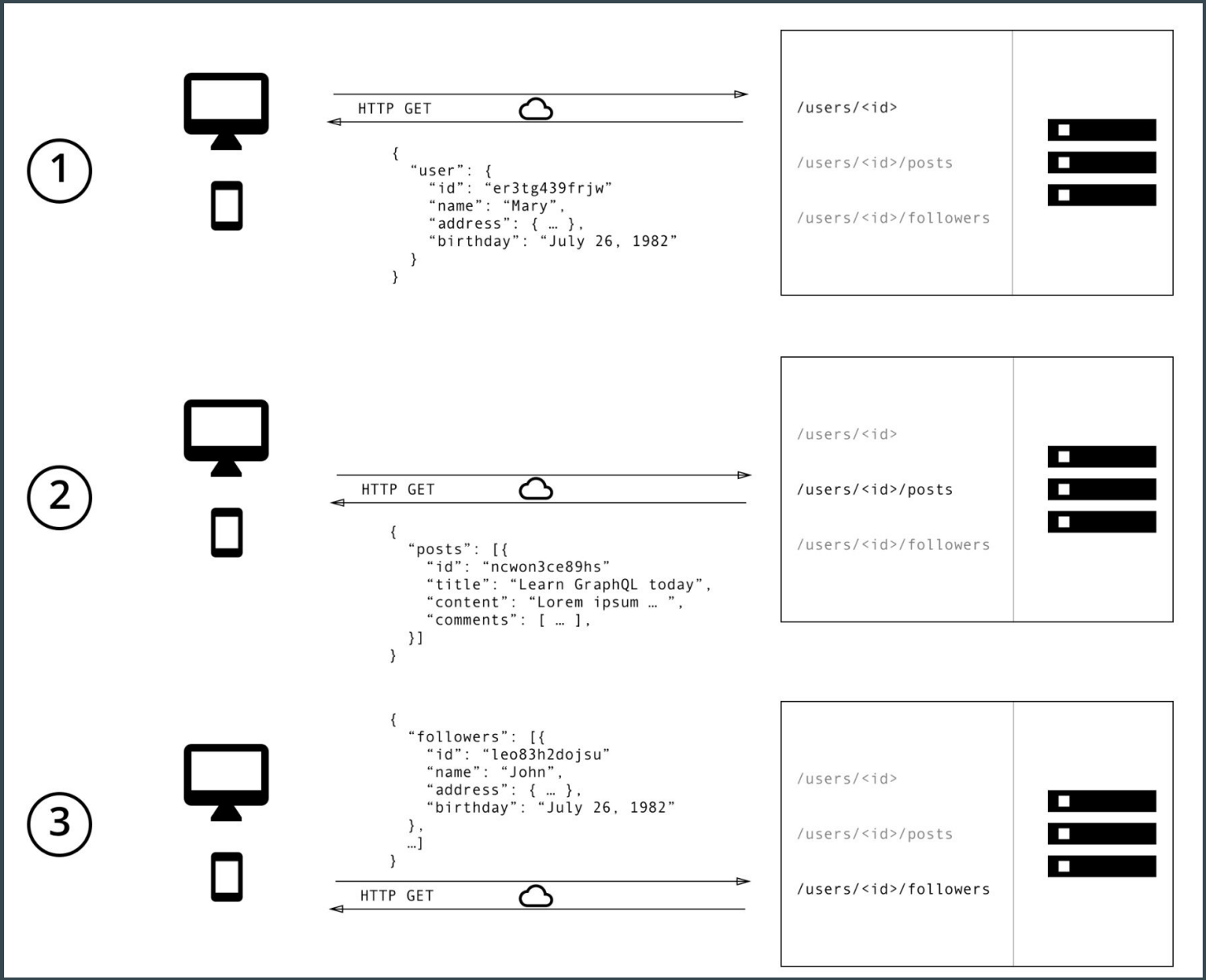
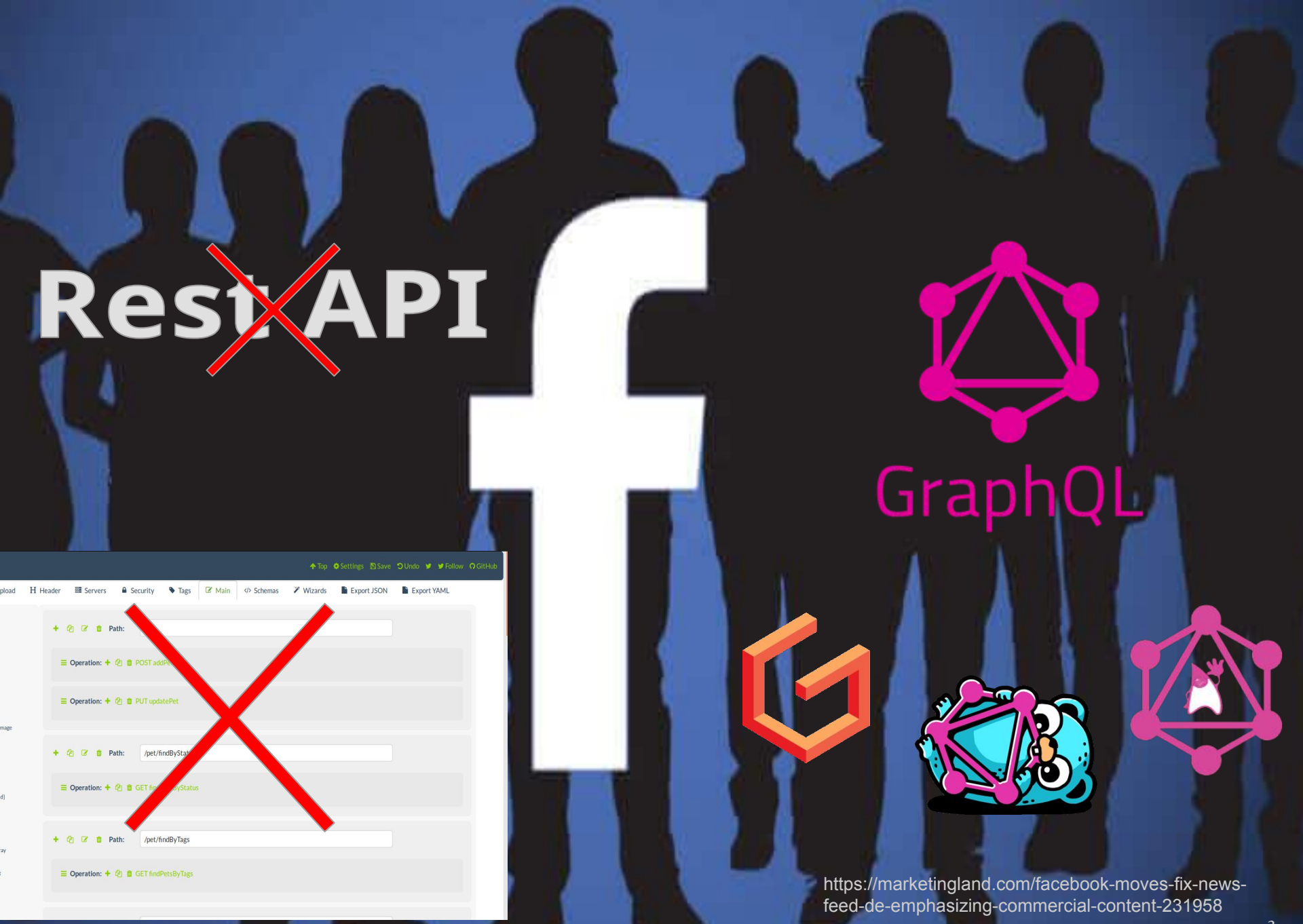
GraphQL-Based Access to Virtual Datasets Exposed by RML Mappings

...

Freddy Priyatna, David Chaves, Ahmad Alobaid, Oscar Corcho
Ontology Engineering Group, UPM, Spain
schema:email <- lower(substr({nombre},1,1) || {apellido} || '@fi.upm.es')
Ghent University, Ghent, Belgium
October 2018

REST: Get the User's full name + Posts + Followers' names





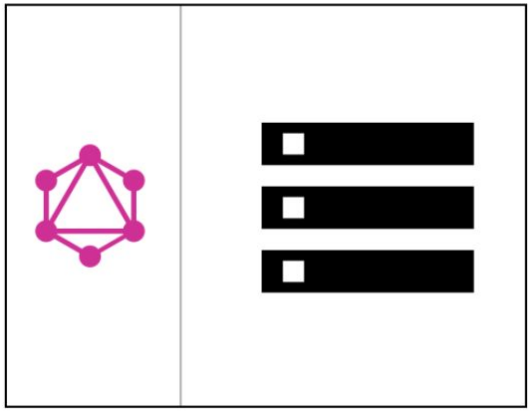
https://marketingland.com/facebook-moves-fix-news-feed-de-emphasizing-commercial-content-231958

https://github.com/Mermade/openapi-gui

Product Hunt
PROJECT SEPTEMBER
PROTEL
1stdibs
20 minutes
adayroi.com
Goalify
GRAPHCMS
GRAPHCO
Quri
REDBUBBLE
Reindex
ALEMBIC
ALLOCINE
AlphaSights
HACKAGES
HASURA
HSL HRT
Amplitude
ANTS
Appier
Restorando
SALE STOCK
scaphold
HIJUP
HOUSING ANYWHERE
hudl
ArangoDB
ART SY
ATLASSIAN
SERVERLESS
sky
Icon Systems
idobata
Indonesia
attendify
bazinga!
Blender Bottle
SKYARCH
SMARKETS
stackshare
Bright
Buildkite
bynder
inerva
INTUIT
Jusbrasi
startups.co
stem
swapcard
cheddar
HD CircleHD
Club Med
KLM
leanIX
LEGENDS OF LEARNING
SYZYGY
taller
Teachers Pay Teachers
Colectica
compara online
Conduit
lelivrescolaire.fr
Lets.events
teselagen
THE HUNT
coursera
credit karma
curio
MAKE SCHOOL
MEDALLIA
Trove
UC Trend
dailymotion
digitransit
METEOR
metric:ai
Mixclou
MyHeritage
NEWS
Unigraph:io
universe
UST Global
Drift
DueDil
eastview christian church
neo4j
NEWSPRING CHURCH
NINGENSOF
vanila.
waldo
wayfair
easy carros
ediket
Nova-Ideo
OK GROW!
OVOS
whitescape
wirtualna polska
wishlife
Fairfax Media
gentux
GetNinjas
PayPal
PERSADO[GO]

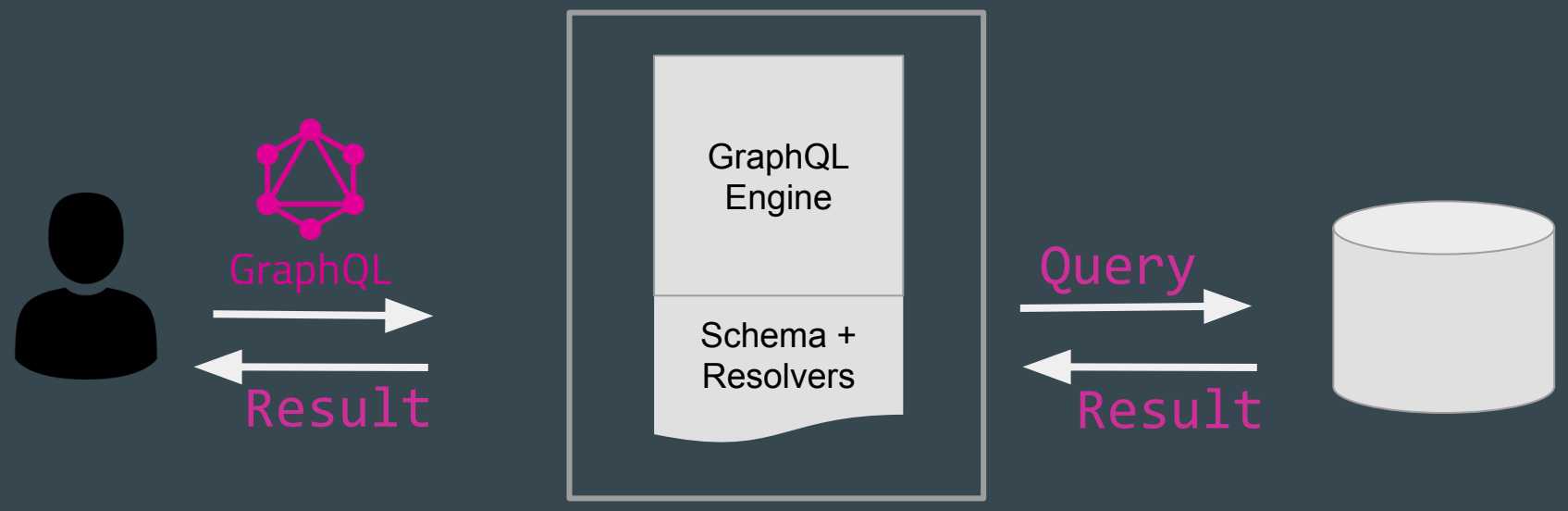
Equivalent GraphQL Example

```
query {
  User(id: “er3tg439frjw”) {
    name
    posts {
      title
    }
    followers(last: 3) {
      name
    }
  }
}
```



```
{
  “data”: {
    “User”: {
      “name”: “Mary”,
      “posts”: [
        { title: “Learn GraphQL today” }
      ],
      “followers”: [
        { name: “John” },
        { name: “Alice” },
        { name: “Sarah” },
      ]
    }
  }
}
```

GraphQL Workflow



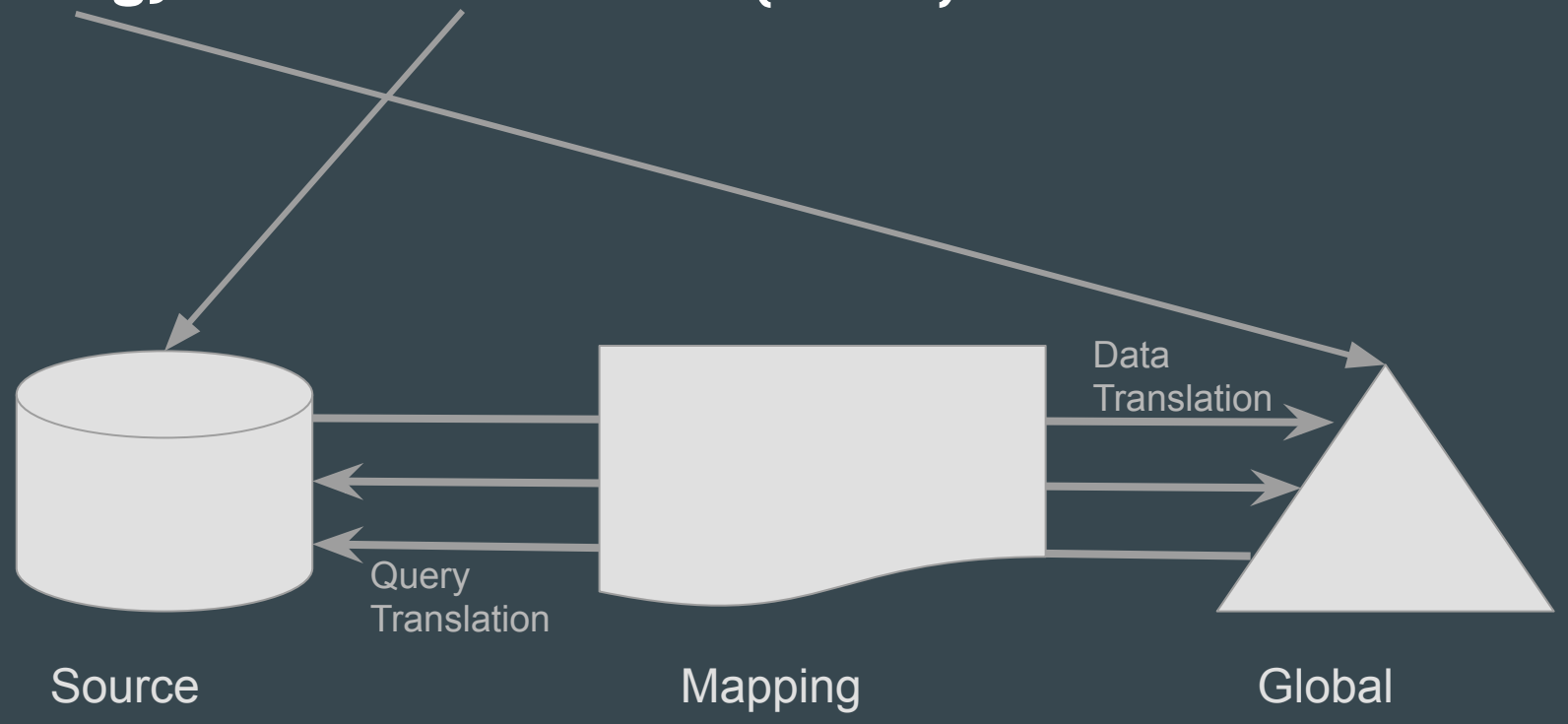
Schema and Resolvers

```
schema {
   query: Query
   mutation: Mutation
}

type Query {
   person(id: String!): Person
}

type Mutation {
   createPerson(name: String!,
      occupation: String): Person
}
```

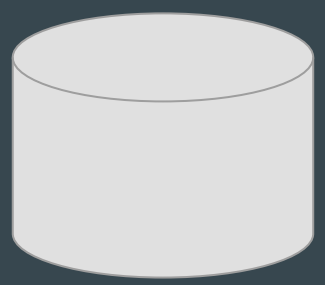
```
public Person person(String id) {
   ...
}

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

OBDA

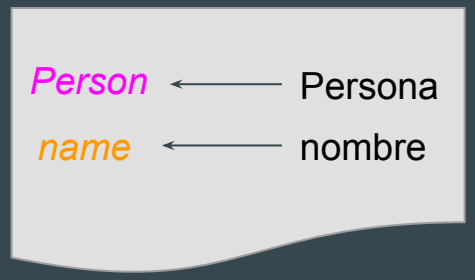
Ontology Based Data Access (OBDA)



OBDA Techniques: Data Translation



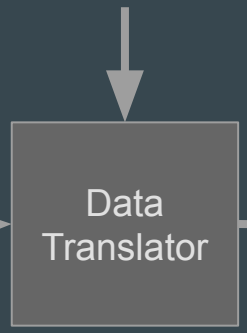
Source



Mapping

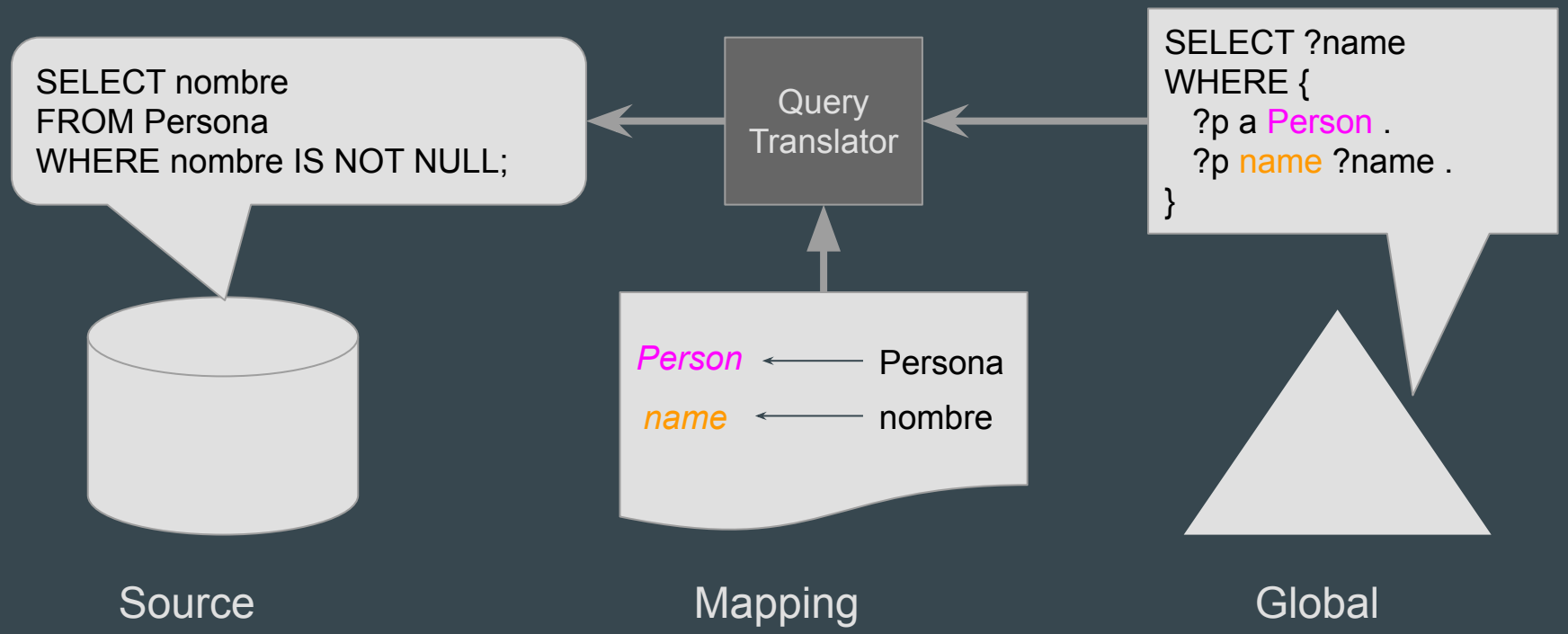
Global

Persona		
id	**nombre**	**ocupación**
1	Ahmad	estudiante
2	Freddy	investigador
3	Oscar	profesor




```
_:1 a Person .
_:1 name "Ahmad" .

_:2 a Person .
_:2 name "Freddy" .

_:3 a Person .
_:3 name "Oscar" .
```

OBDA Techniques: Query Translation



Persona		
id	**nombre**	**ocupación**
1	Ahmad	estudiante
2	Freddy	investigador
3	Oscar	profesor

State of the Art

Relational Databases

R2RML

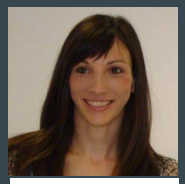
Data Translation ✓

Query Translation ✓

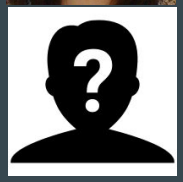
Semi-structured data (XML, Json, CSV)

RML

Data Translation ✓



Query Translation ✗



CSV with functions

RMLC

Data Translation ✓

Query Translation ✓

Questions Time ...

- What are the **similarities** between GraphQL and OBDA?

- What are the **differences** between GraphQL and OBDA?

GraphQL vs OBDA (Similarities)

GraphQL vs OBDA (Differences - Global Layer)

- Query Language
  - GraphQL Query
  - Input Structure = Output Structure
- Query Translator
  - Various Implementations
  - Industry-grade
  - Read and Write

- Query Language
  - SPARQL
  - Input as Graph, Output as Table
- Query Translator
  - Few Implementations
  - Academic-grade
  - Read Only

GraphQL vs OBDA (Differences - Mapping Layer)

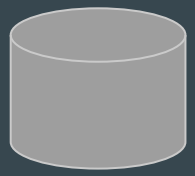
- Resolvers
- Code Involved
- Non Reusable

- W3C Standard & Extensions
- No Code Involved
- Reusable

GraphQL vs OBDA (Differences - Data Layer)

- Anything*

**As long as it is supported by the resolver*



- RDB
- CSV/JSON/XML
- MongoDB

GraphQL and OBDA: A Proposal

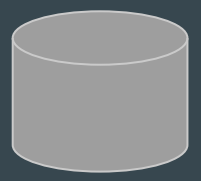
GraphQL Query

SPARQL

Resolvers
(custom code)

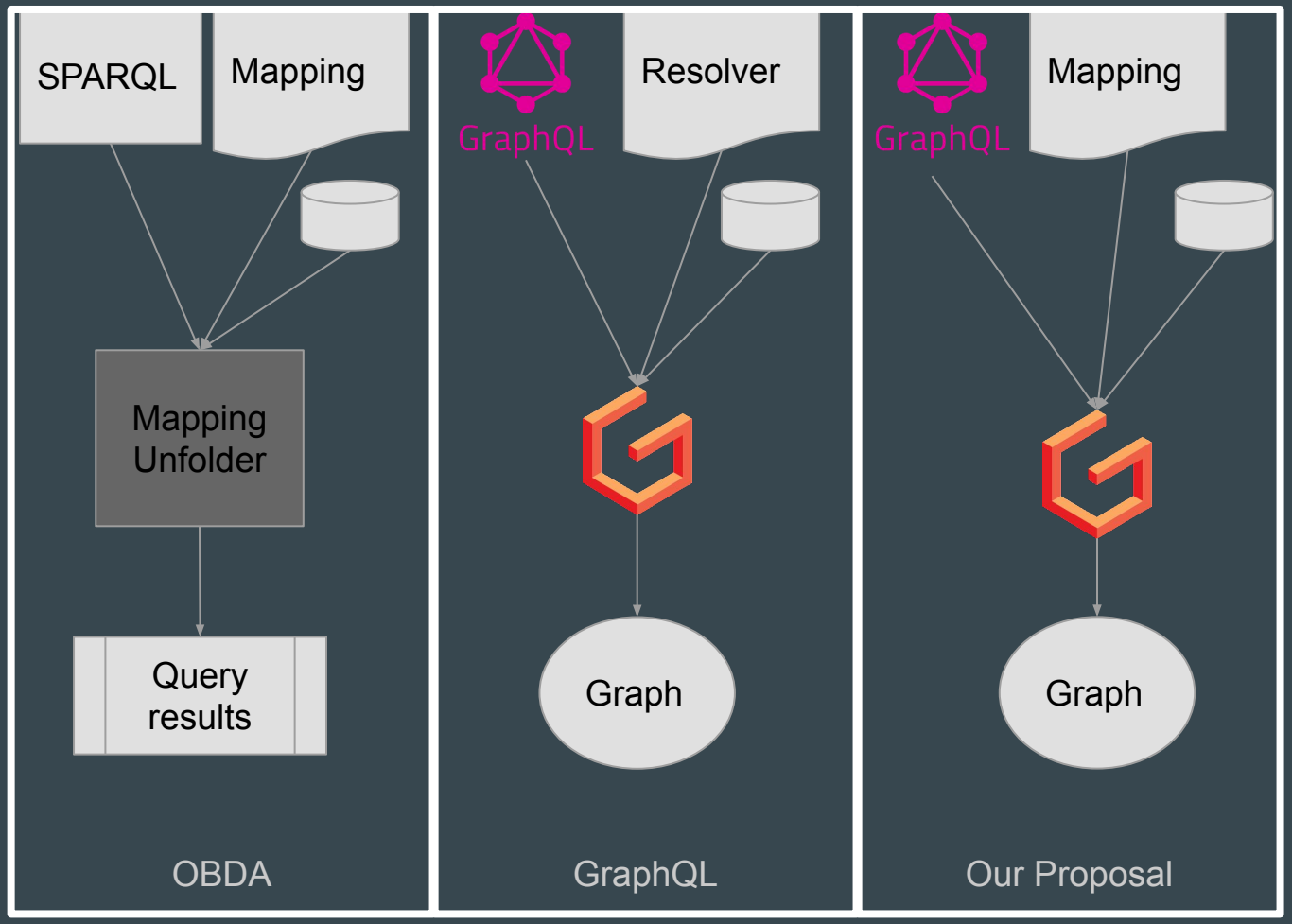
W3C Standard and its extensions

Anything *

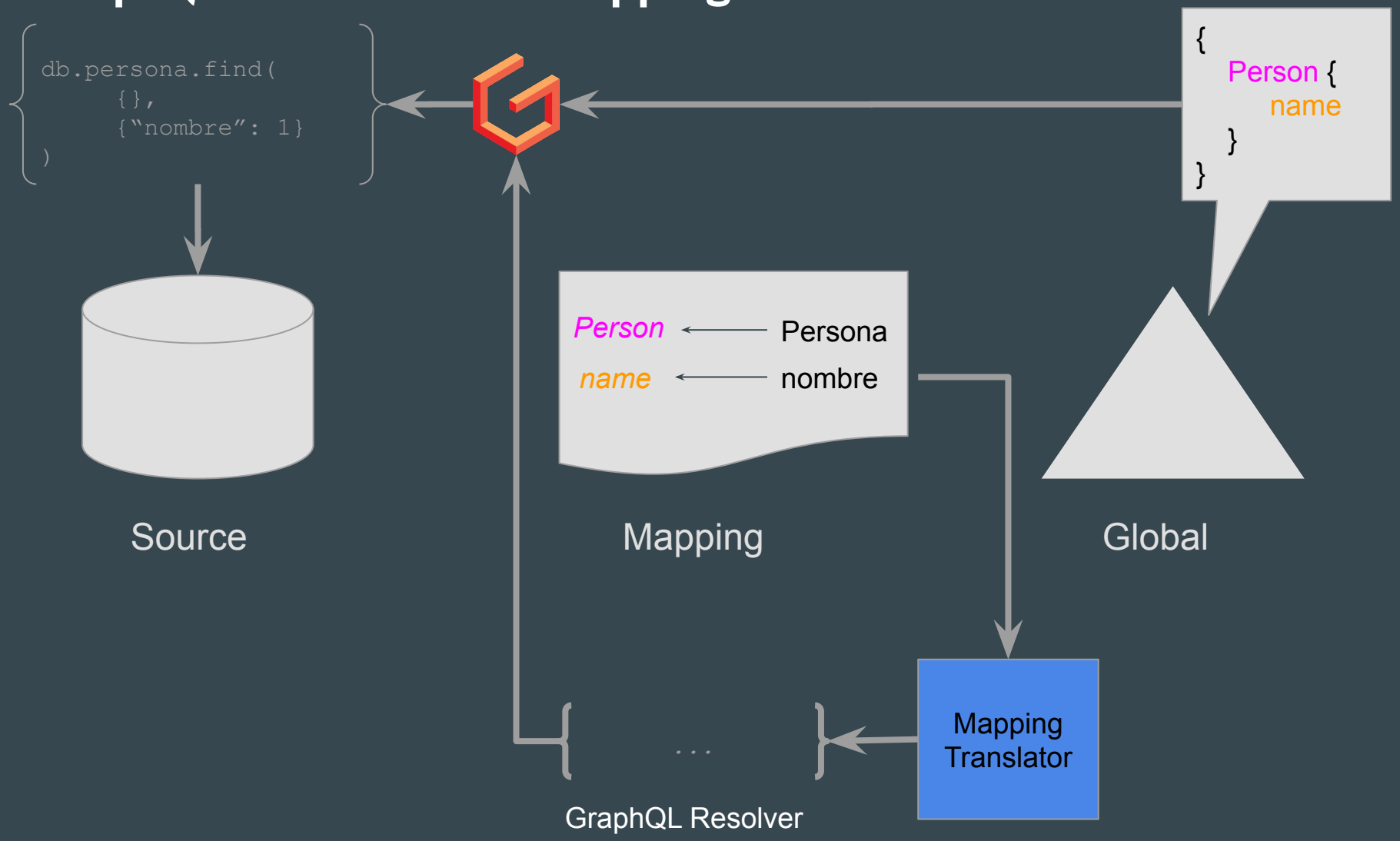


RDB
CSV/JSON/XML
MongoDB

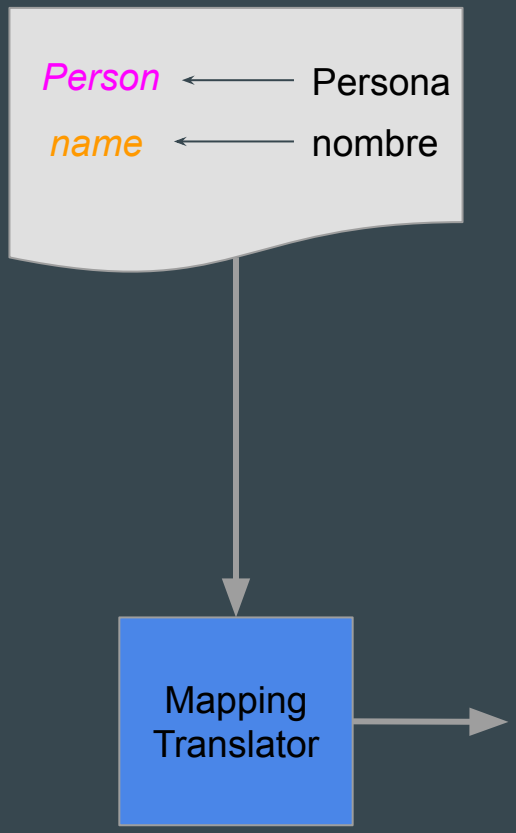
Different Query Translation Workflows



GraphQL-based OBDA: Mapping Translator



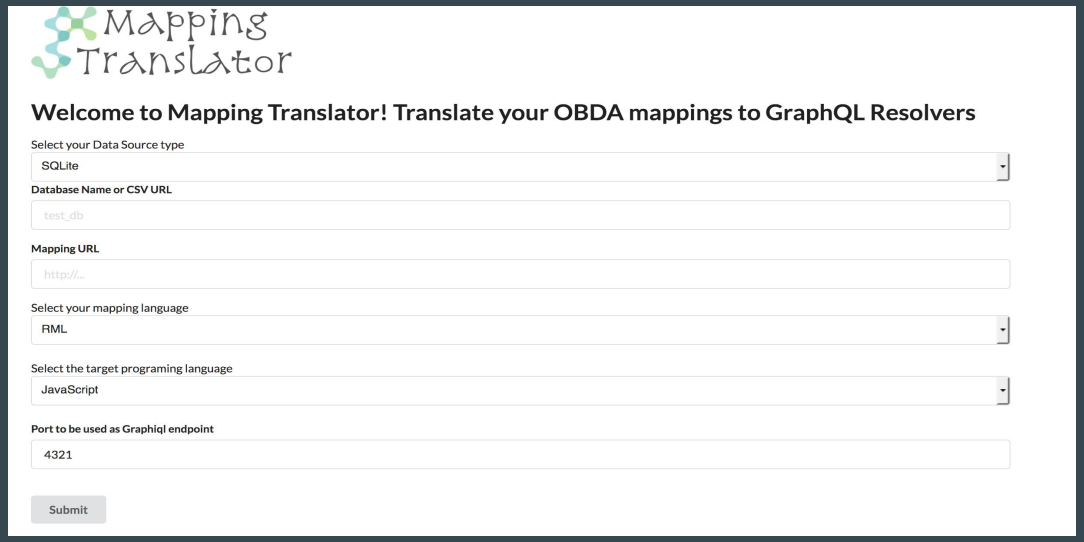
GraphQL-based OBDA: Mapping Translator (Java example)




```java
public class PersonResolver {
    private final MongoCollection<Document> persons;

    //map Person to persona
    static final String COLLECTION_SOURCE = "persona";

    public void savePerson(Person person) {
        Document doc = new Document();

        //map Person.name to personas.nombre
        doc.append("nombre", person.getName());

        //map Person.occupation to personas.ocupacion
        doc.append("ocupacion", person.getOccupation());
        persons.insertOne(doc);
    }
```

Current Status

- Implementation available at: https://github.com/oeg-upm/mapping-translator
- Vocabulary supported: schema.org
- Functionalities: read, write, filter
- Mapping Expressivity:
  - Column: name <- nombre
  - Template: name <- {nombre} || {apellido}
  - Function: email <- lower(substr({nombre},1,1) || {apellido} || '@fi.upm.es')
- Join between multiple mappings: Mapping Person and Mapping Posts



What do we have

	Ghent	**OEG**
Language	RML	RMLC, RMLC-Iterator
Approach	Linked Data Generation	Access via Query
Function implementation (how)	Programming Language (Java, Python, ...)	RDB built-in functions
Function definition (where)	Fno	Inside mappings
Means of supporting vocabularies	Multiple & Explicit with GraphQL-LD	Single & Implicit with schema.org
Translation	GraphQL to SPARQL (SPARQL to GraphQL?)	RML[C] to GraphQL Resolvers
Mapping Editor	Matey, RML Editor	Simple OME

Work in Progress

- Support for multiple vocabularies
- Join between multiple mappings: Mapping Person and Mapping Country
- Support for more dataset types: XML, JSON, etc
- Join between multiple datasource types: Person in CSV and Country in XML
- Support for more programming languages: Java, Scala, etc

	JavaScript	Python	Java
MongoDB		Yes	
SQLite	Yes		
CSV	Yes		
JSON			
XML			

Discussion

How do we collaborate?

- SPARQL to GraphQL?
- GraphQL-LD
- ...