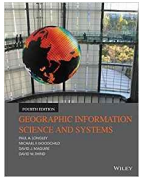
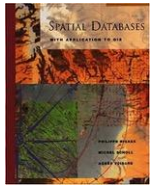
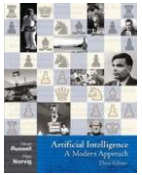
RDF and SPARQL extensions for geospatial and temporal data

Background



- Geographic information systems
- Spatial database research
- Spatial logics and reasoning
- Industry standards and implemented systems





OGC®
Making location count.
www.opengeospatial.org

ArcGIS ESRI

QGIS

PostGIS

Overview

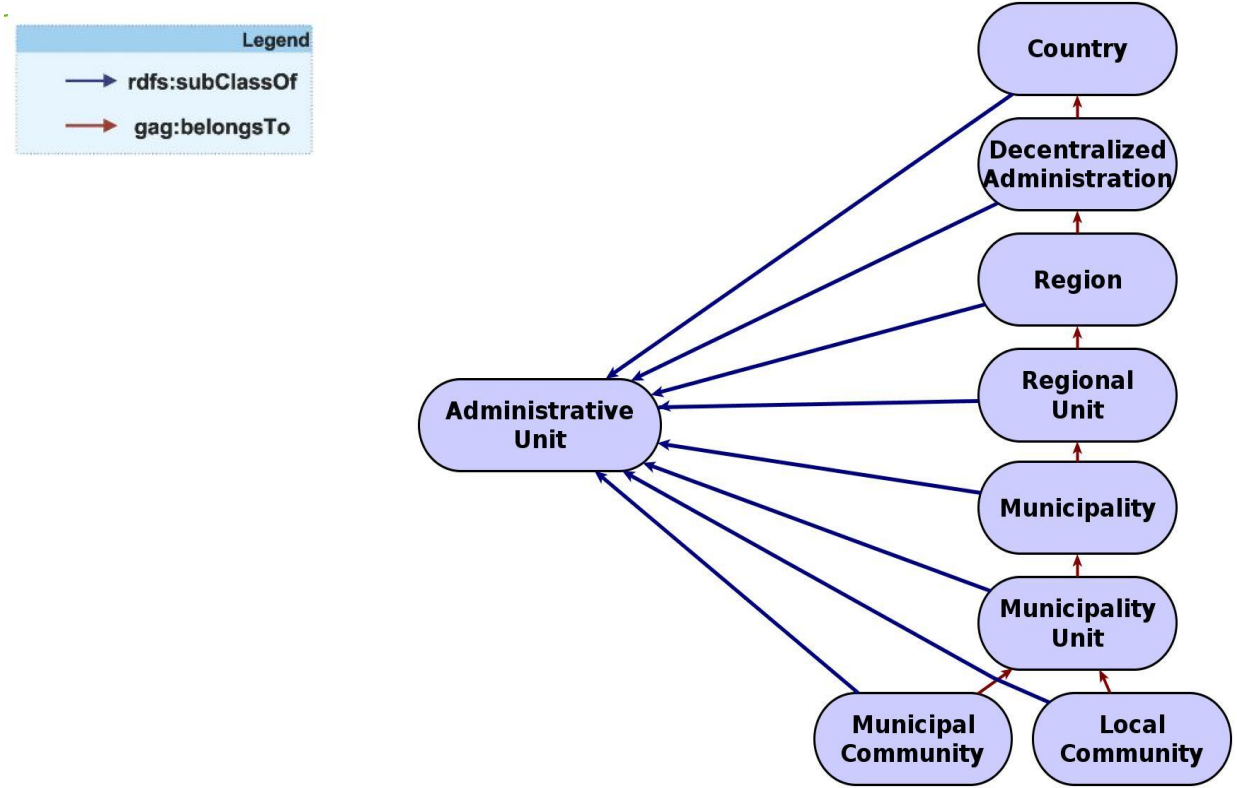
- The OGC standard GeoSPARQL (2012)
- The data model stRDF/stSPARQL (2012)
- The framework RDFi (2013)
- GeoSPARQL+ (2020)

The Model stRDF

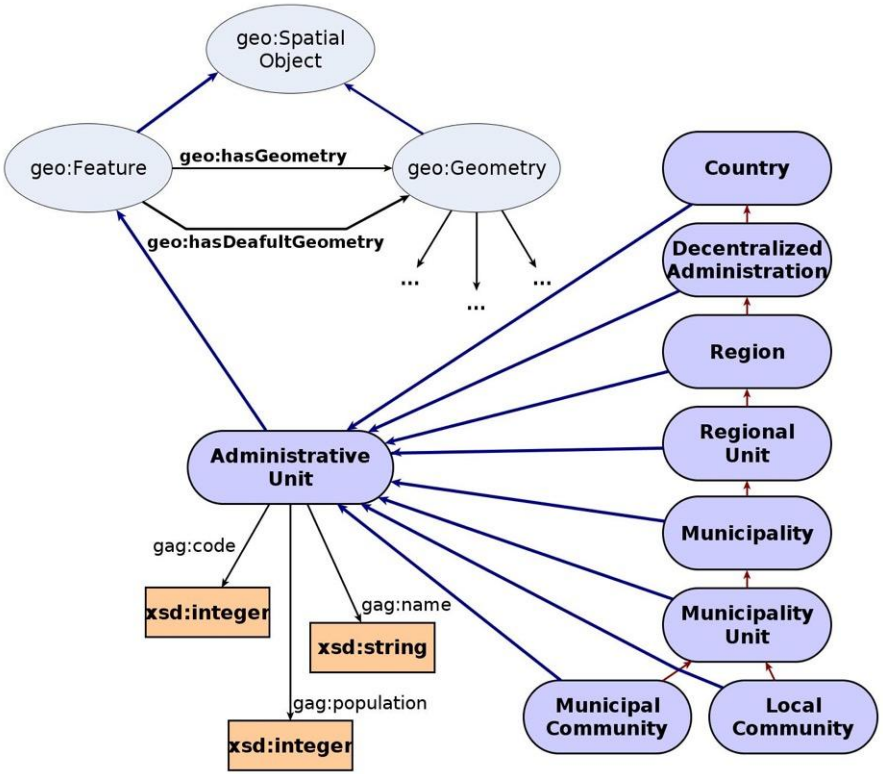
- An extension of RDF for the representation of **geospatial information that changes over time**.
- Geospatial dimension:
  - Two **spatial data types** are introduced.
  - Geospatial information is represented using **spatial literals** of these datatypes.
  - OGC standards **WKT** and **GML** are used for the serialization of spatial literals.
- **Temporal dimension** (more later)

Example: Greek Administrative Geography

Domain Ontology



Connect to a Top Level Geospatial Ontology + Introduce Some Properties



Example of stRDF (geospatial dimension)

```
gag:Olympia
 rdf:type gag:MunicipalCommunity;
 gag:hasName "Ancient Olympia";
 gag:hasPopulation "184"^^xsd:int;
 strdf:hasGeometry "MULTIPOLYGON(((308511.906249999
  4201042,308763.8125 4200714,
  308840.09375 4200629,308939.3125 4200545,.....308390.000000001
  4201276,308451.593749999 4201167,308467 4201125,308511.906249999
  4201042)));<http://www.opengis.net/def/crs/EPSG/0/2100>"^^strdf:WKT.
```

The Query Language stSPARQL (geospatial dimension)

- It is an **extension of SPARQL 1.1**
- It offers **families of functions for querying geometries**.
- The functions are taken mostly from the **OGC standard** “OpenGIS Simple Feature Access - Part 2: SQL Option”.
- They are similar to the ones offered by **spatially-enabled relational database management systems** (e.g., PostGIS).

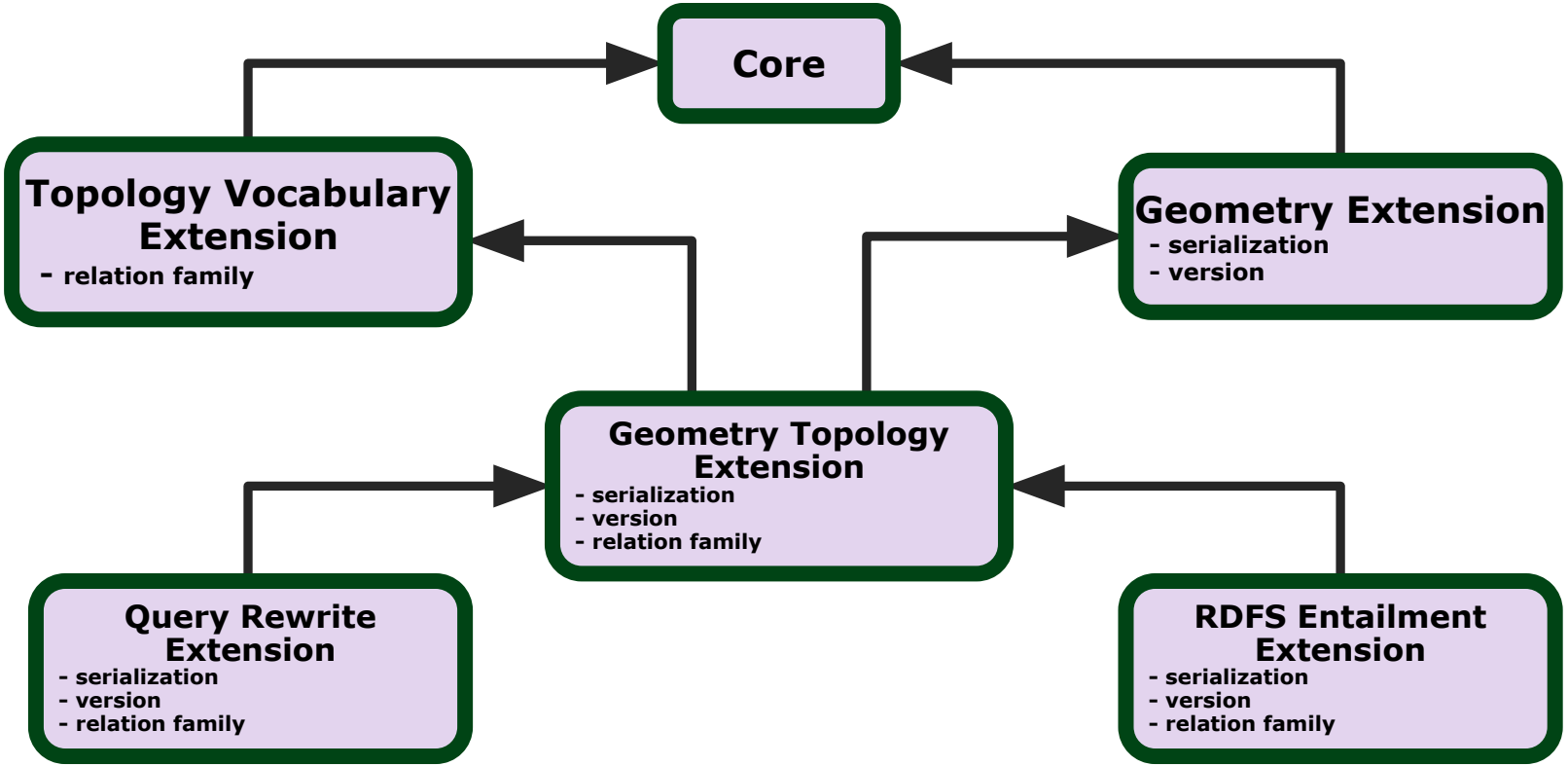
Example of stSPARQL (geospatial dimension)

Query: Compute the parts of burnt areas that lie in coniferous forests

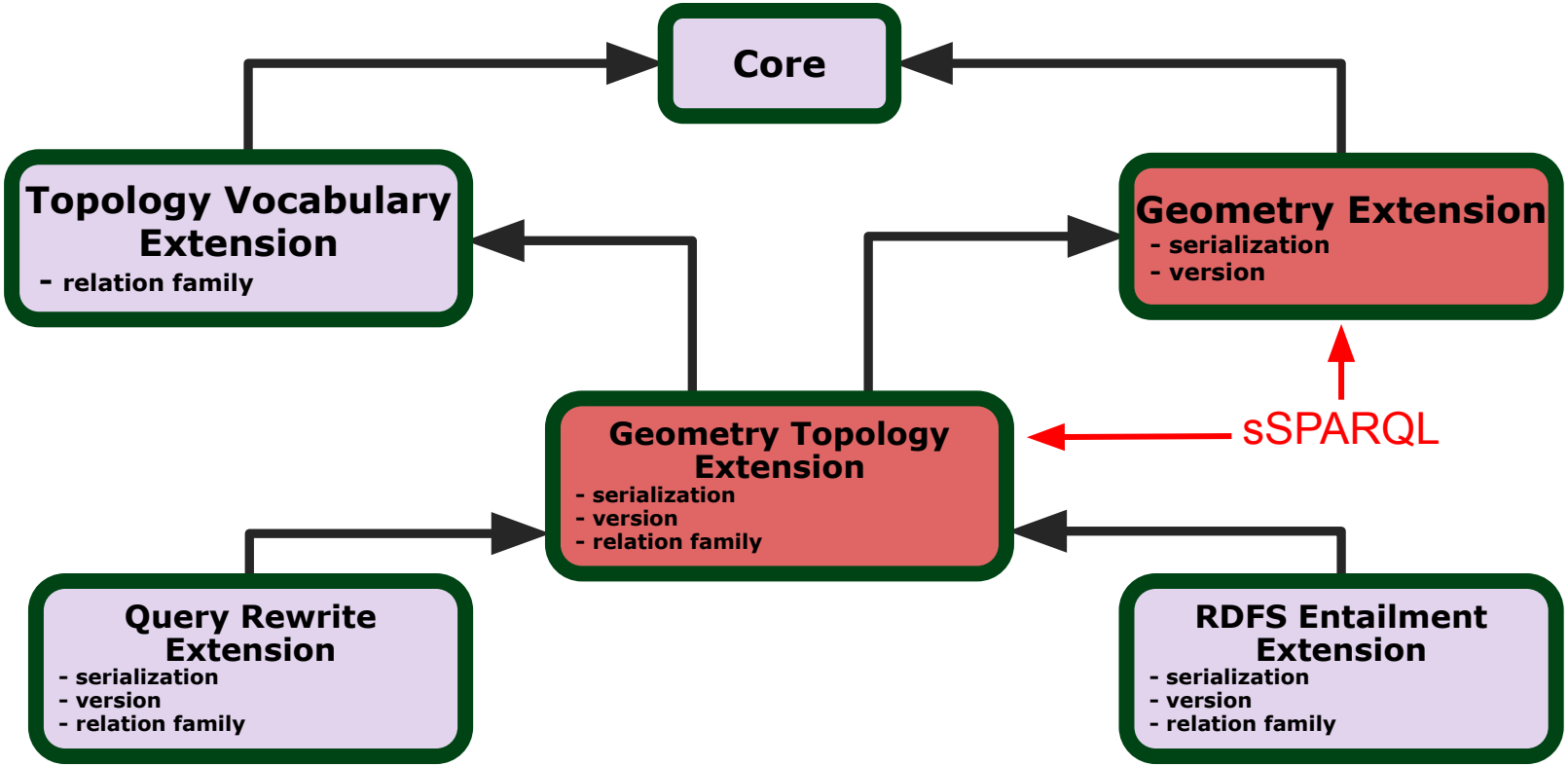

```
SELECT ?burntArea (strdf:intersection(?baGeom,
                   strdf:union(?fGeom)) AS ?burntForest)
WHERE {
      ?burntArea  rdf:type noa:BurntArea;
                  strdf:hasGeometry ?baGeom.
        ?forest   rdf:type clc:Region;
                  clc:hasLandCover
clc:ConiferousForest;
                    strdf:hasGeometry ?fGeom.

FILTER  (strdf:intersects(?baGeom,?fGeom)) }
GROUP BY ?burntArea ?baGeom
```

The OGC Standard GeoSPARQL (2012)



GeoSPARQL vs. stSPARQL



Example of the Topology Vocabulary Extension

Triples:

```
gag:Olympia rdf:type gag:MunicipalCommunity .
gag:OlympiaMunicipality rdf:type gag:Municipality .
gag:WesternGreece rdf:type gag:Region .

gag:Olympia   geo:sfWithin  gag:OlympiaMunicipality .
gag:OlympiaMunicipality   geo:sfWithin   gag:WesternGreece .
```

Query: Find the **region** in which Ancient Olympia is located.

Answer: `gag:WesternGreece`

Example of the Topology Vocabulary Extension

Triples:

```
gag:Olympia rdf:type gag:MunicipalCommunity .
gag:OlympiaMunicipality rdf:type gag:Municipality .
gag:WesternGreece rdf:type gag:Region .

gag:Olympia   geo:sfWithin  gag:OlympiaMunicipality .
gag:OlympiaMunicipality   geo:sfWithin   gag:WesternGreece .
```

Query: Find the **region** in which Ancient Olympia is located.

Answer: `gag:WesternGreece`

Method: By transitivity of geo:sfWithin.

Not supported by GeoSPARQL!

The Query Rewrite Extension

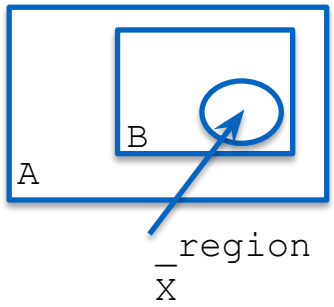
- Enables the **translation of qualitative topological information appearing in a query to quantitative.**
- This is done by rewriting of queries with triple patterns involving topological relations into queries with topological functions on geometries.
- The rewriting is based on **RIF rules.**

Beyond the Topology Vocabulary Extension

Triples:

```
ex:regionA strdf:hasGeometry "POLYGON(A)"^^strdf:WKT .
ex:regionB strdf:hasGeometry "POLYGON(B)"^^strdf:WKT .

_:regionX geo:sfWithin ex:regionB
```



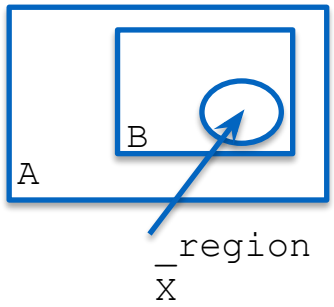
Query: Is regionX contained in regionA?

Beyond the Topology Vocabulary Extension (cont'd)

Triples:

```
ex:regionA strdf:hasGeometry "POLYGON(A)"^^strdf:WKT .
ex:regionB strdf:hasGeometry "POLYGON(B)"^^strdf:WKT .

_:regionX geo:sfWithin ex:regionB
```



Query: Is regionX contained in regionA?

Answer: **Yes**

Not supported by GeoSPARQL.

The Framework RDFi (RR2013, AIJ 2016)

- Extension of RDF with incomplete information.
- New kind of literals (**e-literals**) for each datatype.
  - Property values that exist but are unknown or partially known.

- **Partial knowledge**: captured by constraints (appropriate constraint language *L)*.

- RDF graphs extended to **RDF^i databases**: pair (G, φ)

  G: RDF graph with e-literals

  φ: quantifier-free formula of *L*

The Framework RDFi (cont'd)

- **Formal semantics** for RDF^{i} and SPARQL query evaluation.
- **Representation Systems**:
  - CONSTRUCT queries
    - Without blank nodes in their templates
    - With monotone graph patterns (using only operators AND, UNION and FILTER).
  - CONSTRUCT queries
    - Without blank nodes in their templates
    - With well-designed graph patterns (graph patterns using only AND, FILTER and OPT plus some intuitive conditions).
- **Computational Complexity**:
  - Query answering is **coNP-complete (data complexity)** for certainty queries and various interesting classes of spatial constraints. Compare this with LOGSPACE complexity for the standard SPARQL case.

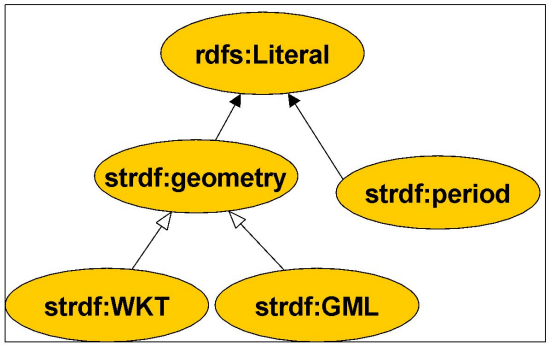
The Temporal Dimension of stSPARQL (valid time of triples, ESWC 2013)

The following extensions to RDF are introduced:

- **Timeline**: the (discrete) value space of the datatype `xsd:dateTime` of XML-Schema.

- Two kinds of time primitives are supported: time instants and time periods.
  - A **time instant** is an element of the time line.
  - A **time period** is an expression of the form [B, E) or [B, E] or (B, E] or (B, E) where B and E are time instants called the beginning and ending time of the period.

- The **new datatype** `strdf:period` is introduced.



The Temporal Dimension of stRDF (cont'd)

- Triples are extended to quads.

- A temporal triple (quad) is an expression of the form

  `s p o t.`

  where `s p o.` is an RDF triple and `t` is a time instant or time period called the **valid time of the triple**.

- The temporal constants **NOW** and **UC** ("until changed") are introduced.

- This approach was first used in the knowledge representation language Telos in 1988 (Koubarakis, M.Sc.)

Example


```
clc:region1 clc:hasLandCover clc:Forest
   "[2006-08-25T11:00:00+02,2007-08-25T11:00:00+02)"^^strdf:period .
noa:ba1 rdf:type noa:BurntArea
     "[2007-08-25T11:00:00+02,2009-08-25T11:00:00+02)"^^strdf:period .
clc:region1 clc:hasLandCover clc:AgriculturalArea
                      "[2009-08-25T11:00:00+02, "UC")"^^strdf:period .
```

The Temporal Dimension of stSPARQL

The following extensions to SPARQL are introduced:

- Triple patterns are extended to **quad patterns** (the last component is a temporal term: variable or constant)

- **Temporal extension functions** are introduced:
  - Allen's temporal relations (e.g., `strdf:after`)
  - Period constructors (e.g., `strdf:period_intersect`)
  - Temporal aggregates (e.g., `strdf:maximalPeriod`)

Example Query

Query: Find the current land cover of all areas in the dataset.

```
SELECT ?clc
WHERE {
    ?R rdf:type clc:Region .
    ?R clc:hasLandCover ?clc ?t1 .
    FILTER (strdf:during("NOW", ?t1))
   }
```

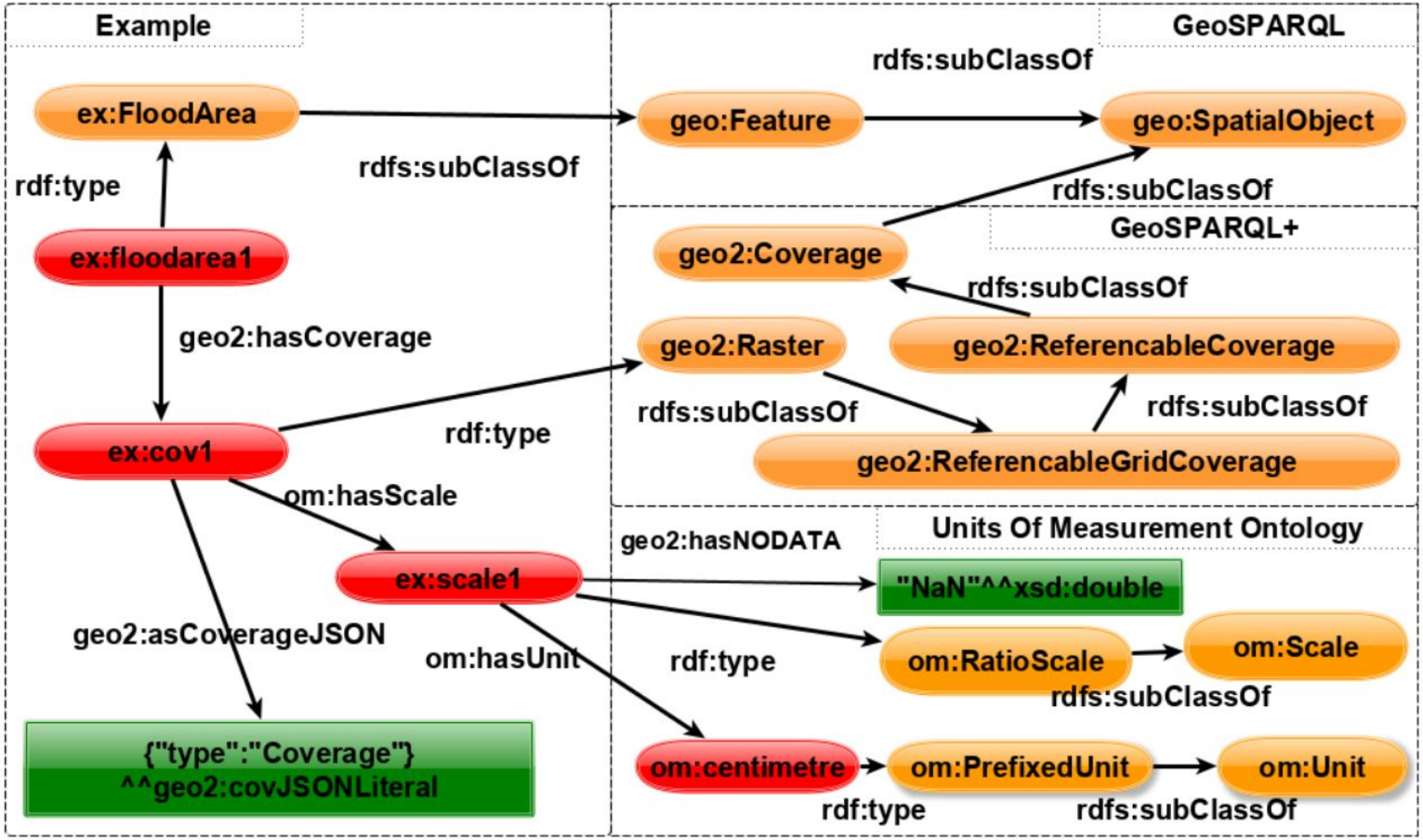
GeoSPARQL+ (Homburg et al. ISWC 2020)

- GeoSPARQL+ ontology in order to integrate geospatial **raster data** into the Semantic Web
  - New type of geospatial data for raster

- Extension of GeoSPARQL query language
  - **New filter functions** based on raster algebra operations, e.g. **rasterPlus**, **rasterSmaller**

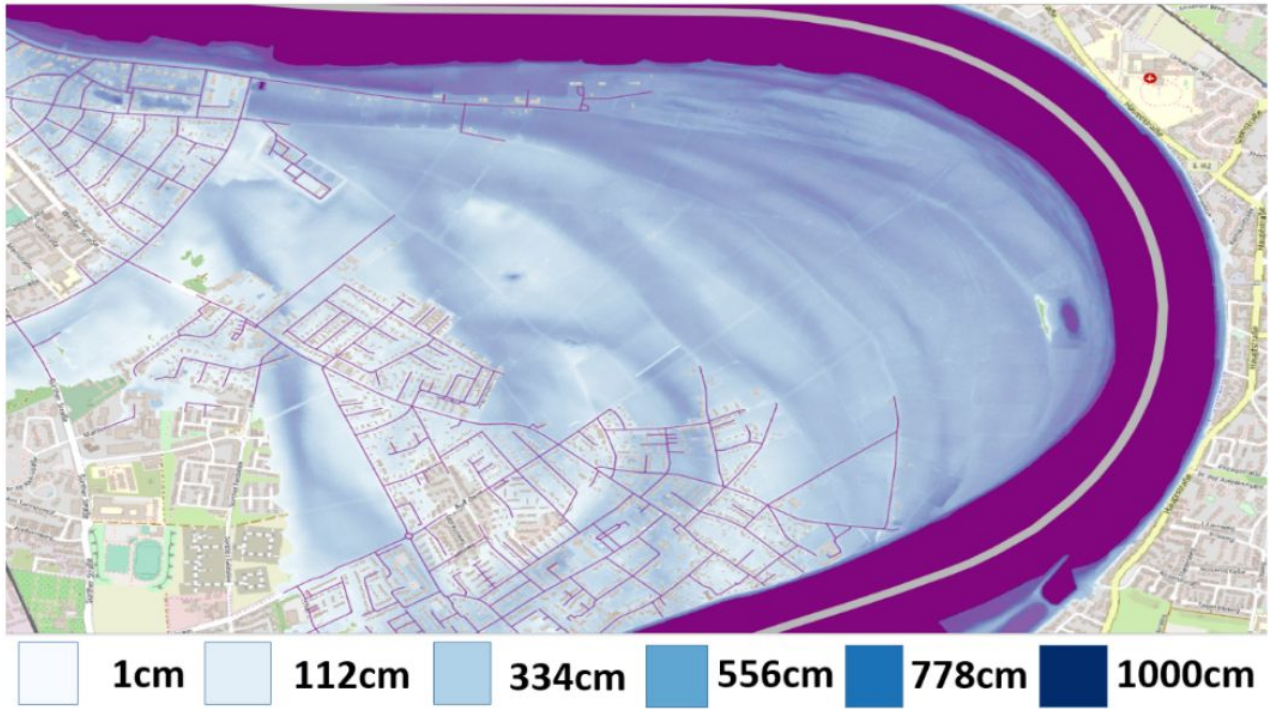
- **Combined vector and raster data analysis** can be achieved from a single query

GeoSPARQL+ Ontology (Homburg et al. ISWC 2020)



GeoSPARQL+ Example (Homburg et al. ISWC 2020)

- Give me all roads which are not flooded by more than 10cm
  - Road network vector dataset
  - Flood altitude raster dataset



GeoSPARQL+ Example (Homburg et al. ISWC 2020)

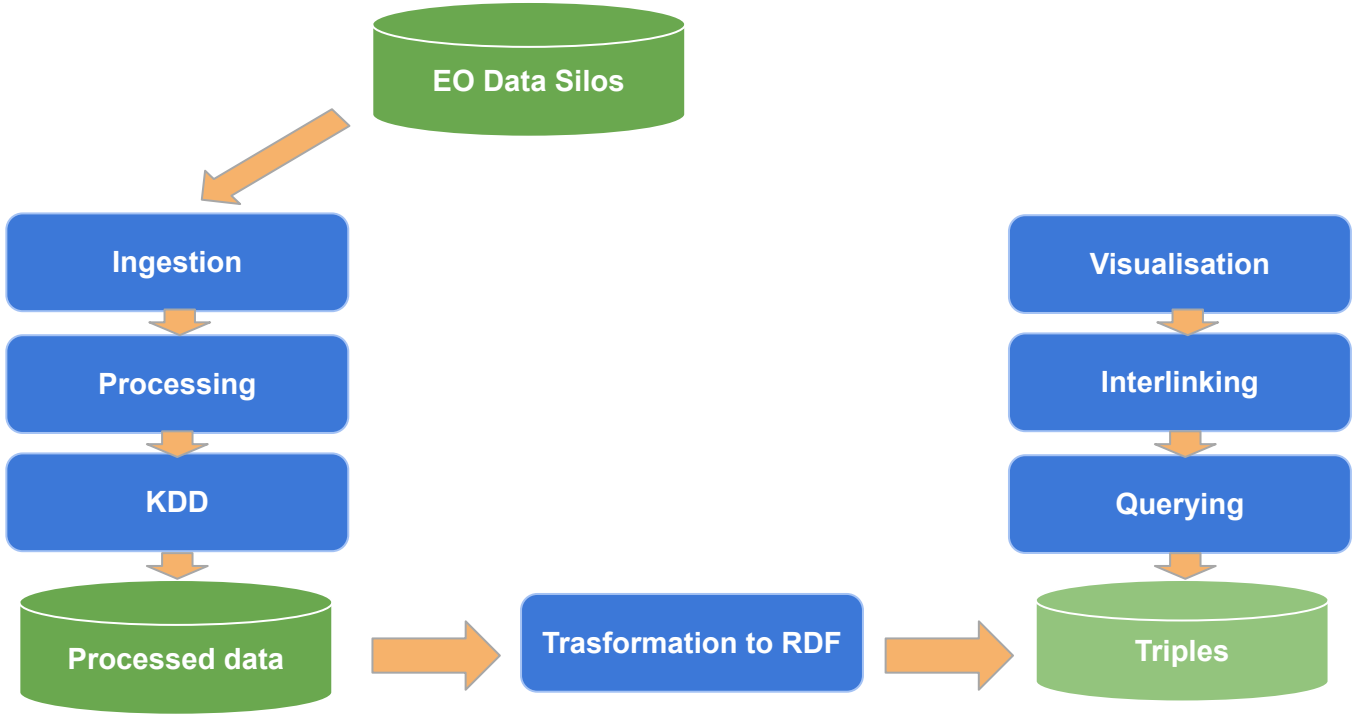
- Give me all roads which are not flooded by more than 10cm
  - Road network vector dataset
  - Flood altitude raster dataset

```
SELECT ?road
WHERE      {
           ?road a ex:Road ; geo:hasGeometry ?roadseg .
           ?roadseg geo:asWKT ?roadseg_wkt .
           ?floodarea a ex:FloodRiskArea ;
                      geo2:asCoverage ?floodarea_cov .
           ?floodarea_cov geo2:asCoverageJSON ?floodarea_covjson.
           BIND(geo2:rasterSmaller(?floodarea_covjson,10) AS? relfloodarea)
           FILTER(geo2:intersects(?roadseg_wkt,?relfloodarea))
           }
```

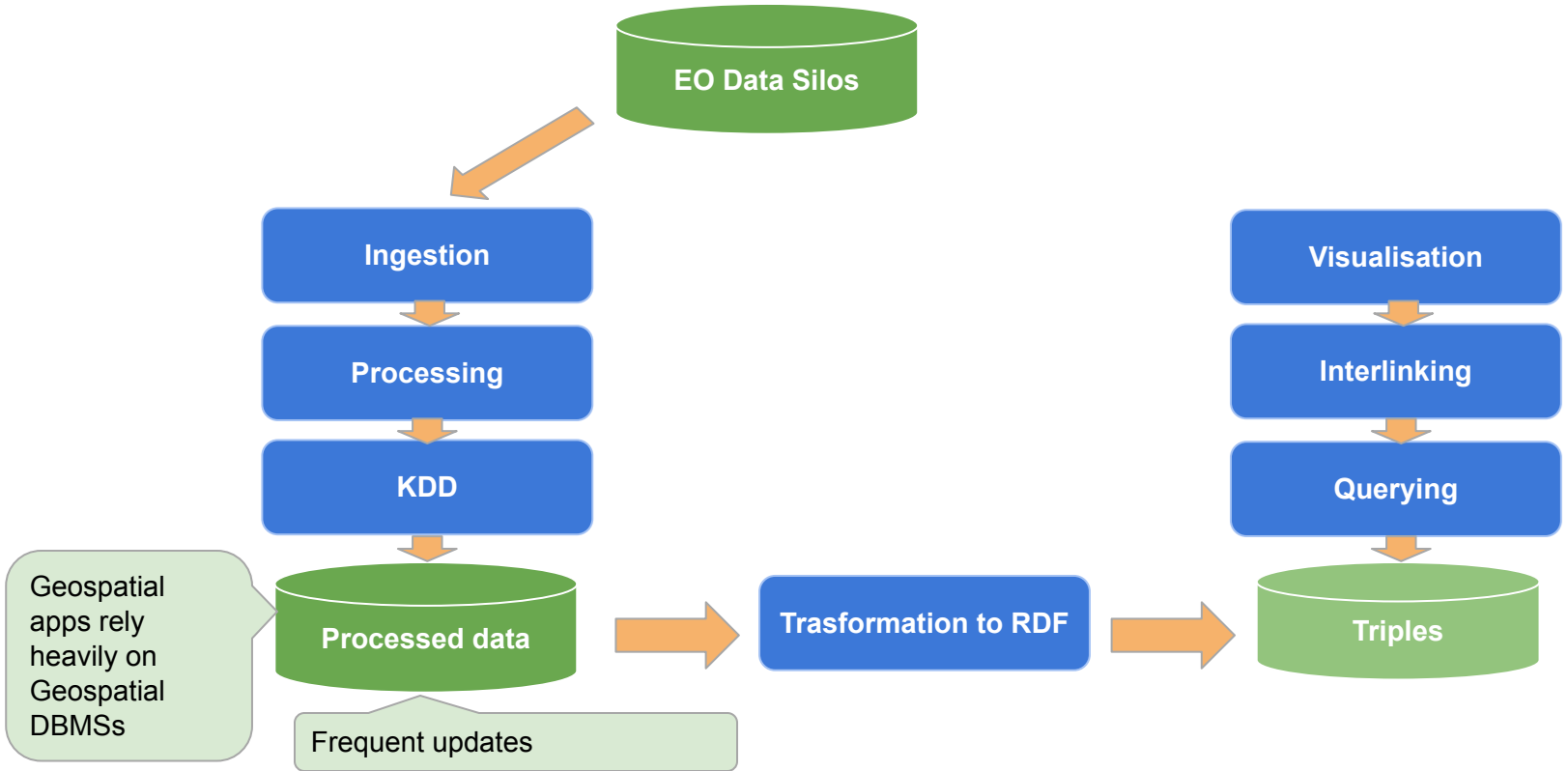
Geospatial ontology-based data access

Motivation

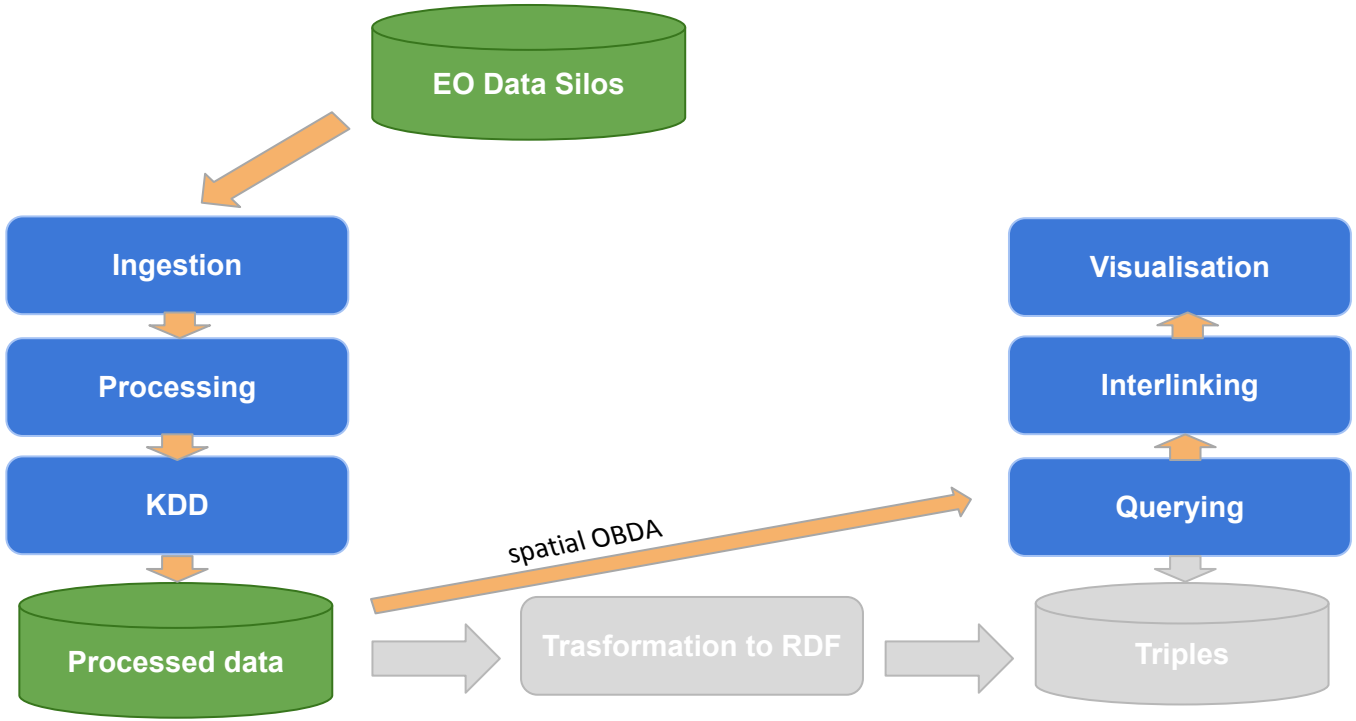
Data Science Pipeline for Big Linked EO Data



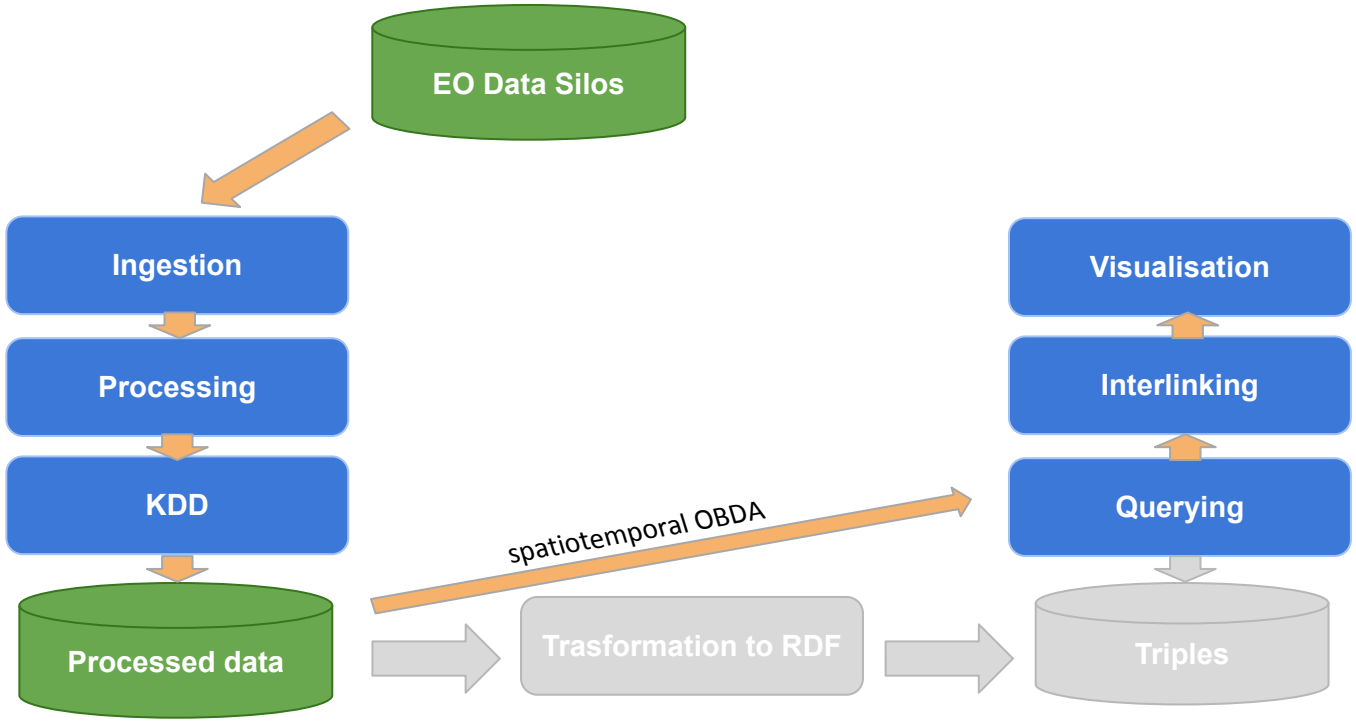
Data Science Pipeline for Big Linked EO Data



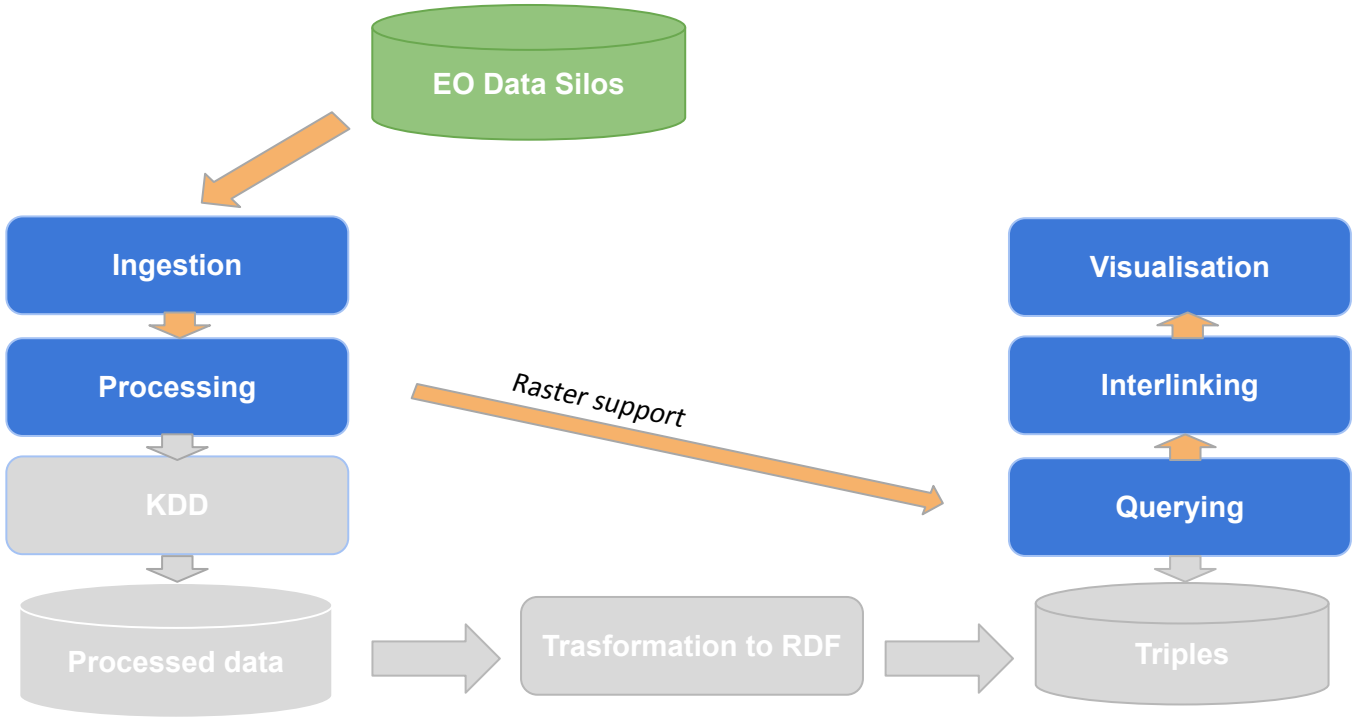
Data Science Pipeline for Big Linked EO Data



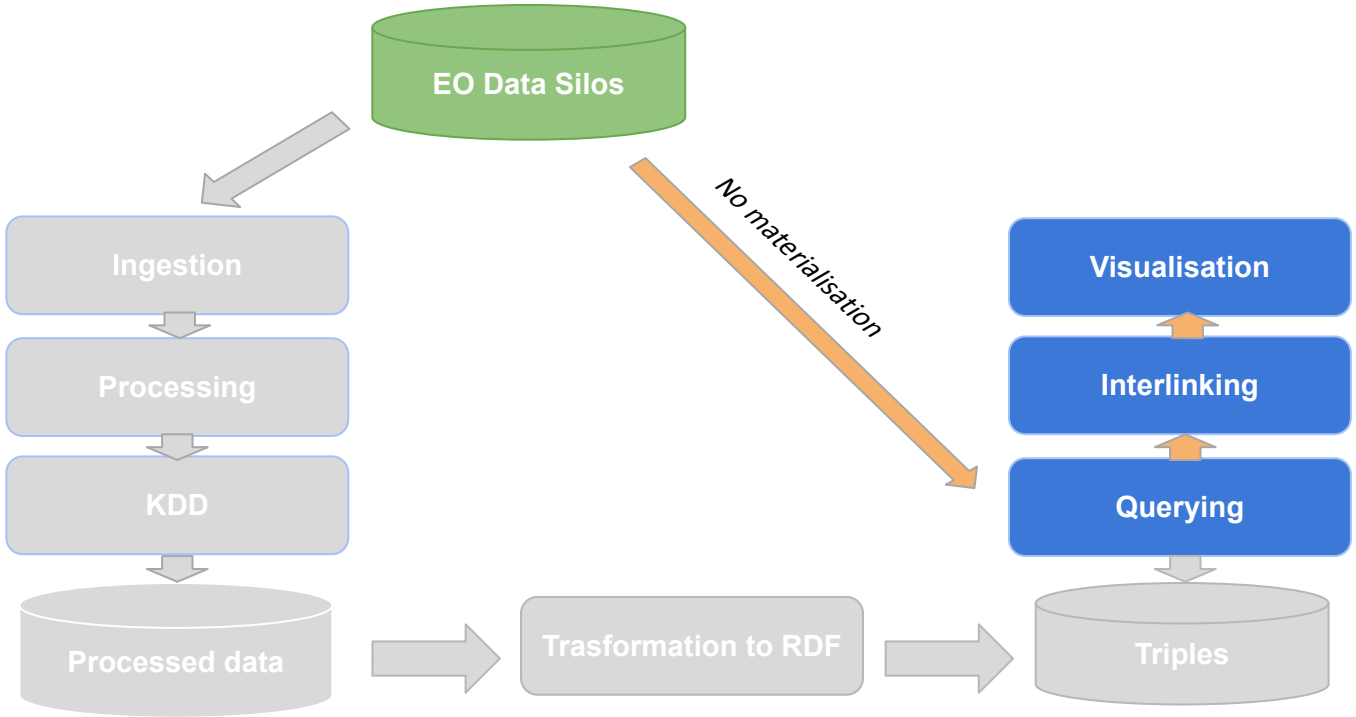
Data Science Pipeline for Big Linked EO Data



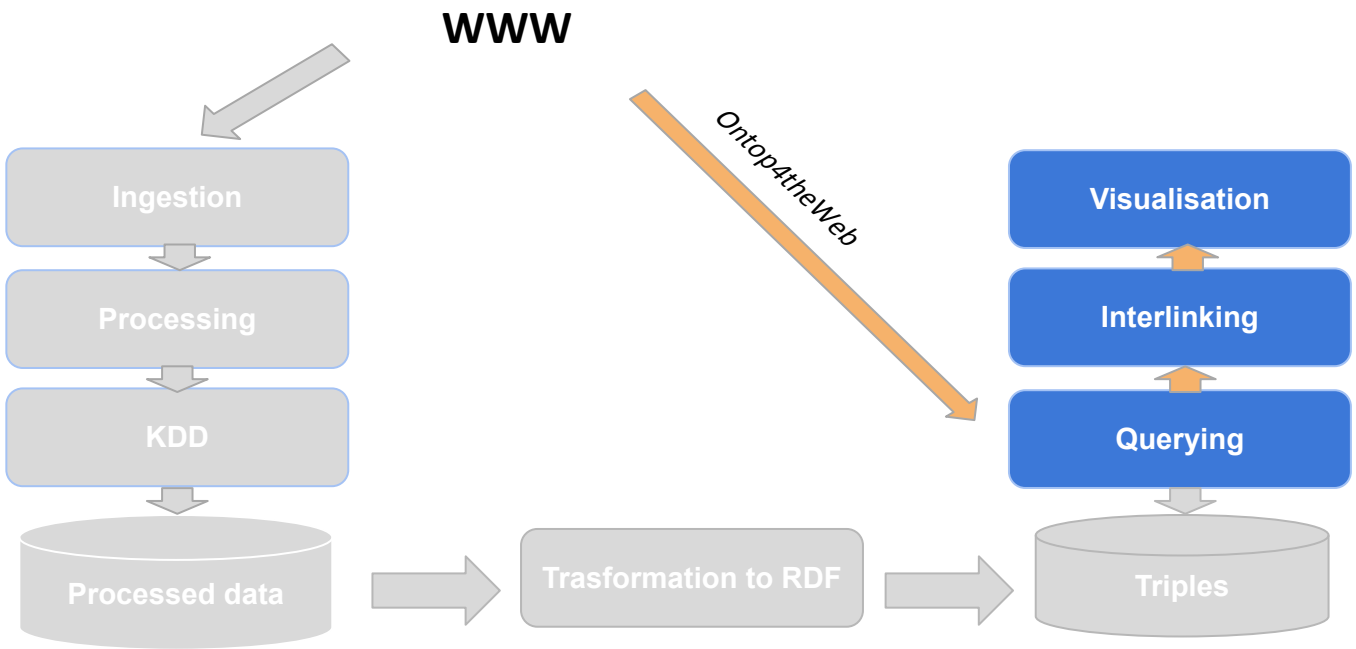
Data Science Pipeline for Big Linked EO Data



Data Science Pipeline for Big Linked EO Data

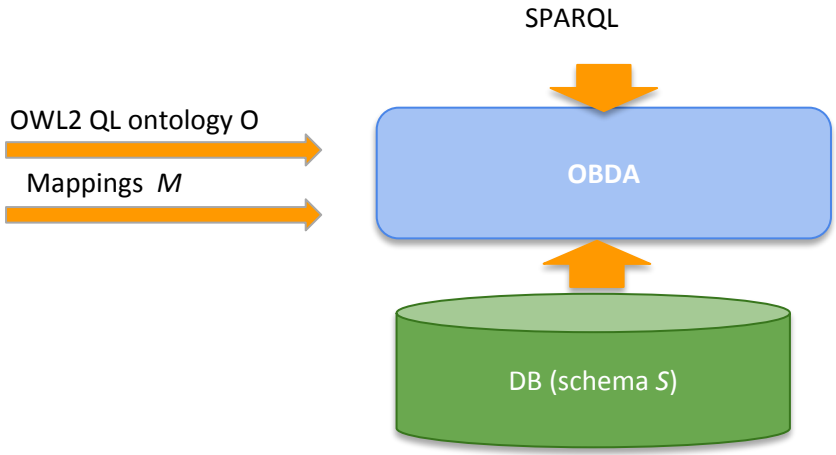


Data Science Pipeline for Big Linked EO Data



OBDA Paradigm

[Poggi et. al, JDS2008]
[Xiao et. al, IJCAI2018]



O: Ontology

S: DB schema

M: S -> O

Related work- OBDA

- Geospatial RDB2RDF systems: GeoTriples, TriplesGeo
- Mapping languages: R2RML (W3C standard), OBDA
- OBDA systems (virtual OBDA):
  - Ontop [Calvanese., SWJ'17; Latest version: ISWC2020]
  - Ultrawrap [Sequeda et al., JWS'13]
  - MASTRO [Calvanese, 2011]
- Geospatial OBDA systems:
  - Oracle Spatial and Graph 12c release 2

OBDA Mappings

[MappingDeclaration] [[mappingId gag_geometry

target gag:geometry/{gid}/ gag:asWKT {geom}^^geo:wktLiteral .

source select distinct gid,geom from gag

mappingId clc_geometry

Target clc:/{gid}/ clc:hasGeometry clc:/geometry/{id}/ . clc:/{gid}/ clc:asWKT {geom}^^geo:wktLiteral .

source select distinct gid, geom,id from clc

mappingId clc_id

target clc:/{gid}/ clc:hasID {gid} . clc:/{gid}/ clc:hasLandUse {code_00} .

source select distinct gid, geom, code_00 from clc

mappingId clc_type

target clc:/{gid}/ clc:type clc:type . clc:/{gid}/ rdf:type clc:Area .

source select distinct gid, geom from clc]]

R2RML example

```
[ a rr:TriplesMap ;
rr:logicalTable [ a rr:R2RMLView ;
rr:sqlQuery "select distinct gid,geom from clc" ] ; rr:predicateObjectMap [ a rr:PredicateObjectMap ;
rr:objectMap [ a rr:ObjectMap , rr:TermMap ; rr:column "geom" ;
rr:termType rr:Literal ] ; rr:predicate clc:asWKT ] ;
rr:subjectMap [ a rr:TermMap , rr:SubjectMap ; rr:template gag:{gid} ; rr:termType rr:IRI ]] .
```

Virtual Triples

clc:20440 rdf:type geo:Geometry;

geo:asWKT “POLYGON(…)”^^geo:wktLiteral .

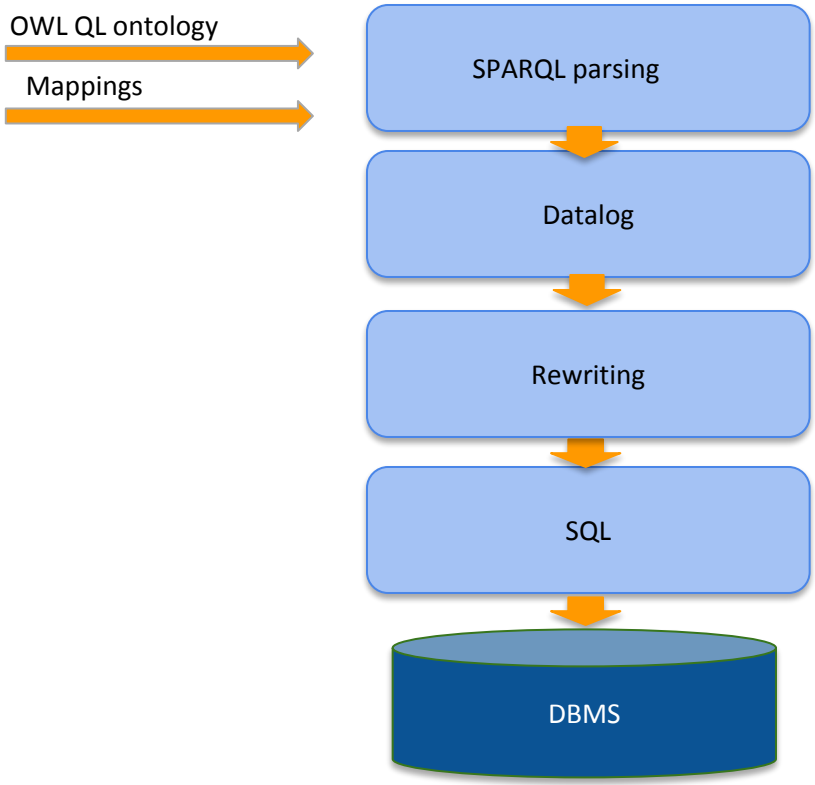
Clc:20512 rdf:type geo:Geometry;

geo:asWKT “POLYGON(…)”^^geo:wktLiteral .

…

	gid integer	code_00 character varying(100)	id character varying(18)	remark character varying(20)	area_ha numeric	shape_leng numeric	shape_area numeric	geom geometry
1	20440	BroadLeavedForest	EU-1900387		9169698	72.0513230	5691.69698	010300002
2	20512	BroadLeavedForest	EU-1900769		3331793	35.0022857	9833.31793	010300002
3	20543	BroadLeavedForest	EU-1900881		9247076	6.17039328	189.247076	010300002
4	20797	BroadLeavedForest	EU-1901587		7822436	3.55011923	107.822436	010300002
5	20904	BroadLeavedForest	EU-1901816		1899830	97.0454395	0618.99830	010300002

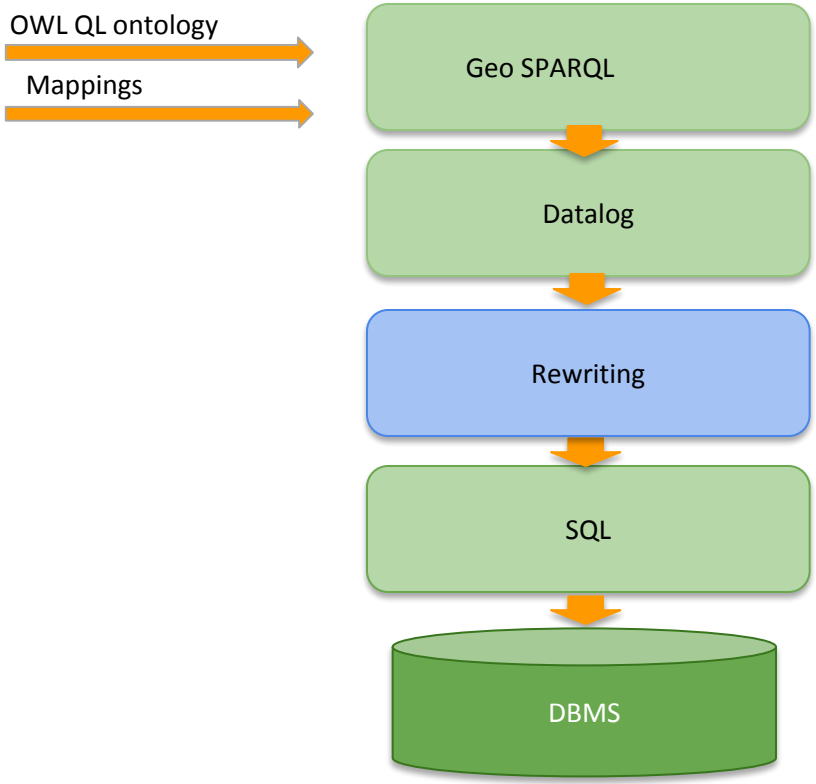
Ontop Architecture



[Calvanese et. al, SWJ2017]

https://ontop.inf.unibz.it/

Ontop-spatial Architecture



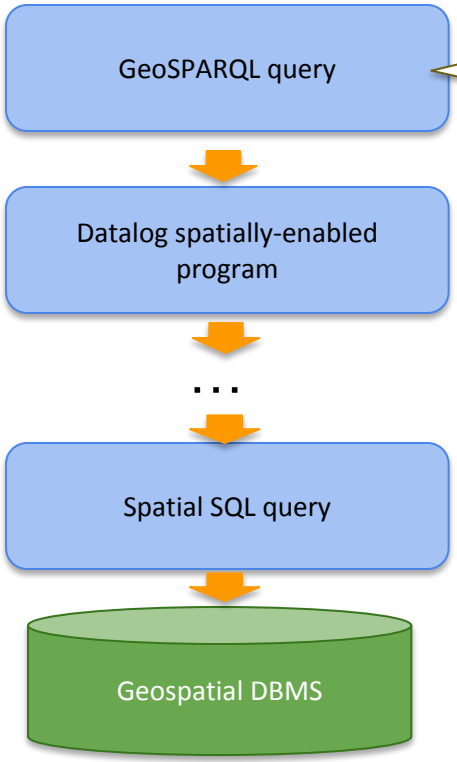
http://ontop-spatial.di.uoa.gr

GeoSPARQL-to-SQL translation

Input: GeoSPARQL query q, Ontology $\mathcal{T}$, Mapping $\mathcal{M}$
Output: An SQL expression

// offline phase
$\mathcal{T}_{classified} = \text{classify}(\mathcal{T} \cup geo)$ ▷ ontology classification
$\mathcal{M}_{\mathcal{T}} \leftarrow saturate(\mathcal{M}, \mathcal{T}_{classified})$ ▷ mapping saturation
// online phase
$Q \leftarrow rew_{geo}(q)$ ▷ GeoSPARQL query write
$S \leftarrow$ list of nodes in Q in a bottom-up topological order
sql $\leftarrow$ empty map from nodes to SQL expressions
for node $n \in S$ **do**
  if n is triple pattern **then** ▷ translating leaves
    sql$[n] \leftarrow$ replace-Tmap-def$(n, \mathcal{M}_{\mathcal{T}})$
  else ▷ translating non-leaf nodes
    if n = JOIN(n_1, n_2) **then**
      sql$[n] \leftarrow$ InnerJoin(sql$[n_1]$, sql$[n_2]$)
    else if n = OPTIONAL(n_1, n_2, e) **then**
      sql$[n] \leftarrow$ LeftJoin(sql$[n_1]$, sql$[n_2]$, e)
    else if n = UNION(n_1, n_2) **then**
      sql$[n] \leftarrow$ Union(sql$[n_1]$, sql$[n_2]$)
    else if n = FILTER(n_1, e) **then**
      sql$[n] \leftarrow$ Filter(sql$[n_1]$, e)
    else if n = PROJECT(n_1, p) **then**
      sql$[n] \leftarrow$ Project(sql$[n_1]$, p)
    end if
  end if
end for
return sql$[S.\text{last}()]$

GeoSPARQL-to-SQL translation

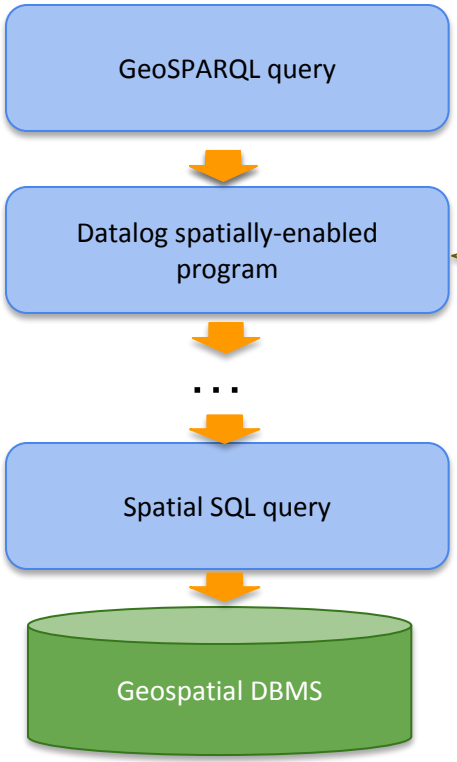



```
SELECT ?x1 ?x2
 WHERE{
   ?x1 geo:asWKT ?g1 .
    ?x2 geo:asWKT ?g2 .
    FILTER (geof:sfOverlaps(?g1,?g2)) }
```

OR

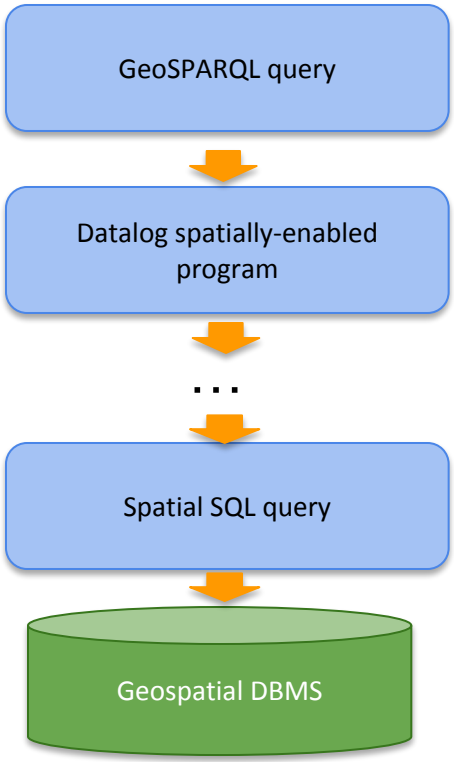
```
SELECT ?x1 ?x2
WHERE{
  ?x1 geo:sfOverlaps ?x2 }
```

GeoSPARQL-to-SQL translation



ans1(x1,g1) :- ans2(g1,g2,x1,x2)
ans2(g1,g2,x1,x2) :- ans4(g1,g2,x1,x2), **SF-OVERLAPS**(g1,g2)
ans4(g1,g2,x1,x2) :- ans8(g1,x1), ans9(g2,x2)
ans8(g1,x1) :- http://www.opengis.net/ont/geosparql#asWKT(x1,g1)
ans9(g2,x2) :- http://www.opengis.net/ont/geosparql#asWKT(x2,g2)

GeoSPARQL-to-SQL translation




```
SELECT *
FROM (
SELECT DISTINCT
   1 AS "x1QuestType", ...
   10 AS "g1QuestType", NULL AS "g1Lang",
CONCAT('<http://www.opengis.net/def/crs/EPSG/0/4326> ' ,
ST_AsText(QVIEW2."strdfgeo")) AS "g1"
 FROM
geo_values QVIEW1,
geo_values QVIEW2
WHERE
(ST_Overlaps(QVIEW2."strdfgeo",QVIEW1."strdfgeo")) AND
QVIEW1."id" IS NOT NULL AND
QVIEW1."strdfgeo" IS NOT NULL AND
QVIEW2."id" IS NOT NULL AND
QVIEW2."strdfgeo" IS NOT NULL
) SUB_QVIEW
```

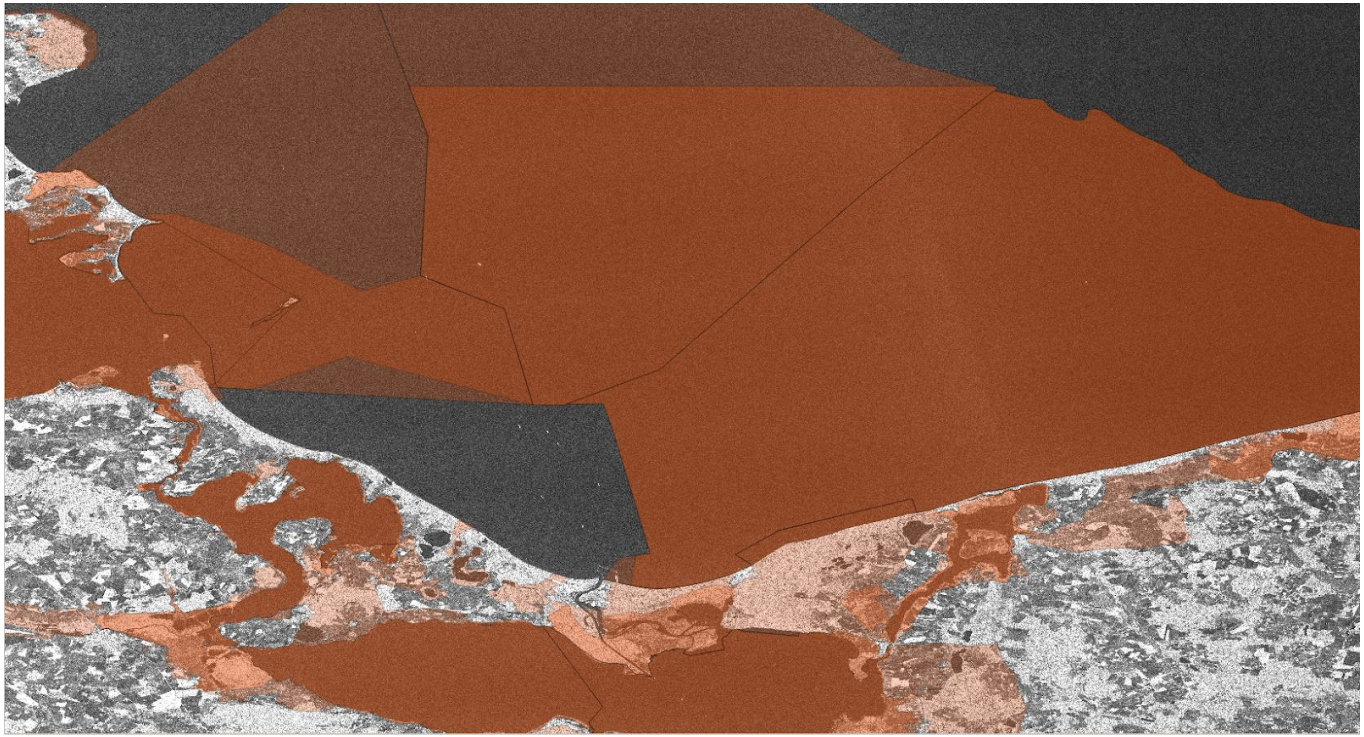
Example GeoSPARQL Query

```
PREFIX gag: <http://geo.linkedopendata.gr/gag/ontology/>
PREFIX clc: <http://geo.linkedopendata.gr/corine/ontology#>

SELECT distinct ?x1 ?lu
WHERE {
        ?x1 geo:asWKT ?g1 .
        ?x2 geo:asWKT ?g2 .
        ?x2 clc:hasLandUse ?lu .
        FILTER(geof:sfIntersects(?g1,?g2)) }
```

Raster data support

Natura 2000 sites and Sentinel-1 image



Raster Data Sources

```
[[ mappingId      sentinel1
   target         :{rid} rdf:type :rasterCell ;  :hasGeometry {rast} .
   source         select rid, ST_DumpAsPolygons(rast).geom as geom from S1;
mappingId       natura
  target         : {id} rdf:type :NaturaSite; geo:hasGeometry :{gid} .
                :{gid} geo:asWKT {geom}^^geo:WKTLiteral .
  source         select * from Natura ]]
```

Data sources

S1 [rid | rast]

Natura [gid | id | sitename | sitetype | sitecode| ms |geom]

GeoTIFF Sentinel-1 image imported in PostGIS as table (raster geometries)

Shapefile describing NATURA 2000 sites and boundaries (vector geometries)

Example Query: protected area monitoring

Retrieve NATURA sites that intersect with raster cells of the Sentinel-1 image.

```
SELECT ?nat
WHERE{
   ?r rdf:type :rasterCell .
   ?r :hasGeometry ?rast .
   ?nat rdf:type :NaturaSite .
   ?nat geo:hasGeometry ?g .
  ?g geo:asWKT ?geom .
FILTER(geof:sfIntersects(?geom,?rast))
```

Vector geometries will be bound

Raster geometries will be bound

Evaluation

Dataset	*Table Size*	*No. of rows/geometries*	*Avg #points/ geometry*
Corine Land Cover (CLC)	283MB	44834	187.84
Hotspots	35 MB	37048	5
Global Administrative Geography (GAG)	24 MB	326	3020.14
OSM-Buildings	42 MB	155474	6.5
OSM-Landuse	20 MB	40220	19.4
OSM-places	2.4 MB	13043	1
OSM-points	12 MB	61664	1
OSM-railways	2 MB	4996	13.3
OSM-roads	250 MB	514403	19
OSM-waterways	16 MB	20565	39.84

Queries

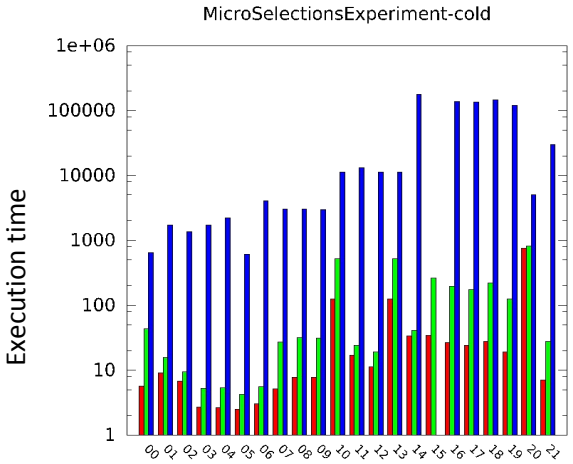
No	Query	#BGPs	results
00	Equals_GADM_P	1	0
01	Contains_GADM_P	1	9
02	Contains_GADM_P	1	0
03	Equals_GADM_L	1	1
04	Overlaps_GADM_L	1	0
05	Contains_GADM_L	1	0
06	Intersects_CLC_L	1	5
07	Contains_CLC_L	1	0
08	Equals_CLC_L	1	5
09	Overlaps_CLC_L	1	0
10	Overlaps_CLC_P	1	132
11	Intersects_CLC_P	1	533
12	Contains_CLC_P	1	401
13	Equals_CLC_P	1	0
14	Intersects_LGD_P	2	2749
15	Intersects_LGD_B	2	2749
16	Intersects_LGD_PL	2	2626
17	Intersects_LGD_P	2	2522
18	Intersects_LGD_LU	2	2722
19	Intersects_LGD_ROA	2	2387
20	Intersects_LGD_bigP	1	729189
21	Intersects_LGD_P2	3	5

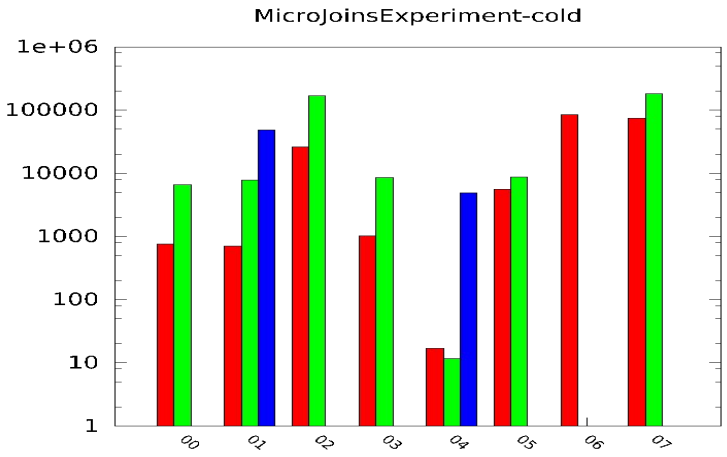
No	Query	#BGPs	results
00	Within_CLC_GADM	2	34114
01	Intersects_GADM_GADM	2	1556
02	Overlaps_GADM_CLC	2	17035
03	Intersects_LGD_GADM	3	154725
04	Intersects_LGD_LGD_Mus	4	2
05	Intersects_LGD_GADM	2	819319
06	Intersects_LGD_LGD	1	3686229
07	Crosses_LGD_LGD_Roads	4	178602

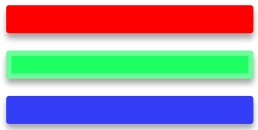
Highly selective query

Poorly selective query

Experiments- cold cache





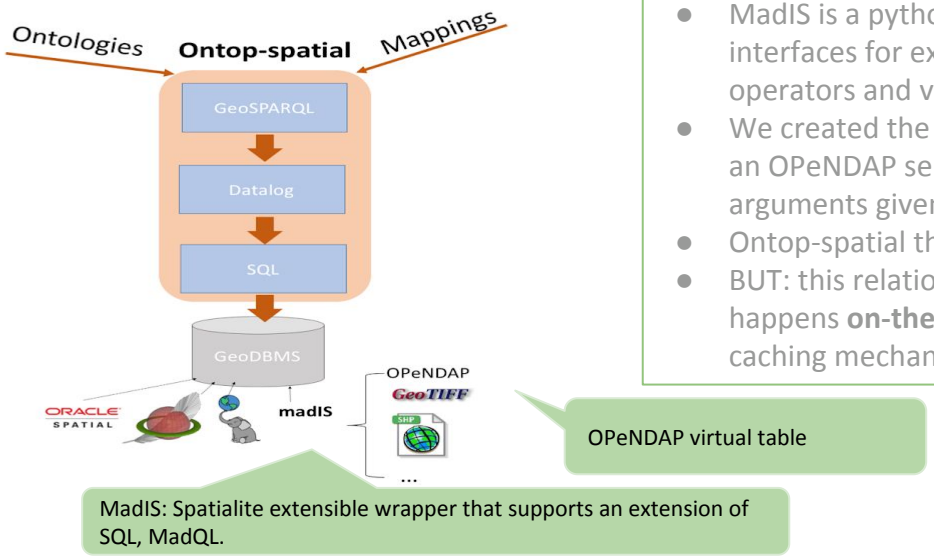


Performance evaluation conclusion

- Ontop-spatial has better performance than Strabon and System-O (JWS paper).
- Performance difference also depends on the relational schema, complexity of geometries, and number of geometries

Ontop-spatial for Copernicus Data

Querying OPENDAP data on-the-fly



- Support for madIS as a back-ed DBMS. For this, the JDBC connector of madIS was extended
- MadIS is a python wrapper for Sqlite and provides high level interfaces for extending new Sqlite functions (row, aggregate operators and vtables) using Python.
- We created the opendap operator as a MadIS vtable that connects to an OPeNDAP server and displays the results (according to the arguments given) as relational data
- Ontop-spatial then handles this data as a relational data source
- BUT: this relational view is not created until a query is triggered. It all happens **on-the-fly without materializing anything** (unless the caching mechanism is enabled)

The OPeNDAP virtual table

```
[MappingDeclaration]
@collection [[
mappingId      opendap_mapping
 target        lai:{id} rdf:type sosa:Observation;
                 lai:LAI {LAI}^^xsd:float ;  lai:detectionTime {time}^^xsd:dateTime;
                 geosparql:asWKT {wkt}^^geosparql:wktLiteral
Source        select id, LAI, time, wkt  from  (ordered dap
url:https://analytics.ramani.ujuizi.com/%28http://analytics.ramani.ujuizi.com/SOURCES/%2528http://dap.vgt.vito.be/thredds/dodsC/Copernicus/LAI%2529readdods/.LAI/lon/2.22422456741344/2.46725964546204/RANGEEDGES/lat/48.815757751465/48.901653289795/RANGEEDGES/dods%29readdods/.LAI/time/%28Aug%202017%29VALUES/lat/%2848.815757751465N%29%2848.901653289795N%29RANGEEDGES/lon/%282.22422456741344E%29%282.46725964546204E%29RANGEEDGES/lon//x/renameGRID/lat//y/renameGRID/time//T/renameGRID/dods rate:10) where LAI < 1
]]
```

dap vtable operator

Cached results can be used using the rate perameter: New results are fetched every 10 minutes.

Cleaning data on-the-fly using SQL statements

Project LAI values and locations

GeoSPARQL

```
select ?s ?g ?lai
        where {
        ?s lai:hasLai ?lai .
        ?s geo:asWKT ?g . }
```

SQL

```
SELECT
  1 AS "pQuestType", NULL AS "pLang", 'http://geo.linkedopendata.gr/lai/ontology#hasAirTemperature' AS "p"FROM
(select id, airt , time, wkt  as loc from (dap
      url:https://analytics.ramani.ujuizi.com/%28http://analytics.ramani.ujuizi.com/SOURCES/%2528http://dap.vgt.vito.be/thredds/dodsC/Copernicus/LAI%2529readdods/.LAI/lon/2.22422456741344/2.46725964546204/RANGEEDGES/lat/48.815757751465/48.901653289795/RANGEEDGES/dods%29readdods/.LAI/time/%28Aug%202017%29VALUES/lat/%2848.815757751465N%29%2848.901653289795N%29RANGEEDGES/lon/%282.22422456741344E%29%282.46725964546204E%29RANGEEDGES/lon//x/renameGRID/lat//y/renameGRID/time//T/renameGRID/dods rate:10) QVIEW1
WHERE
QVIEW1.id IS NOT NULL AND
QVIEW1.airt IS NOT NULL
```

The requested dataset is downloaded and a virtual table is constructed on-the-fly and is populated with this data. The downloaded data from OPeNDAP is cached for 10 minutes (rate parameter)

Current and future work

Recent updates:

- GeoSpark support for scalable spatial analysis
- Store for RDF data on Apache Hive using vertical partitioning
- On-the-fly creation of mappings based on VP schema
- Support for more GeoSPARQL operators (non binary operators such as distance)

Future work

- Ontop-spatial will be extended with support for Data Cubes