



# Lenguajes de Mapeo para el acceso a datos heterogéneos

**David Chaves-Fraga, Ontology Engineering Group  
Universidad Politécnica de Madrid, Spain**

Oscar Corcho, OEG-UPM

Freddy Priyatna, OEG-UPM

Ahmad Alobaid, OEG-UPM

Andrea Cimmino, OEG-UPM

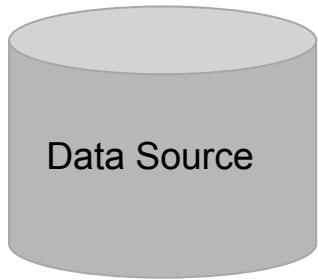
✉ [dchaves@fi.upm.es](mailto:dchaves@fi.upm.es)

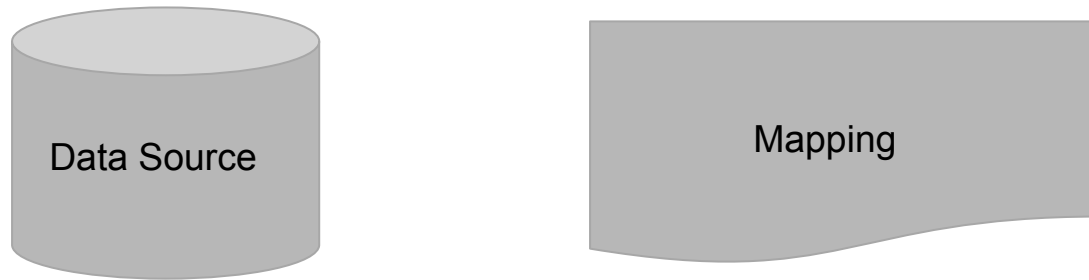
🐦 @dchavesf

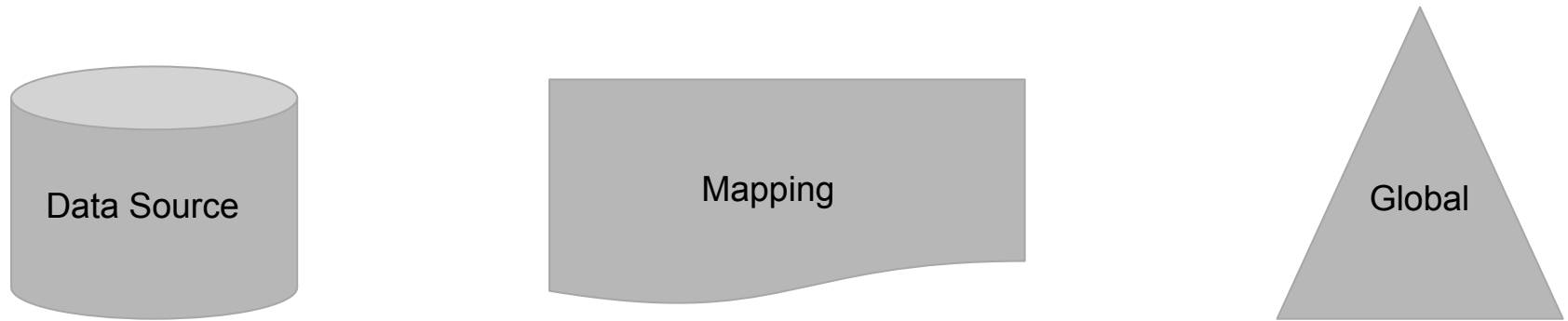
📅 22-23/01/2019

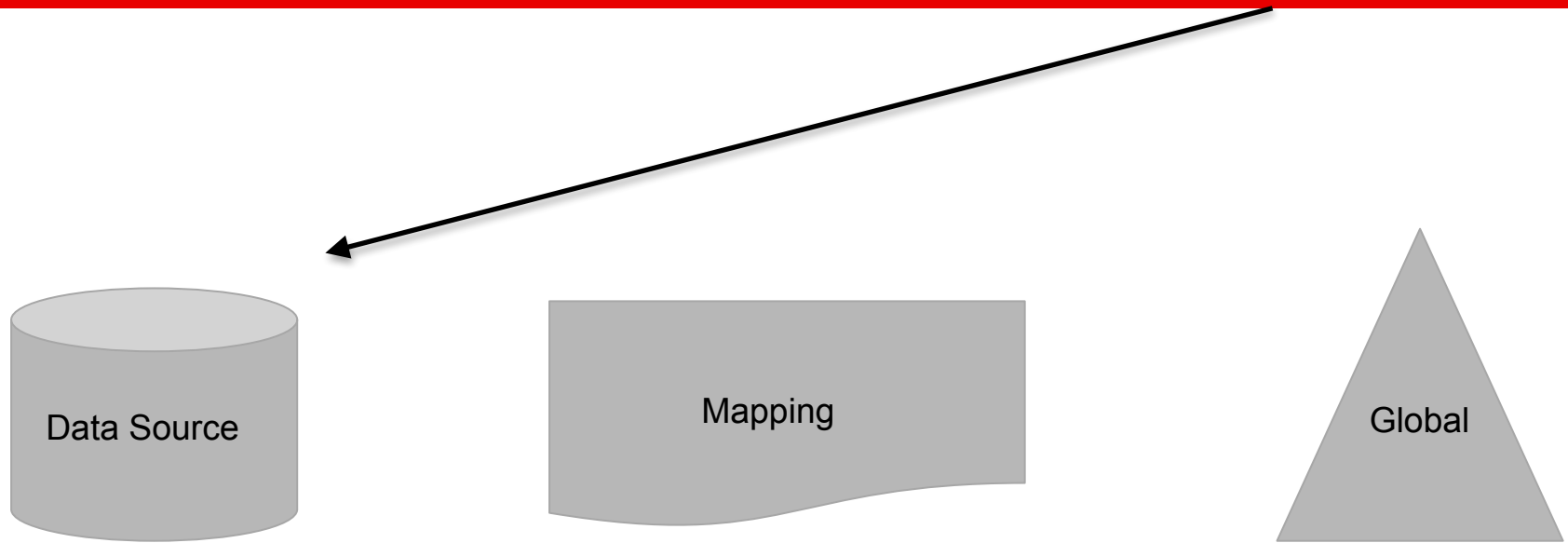
📍 Reunión Datos 4.0

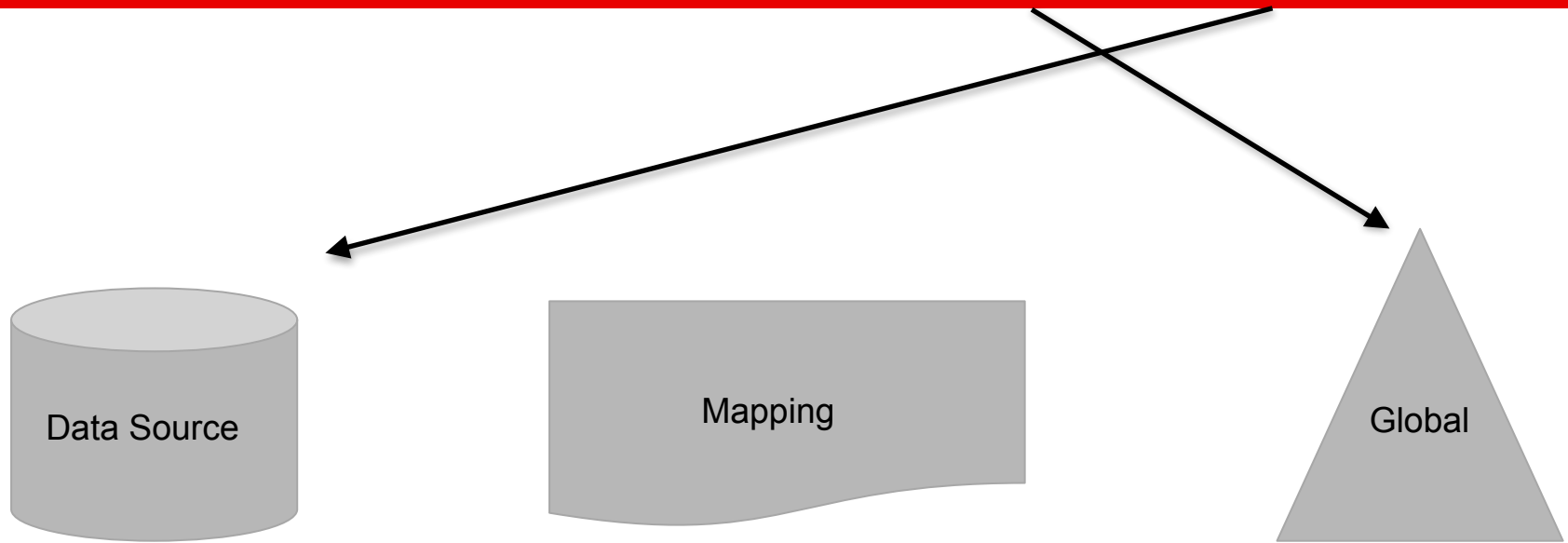


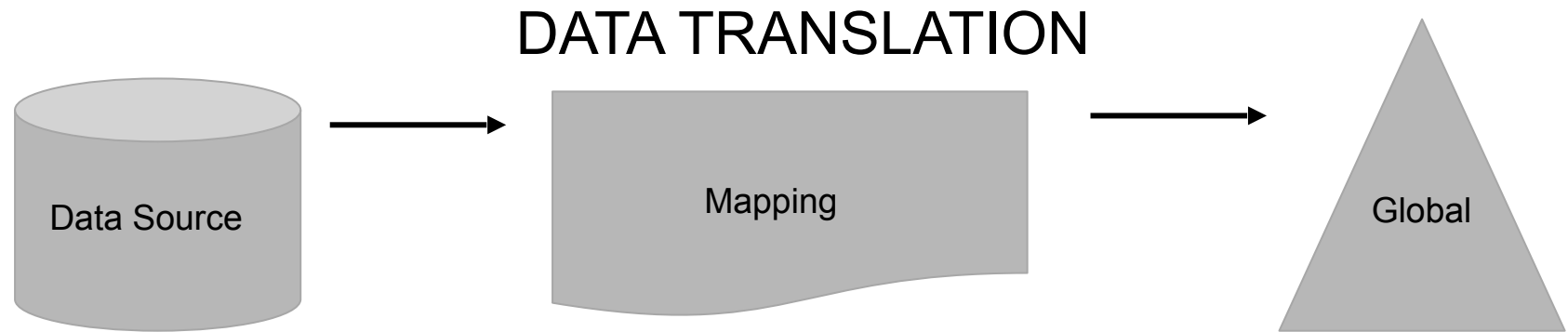




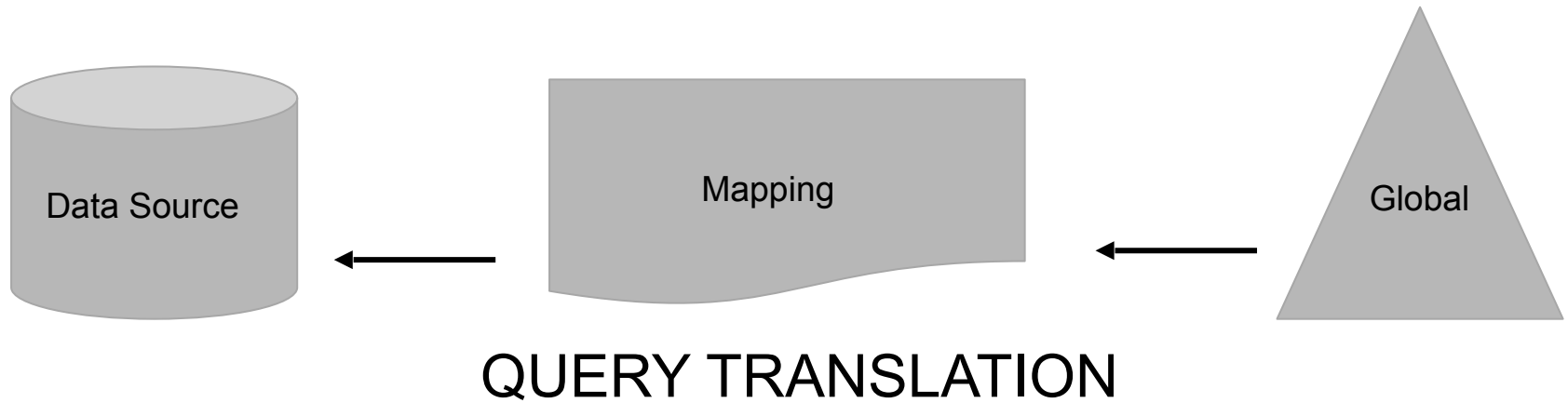


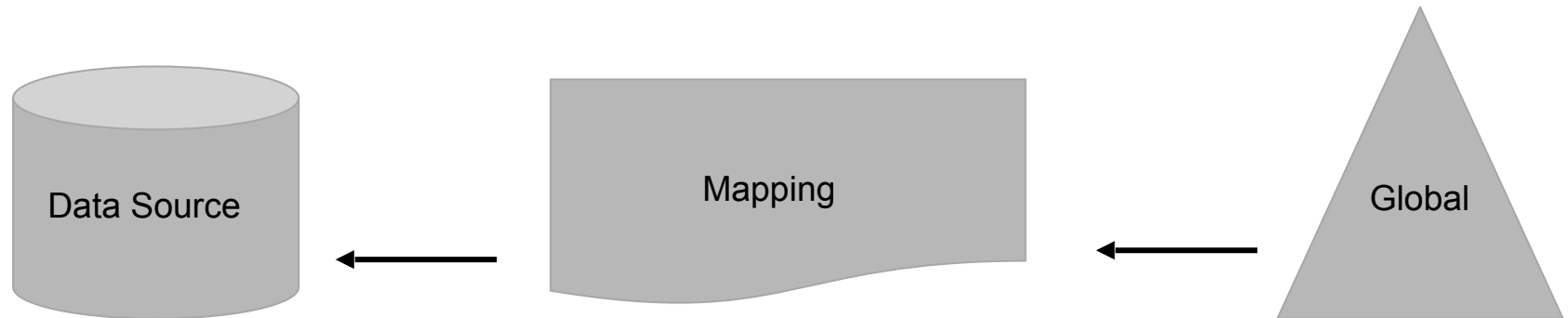




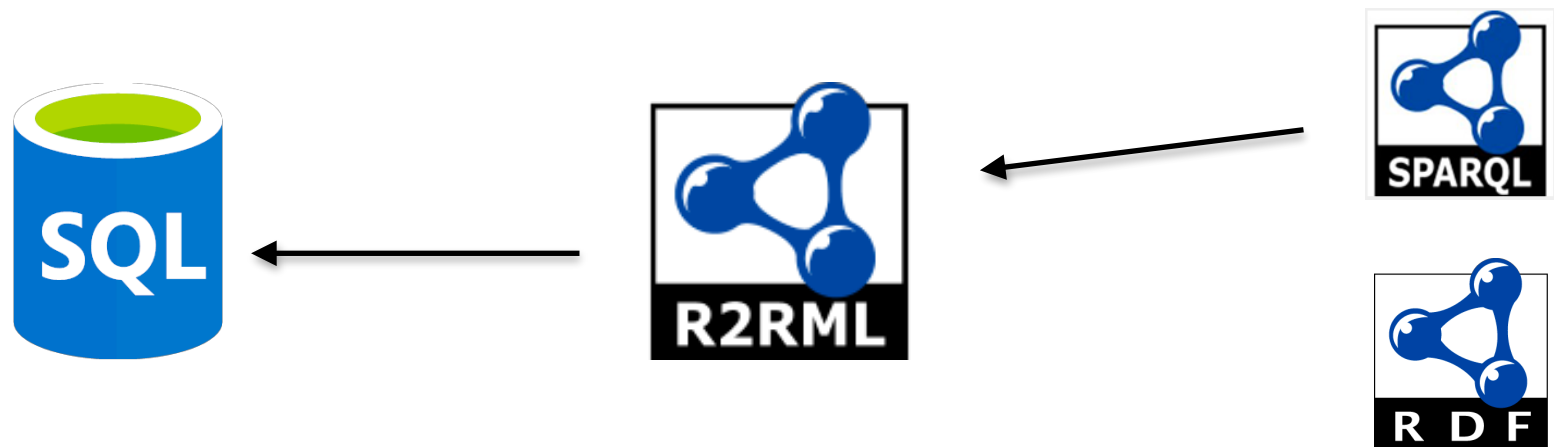


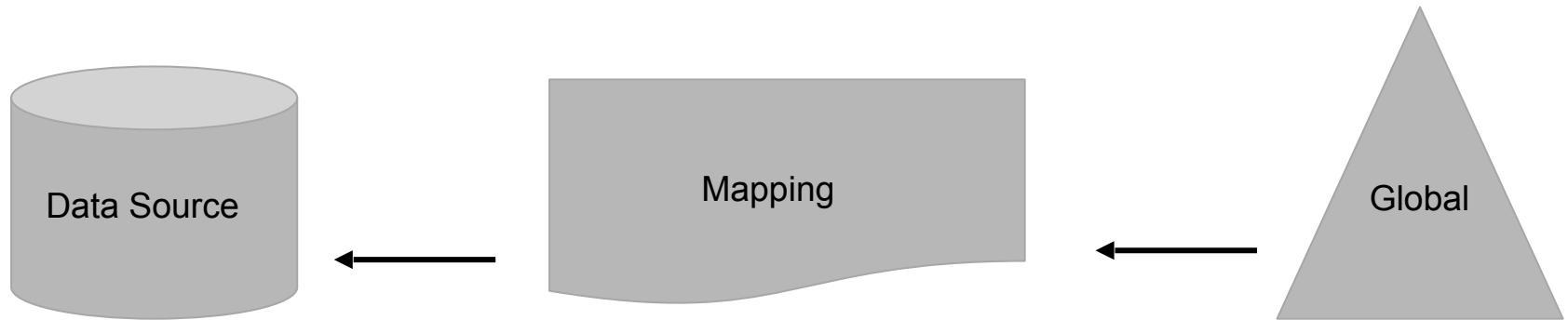




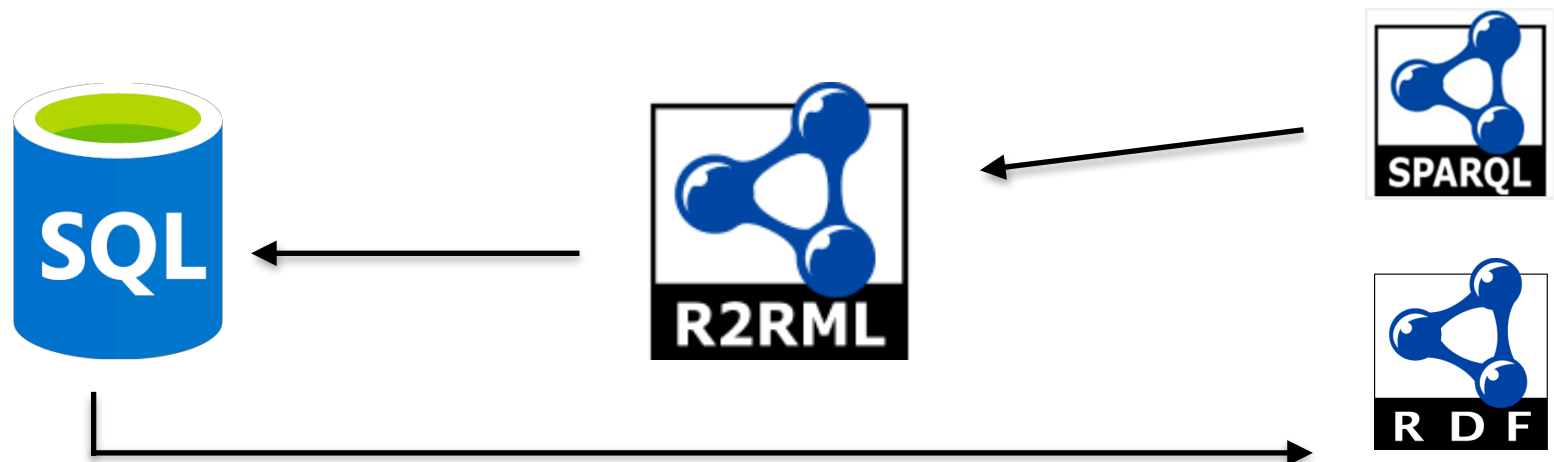


## QUERY TRANSLATION





## QUERY TRANSLATION



¿Cuáles son los formatos más utilizados para exponer datos en la **WEB**?

¿Cuáles son los formatos más utilizados para exponer datos en la **WEB**?

Formato
CSV (8854)
XLS (6339)
JSON (4517)
HTML (3540)
XML-APP (2593)
PDF (2094)
ASCII (1909)
PC-Axis (1302)
RDF-Turtle (1192)
XLSX (1186)
<a href="#">Mostrar más</a>

¿Cuáles son los formatos más utilizados para exponer datos en la **WEB**?

Formato
CSV (8854)
XLS (6339)
JSON (4517)
HTML (3540)
XML-APP (2593)
PDF (2094)
ASCII (1909)
PC-Axis (1302)
RDF-Turtle (1192)
XLSX (1186)
<a href="#">Mostrar más</a>

¿Cuáles son los formatos más utilizados para exponer datos en la **WEB**?

Formato
CSV (8854)
XLS (6339)
JSON (4517)
HTML (3540)
XML-APP (2593)
PDF (2094)
ASCII (1909)
PC-Axis (1302)
RDF-Turtle (1192)
XLSX (1186)
<a href="#">Mostrar más</a>

Formats
CSV (86834)
TXT (65699)
ZIP (51600)
JSON (45900)
GMZ (44712)
HTML (42959)
PDF (35421)
XLS (24418)
WMS (21800)
SHP (19507)
<a href="#">Show More Formats</a>

¿Cuáles son los formatos más utilizados para exponer datos en la **WEB**?

Formato
CSV (8854)
XLS (6339)
JSON (4517)
HTML (3540)
XML-APP (2593)
PDF (2094)
ASCII (1909)
PC-Axis (1302)
RDF-Turtle (1192)
XLSX (1186)
<a href="#">Mostrar más</a>

Formats
CSV (86834)
TXT (65699)
ZIP (51600)
JSON (45900)
GMZ (44712)
HTML (42959)
PDF (35421)
XLS (24418)
WMS (21800)
SHP (19507)
<a href="#">Show More Formats</a>



¿Cuáles son los formatos más utilizados para exponer datos en la **WEB**?

Formato
CSV (8854)
XLS (6339)
JSON (4517)
HTML (3540)
XML-APP (2593)
PDF (2094)
ASCII (1909)
PC-Axis (1302)
RDF-Turtle (1192)
XLSX (1186)
<a href="#">Mostrar más</a>

Formats
CSV (86834)
TXT (65699)
ZIP (51600)
JSON (45900)
GMZ (44712)
HTML (42959)
PDF (35421)
XLS (24418)
WMS (21800)
SHP (19507)
<a href="#">Show More Formats</a>

Formatos
CSV (371)
XLS (169)
XML (129)
XLSX (122)
WMS (29)
RDF (21)
prj (7)
SHP (7)
SHX (7)
ZIP (7)
dBase (6)
WMTS (5)
JSON (4)
KML (3)
BDF (2)
DAT (2)

¿Cuáles son los formatos más utilizados para exponer datos en la **WEB**?

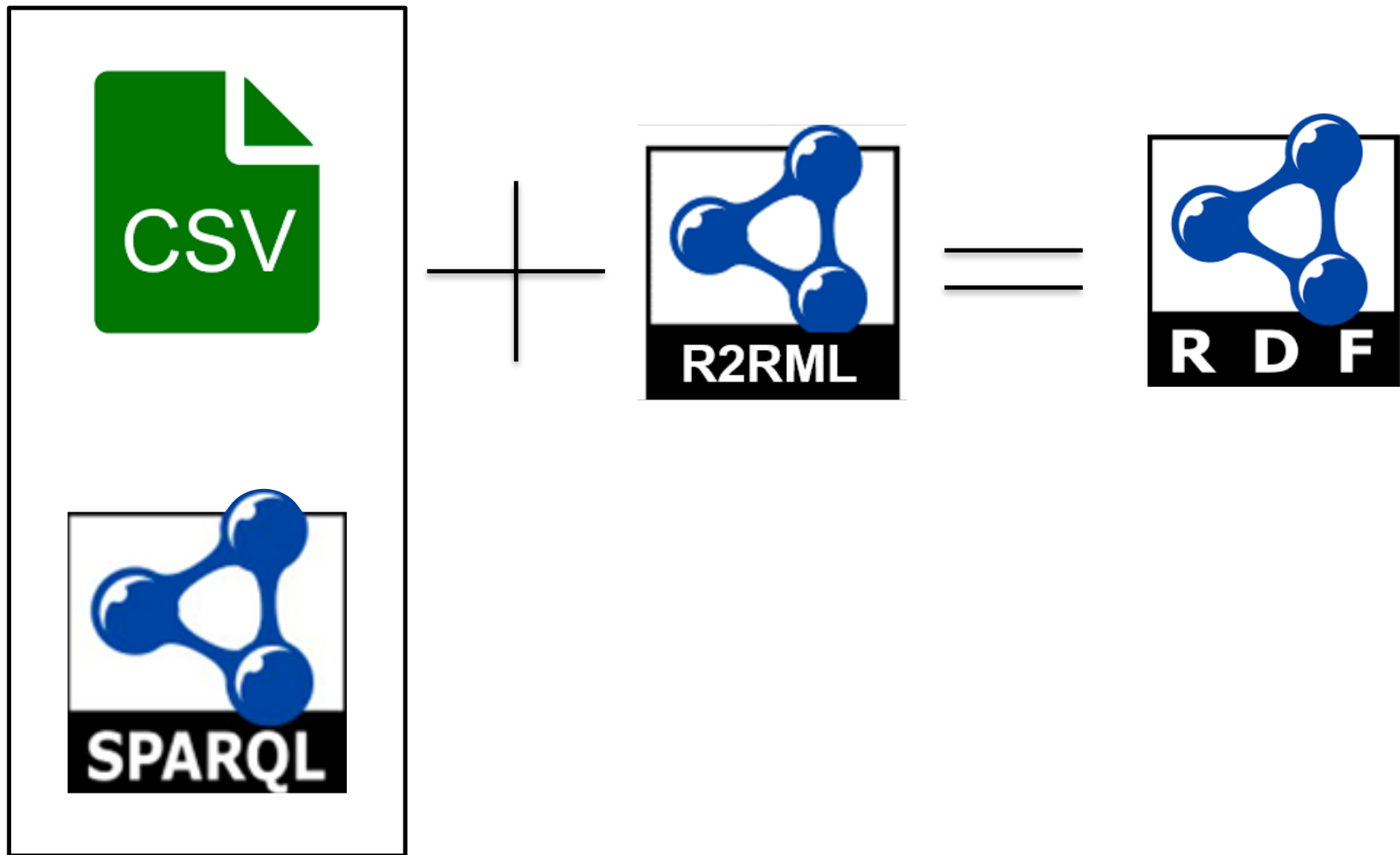
Formato
CSV (8854)
XLS (6339)
JSON (4517)
HTML (3540)
XML-APP (2593)
PDF (2094)
ASCII (1909)
PC-Axis (1302)
RDF-Turtle (1192)
XLSX (1186)
<a href="#">Mostrar más</a>

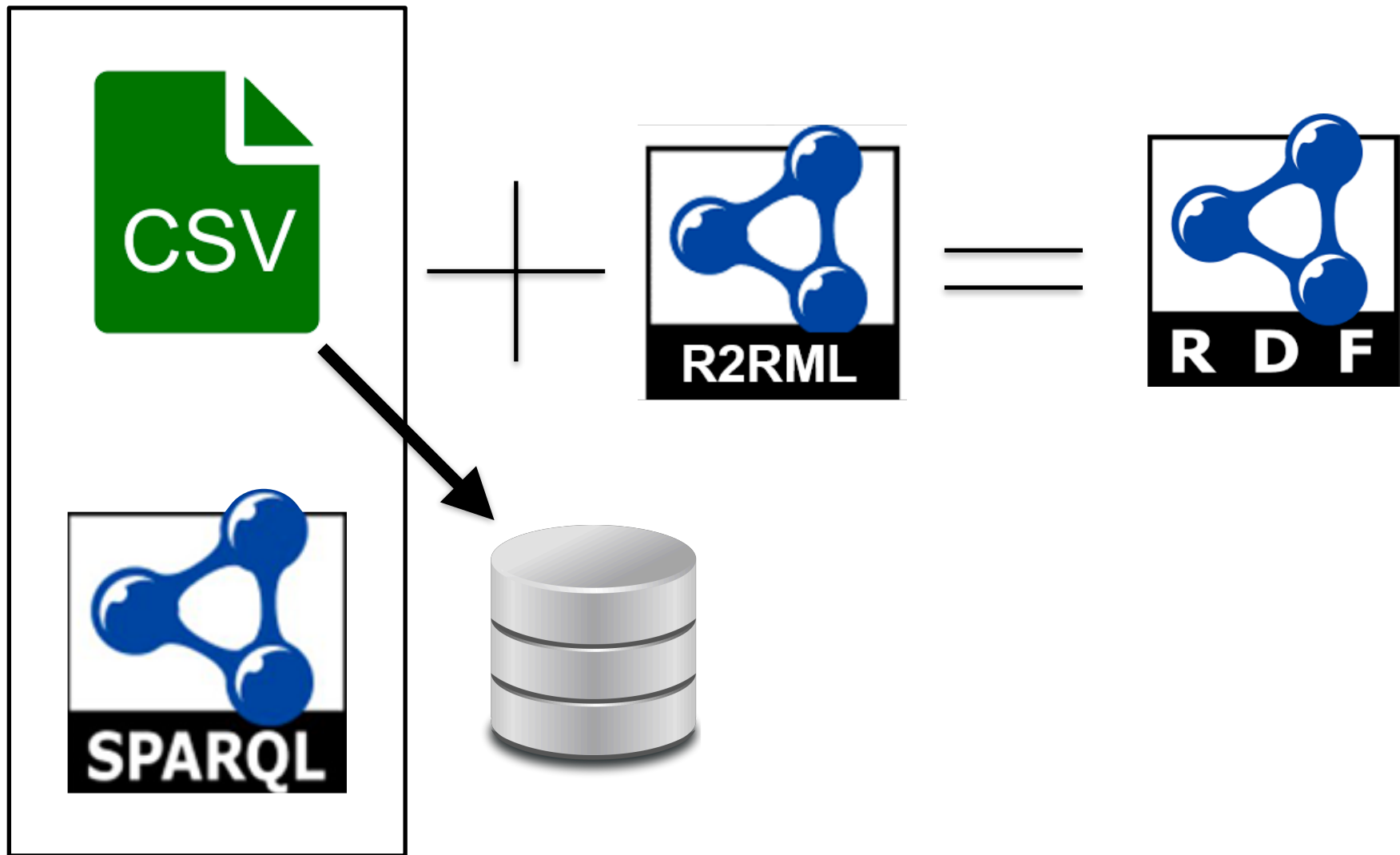
Formats
CSV (86834)
TXT (65699)
ZIP (51600)
JSON (45900)
GMZ (44712)
HTML (42959)
PDF (35421)
XLS (24418)
WMS (21800)
SHP (19507)
<a href="#">Show More Formats</a>

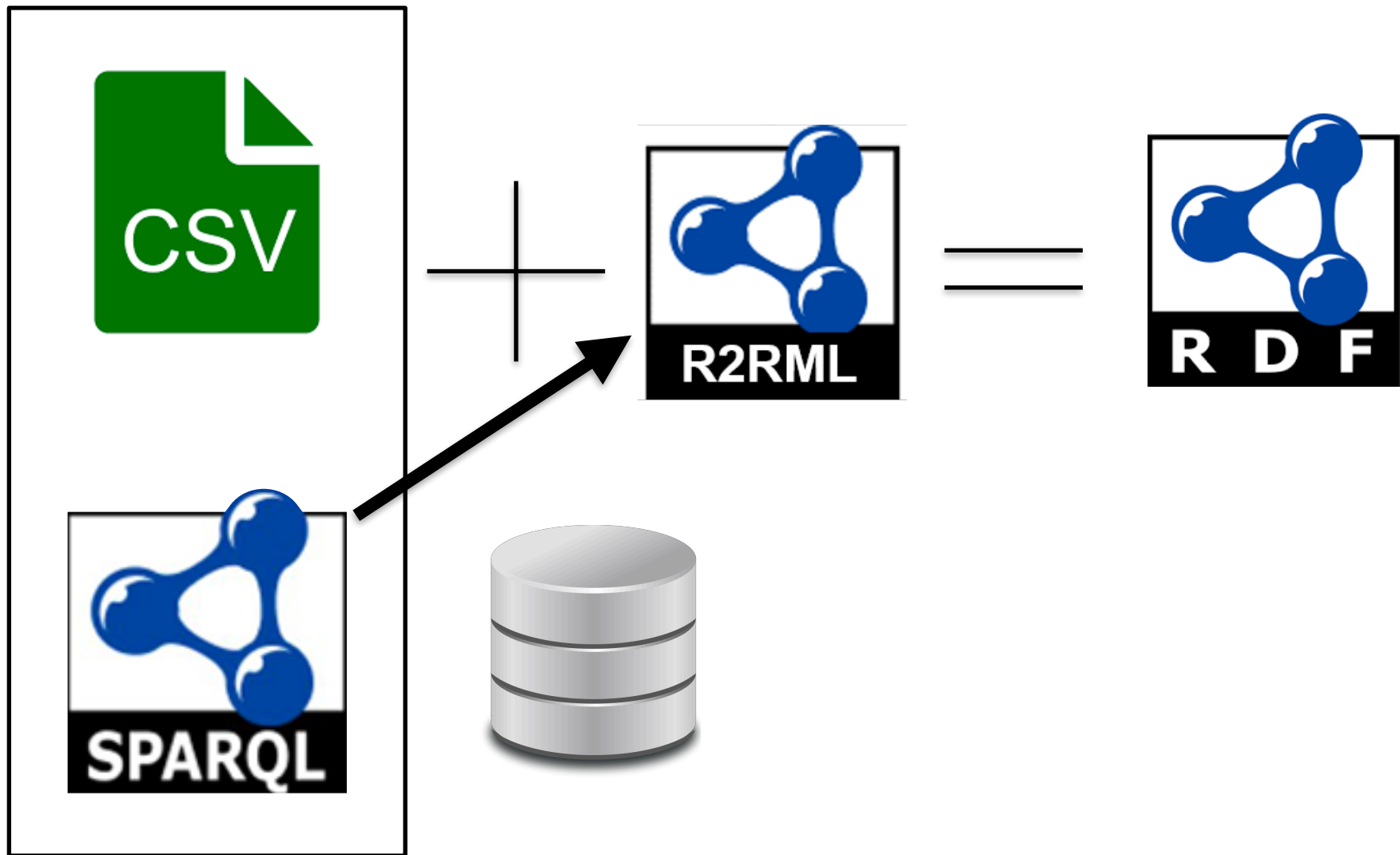
Formatos
CSV (371)
XLS (169)
XML (129)
XLSX (122)
WMS (29)
RDF (21)
prj (7)
SHP (7)
SHX (7)
ZIP (7)
dBase (6)
WMTS (5)
JSON (4)
KML (3)
BDF (2)
DAT (2)

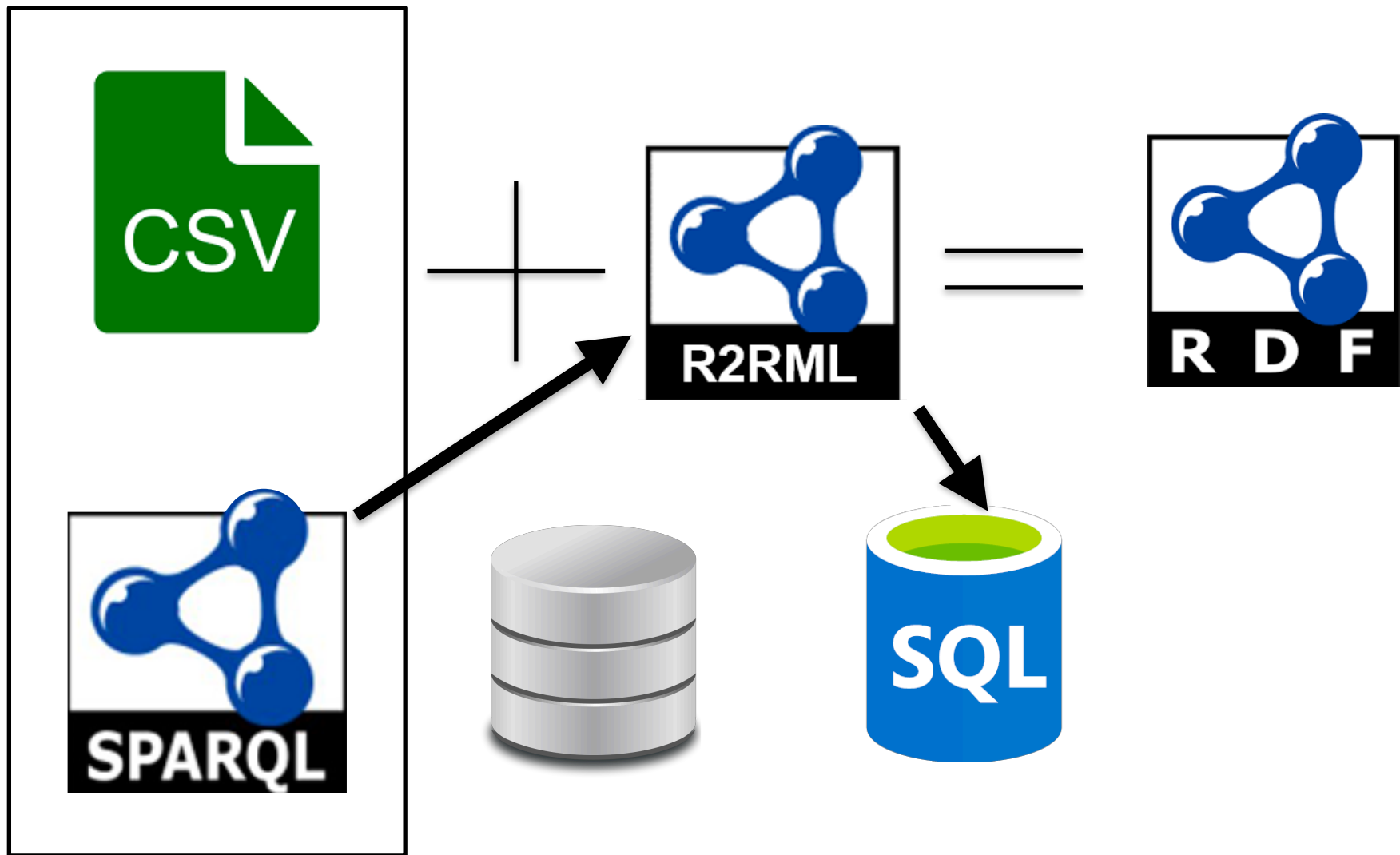
- ¿Cómo permitir **acceso a múltiples archivos CSV** utilizando un enfoque **OBDA**?
- ¿Cómo acercar las **tecnologías de la Web Semántica** a los **desarrolladores** con el fin de ofrecer acceso unificado a **fuentes de datos heterogéneas**?

¿Cómo permitir acceso a múltiples archivos CSV  
utilizando un enfoque OBDA?

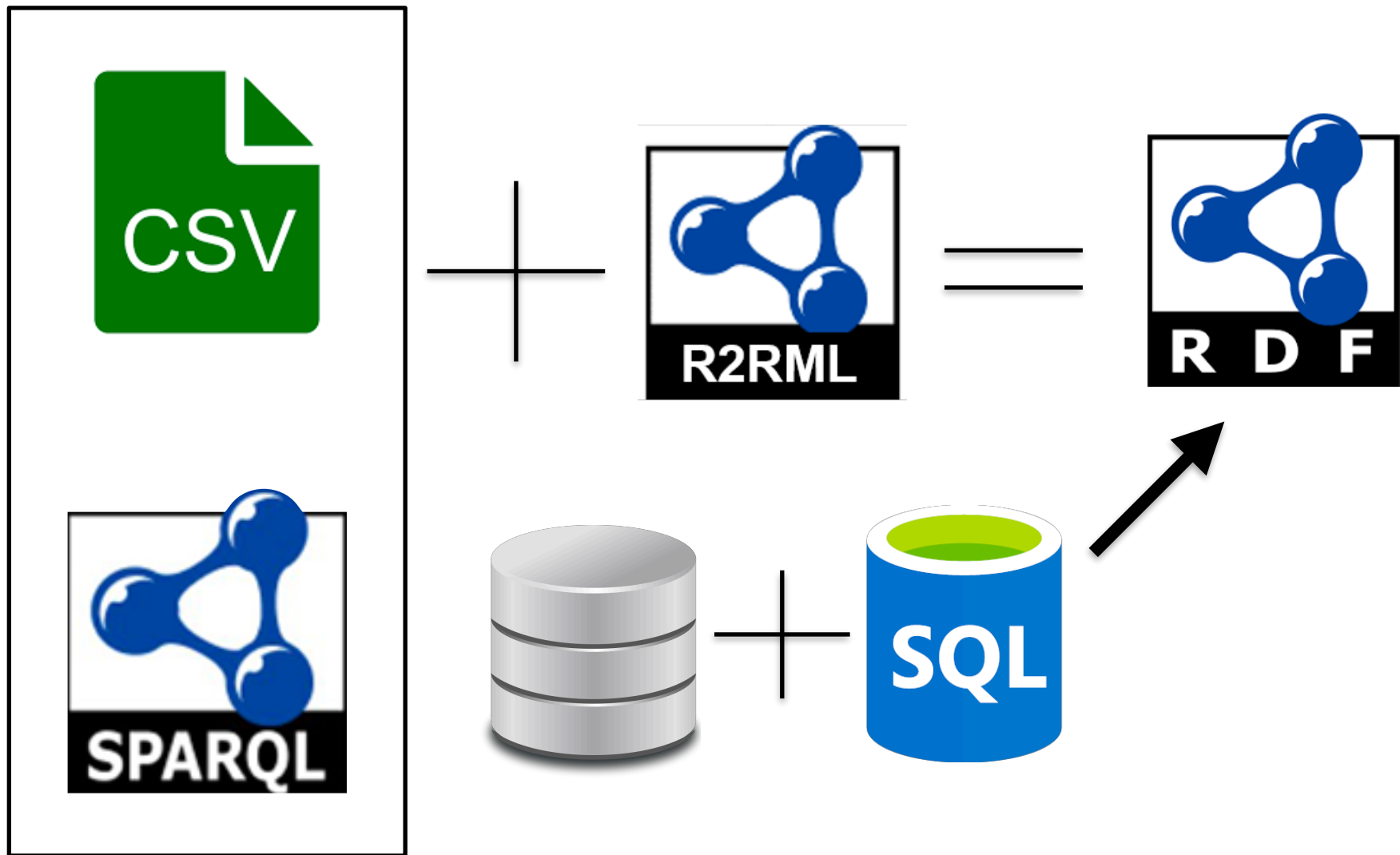












```

SELECT ?comment ?email ?country ?mod ?cour
WHERE {
  ?comment rdf:type ex:Comment;
  schema:author ?author;
  schema:dateModified ?m .
  ?m ex:hasValue ?mod .
  ?author ex:email ?email ;
  ex:hasCountry ?country;
  ex:nOfCourses ?cour }

```



date, username, comment, modifiedDates,nOfLikes	<b>comments.csv</b>
"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	

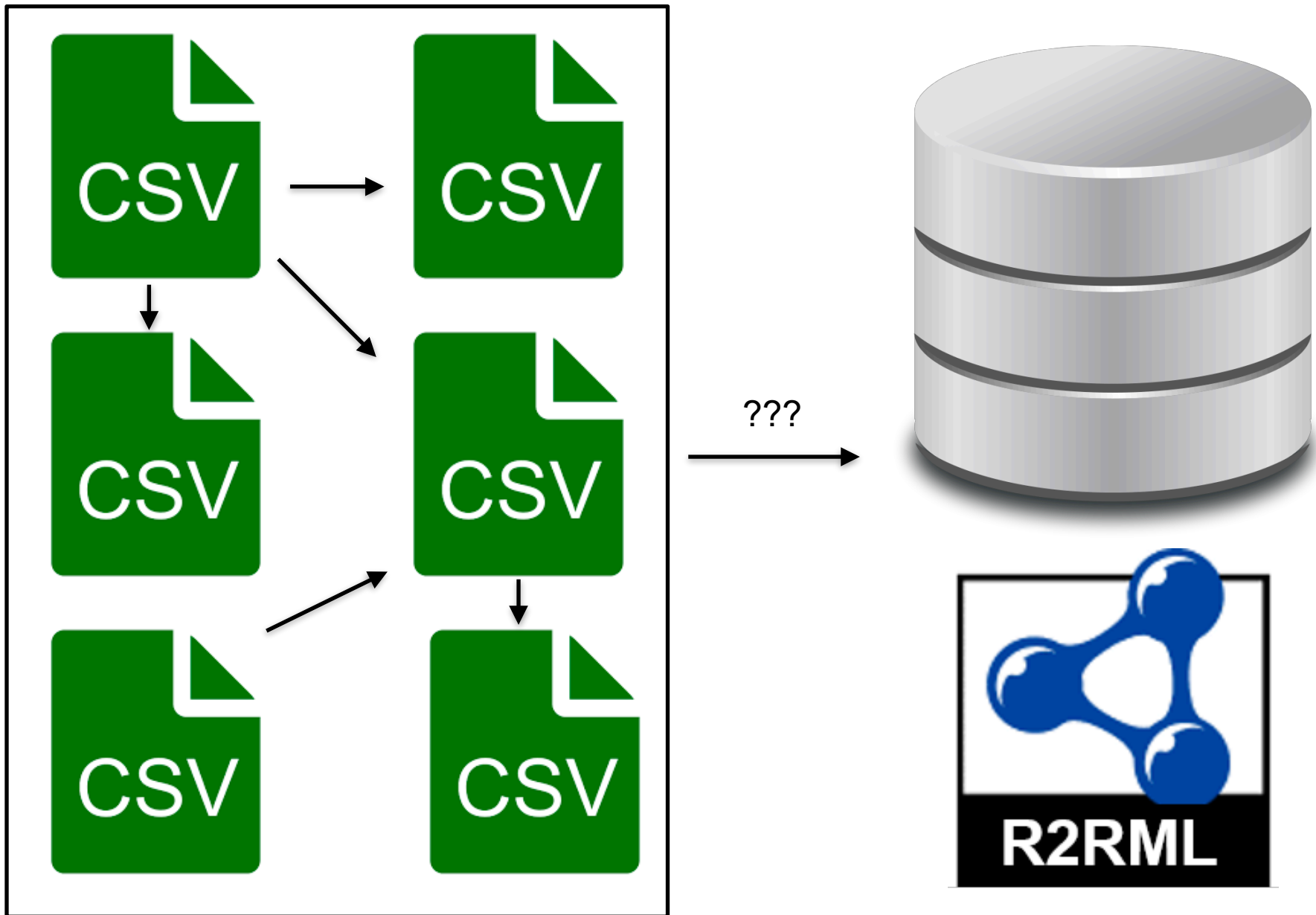
**people.csv**

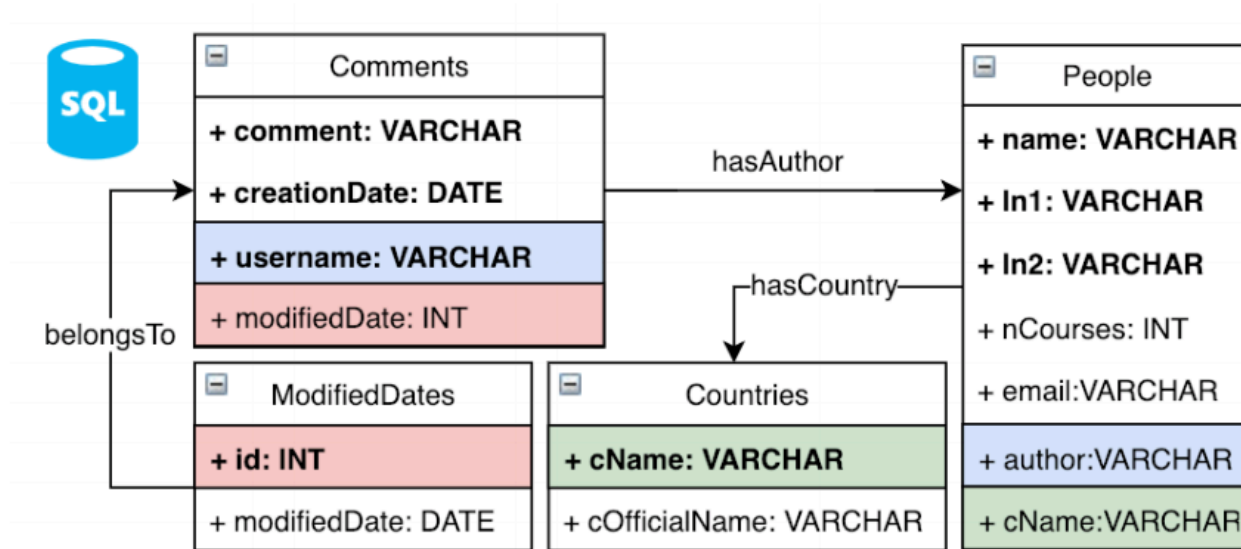
```

      ?comment, ?email, ?country, ?mod, ?cour
"Hallo Dunia", "fpriyatna@fi.upm.es", dbr:Indonesia, 2018-10-01, 0
"Hallo Dunia", "fpriyatna@fi.upm.es", 2018-10-01, dbr:Indonesia, 2018-11-01, 0
"Hola Mundo", "dchaves@fi.upm.es", 2018-10-02, dbr:Spain, 2018-12-04, 3
...

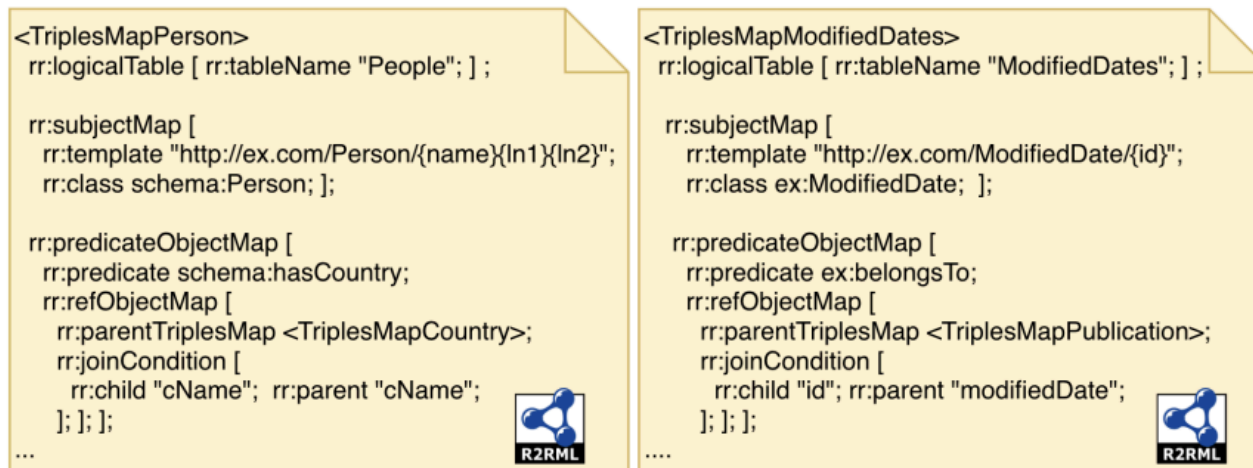
```

...





(a) Enriched RDB schema from CSV files



(b) R2RML mappings that allow exploiting the data in the enriched RDB

## Los CSVs...

- No proporcionan un schema
- No están normalizados
- No tienen un formato normalizado para los datos
- No suelen contener joins explícitos
- Pueden no incorporar algunos [Meta]datos

## Los CSVs...

- No proporcionan un schema
- No están normalizados
- No tienen un formato normalizado para los datos
- No suelen contener joins explícitos
- Pueden no incorporar algunos [Meta]datos

¿Cómo aprovechar las optimizaciones en la traducción de consultas de SPARQL a SQL y el uso de R2RML para el acceso a CSVs?

## Lenguajes de mapping o anotaciones para CSV:

- **RML:** RDF Mapping Language para datos heterogéneos
  - Permite traducción de datos a RDF
  - Múltiples formatos (XML, JSON, RDB, CSV)
- **CSVW:** Metadatos para caracterizar CSVs en la Web (utilizado por Google)
  - Permite traducción de datos a RDF
  - Propiedades específicas del formato
- **The Function Ontology:** Ontología para la definición de funciones (e.g., transformarXtoY)
  - Integración con RML





```

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, $(ln1), o]
  country:
    source:
      - [people.csv~csv]
    s: http://ex.org/country/$(cName)
    po:
      - [a, schema:Country]
      - [ex:officialName, $(cOfficialName)]
  
```

mappings:



```

  person:
    source:
      - [people.csv~csv]
    s: http://ex.org/Person/$(name)$(ln1)$(ln2)
    po:
      - [a, schema:Person]
      - [schema:givenName, $(name)]
      - [schema:familyName, "$(ln1) $(ln2)"]
      - [schema:name, "$(name) $(ln1) $(ln2)"]
      - [ex:nOfCourse, $(nCourses)]
      - p: schema:email
    o:
      - function: sql:lower
      parameters:
        - parameter: sql:valueParameter
        value:
          function: sql:concat
          parameters:
            - parameter: sql:valueParameter
            value:
              function: sql:substring
              parameters:
                - [sql:valueParameter, $(name)]
                - [sql:valueParameter4, "1"]
                - [sql:valueParameter5, "1"]
            - parameter: sql:valueParameter2
            value:
              function: sql:concat
              parameters:
                - [sql:valueParameter, $(ln1)]
                - [sql:valueParameter2, "@fi.upm.es"]
      - p: ex:hasCountry
    o:
      mapping: country
      condition:
        function: equal
        parameters:
          - [str1, "$(name)$(ln1)$(ln2)"]
          - [str2, "$(name)$(ln1)$(ln2)"]
  
```



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



- Definición de funciones de transformación en SQL con  $\text{FnO}_{\text{SQL}}$
- Generación de una base de datos enriquecida usando la información proporcionada por las anotaciones
- Generación de R2RML a partir de RML

## Reglas para la generación de la base de datos enriquecida

- Generación del schema básico
  - Definición de los títulos de las columnas
  - Formalización (csvw:separator)
  - Nuevas columnas con las funciones aplicadas
- Generación de las restricciones
  - Formato de fechas/números
  - Máximos y mínimos
  - NULL y default
- Generación de relaciones
  - Nuevas columnas con las funciones aplicadas
  - PK, FK e índices

## Reglas para la creación de R2RML

- Transformación de las propiedades correspondientes de RML a R2RML (e.g, `rml:logicalSource` to `rr:logicalTable`)
- Propiedades que contienen funciones se transforman a las correspondientes referencias de las columnas creadas en la generación de la RDB

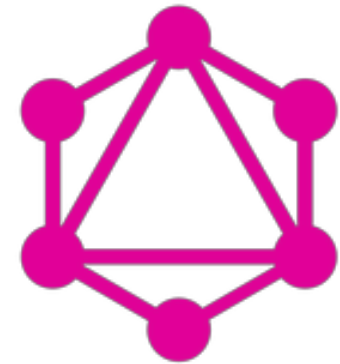
## Resumen:

- Framework de acceso a CSV sin materialización
- Incorporación de la heterogeneidad del formato al proceso
- Uso de tecnologías semánticas del estado del arte
- Aprovechamiento de las optimizaciones de SPARQL-to-SQL
- Uso de cualquier herramienta que procese R2RML
- Nuevo concepto: Mapping Translation

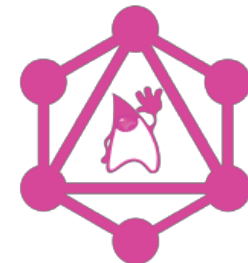
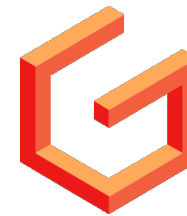
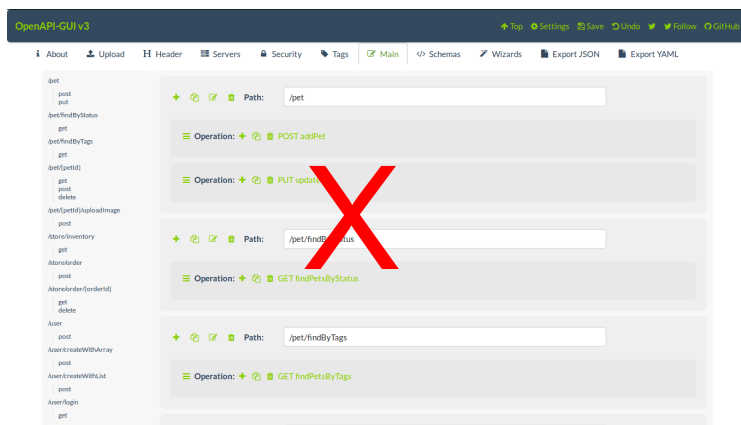
¿Cómo acercar las tecnologías de la Web Semántica a los desarrolladores con el fin de ofrecer acceso unificado a fuentes de datos heterogéneas?



# REST API

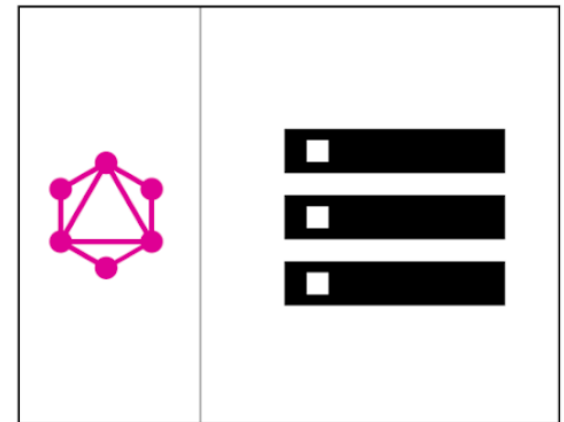
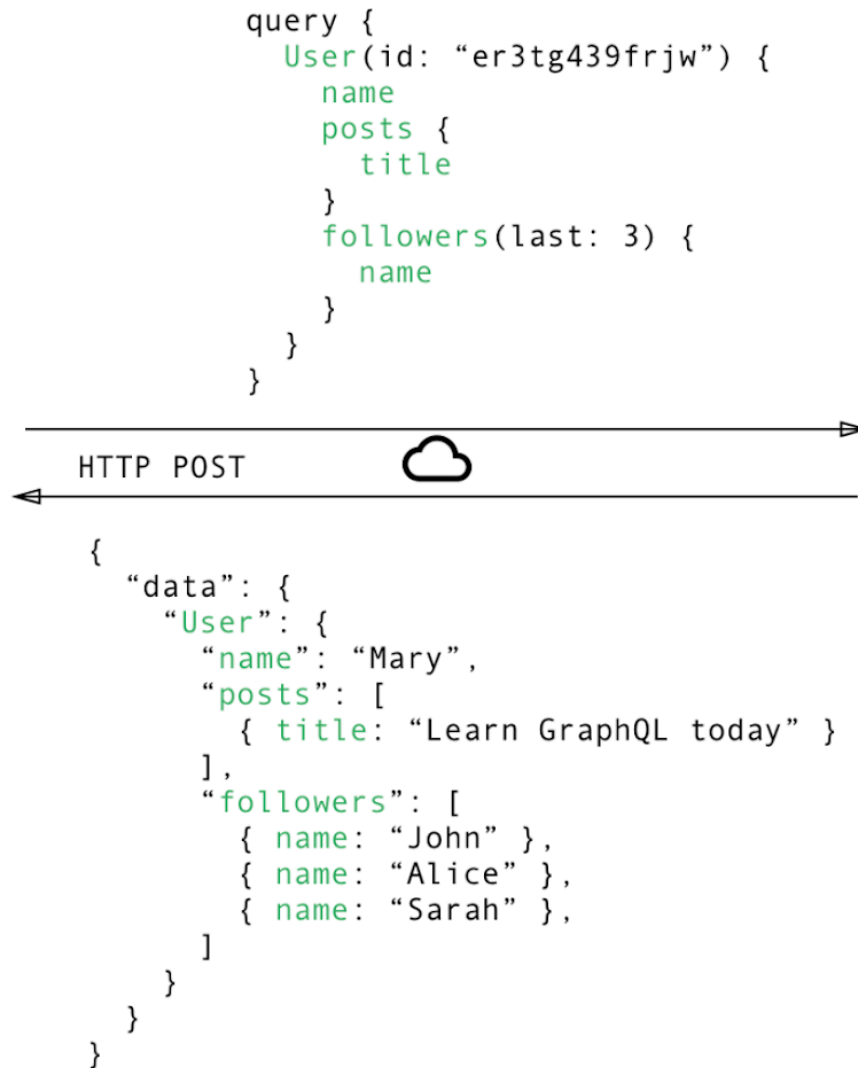
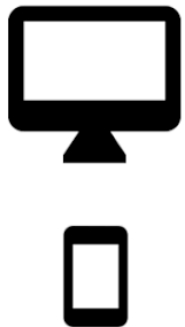


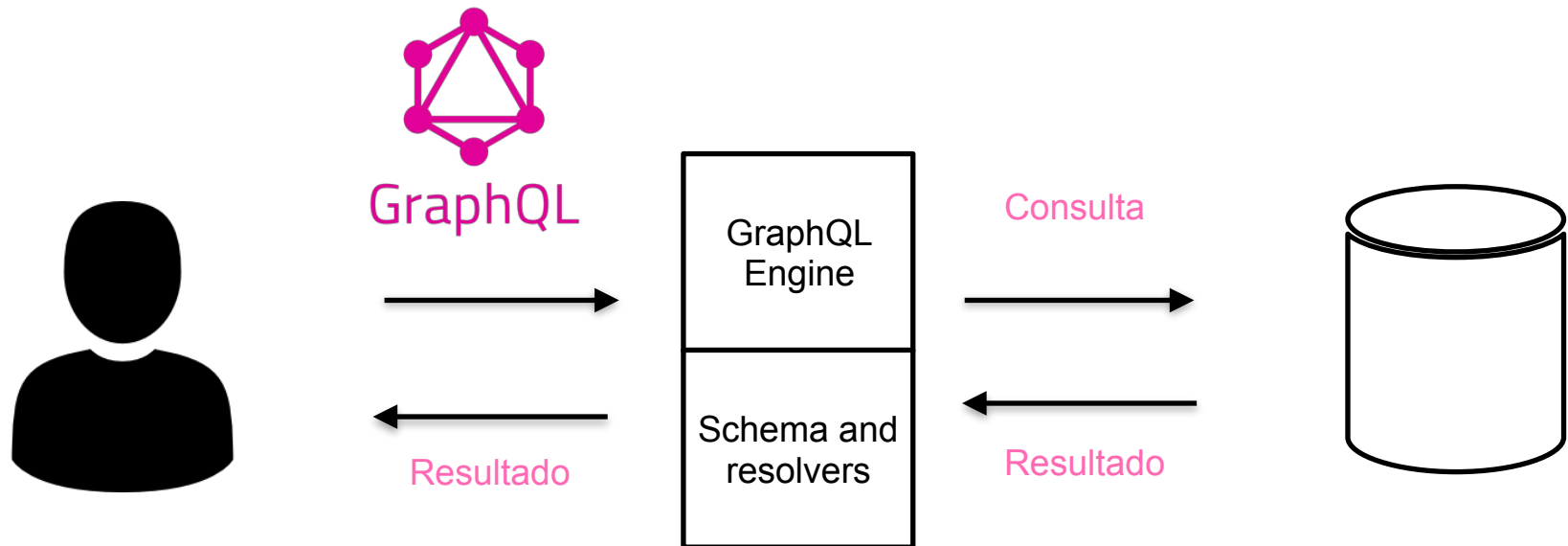
# GraphQL









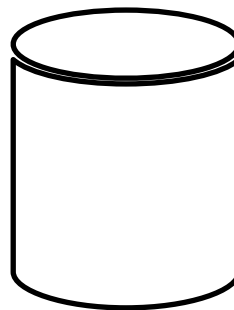
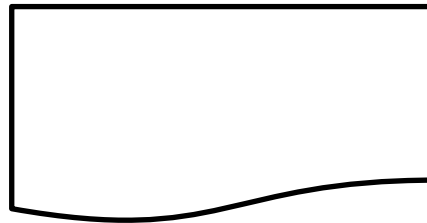
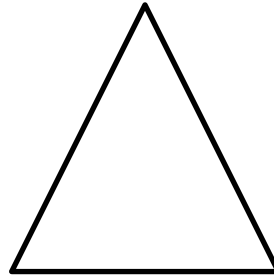


```
schema {  
  query: Query  
  mutation: Mutation  
}  
  
type Query {  
  person(id: String!): Person  
}  
  
type Mutation {  
  createPerson(name: String!,  
    occupation: String): Person  
}
```

```
public Person person(String id) {  
  ...  
}  
  
public Person createPerson(String name,  
  String occupation) {  
  ...  
}
```

# GraphQL

# OBDA



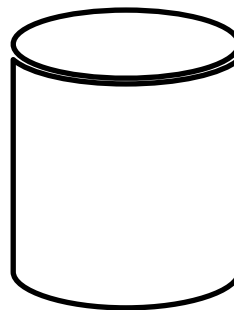
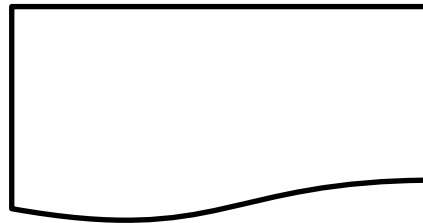
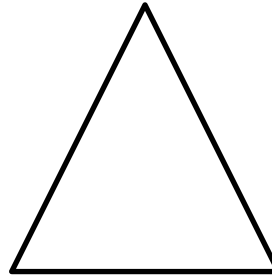
# GraphQL

## Query Language

- GraphQL Query
- Input Structure = Output Structure

## Query Translator

- Various Implementations
- Industry-grade
- Read and Write



# OBDA

## Query Language

- SPARQL
- Input as Graph, Output as Table

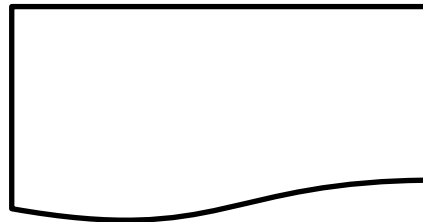
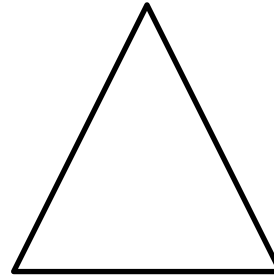
## Query Translator

- Few implementations
- Academic-grade
- Read-only

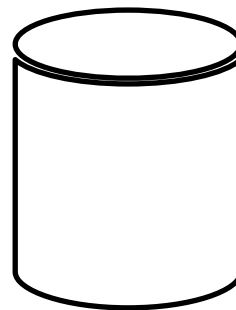
# GraphQL

# OBDA

- Resolvers
- Code Involved
- Non Reusable

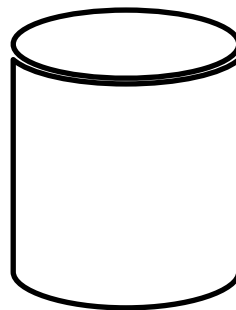
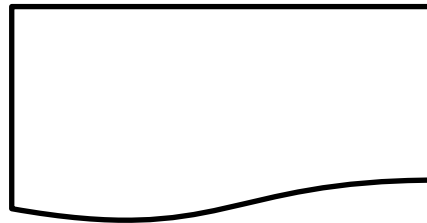
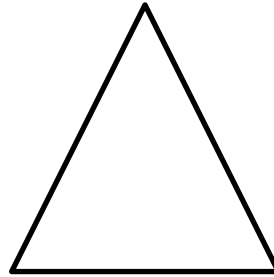


- W3C Standard & Extensions
- No Code Involved
- Reusable



# GraphQL

# OBDA

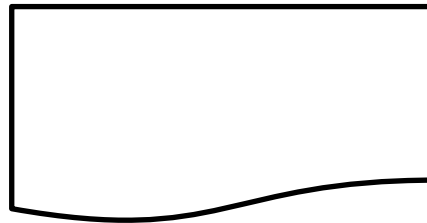
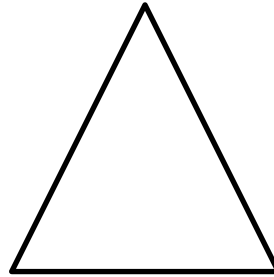


- Anything

- RDB
- CSV/JSON/XML
- MongoDB

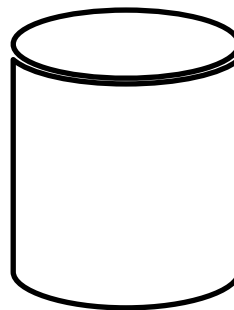


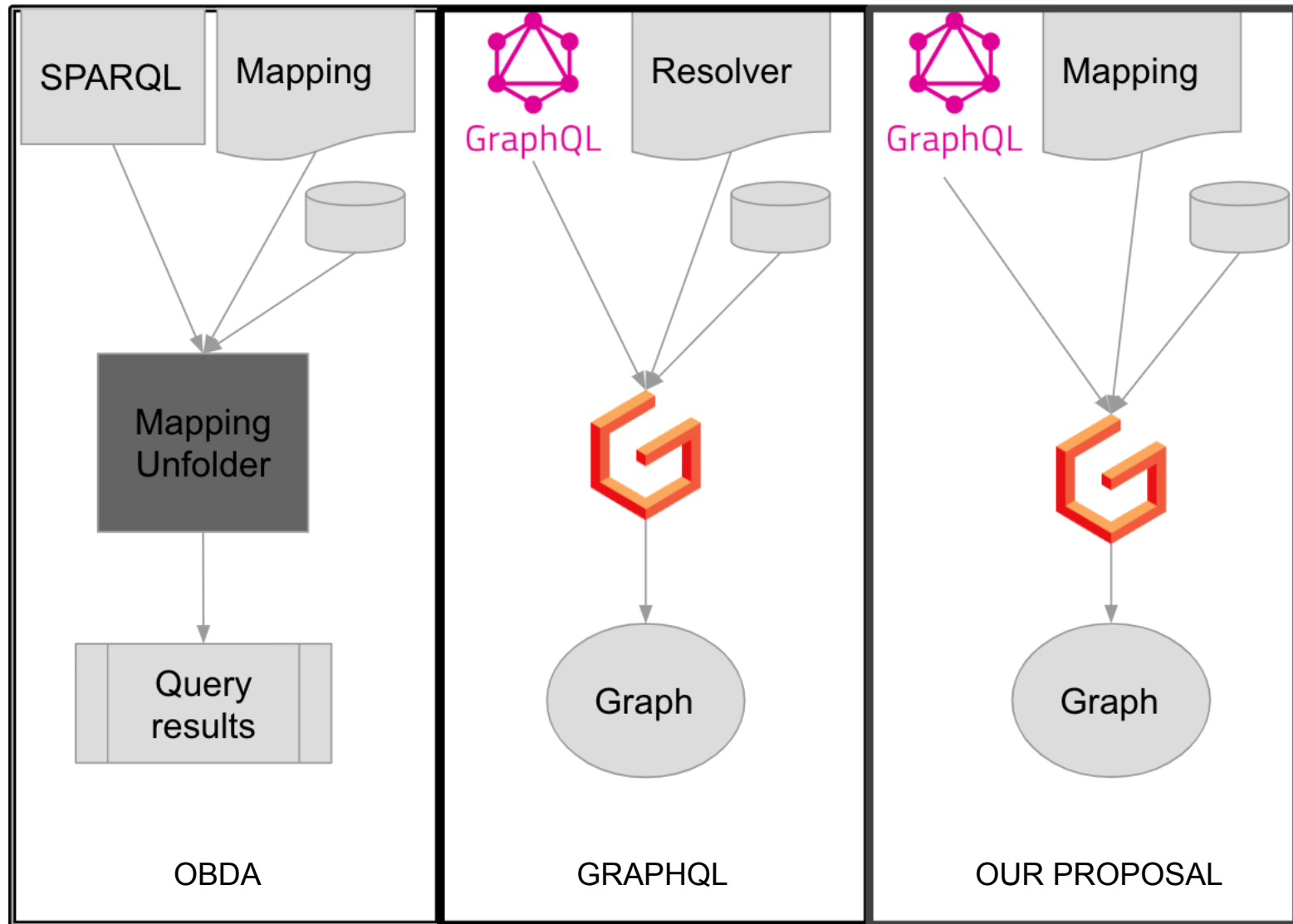
**GraphQL Query**

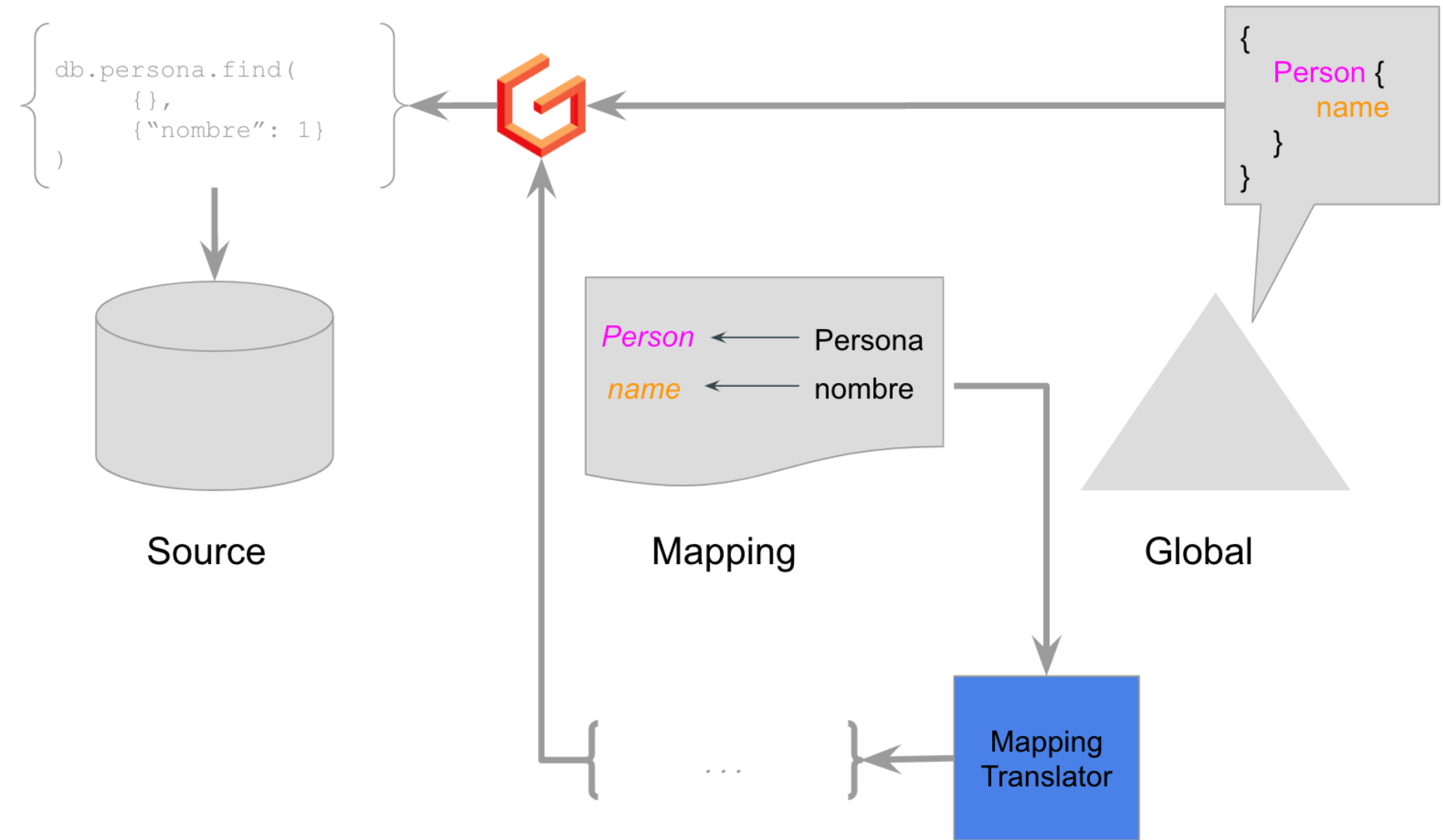


**W3C Standard and extensions**

**Anything**







## Resumen:

- GraphQL y Mappings para desarrolladores
- Generación automática de los GraphQL resolvers
- SPARQL != GraphQL
- Soporte industrial
- Nuevo concepto: Mapping Translation

	JavaScript	Python	Java
MongoDB		Yes	
SQLite	Yes		
CSV	Yes		
JSON			
XML			



# Lenguajes de Mapeo para el acceso a datos heterogéneos

**David Chaves-Fraga, Ontology Engineering Group  
Universidad Politécnica de Madrid, Spain**

Oscar Corcho, OEG-UPM

Freddy Priyatna, OEG-UPM

Ahmad Alobaid, OEG-UPM

Andrea Cimmino, OEG-UPM

✉ [dchaves@fi.upm.es](mailto:dchaves@fi.upm.es)

🐦 [@dchavesf](https://twitter.com/dchavesf)

📅 22-23/01/2019

📍 Reunión Datos 4.0