workflow



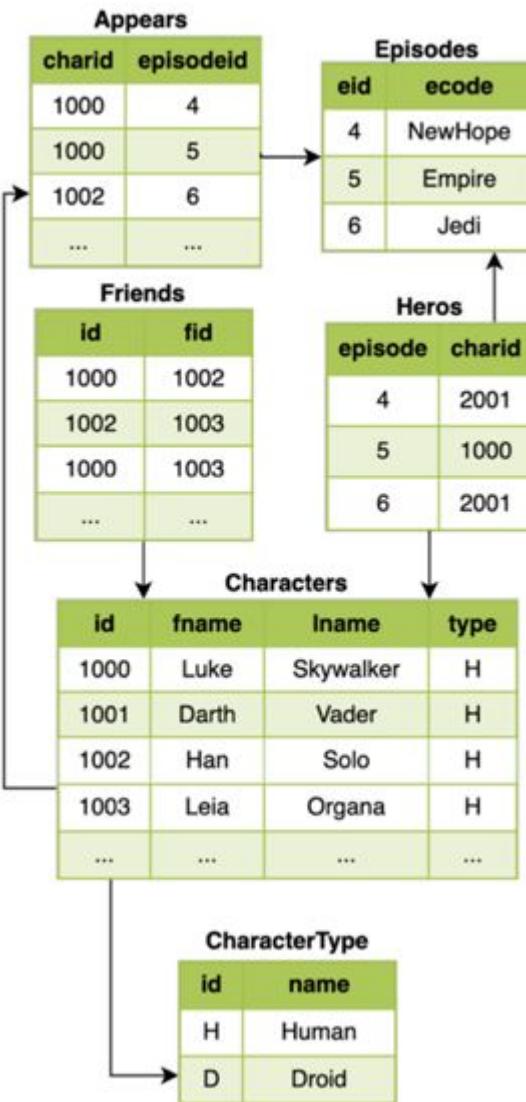
- Source code: <https://github.com/oeg-upm/morph-graphql>
- Deployment: <http://graphql.morph.oeg-upm.net>



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

```
1 {  
2   listHeroes {  
3     episode {  
4       identifier  
5       code  
6     }  
7     hero {  
8       identifier  
9       name  
10    }  
11  }  
12}  
  
{  
  "episode": [  
    {  
      "identifier": "http://starwars.mappingpedia.linkeddata.es/episode/4",  
      "code": "NewHope"  
    }  
  ],  
  "hero": [  
    {  
      "identifier": "http://starwars.mappingpedia.linkeddata.es/character/2001",  
      "name": "R2 D2"  
    }  
  ]  
},  
{  
  "episode": [  
    {  
      "id": 1  

```

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

GraphsiQL



Prettify

< Docs

```
1 {  
2   listCharacter(name: "R2 D2")  
3     identifier  
4     name  
5     friends {  
6       identifier  
7       charid  
8       friendId  
9     }  
10   }  
11 }  
  
{  
  "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

```
1 {  
2   listCharacter(  
3     identifier: "http://starwars.mappingpedia.linkeddata.es/character/1000")  
4   {  
5     name  
6   }  
7 }
```

```
{  
  "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/' || "listCharac".
```

Q4: to query Luke & Leia

GraphsiQL



Prettify

< Docs

```
1 query FetchLukeAndLeiaAliased {  
2   luke: listCharacter(  
3     identifier: "http://starwars.mappingpedia.linkeddata.es/character/1000")  
4   {  
5     name  
6   }  
7   leia: listCharacter(  
8     identifier: "http://starwars.mappingpedia.linkeddata.es/character/1003")  
9   {  
10     name  
11   }  
12 }
```

```
{  
  "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/' || "listCharac".
```

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

QUERY VARIABLES

Q5: to verify that D2-R2 is a droid

GraphsQL



Prettify

< Docs

```
1 {  
2   listCharacter(name: "R2 D2") {  
3     identifier  
4     name  
5     type(name: "Droid") {  
6       identifier  
7       name  
8     }  
9   }  
10 }
```

```
{  
  "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 "identifier",  
  '' || "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

GraphsQL



Prettify

< Docs

```
1 {  
2   listHeroes {  
3     identifier  
4     hero {  
5       identifier  
6       name  
7       type(name: "Human") {  
8         identifier  
9         name  
10      }  
11    }  
12    episode(code: "Empire") {  
13      identifier  
14      code  
15    }  
16  }  
17 }  
  
{  
  "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"  
  
QUERY VARIABLES
```

How to generate mappings?



TADA

Ahmad Alobaid

Ontology Engineering Group
Universidad Politécnica de Madrid

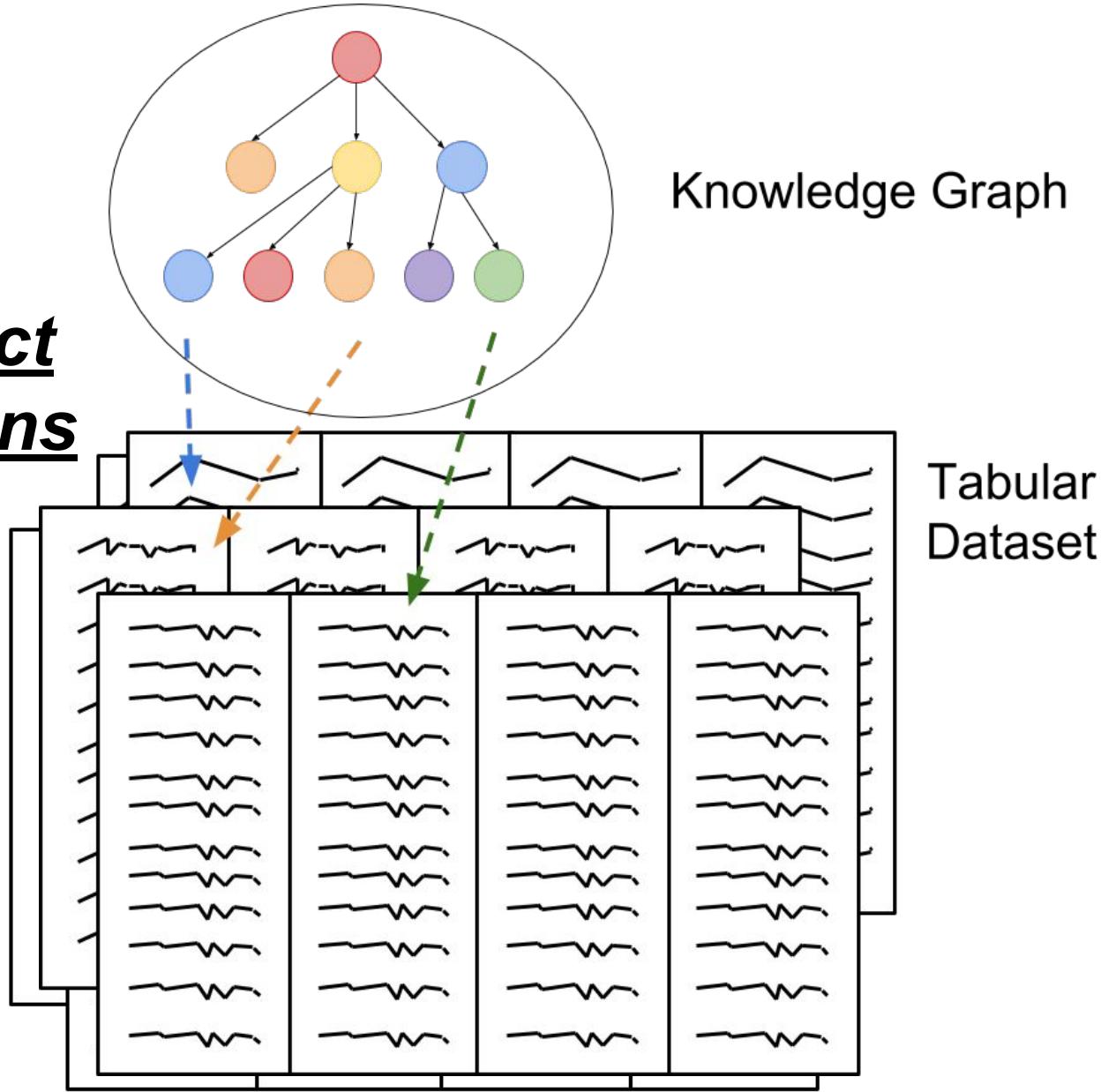
 aalobaid@fi.upm.es

 [@oeg-upm](https://twitter.com/oeg-upm)

 02/06/2019

 ESWC2019 - Potoroz

Subject Columns



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



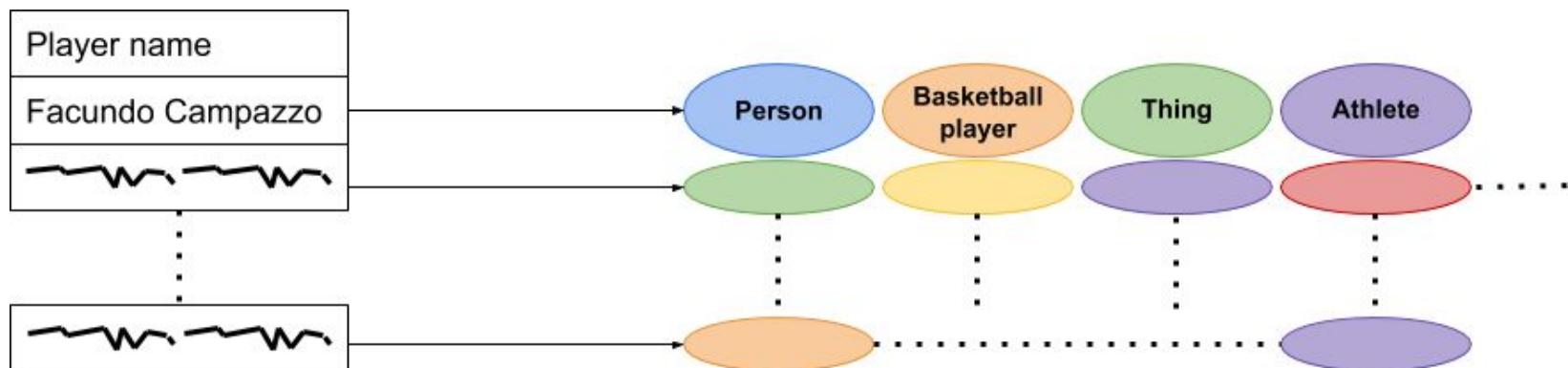
How? link cells to entities?

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?

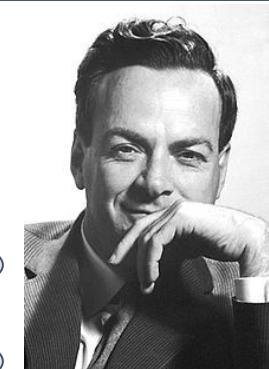
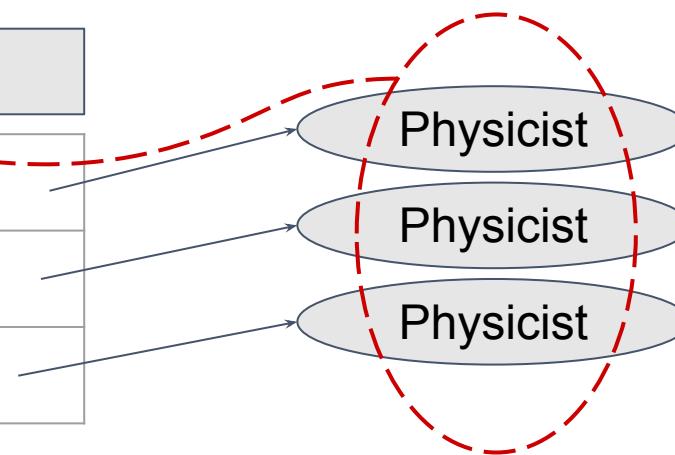


4GIFs.com



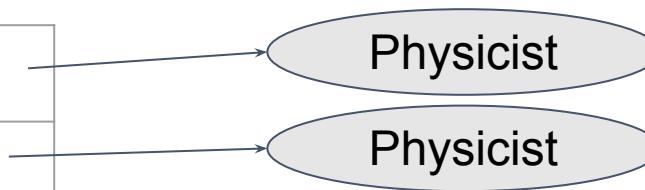
1. Obvious case:

| |
|-----------------|
| Physicist |
| Richard Feynman |
| Stephen Hawking |
| Max Planck |



2. Not-so-obvious case:

| |
|-----------------|
| Richard Feynman |
| Stephen Hawking |
| Brian May |



<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

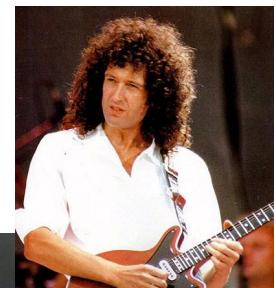
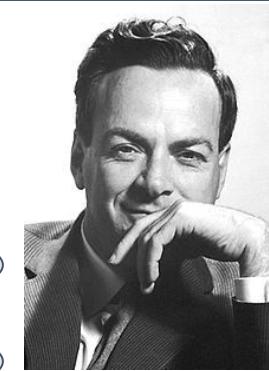
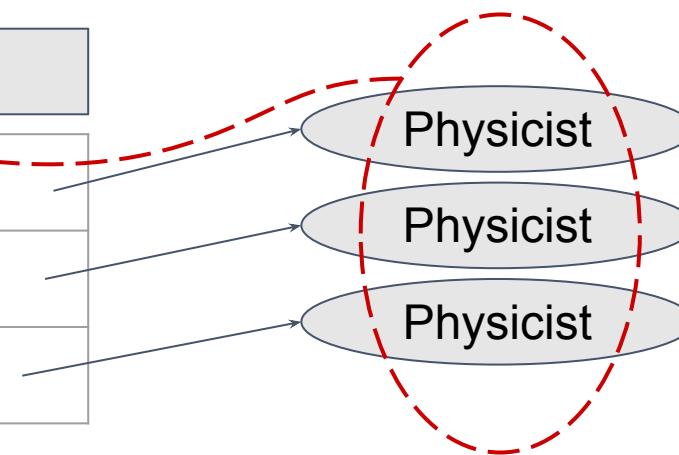
<https://www.queenie.cz/storage/temp/8f2b24db44d09bd8e3d563d0bb099fc2-400x800x1.jpg>

<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>



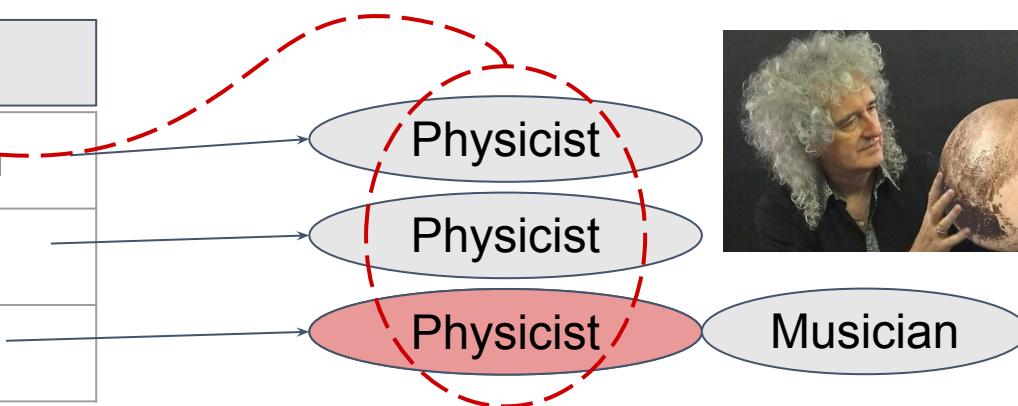
1. Obvious case:

| |
|-----------------|
| Physicist |
| Richard Feynman |
| Stephen Hawking |
| Max Planck |



2. Not-so-obvious case:

| |
|-----------------|
| Physicist |
| Richard Feynman |
| Stephen Hawking |
| Brian May |



<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>

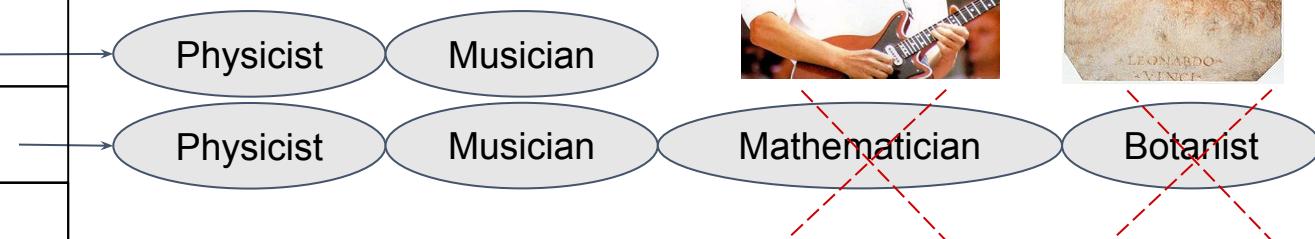
<https://www.thetimes.co.uk/imageserver/image/methode%2Ftimes%2Fprod%2Fweb%2Fbin%2F3f44abc8-2774-11e8-acc5-262aff1ca7a6.jpg?crop=988%2C556%2C910%2C214&resize=685>

<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

| |
|-------------------|
| ? |
| Brian May |
| Leonardo da Vinci |
| Pharrell Williams |



<https://cdn.britannica.com/s:300x300/73/20973-004-F71E20CB.jpg>

https://upload.wikimedia.org/en/thumb/4/42/Richard_Feynman_Nobel.jpg/220px-Richard_Feynman_Nobel.jpg

<https://www.queenie.cz/storage/temp/8f2b24db44d09bd8e3d563d0bb099fc2-400x800x1.jpg>

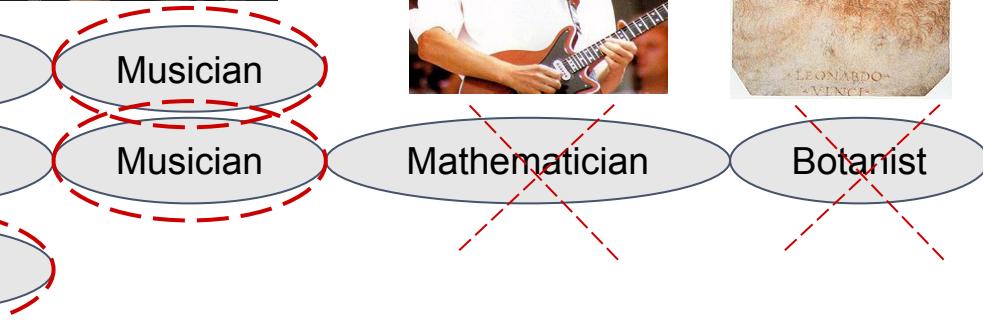
<https://www.thetimes.co.uk/imageserver/image/methode%2Ftimes%2Fprod%2Fweb%2Fbin%2F3f44abc8-2774-11e8-acc5-262aff1ca7a6.jpg?crop=988%2C556%2C910%2C214&resize=685>

<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 |
| Leonardo da Vinci | Physicist |
| Pharrell Williams | Musician |



<https://cdn.britannica.com/s:300x300/73/20973-004-F71E20CB.jpg>

https://upload.wikimedia.org/en/thumb/4/42/Richard_Feynman_Nobel.jpg/220px-Richard_Feynman_Nobel.jpg

<https://www.queenie.cz/storage/temp/8f2b24db44d09bd8e3d563d0bb099fc2-400x800x1.jpg>

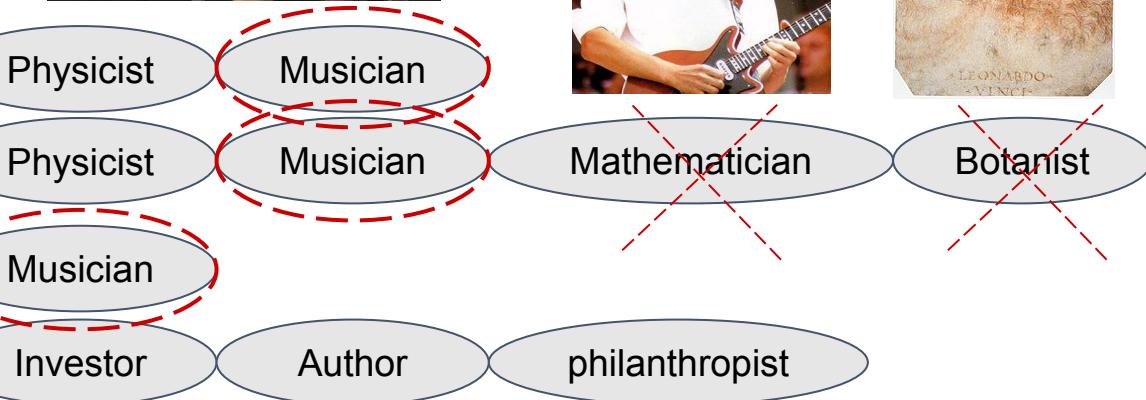
<https://www.thetimes.co.uk/imageserver/image/methode%2Ftimes%2Fprod%2Fweb%2Fbin%2F3f44abc8-2774-11e8-acc5-262aff1ca7a6.jpg?crop=988%2C556%2C910%2C214&resize=685>

<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

| |
|-------------------|
| ? |
| Brian May |
| Leonardo da Vinci |
| Pharrell Williams |
| Bill Gates |



<https://cdn.britannica.com/s:300x300/73/20973-004-F71E20CB.jpg>

https://upload.wikimedia.org/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



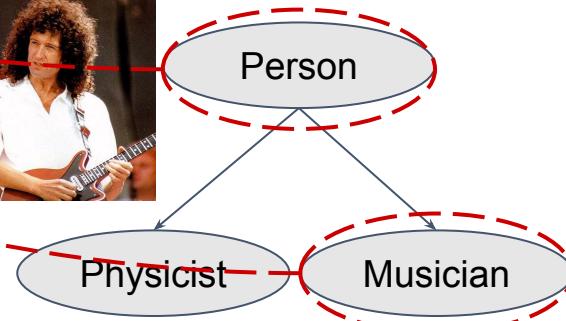
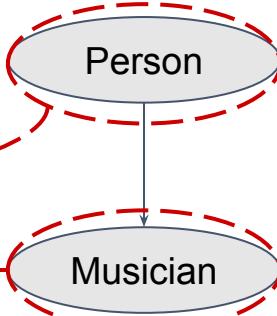
Physicist

Musician

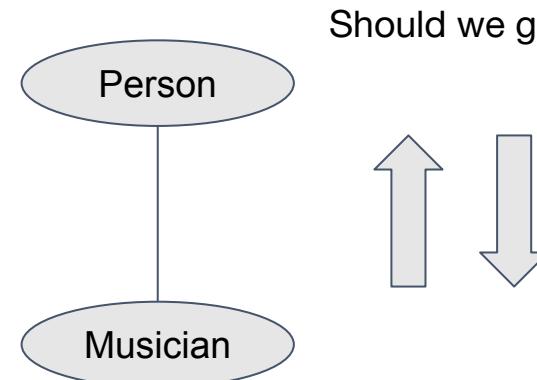
<http://cdn01.cdn.justjared.com/wp-content/uploads/2014/03/williams-shorts/pharrell-williams-wear-shorts-on-oscars-2014-red-carpet-03.jpg>
https://pbs.twimg.com/profile_images/988775660163252226/XpgonN0X.jpg
<https://www.queenie.cz/storage/temp/8f2b24db44d09bd8e3d563d0bb099fc2-400x800x1.jpg>
<https://www.thetimes.co.uk/imageserver/image/methode%2Ftimes%2Fprod%2Fweb%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>





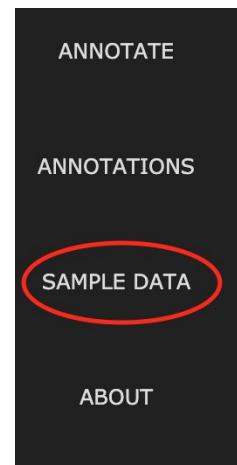
Questions?





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

2. Pick a sample CSV:



3. Add the CSV file:

Sample CSV files can be found here

- [Olympic Games](#)
- [Data web commons](#)

Name

Boxers

Prefix (optional)

<http://dbpedia.org/ontology/>

Upload CSV file

aaaboxers.csv

Subject column index (default is 0)

0

Force title case



4. Go to the annotation page:

List of Entity Annotations

To see the annotation result of your file wait until the corresponding status is "**Annotation is complete**"

| ID | Name | Status | See raw results | Annotation results |
|----|----------------|------------------------|---------------------|-------------------------|
| 14 | Boxers (id=25) | Annotation is complete | raw | results |

5. Balance the Coverage and Score:

alpha

Annotation Model

If you don't know what to chose then go for 3.

Specificity function

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)

alpha **0.01**

Annotation Model **14 - Boxers** 

If you don't know what to chose then go for 3.

Specificity function **3** 

Score

(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

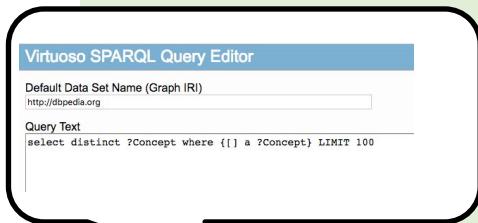
Andrea Cimmino
Ontology Engineering Group
Universidad Politécnica de Madrid, Spain

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



Horizon 2020
European Union funding
for Research & Innovation

From Heterogeneous Data Sources to Linked Data Services



SPARQL Endpoint

Resources via http

Dump File

Linked Data Service
(Homogeneous Datasource)

Linked Data Service

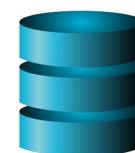
CSV



JSON

{ REST }

TABULAR

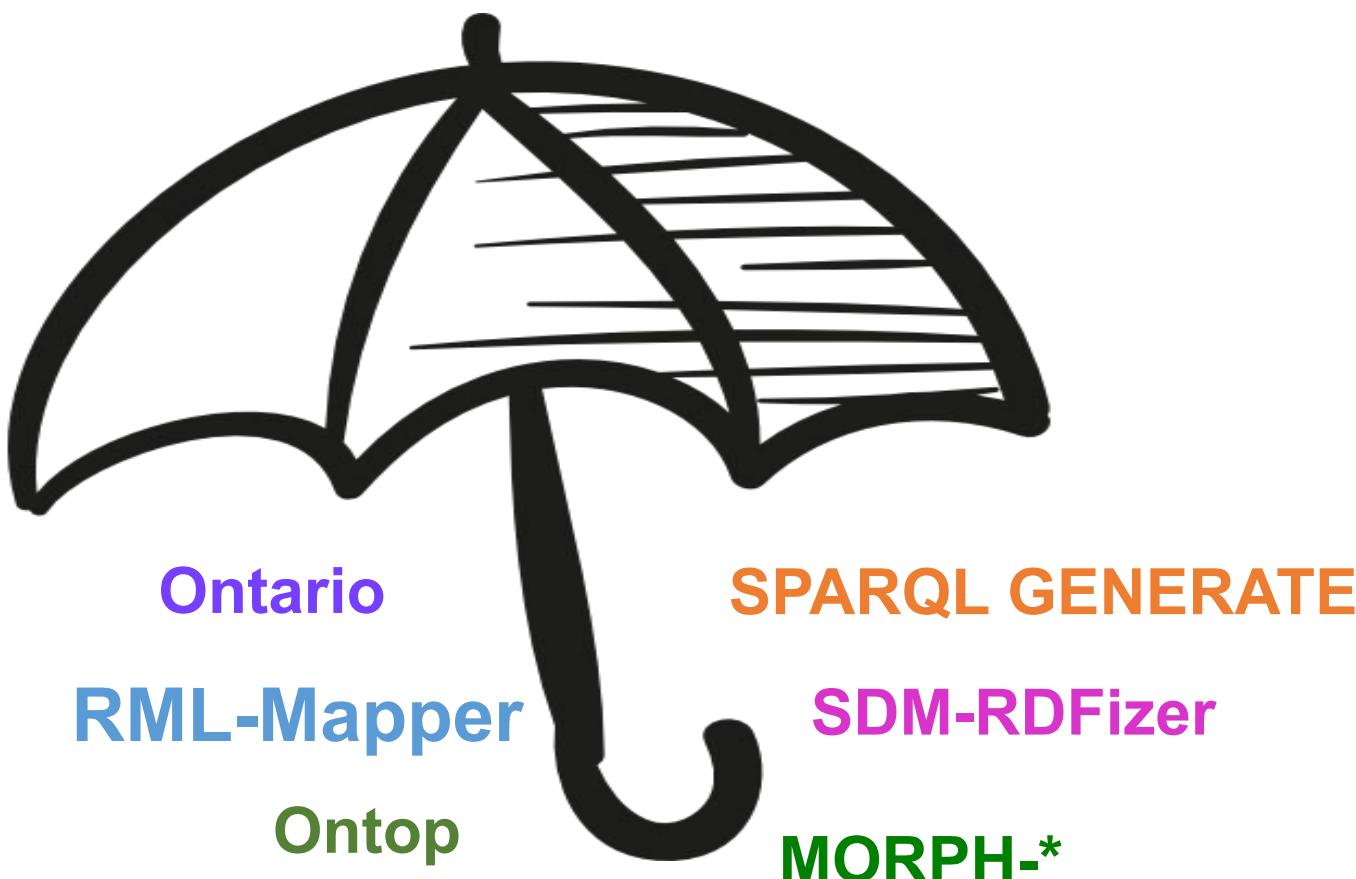


HTML



Heterogeneous Datasources

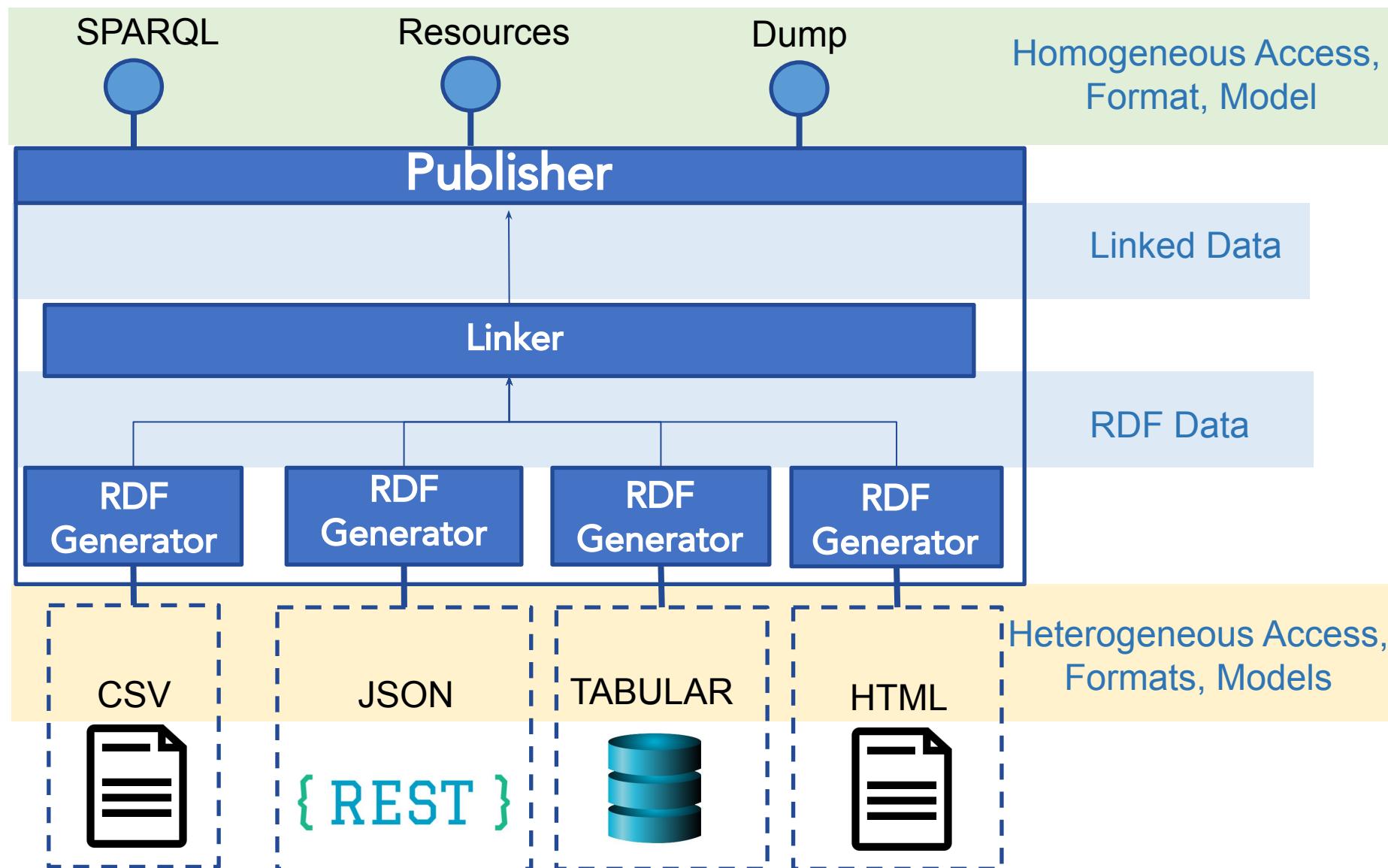
Integrating existing S.W. RDF generators

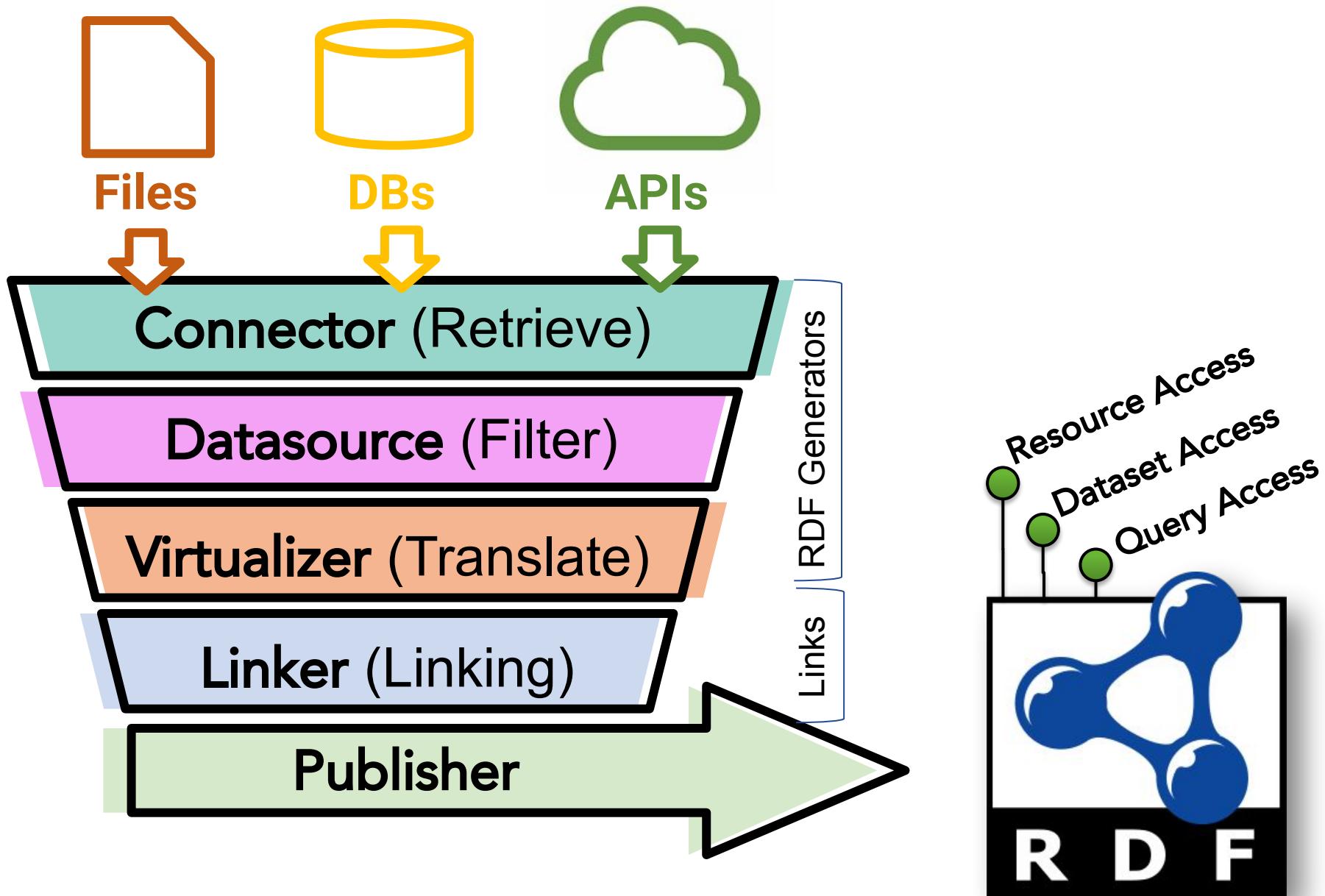




- Cope with the different data sources
 - Access methods, i.e., API, file, DB
 - Formats, i.e., JSON, CSV, Tabular, HTML
 - Security, e.g., APIs with OAuth, files with passwords
- Clean data
 - Lowercase, missing values
- Relate data
 - Interlink data from different sources
- Publish as an RDF view the 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



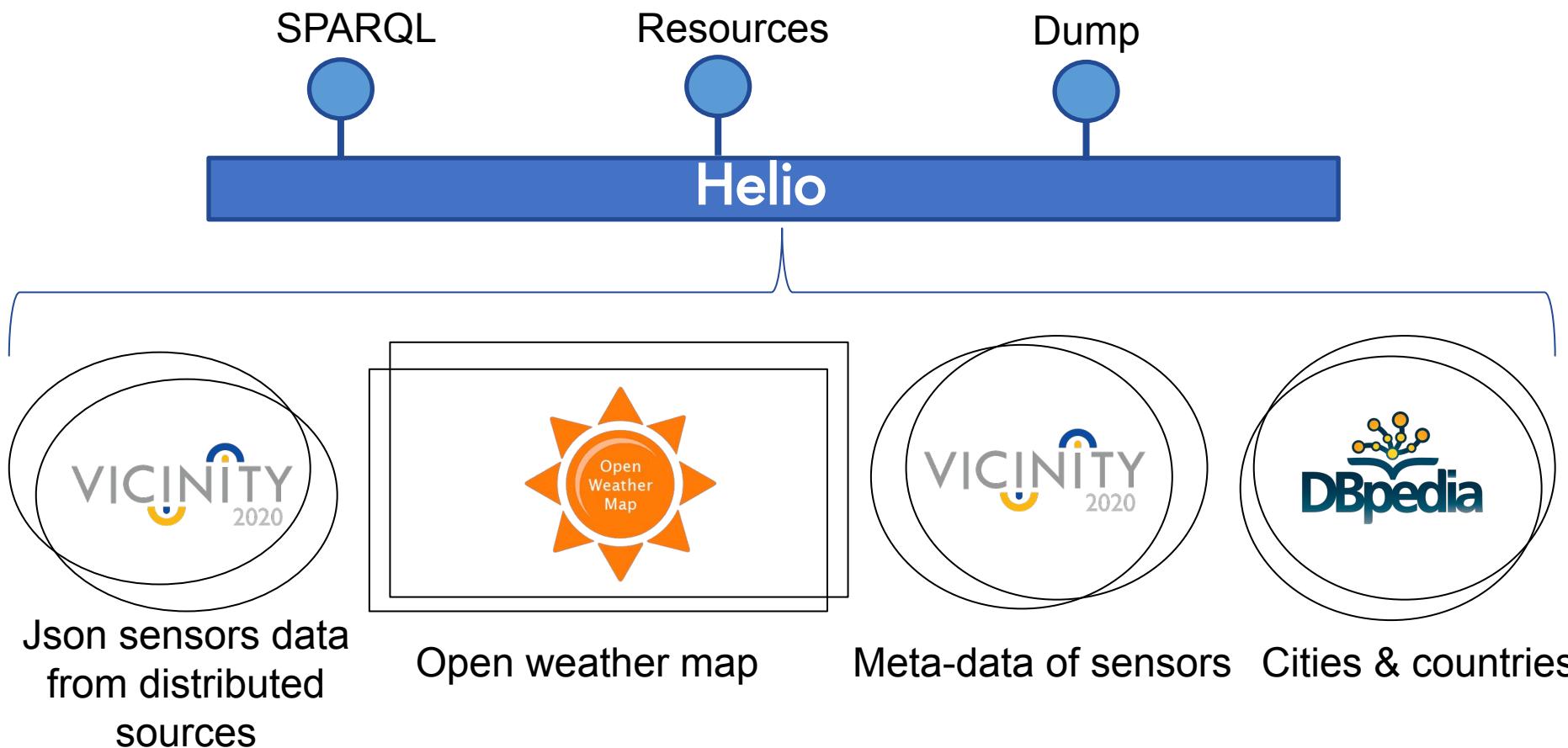




```
"datasources" : [  
    {  
        "id" : "Table PanTHERIA",  
        "type" : "CsvDatasource",  
        "arguments" : [";"],  
        "refresh" : "22118400000",  
        "connector" : {  
            "arguments" : ["./mappings/data/PanTHERIA.csv"],  
            "type" : "LocalFileConnector",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

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

| | Connector | Datasource | Translation | Linking |
|------------|-----------|------------|-------------|---------|
| Challenges | ✓ | ✓ | ✗ | ✓ |



- 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

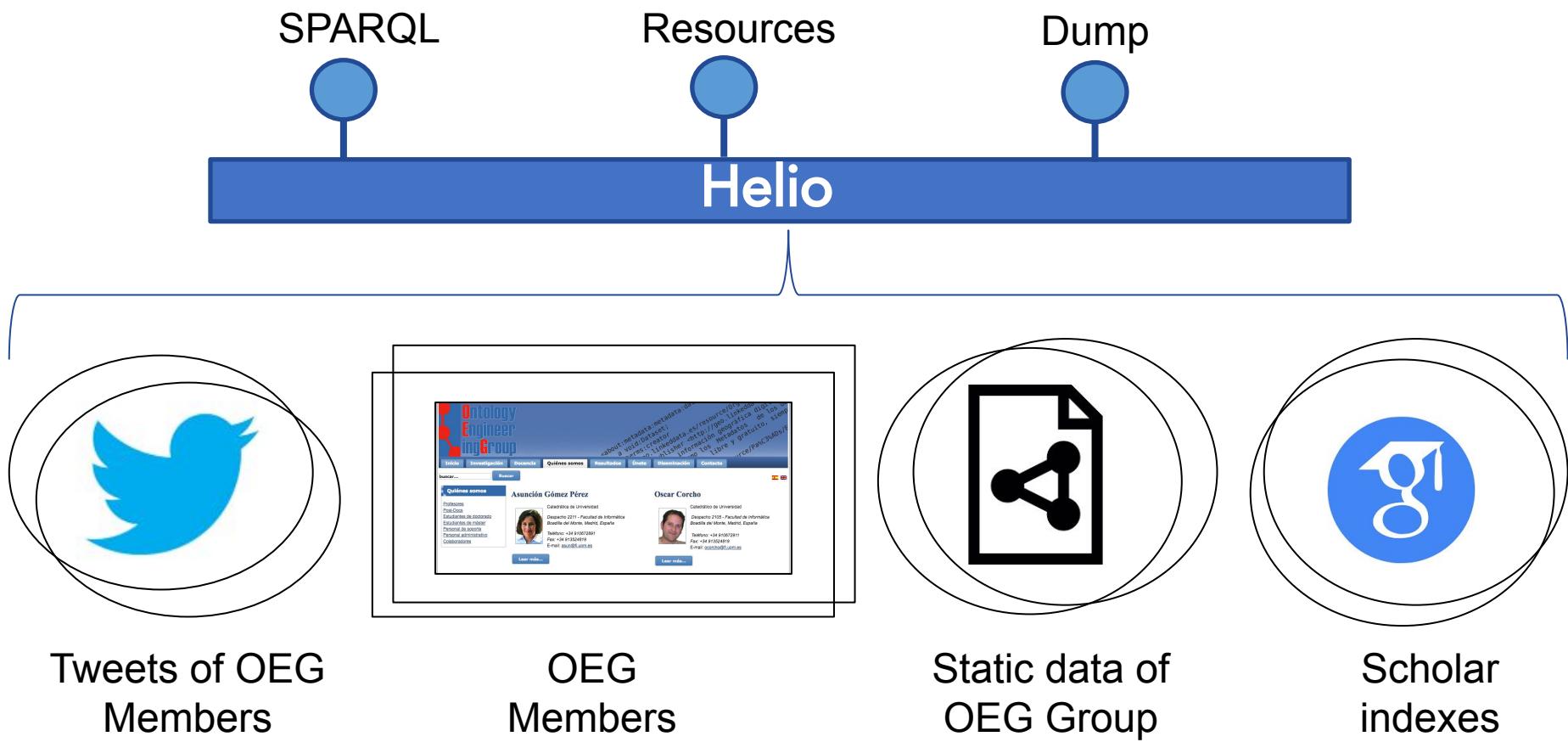
```
{  
  "id" : "DBPedia Cities and countries",  
  "type" : "JSONDatasource",  
  "refresh" : "36000000",  
  "arguments" : [ "..."],  
  "connector" : {  
    "arguments" : [ "..."],  
    "type" : "GetConnector",  
  }  
}
```

- 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 | ✓ | ✓ | ✓ | ✓ |



- 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 asynchronously 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

Javier Bajo



Profesor Titular de Universidad

Despacho 2105 - Facultad de Informática
Boadilla del Monte, Madrid, España

Teléfono: +34 910672881
Fax: +34 913524819
E-mail: jbajo@fi.upm.es

Leer más...

Raúl García Castro

```
<script language="JavaScript" type="text/javascript">
  <!--
  var prefix = 'm&#97;&#105;lt;&#111;:';
  var suffix = '';
  var attrs = '';
  var path = 'hr' + 'ef' + '=';
  var addy83780 = 'jb&#97;j&#111;' + '&#64;';
  addy83780 = addy83780 + 'f&#105;' + '&#46;' + '&#117;pm' + '&#46;' + '&#101;s';
  document.write( '<a ' + path + '\'' + prefix + addy83780 + suffix + '\'' + attrs + '>');
  document.write( addy83780 );
  document.write( '</a>' );
  //-->

</script>
```

Guadalupe Aguado-de-Cea

Profesora Ad Honorem

| | | |
|----|-----------------------------|-------------------------|
| 20 | Mari Carmen Suárez-Figueroa | mcsuarez at fi.upm.es |
| 21 | Elena Montiel Ponsoda | emontiel at fi.upm.es |
| 22 | Emilio Serrano | emiloserra at fi.upm.es |
| 23 | Ahmad AlObaid | aalobaid at fi.upm.es |
| 24 | Carlos Badenes | cbadenes at fi.upm.es |

Elena Montiel Ponsoda

ratada Doctor

- Facultad de Informática
Boadilla del Monte, Madrid, España
Teléfono: +34 910673051
Fax: +34 913524819
E-mail: emontiel@fi.upm.es

- 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

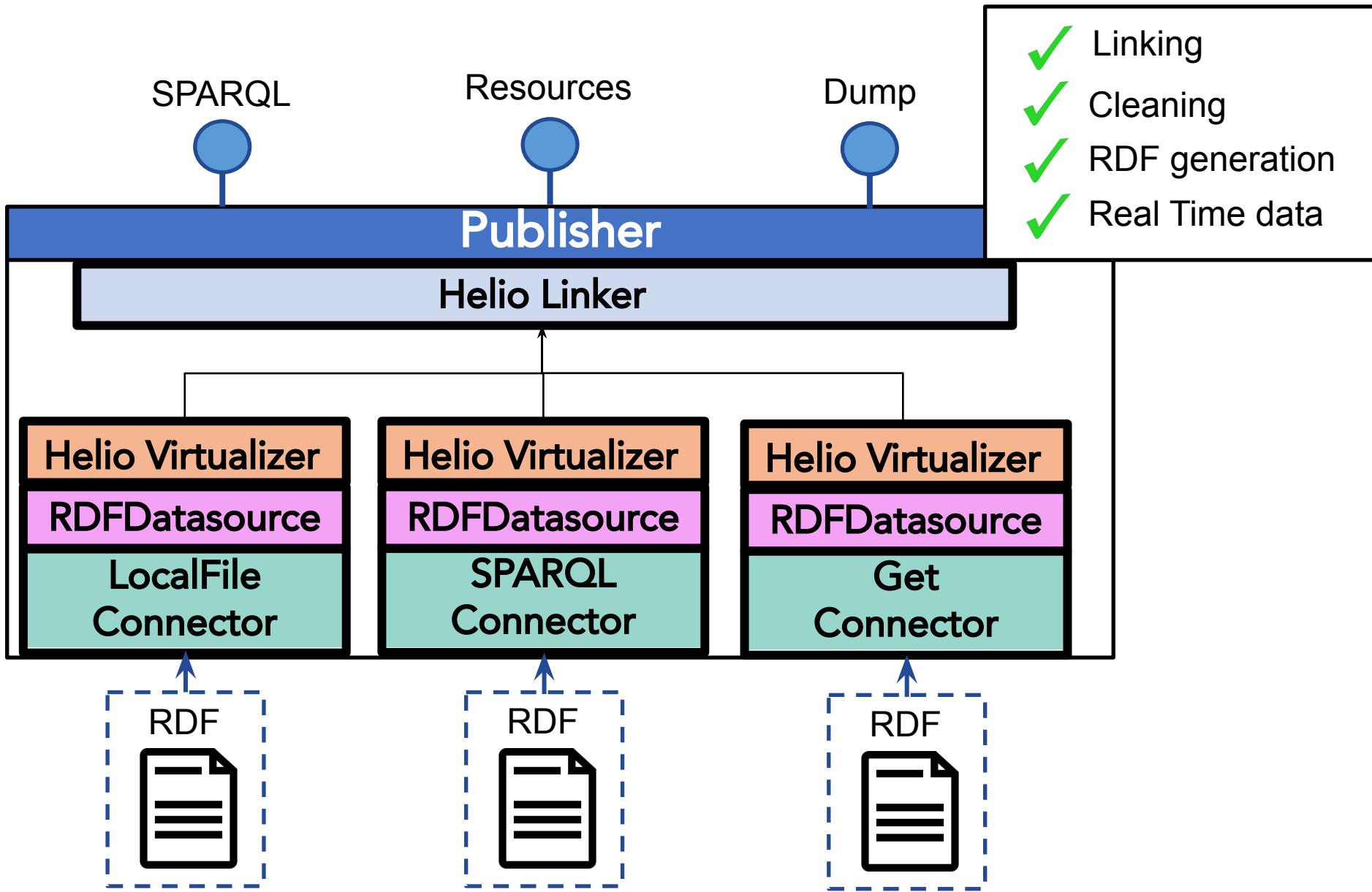
Equality of names does not solve this problem, Fuzzy rules required

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

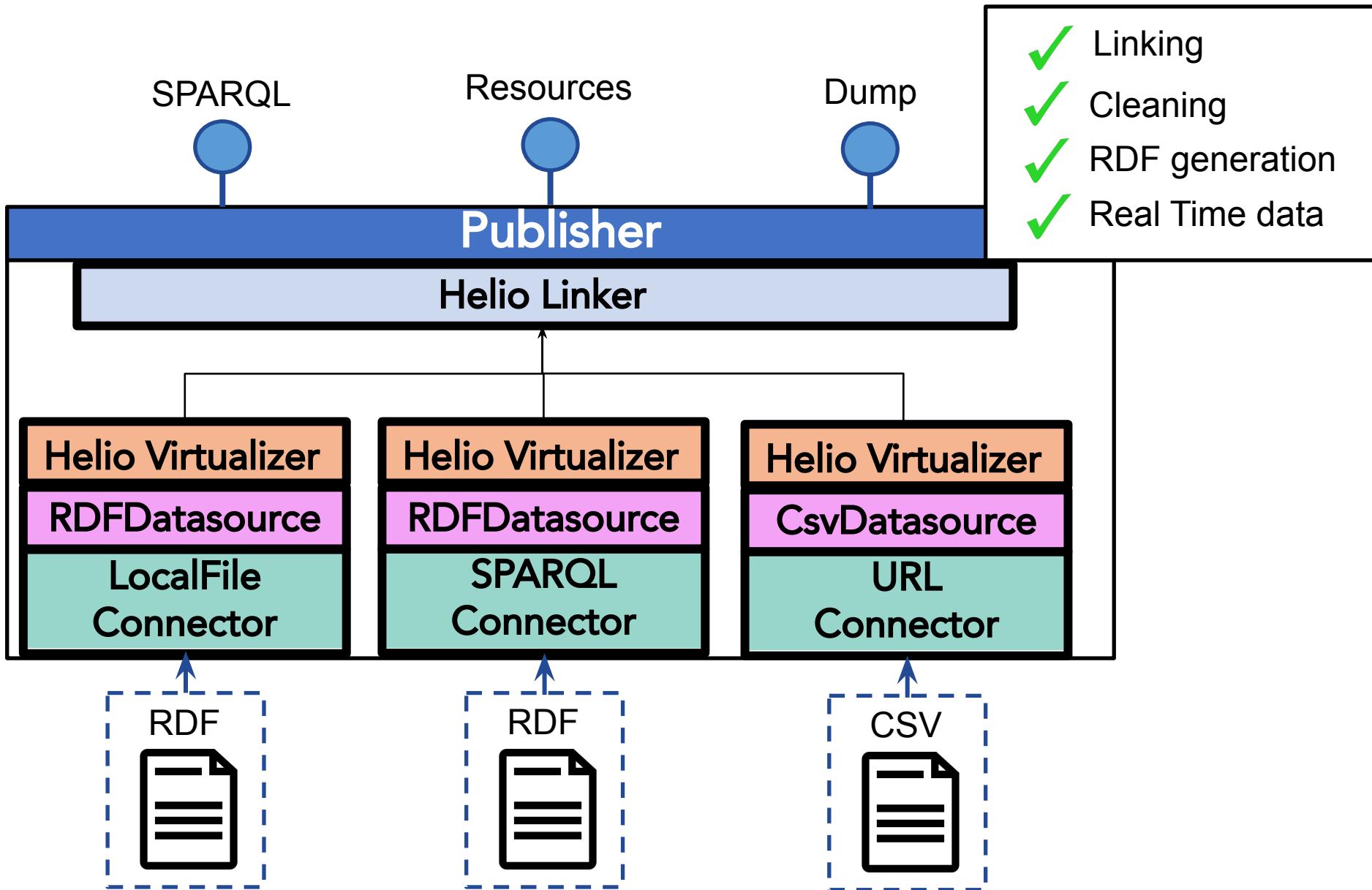
| | tweetAuthor | tweetText | oegMember | oegWebPageName |
|---|------------------|---|---|--------------------------|
| 1 | Esteban González | La nebulosa Roseta en Ha OIII Luminancia y RGB (HaOIIILRGB). Esta imagen es el resultados de la unión de banda estrecha y L... | http://helio.linkeddata.es/oeg/people/Esteban%20Gonzalez%20Guardia | 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 @Aydame3D para mejorar la vida de las personas. Ilusión y ciencia. Buena mezcla. #AYUDAME3D #learnwitht3chfest https://t.co/dKou42HQEO | http://helio.linkeddata.es/oeg/people/Esteban%20Gonzalez%20Guardia | Esteban González Guardia |

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

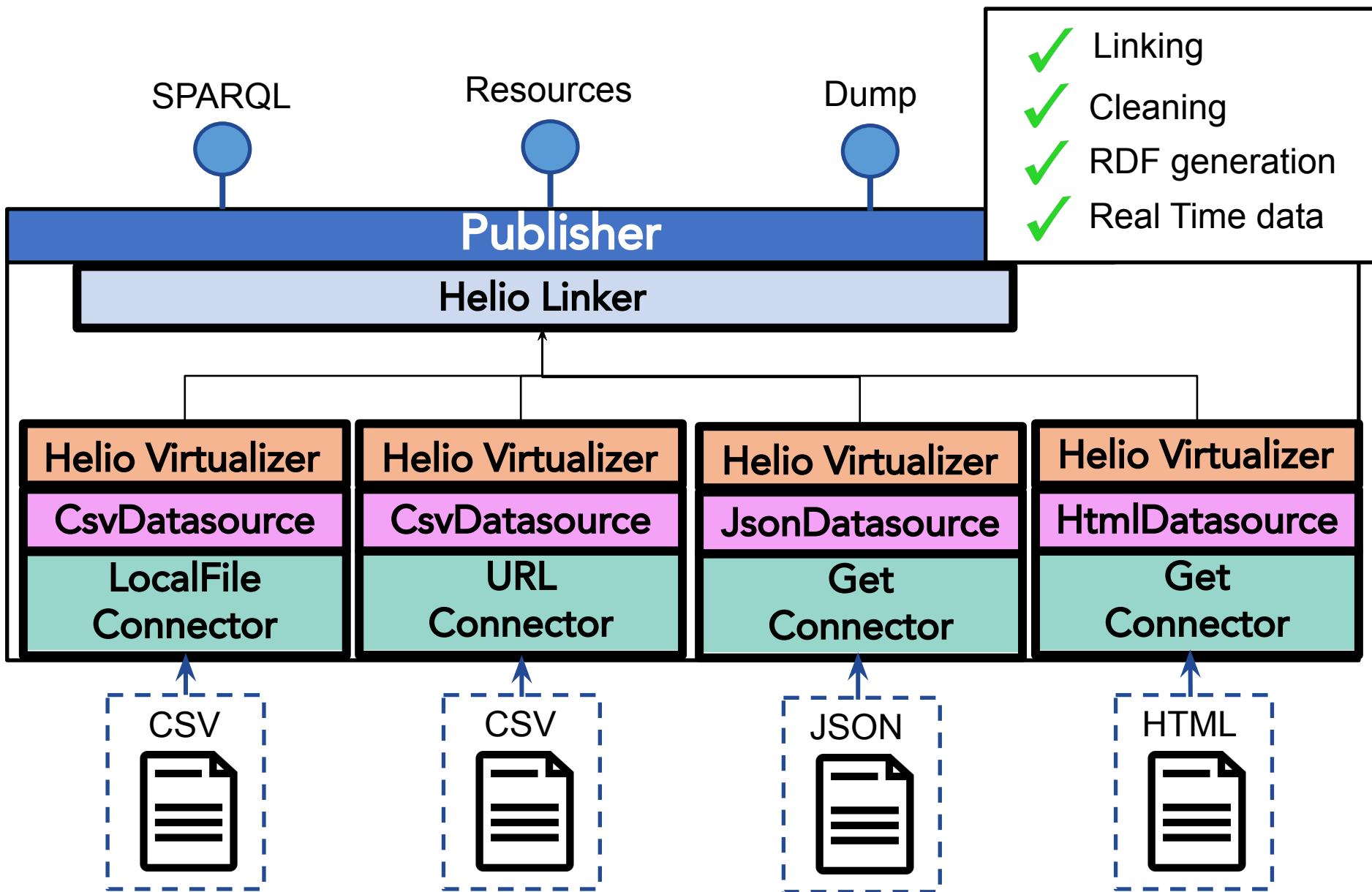
Helio Scenario 1: Refining RDF

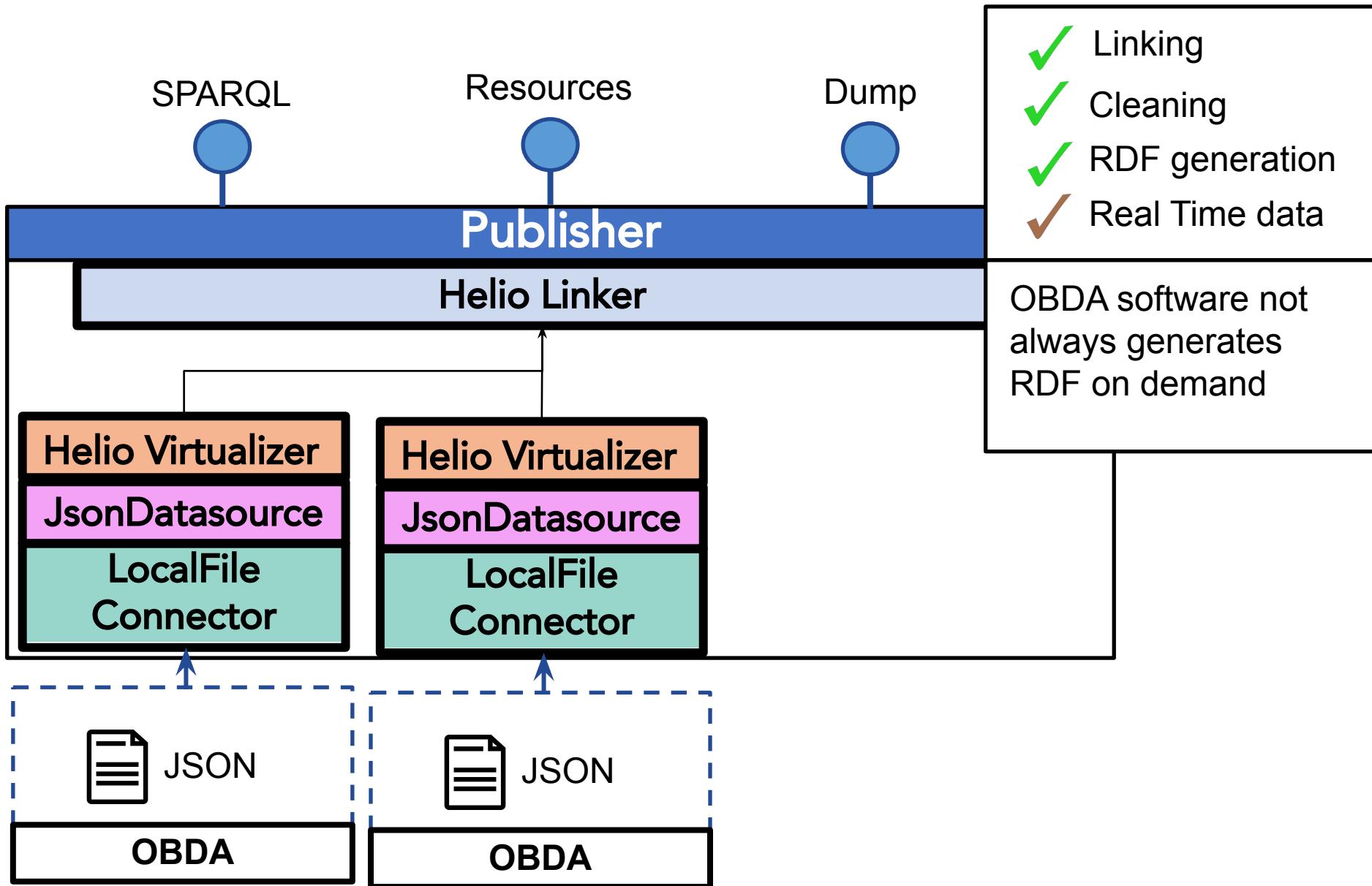


Helio Scenario 2: Enhancing RDF

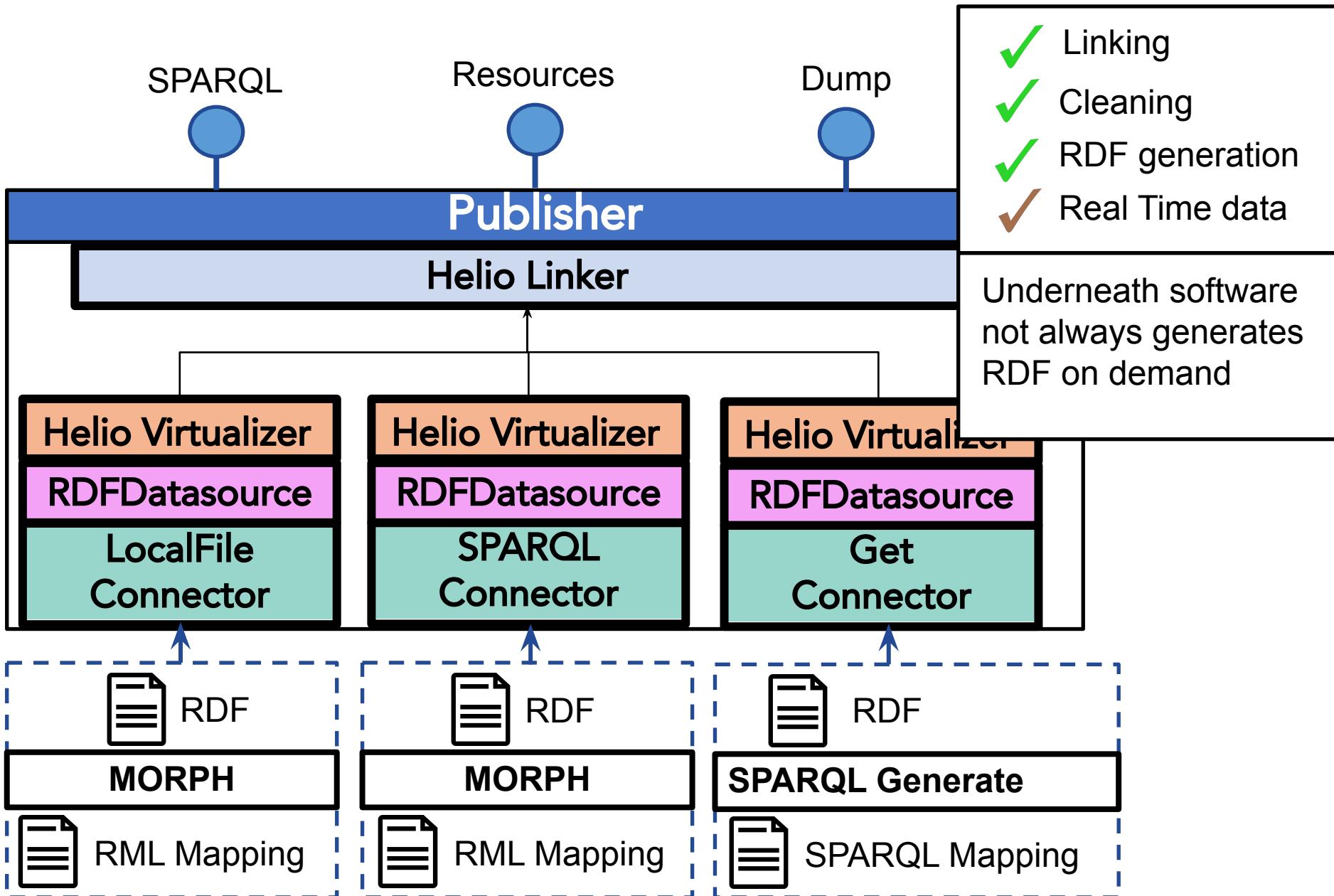


Helio Scenario 3: Data Integration

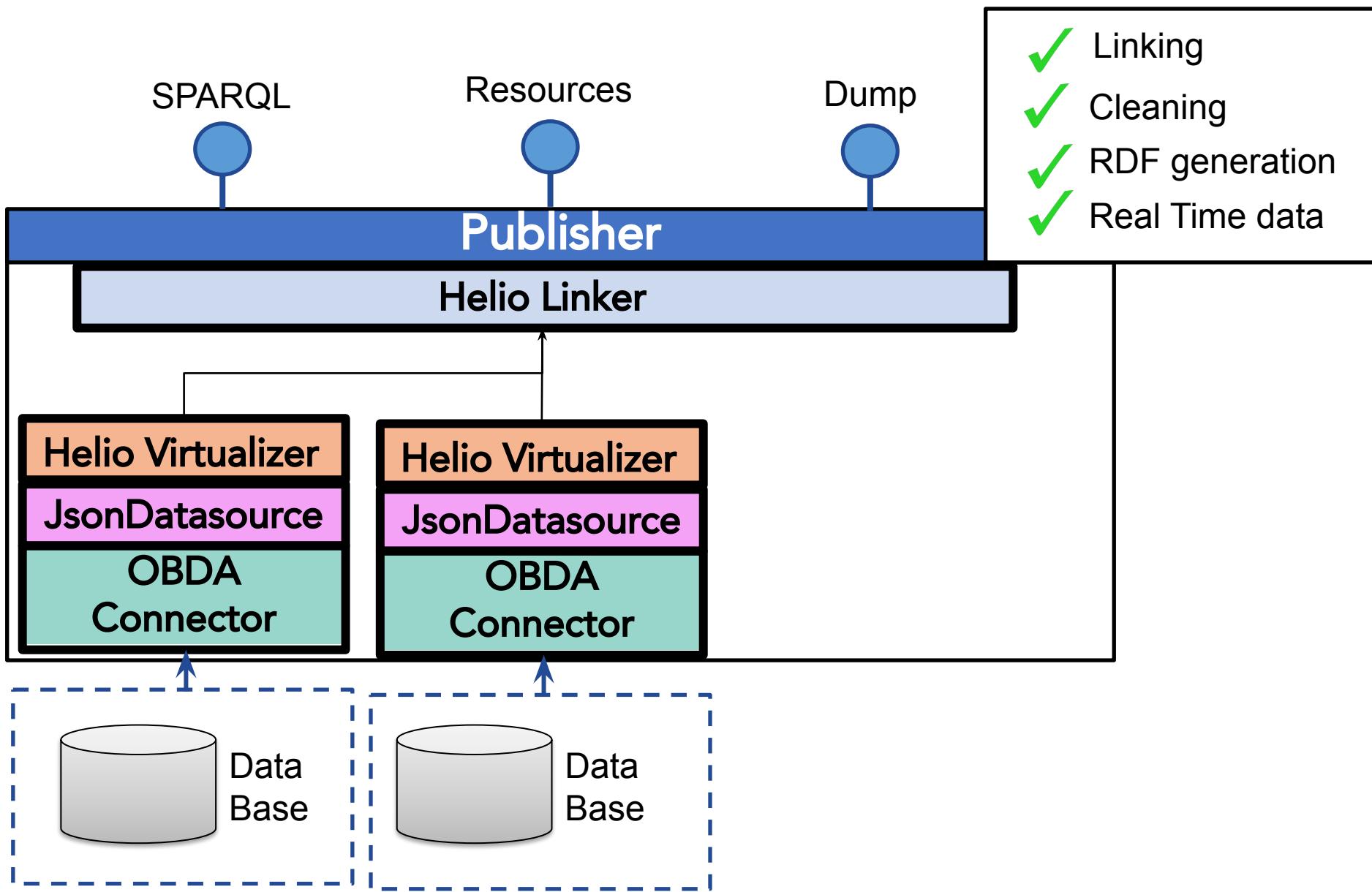




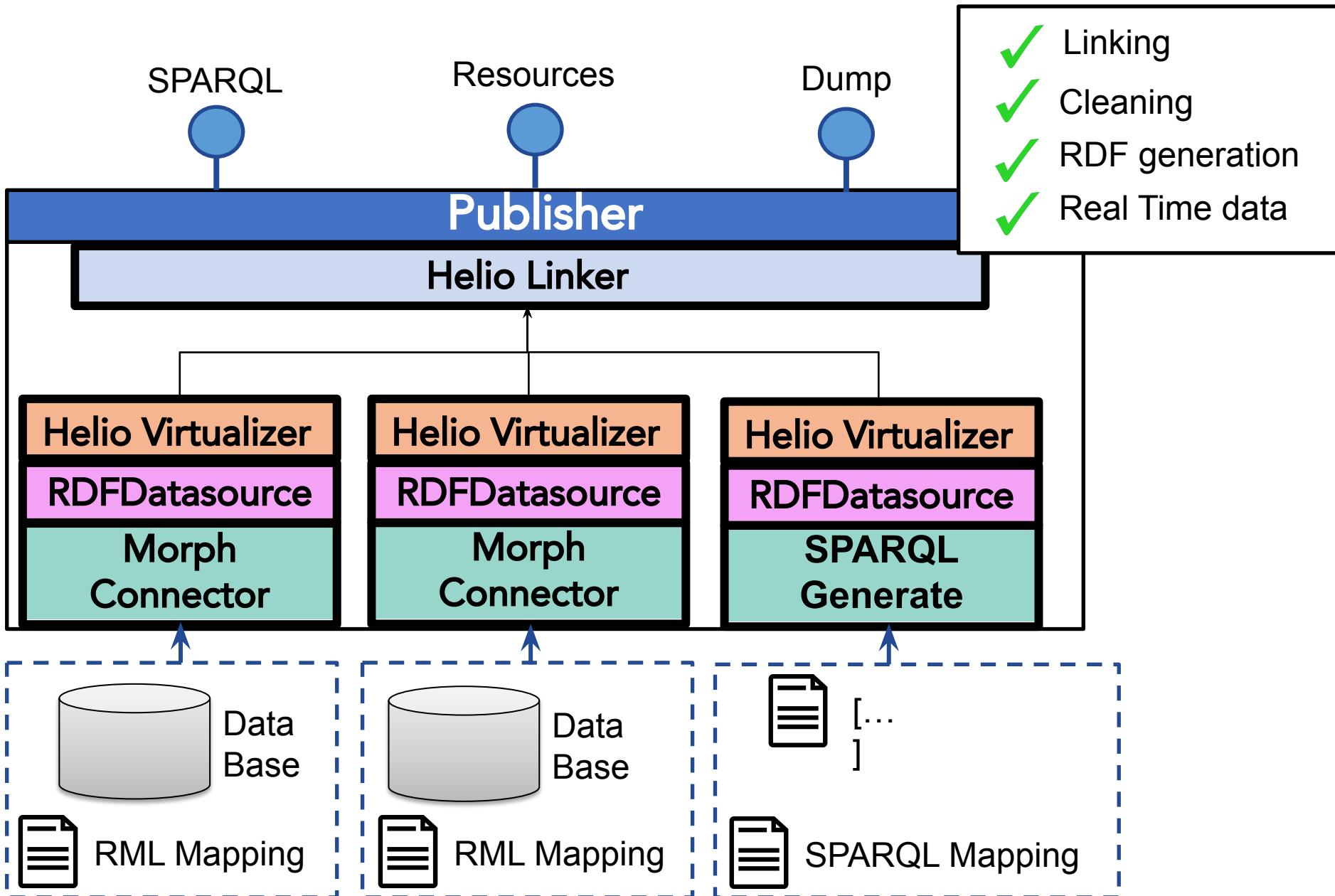
Helio scenario 5: Using third-part RDF generators



Helio Scenario 6: Integrating OBDA



Helio scenario 7: Integrating third-part RDF generators





2

He

Helio

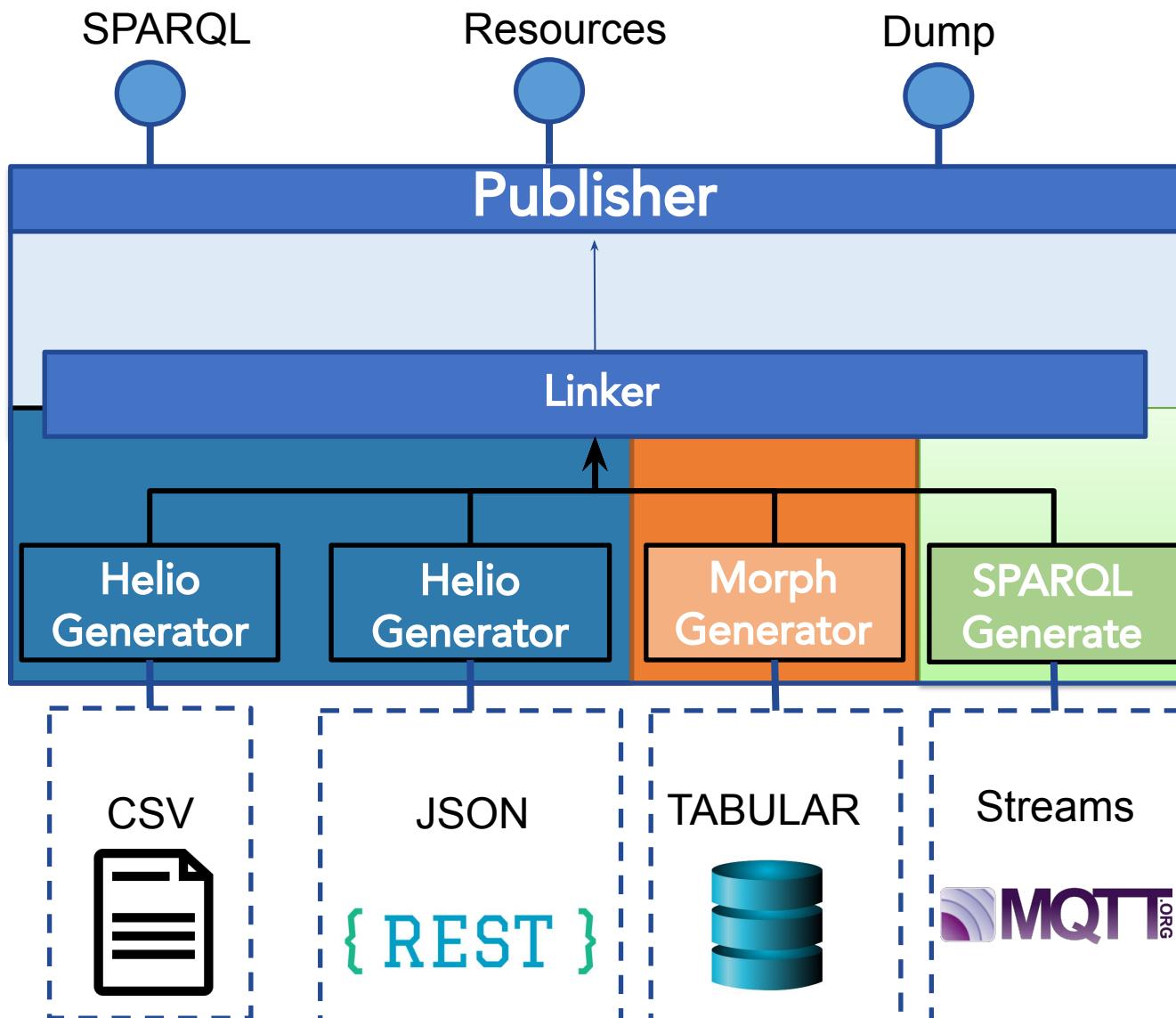


VKG 2019

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



| 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 |