Towards Linked Data

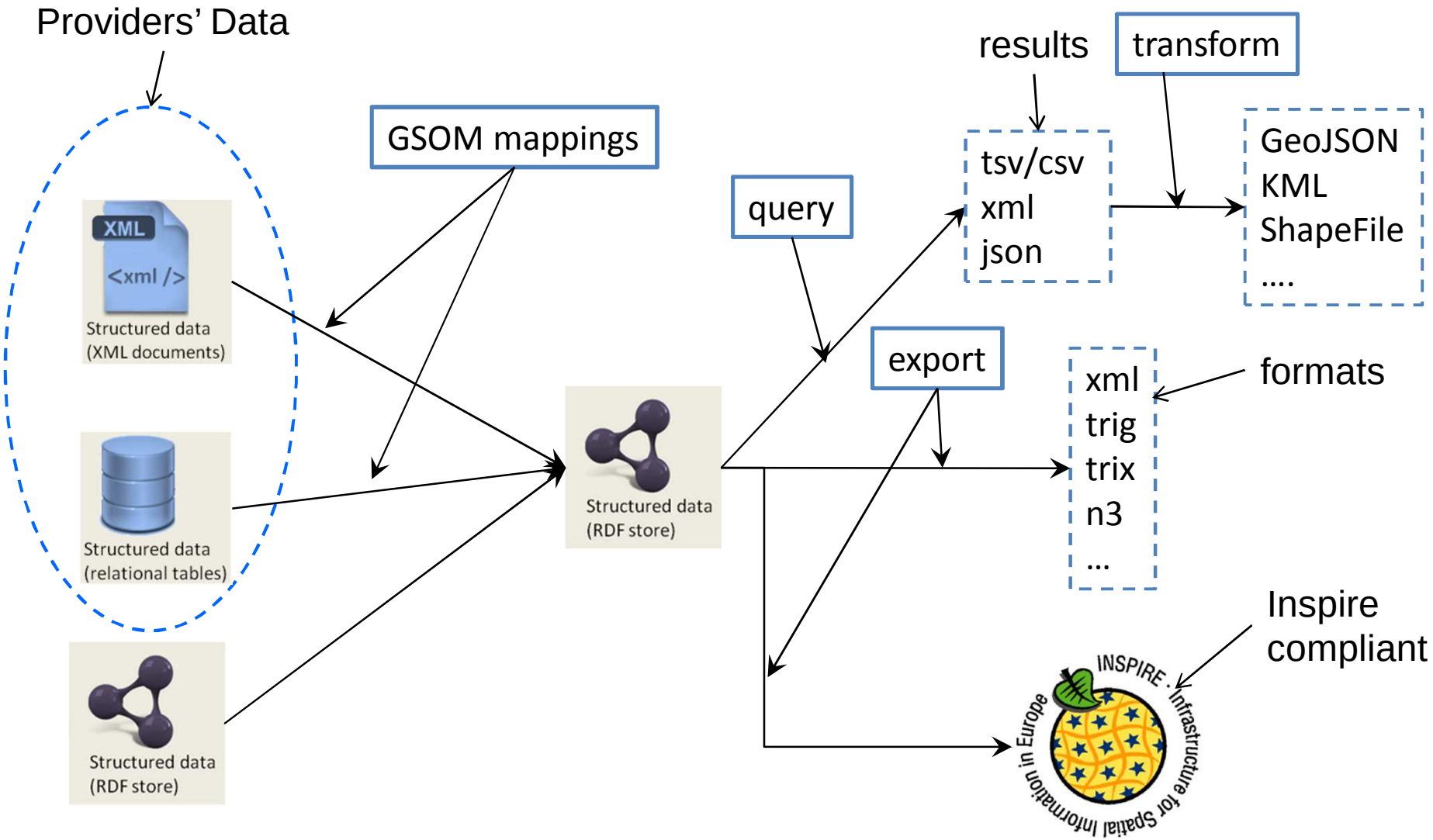
Yannis Roussakis, Athina Kritsotaki, Martin Doerr, Gerald Hiebel, Kyriakos Kritikos, Vassilis Christophides, Dimitris Kotzinos

FORTH-ICS

Outline

- Transformation into Linked Data
- Querying the Linked Data
- Export Linked Data into INSPIRE

Overview


Providers' Data
GSOM mappings
query
results
transform
tsv/csv
xml
json
GeoJSON
KML
ShapeFile
....
export
formats
xml
trig
trix
n3
...
Inspire
compliant
XML
<xml />
Structured data
(XML documents)
Structured data
(relational tables)
Structured data
(RDF store)
Structured data
(RDF store)
INSPIRE - Infrastructure for Spatial Information in Europe

Linked Data Transformation Steps

- In this project all providers were offering XML/relational data
  - we focused with their appropriate transformation and synchronization into RDF data
- In order to transform providers' data into Linked Data w.r.t. the GSOM we have to follow the next steps:
  - Separate the schema-level information from the data-level information
  - Create appropriate mappings from the schema notions to the GSOM notions (considering the GSOM scope notes) :
    - Relational-to-Linked data mappings have to be formalized using R2RML language (http://www.w3.org/TR/r2rml/)
    - XML-to-Linked data mappings have to be formalized using XSLT

Linked Data Transformation Steps

- Providers' data transformation (cont'd)
  - Connect Virtuoso Triple Store (http://virtuoso.openlinksw.com) to the relational data source either directly by copying the relational data from that source or indirectly by linking to the source Relational Database (e.g., JDBC). (only applies to R2RML mapping)
  - Register the R2RML mappings into Virtuoso using the ***addR2RMLMappings*** API method when dealing with Relational source data
  - Apply the XSL mappings using the ***addXSLMappings*** API method when dealing with XML source data and create the respective RDF data

Mapping Methods Benefits

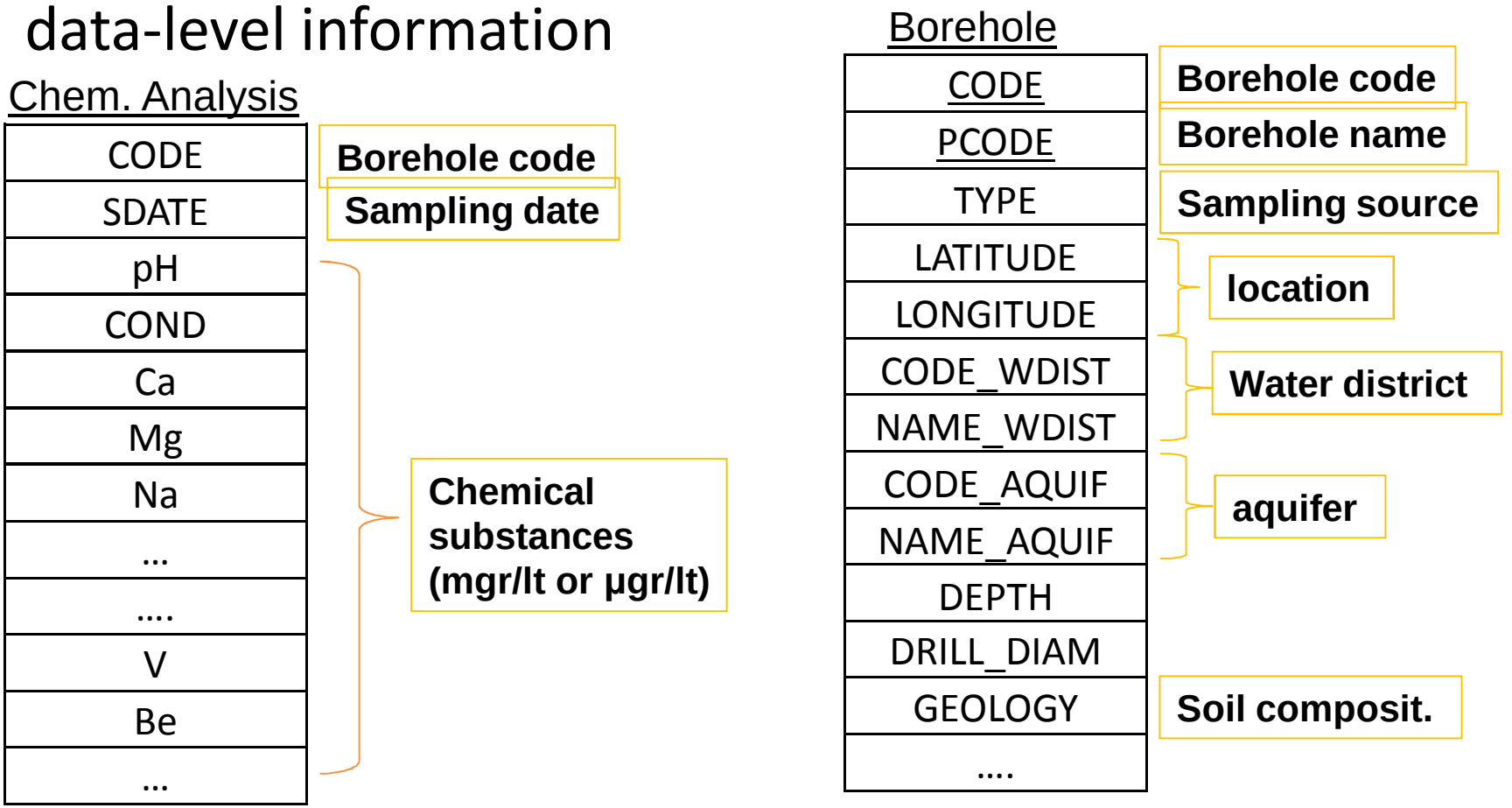
- After one of ***addXSLMappings***, ***addR2RMLMappings*** methods is called, the corresponding Linked Data are created and imported into Virtuoso
- One benefit of the use of R2RML (and Relational Data) is that the Linked Data are automatically updated when changes over the source data occur
- The use of XSLT is recommended if provider's data are in XML and only new insertions are performed on the original XML data.
  - Modifications/deletes should be accompanied with corresponding SPARUL statements by calling the ***ldupdate*** API method to update accordingly the respective RDF Data

R2RML-based Linked Data Transformation Scenario (I)

- We will show how the previously defined steps (for R2RML-based mappings) can be followed to create and upload the corresponding Linked Data for a specific data provider – IGMEM
- IGMEM data
  - Original data are stored in relational tables
  - Ground waters sample takings taken from the regions of Mygdonia and Thriasio in Greece
  - Chemical analyses results over sample takings
  - Information about the sampling date and origin (e.g., boreholes, springs, etc)

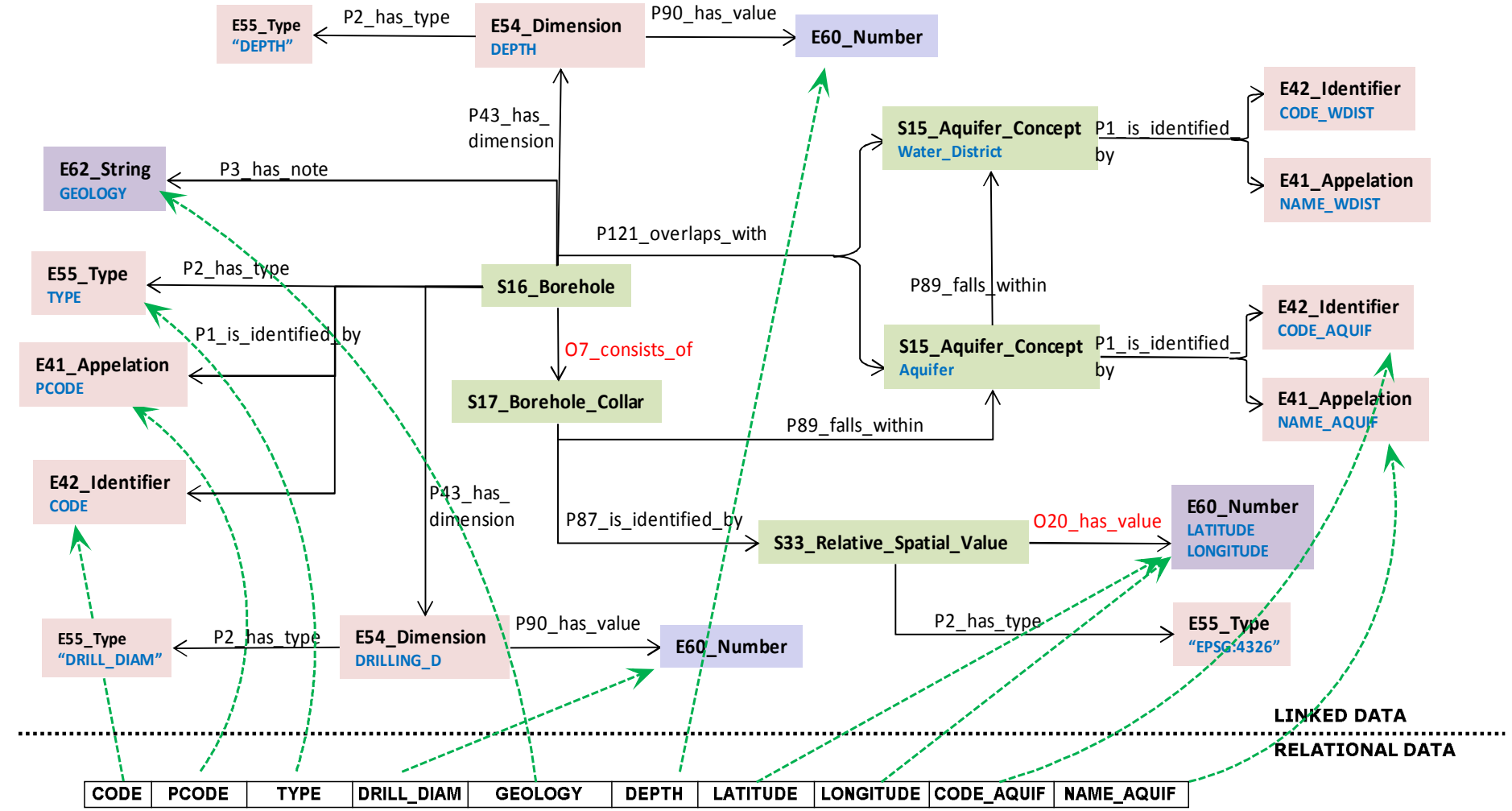
R2RML-based Linked Data Transformation Scenario (II)

- Separate the schema-level information from the data-level information



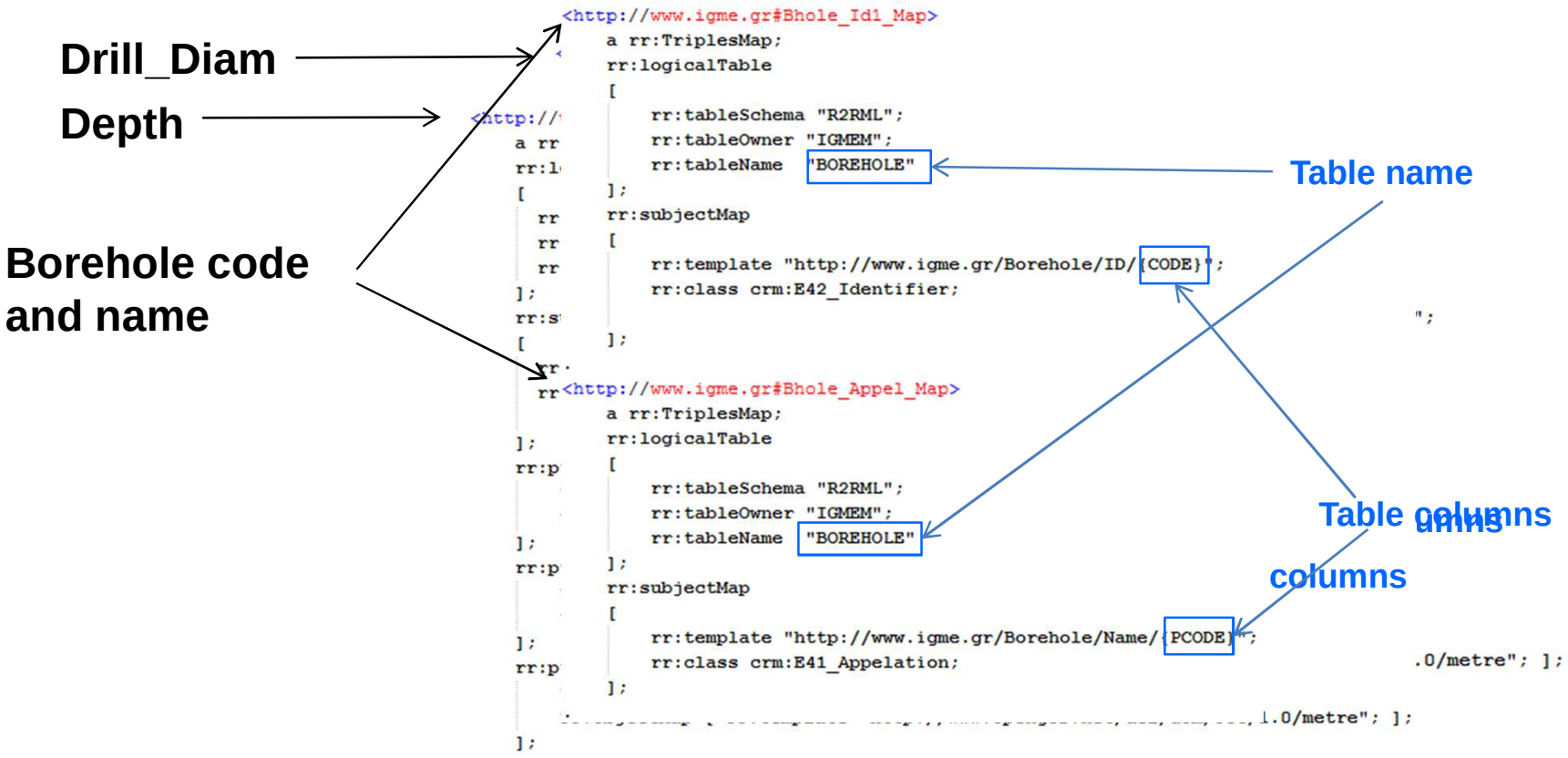
R2RML-based Linked Data Transformation Scenario (III)

- Create appropriate mappings from the schema notions to the GSOM notions (considering the GSOM scope notes) in an informal manner (e.g., via a diagram – see next slide)


E55_Type
"DEPTH"
P2_has_type
E54_Dimension
DEPTH
P90_has_value
E60_Number
P43_has_
dimension
E62_String
GEOLOGY
P3_has_note
S15_Aquifer_Concept
Water_District
P1_is_identified_
by
E42_Identifier
CODE_WDIST
E41_Appelation
NAME_WDIST
P121_overlaps_with
E55_Type
TYPE
P2_has_type
S16_Borehole
P89_falls_within
P1_is_identified_by
E42_Identifier
CODE_AQUIF
E41_Appelation
PCODE
O7_consists_of
S15_Aquifer_Concept
Aquifer
P1_is_identified_
by
E41_Appelation
NAME_AQUIF
S17_Borehole_Collar
P89_falls_within
E42_Identifier
CODE
P43_has_
dimension
P87_is_identified_by
S33_Relative_Spatial_Value
O20_has_value
E60_Number
LATITUDE
LONGITUDE
E55_Type
"DRILL_DIAM"
P2_has_type
E54_Dimension
DRILLING_D
P90_has_value
E60_Number
P2_has_type
E55_Type
"EPSG:4326"
LINKED DATA
RELATIONAL DATA
CODE
PCODE
TYPE
DRILL_DIAM
GEOLOGY
DEPTH
LATITUDE
LONGITUDE
CODE_AQUIF
NAME_AQUIF

R2RML-based Linked Data Transformation Scenario (IV)

- Relational-to-Linked data mappings have to be formalized using R2RML language
  - Due to lack of space we show a part of the R2RML file



R2RML-based Linked Data Transformation Scenario (V)

- Formalizing mappings using R2RML (cont'd)
  - Putting it all together into the borehole R2RML map

```
<http://www.igme.gr#BoreholeMap>
    a rr:TriplesMap;

    rr:logicalTable
    [
      rr:tableSchema "R2RML";
      rr:tableOwner "IGMEM";
      rr:tableName "BOREHOLE"
    ];
    rr:subjectMap
    [
      rr:template "http://www.igme.gr/Borehole/{CODE}";
      rr:class sci:S16_Borehole;

    ];
    rr:predicateObjectMap [
        rr:predicate crm:P1_is_identified_by;
        rr:objectMap [
            rr:parentTriplesMap <http://www.igme.gr#Bhole_Id1_Map>;
            rr:joinCondition [
                rr:child "CODE";
                rr:parent "CODE";
            ];
        ];
    ];
```

Table name

Code

```
rr:predicateObjectMap [
    rr:predicate crm:P1_is_identified_by;
    rr:objectMap [
        rr:parentTriplesMap <http://www.igme.gr#Bhole_Appel_Map>;
        rr:joinCondition [
            rr:child "PCODE";
            rr:parent "PCODE";
        ];
    ];
];
rr:predicateObjectMap [
    rr:predicate crm:P43_has_dimension;
    rr:objectMap [
        rr:parentTriplesMap <http://www.igme.gr#depth>;
        rr:joinCondition [
            rr:child "CODE";
            rr:parent "CODE";
        ];
    ];
];
rr:predicateObjectMap [
    rr:predicate crm:P43_has_dimension;
    rr:objectMap [
        rr:parentTriplesMap <http://www.igme.gr#dril_diam>;
        rr:joinCondition [
            rr:child "CODE";
            rr:parent "CODE";
        ];
    ];
];
rr:predicateObjectMap [
    rr:predicate crm:P3_has_note;
    rr:objectMap [ rr:template "<GEOLOGY>{GEOLOGY}</GEOLOGY>"; ]
];
```

Name

Depth

Drill_Diam

R2RML-based Linked Data Transformation Scenario (VI)

- Connection with Virtuoso was made by copying the Relational source data into Virtuoso
- The R2RML mappings were developed and stored in a particular file
- The final step is to call ***addR2RMLMappings*** API method**,** with the R2RML file and the URI of the RDF graph to be created as input, to enforce the R2RML mappings and create the corresponding Linked Data

R2RML Import via API

- The method takes as input two parameters:
  - ***mappingsURI***: The URL from which the R2RML specification can be downloaded. Does not need to be provided (as in our case) if the R2RML specification is given inline in the request.
  - ***graphURI***: The URI of the RDF graph to be created (http://igmem.data for our case)
- Note: the R2RML specification must be in the N3 format

Client Realizing Final Scenario Step

```
//Set URI of the API method
WebResource r = c.resource("http://ec2-54-216-181-30.eu-west-
    1.compute.amazonaws.com:8080/linkeddata-webapp-1.0-
    SNAPSHOT/rest/ld/addR2RMLMappings");

// Create a cookie to hold the provider's authentication token
Cookie cookie = new Cookie("iPlanetDirectoryPro", "<auth_token>");

//Create an InputStream to hold the content of the input R2RML specification file
InputStream is = new FileInputStream(new File("C://Documents and Settings/Desktop/r2rml.n3"));

//Create a MultivaluedMapImpl to hold the content of the input parameters
MultivaluedMap<String,String> params = new MultivaluedMapImpl();

//Add the graphURI input parameters to the MultivaluedMapImpl
params.add("graphURI","http://igmem.data");

//Call the API method with the provided media type (N3) and get back the result
ClientResponse response = r.queryParams(params).entity(is,new
    MediaType("text","rdf+n3")).cookie(cookie).post(ClientResponse.class);
```

R2RML file

Graph URI

Querying the Linked Data

- The transformation of providers' data into Linked Data enables the evaluation of uniform queries over the underlying triple store (Virtuoso) which can be of the following types:
  - Provider-specific queries
  - Cross-theme or cross-provider queries
- Queries are expressed via the W3C's SPARQL language (http://www.w3.org/TR/rdf-sparql-query/)

Querying the Linked Data – 1st Provider-Specific Scenario

- **Query Description:** Select the boreholes(names), depth, location and elevation and the date in which samples were taken. Depth must be over 100m and date must be after 1/1/2005.
- This query is applied only over IGMEM data (provider specific)
- Query must first be expressed in SPARQL (see next slide) and then the **ldquery** API method must be called

Querying the Linked Data – 1st Provider-Specific Scenario

- Underlying SPARQL query

```
select ?bname ?depth ?asWKT ?elev ?sdate from <http://igmem.data> where {
?bhole crm:P1_is_identified_by ?bname;
       sci:O7_consists_of ?bcollar;
       crm:P43_has_dimension ?d.
?d crm:P90_has_value ?depth.
filter (regex(?d, "Depth")).
?bcollar geo:hasGeometry ?point;
         crm:P87_is_identified_by ?el.
filter(regex(?el,"Elevation")).
?el sci:O20_has_value ?elev.
?point geo:asWKT ?asWKT.
filter(regex(?asWKT,"/4326")).
filter(regex(?bname, "/Name/")).
filter (?depth > 100).
?st a sci:S2_Sample_Taking;
sci:O4_sampled_at ?bhole;
sci:O5_removed ?sample.
?st crm:P4_has_time-span ?sdate.
filter (xsd:date(?sdate) > "2005-01-01"^^xsd:date)
}
```

provider

Select the boreholes(names), depth, location and elevation and the date in which samples were taken. Depth must be over 100m and date must be after 1/1/2005.

Querying the Linked Data – 2nd Provider-Specific Scenario

- **Query Description:** Select the samples and chemical analyses of the compounds with ph>7 and Na>100. Moreover select the locations and the boreholes(names) which were analyzed.
- This query is also applied only over IGMEM data (provider specific)
- Query is first expressed in SPARQL (see next slide) and then ***ldquery*** API method is called

Querying the Linked Data – 2nd Provider-Specific Scenario

- Underlying SPARQL query

```
select ?sample ?meas ?compound ?cavalue ?bname ?asWKT from <http://igmem.data> where {
?st sci:O5_removed ?sample;
    sci:O4_sampled_at ?bhole.
filter(regex(?sample, "/GrWater/")).
?meas crm:P39_measured ?sample;
      crm:P40_observed_dimension ?cadim.
?cadim crm:P2_has_type ?compound;
     crm:P90_has_value ?cavalue.
?bhole a sci:S16_Borehole;
       sci:O7_consists_of ?bcollar;
       crm:P1_is_identified_by ?bname.
filter(regex(?bname, "Name")).
?bcollar geo:hasGeometry ?point.
?point geo:asWKT ?asWKT.
filter(regex(?asWKT,"/4326")).
filter ( (regex(?compound, "pH") && ?cavalue > 7) ||
         (regex(?compound, "Na") && ?cavalue > 100) ).
}
```

provider

Select the samples and chemical analyses of the compounds with ph>7 and Na>100. Moreover select the locations and the boreholes(names) which were analyzed.

Querying the Linked Data – Cross-Provider Scenario

- The previous queries can be applied to additional data providers with similar data (i.e. BRGM, GEUS) leading to cross datasets queries
- **Query Description:** Select the samples and chemical analyses of the compounds with with ph>7 and Na > 100. Moreover select the locations and the boreholes(names) which were analyzed.
- This query is applied over all partners' datasets which are relevant
  - IGMEM, GEUS, BRGM

Querying the Linked Data – Cross-Provider Scenario

- Underlying SPARQL query

```
select ?sample ?meas ?compound ?cavalue ?bname ?asWKT where {
?st sci:O5_removed ?sample.
{
?st sci:O4_sampled_at ?bhole.
} UNION {
?st sci:O4_sampled_at ?place.
?bhole sci:O9_contains_or_confines ?place.
}
filter(regex(?sample, "/GrWater/")).
?meas crm:P39_measured ?sample;
      crm:P40_observed_dimension ?cadim.
?cadim crm:P2_has_type ?compound;
     crm:P90_has_value ?cavalue.
?bhole sci:O7_consists_of ?bcollar;
       crm:P1_is_identified_by ?bname.
filter(regex(?bname, "Name")).
?bcollar geo:hasGeometry ?point.
?point geo:asWKT ?asWKT.
filter(regex(?asWKT,"/4326")).
filter ( (regex(?compound, "pH") && ?cavalue > 7) ||
         (regex(?compound, "Na") && ?cavalue > 100) ).
}
```

Select the samples and chemical analyses of the compounds with ph>7 and Na > 100. Moreover select the locations and the boreholes(names) which were analyzed.

SPARQL Query Support

- Method ***ldquery*** is used for evaluating SPARQL queries such as those previously presented
- Takes four parameters:
  - ***query***: the SPARQL query as a String
  - ***timeout***: maximum amount of time to wait for response
  - ***maxrows***: maximum amount of result rows to return (can also be specified in SPARQL query)
  - ***graphURI***: URI of graph used to resolve relative URIs in the query
- User should indicate in his/her request the format of the returned results: XML, JSON, TSV, CSV

Client for Querying

```
//Set URI of the API method
WebResource r = c.resource("http://ec2-54-216-181-30.eu-west-1.compute.amazonaws.com:8080/linkeddata-webapp-1.0-
      SNAPSHOT/rest/ld/ldquery");

//Create a FormDataMultiPart to hold the content of the input parameters
FormDataMultiPart form = new FormDataMultiPart();

//Create the String SPARQL query by loading it from a file
String query = MyIOUtils.loadStringFromFile("C://Desktop/Case_X_Query.sql");

//Add the input parameters to the form – the parameters name should be left fixed
form.field("query",query); form.field("graphURI","http://igmem.data"); form.field("maxRows","1000");

//Fix the format of the results to be returned – If JSON is needed, then just change the media type as
      MediaType("application","sparql-results+json");
MediaType type = new MediaType("application","sparql-results+xml");

//Call the API method and get back the result
ClientResponse response = r. accept(type).type(MediaType.MULTIPART_FORM_DATA).post(ClientResponse.class,form);

//Store the result in a file by obtaining an InputStream as the response entity
MyIOUtils.storeFile("C://Desktop/results.xml", response.getEntity(InputStream.class));
```

SPARQL query

Graph URI

Max rows

Results format

Results Visualization (I)

- As the results are provided in a particular machine-processable form, user can request a better visualization by transforming them to a format which can be loaded on various GUI tools/services, such as Google Maps
- To this end, if the SPARQL results contain spatial information, the API offers the ***geoldtransform*** method through which they can be transformed into geographical feature collection representations, such as KML, GeoJSON, GML and Shape.

Results Visualization (II)

- The ***geoldtransform*** method takes as input three parameters:
  - ***url***: the URL from which the sparql-results can be downloaded
  - ***inputFormat***: the format of the sparql-results
  - ***srid***: the default SRID (4326 is assumed if not given)
- The first two parameters must not be provided (as in our cases) when the sparql-results are provided inline in the user request
- The preferred output format can also be specified by fixing the accepted media-type

Client Code

//Set URI of the API method

```
WebResource r = c.resource("http://ec2-54-216-181-30.eu-west-
    1.compute.amazonaws.com:8080/linkeddata-webapp-1.0-
    SNAPSHOT/rest/geold/geoldtransform");
```

//Create an InputStream to hold the content of the input SPARQL-results file

```
InputStream is = new FileInputStream(new File("C://Desktop/results.xml"));
```

//Fix the format of the results to be returned as KML – If GeoJSON is needed, then just change the media type as MediaType("application","sparql-results+json");

```
MediaType type = new MediaType("application","vnd.google-earth.kml+xml");
```

//Call the API method with the provided and accepted media type and get back the result

```
ClientResponse response = r.accept(type).entity(is,new MediaType("application","sparql-
    results+xml")).post(ClientResponse.class);
```

//Store the result in a file by obtaining an InputStream as the response entity

```
MyIOUtils.storeFile("C://Desktop/results.kml", response.getEntity(InputStream.class));
```

Export Linked Data into INSPIRE (I)

- Linked Data in GSOM are not INSPIRE-compliant
- However, there is a specific GSOM-to-INSPIRE mapping mechanism allowing to transform GSOM-based RDF data, acquired from SPARQL queries, into XML-based INSPIRE-compliant data
- Mapping mechanism is limited with respect to which data can be transformed as INSPIRE has limitations (see next slide) which do not allow for a complete export of all the Linked Data (GSOM) notions

Export Linked Data into INSPIRE (II)

- The generic INSPIRE conceptual schema is **not event-oriented** (especially the schema that uses the observation-measurement specification)
- Some information from particular themes cannot be modeled

Approach to INSPIRE-based Exporting (I)

- Our approach was to separate our data (from all data providers) into themes.
  - chemical analyses, boreholes, landslides, earthquakes etc.
- Next we investigated the GSOM notions which could be mapped to inspire notions as only these notions will be eventually exported
- Finally, we created a set of API methods which can be used by users to export Linked Data into an INSPIRE compliant format according to two main alternative ways

Approach to INSPIRE-based Exporting (II)

- First export way:
  - User is acquainted with the appropriate info which can be used to exploit the first export API method. This can be performed by calling the ***query_theme_info*** API method with the name of a particular theme as a String value of the "theme" input parameter. The returned info comprises:
    - The main SPARQL query used to draw related RDF data for a particular theme
    - The main data providers and their corresponding RDF graphs
    - The (SPARQL) variables on which FILTER constraints can be created to accompany the main SPARQL query – all appropriate information is returned, such as variable name, value type and allowed range of values

Approach to INSPIRE-based Exporting (III)

- First export way (cont.):
  - Then, the user can call the ***inspire_export*** method to obtain the appropriate INSPIRE-compliant data related to a particular theme. This method takes as input four parameters:
    - ***themes***: a list of Strings indicating the required themes
    - ***constraints***: a list of FILTER constraints on variables mapping to specific theme information
    - ***maxRows***: the maximum amount of results to return
    - ***returnUnfilteredResults***: return results for which some filters do not actually apply (e.g., exist constraints on information that does not belong to a theme)

Approach to INSPIRE-based Exporting (IV)

- Second export way:
  - The user provides the appropriate SPARQL query related to the data he/she desires to obtain (along with an optional limit on results) to the ***inspire_query_export*** API method. This method, by knowing all GSOM-to-INSPIRE mappings, maps the SPARQL query variables to INSPIRE-mapped GSOM notions and then by processing the SPARQL results makes the corresponding result transformation to INSPIRE.
  - The method takes as input two main parameters:
    - ***query***: the SPARQL query as a String
    - ***maxrows***: the maximum amount of results to be returned

    and produces a zipped file containing the INSPIRE-compliant data

Approach to INSPIRE-based Exporting (V)

- Second export way:
  - This method is appropriate for users which are familiar with GSOM and of course with SPARQL
  - Can also be used to obtain related results which span different themes and different data providers – by allowing the user to provide a SPARQL query we enable the description of any data requirement
  - Due to INSPIRE incompleteness issues raised previously, not all SPARQL query variables are guaranteed to be mapped into INSPIRE-compliant information. On the contrary, the first INSPIRE export method provides a more controlled way to produce the required INSPIRE-compliant information (all SPARQL variables are mapped to INSPIRE information)

INSPIRE Exporting Scenarios

- Recall that IGMEM data refer on Boreholes, Samples and Chemical Analyses
- The relevant themes which can be exported are those of: boreholes, chemical analyses
- We will show next how we can use the methods of the Linked Data Management API to export the Linked Data in INSPIRE-compliant form
- The exporting scenarios consider all theme-related data even from other data providers (the latter data can be filtered by providing filtering constraints)

Export Linked Data into INSPIRE – 1st Scenario

- **Scenario 1:** Recall the first provider specific query shown in the previous section transformed into a cross datasets query
  - Select the boreholes(names), depth, location and elevation and the date in which samples were taken. Depth must be over 100m and date must be after 1/1/2005.
- Export INSPIRE data from boreholes w.r.t. the above conditions Parameters
  - Theme: **Borehole**
  - Constraints: depth>100 **and** date > 2005-01-01
  - returnUnfilteredResults: true

Client for 1st Scenario

```
//Set URI of the API method
WebResource r = c.resource("http://ec2-54-216-181-30.eu-west-
    1.compute.amazonaws.com:8080/linkeddata-webapp-1.0-SNAPSHOT/rest/ld/inspire_export");

//Create a FormDataMultiPart to hold the content of the input parameters
FormDataMultiPart form = new FormDataMultiPart();

//Add the input parameters to the form – the parameters name should be left fixed
//For each theme, add a "themes" parameter to the form
form.field("themes","borehole");
//For each constraint, add a "constraints" parameter to the form
form.field("constraints","FILTER (?depth > 100)");
form.field("constraints","FILTER (?date > 2005-01-01)");
form.field("maxRows","" + 1000 );
form.field("returnUnfilteredResults","true");
//Call the API method and get back the result
ClientResponse response = r.
    type(MediaType.MULTIPART_FORM_DATA).post(ClientResponse.class,form);
//Store the result in a file by obtaining an InputStream as the response entity
MyIOUtils.storeFile("C://Desktop/inspire_results1.zip", response.getEntity(InputStream.class));
```

theme

filters

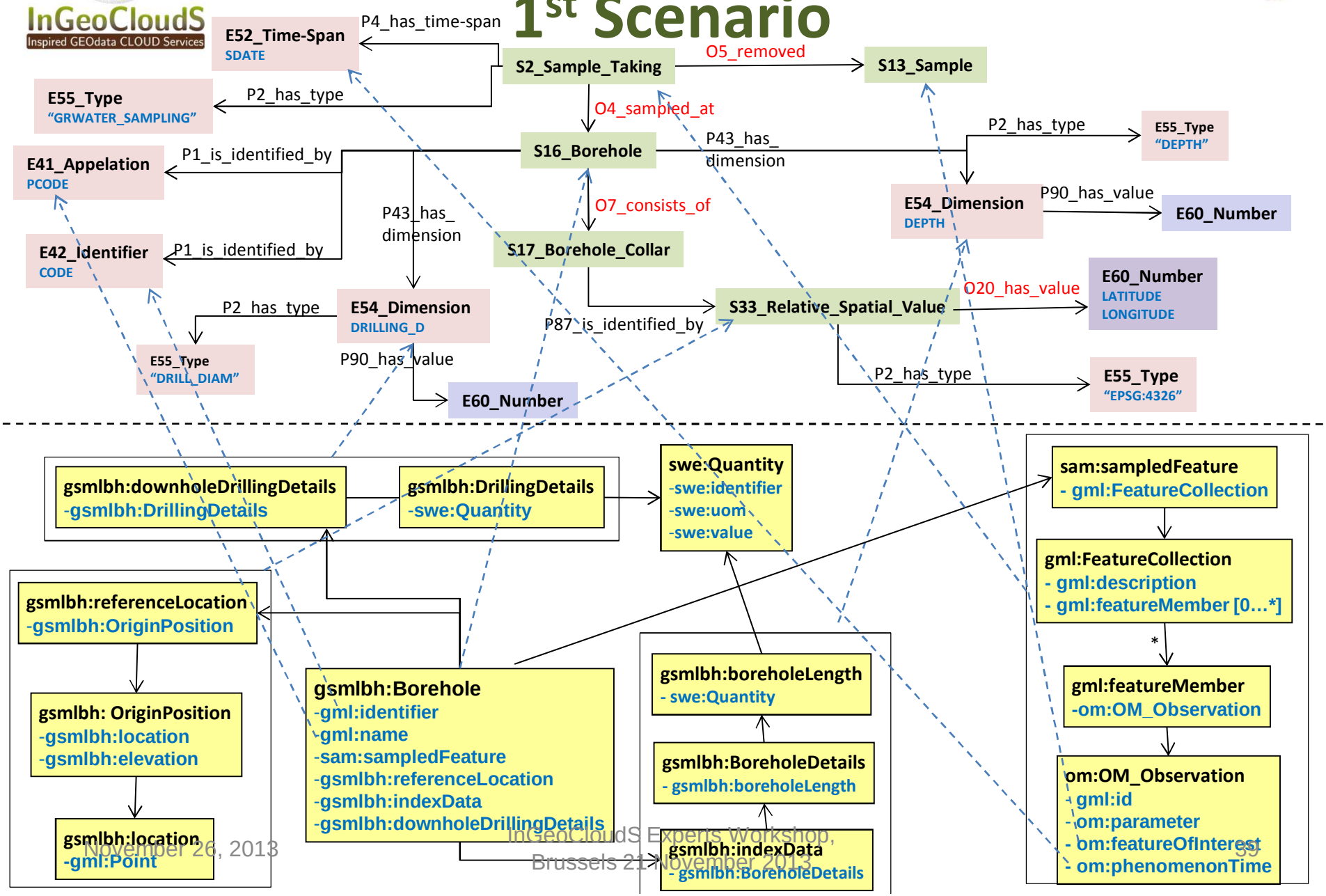
Max rows

Choice for unfiltered results

Export Linked Data into INSPIRE – 1st Scenario

- As a result, a zipped file containing the xml specification of the related borehole data is created w.r.t. the http://xmlns.geosciml.org/Borehole/3.1 schema (see next slide which shows the mapping from the GSOM notions to INSPIRE notions based on the Borehole XML Schema)

Export Linked Data into INSPIRE – 1st Scenario



Export Linked Data into INSPIRE – 2nd Scenario

- **Scenario 2:** Recall the second provider specific query shown in the previous section transformed into a cross datasets query
  - Select the samples and chemical analyses of the compounds with ph>7 and Na > 100. Moreover select the locations and the boreholes(names) which were analyzed.
- Export INSPIRE data from chemical analyses w.r.t. the above condition parameters
  - Case 1: Call first INSPIRE export method with parameters:
    - themes: **Chemical Analysis**
    - constraints: (compound = “pH” and value > 7) or (compound = "Na" and value > 200) or compound = “.”
    - returnUnfilteredResults: false

– **Case 2:** Call second INSPIRE export method with parameters:
  - query: the respective query that produces the equivalent results

Export Linked Data into INSPIRE – 2nd Scenario

– **Case 2(cont'd):** The SPARQL query is:

```
select ?sample ?meas ?compound ?cavalue ?caunit ?asWKT
from <http://igmem.data> where {
?st sci:O5_removed ?sample;
    sci:O4_sampled_at ?bhole.
filter(regex(?sample, "/GrWater/")).
?meas crm:P39_measured ?sample;
      crm:P40_observed_dimension ?cadim.
?cadim crm:P2_has_type ?compound;
      crm:P90_has_value ?cavalue.
?bhole a sci:S16_Borehole;
       sci:O7_consists_of ?bcollar.
?bcollar geo:hasGeometry ?point.
?point geo:asWKT ?asWKT.
filter(regex(?asWKT,"/4326")).
filter ( (regex(?compound, "pH") && ?cavalue > 7) ||
         (regex(?compound, "Na") && ?cavalue > 100) ).
OPTIONAL {
?cadim crm:P91_has_unit ?caunit.
}
}
```

2nd Scenario – 1st Case Client

```
//Set URI of the API method
WebResource r = c.resource("http://ec2-54-216-181-30.eu-west-
    1.compute.amazonaws.com:8080/linkeddata-webapp-1.0-SNAPSHOT/rest/ld/inspire_export");

//Create a FormDataMultiPart to hold the content of the input parameters
FormDataMultiPart form = new FormDataMultiPart();

//Add the input parameters to the form
form.field("themes","chemical analysis");
//For each constraint, add a "constraints" parameter to the form
form.field("constraints"," FILTER ( (?compound = "pH" && ?value > 7) ||
          (?compound = "Na" && "value > 200) || (regex(?compound,"."))");
form.field("returnUnfilteredResults","false");

//Call the API method and get back the result
ClientResponse response = r.
    type(MediaType.MULTIPART_FORM_DATA).post(ClientResponse.class,form);

//Store the result in a file by obtaining an InputStream as the response entity
MyIOUtils.storeFile("C://Desktop/inspire_results2_1.zip", response.getEntity(InputStream.class));
```

theme

filters

Choice for unfiltered results

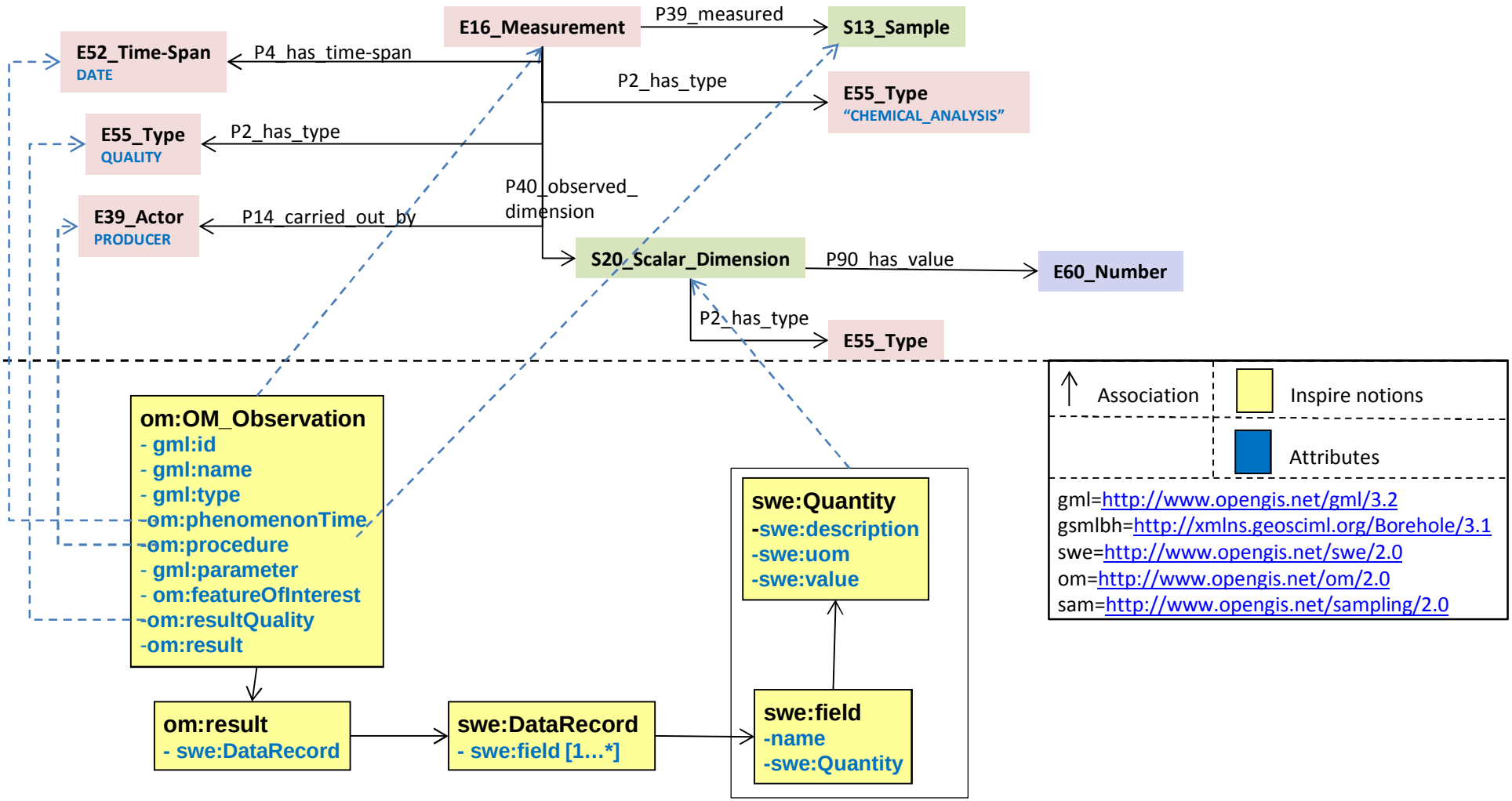
2nd Scenario – 2nd Case Client

```
//Set URI of the API method
WebResource r = c.resource("http://ec2-54-216-181-30.eu-west-
    1.compute.amazonaws.com:8080/linkeddata-webapp-1.0-
    SNAPSHOT/rest/ld/inspire_query_export");

//Create a FormDataMultiPart to hold the content of the input parameters
FormDataMultiPart form = new FormDataMultiPart();

//Create the String SPARQL query by loading it from a file
String query = MyIOUtils.loadStringFromFile("C://Desktop/Scen_2_Case_2_query.sql");

//Add the input parameters to the form – the parameters name should be left fixed
form.field("query",query); form.field("maxRows",1000);

//Call the API method and get back the result
ClientResponse response = r.
    type(MediaType.MULTIPART_FORM_DATA).post(ClientResponse.class,form);

//Store the result in a file by obtaining an InputStream as the response entity
MyIOUtils.storeFile("C://Desktop/inspire_results2_2.zip", response.getEntity(InputStream.class));
```

query

Max rows

Export Linked Data into INSPIRE – 2nd Scenario



Conclusions

- The Linked Data Management API:
  - Provides high-level functionality hiding the low-level complexities/pecularities of the underlying RDF Store
  - Enables the complete management of Linked Data (import, update, query, transform)
    - Direct importing of RDF data
    - Indirect importing of XML/relational data
  - Manipulates I/O streams for better scalability and performance & exploits a sophisticated connection management handler

Thanks for your attention

This content by the InGeoCloudS consortium members is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License. Based on a work at http://www.ingeoclouds.eu/.