GraphQL-Based Access to Virtual Datasets Exposed by RML Mappings

Freddy Priyatna, David Chaves, Ahmad Alobaid, Oscar Corcho
Ontology Engineering Group, UPM, Spain

\[
\text{schema:email} \leftarrow \text{lower} (\text{substr} (\text{name}, 1, 1) \parallel \text{apellido} \parallel '@\text{fi.upm.es}')
\]

Ghent University, Ghent, Belgium
October 2018
REST: Get the User’s full name + Posts + Followers’ names

1. HTTP GET

```
{  
  "user": {  
    "id": "er3tg439frjw"  
    "name": "Mary",  
    "address": { _ },  
    "birthday": "July 26, 1982"  
  }  
}
```

2. HTTP GET

```
{  
  "posts": [{  
    "id": "ncw0n3c89hjs"  
    "title": "Learn GraphQL today",  
    "content": "Lorem ipsum ...",  
    "comments": [ ... ],  
  }]  
}
```

3. HTTP GET

```
{  
  "followers": [{  
    "id": "leo83h2dojsu"  
    "name": "John",  
    "address": { _ },  
    "birthday": "July 26, 1982"  
  },  
  ...  
}  
```
Equivalent GraphQL Example

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

HTTP POST

```
{
  "data": {
    "User": {
      "name": "Mary",
      "posts": [
        { "title": "Learn GraphQL today" }
      ],
      "followers": [
        { name: "John" },
        { name: "Alice" },
        { name: "Sarah" }
      ]
    }
  }
}
```
GraphQL Workflow
Schema and Resolvers

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

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

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

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

type Mutation {
    createPerson(name: String!, occupation: String): Person
}
```
OBDA
Ontology Based Data Access (OBDA)

Source

Mapping

Global

Data Translation

Query Translation
OBDA Techniques: Data Translation

Source

<table>
<thead>
<tr>
<th>id</th>
<th>nombre</th>
<th>ocupación</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ahmad</td>
<td>estudiante</td>
</tr>
<tr>
<td>2</td>
<td>Freddy</td>
<td>investigador</td>
</tr>
<tr>
<td>3</td>
<td>Oscar</td>
<td>profesor</td>
</tr>
</tbody>
</table>

Mapping

Persona

Persona

Global

_:1 a Person .
_:1 name “Ahmad” .
_:2 a Person .
_:2 name “Freddy” .
_:3 a Person .
_:3 name “Oscar” .
OBDA Techniques: Query Translation

Source

```
SELECT nombre
FROM Persona
WHERE nombre IS NOT NULL;
```

Mapping

```
SELECT ?name
WHERE {
  ?p a Person .
  ?p name ?name .
}
```

Global

```
SELECT ?name
FROM Persona
WHERE nombre IS NOT NULL;
```

<table>
<thead>
<tr>
<th>Persona</th>
<th>id</th>
<th>nombre</th>
<th>ocupación</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>Ahmad</td>
<td>estudiante</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Freddy</td>
<td>investigador</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Oscar</td>
<td>profesor</td>
</tr>
</tbody>
</table>
State of the Art

Relational Databases

Semi-structured data (XML, JSON, CSV)

CSV with functions

Data Translation

Query Translation

R2RML

RML

RMLC
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

- SPARQL Mapping
- OBDA Mapping Unfolder
- Query results

- Resolver
- GraphQL
- Graph

- Mapping
- GraphQL
- Graph
- Our Proposal
GraphQL-based OBDA: Mapping Translator

```
{ 
  db.persona.find( 
    {},
    {"nombre": 1}
  )
}
```

```
{ 
  Person {
    name
  }
}
```
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

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

<table>
<thead>
<tr>
<th></th>
<th>JavaScript</th>
<th>Python</th>
<th>Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>MongoDB</td>
<td></td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SQLite</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSV</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSON</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XML</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Discussion

How do we collaborate?

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