# Web of Data

Master 2 - Data Science

Joe Raad

joe.raad@universite-paris-saclay.fr

université PARIS-SACLAY

# Plan for this session

1. Introduce the GeoSPARQL standard
2. Create a simple RDF graph of French cities
3. Have fun with GeoSPARQL ☺

https://github.com/raadjoe/wod-2023-24

# What you need

- Basic knowledge of SPARQL
- GraphDB (triple store)

  Installation and Tutorial:

  https://github.com/raadjoe/wod-2023-24
- Protégé (ontology editor)

  https://protege.stanford.edu/download/protege/4.3/installanywhere/Web_Installers/

  Or Web Protégé: https://webprotege.stanford.edu/
- Optional: Notepad++ (text editor with syntax highlighting)

# GeoSPARQL

- Standard by the Open Geospatial Consortium (OGC) since 2012
- Goal: representing and querying geospatial data in the Web of Data
- GeoSPARQL are aligned to other Geo standards (outside RDF)
- Contributions
    1. Vocabulary for representing geospatial data in RDF
    2. A set of SPARQL extension functions for spatial computations
    3. A set of RIF rules for spatial reasoning (not covered today)

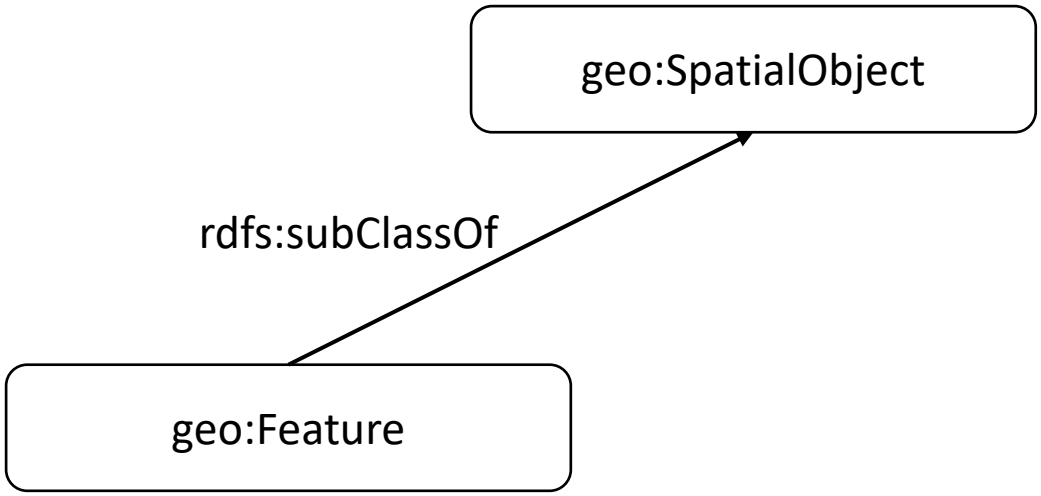
# 1. GeoSPARQL Vocabulary

*geo: <http://www.opengis.net/ont/geosparql#>*


**geo:SpatialObject:** This class represents everything that can have a spatial representation

# 1. GeoSPARQL Vocabulary

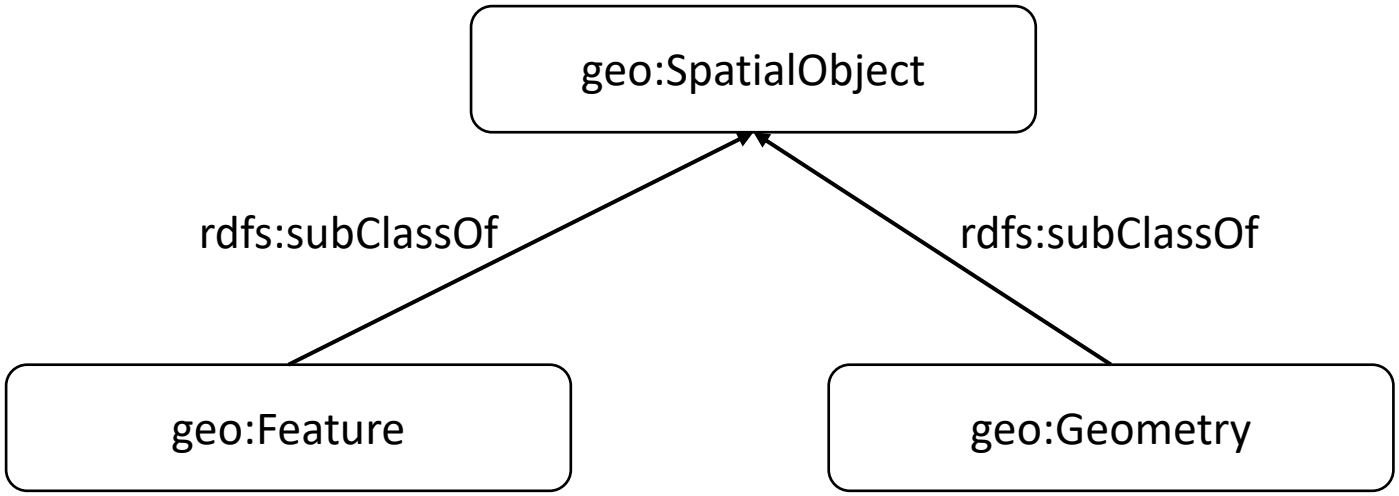
*geo: <http://www.opengis.net/ont/geosparql#>*




**geo:Feature:** This class represents the top-level feature type

# 1. GeoSPARQL Vocabulary

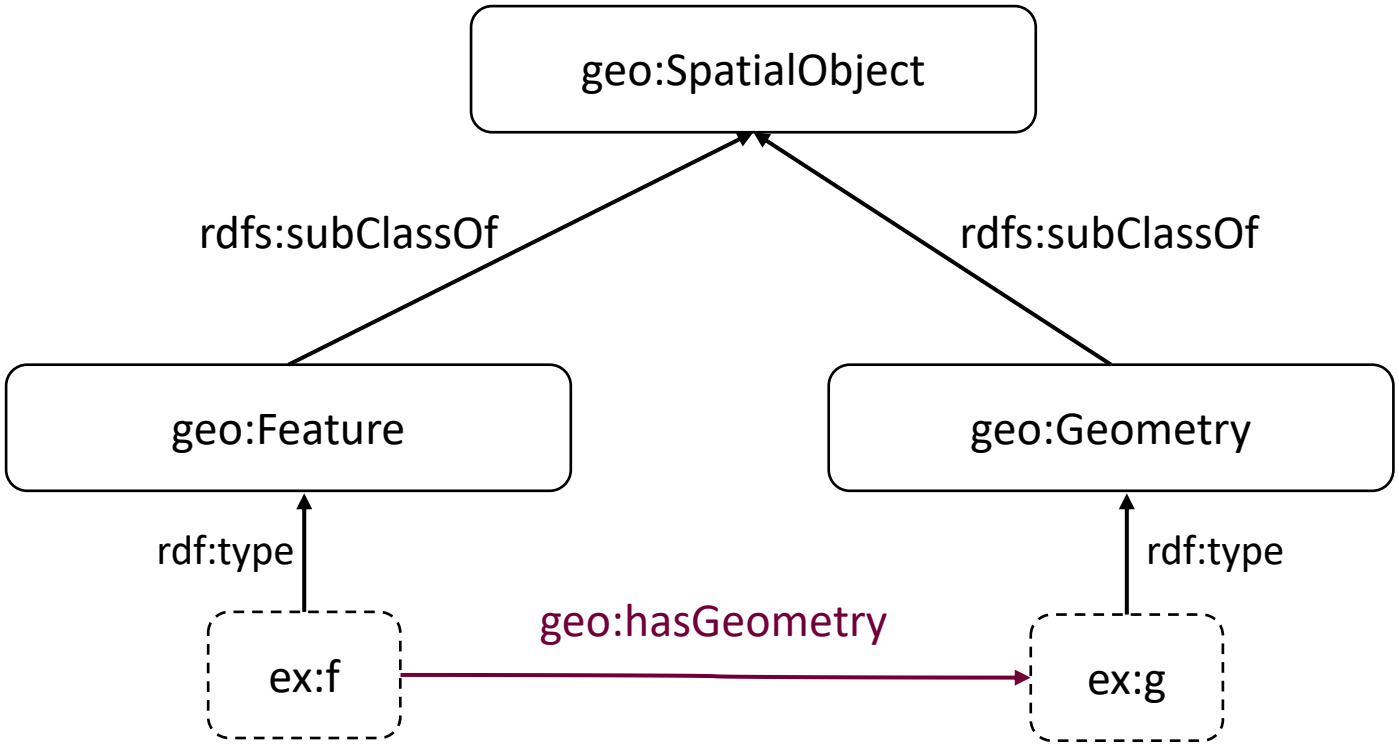
*geo: <http://www.opengis.net/ont/geosparql#>*




**geo:Geometry:** This class represents the top-level geometry type

# 1. GeoSPARQL Vocabulary

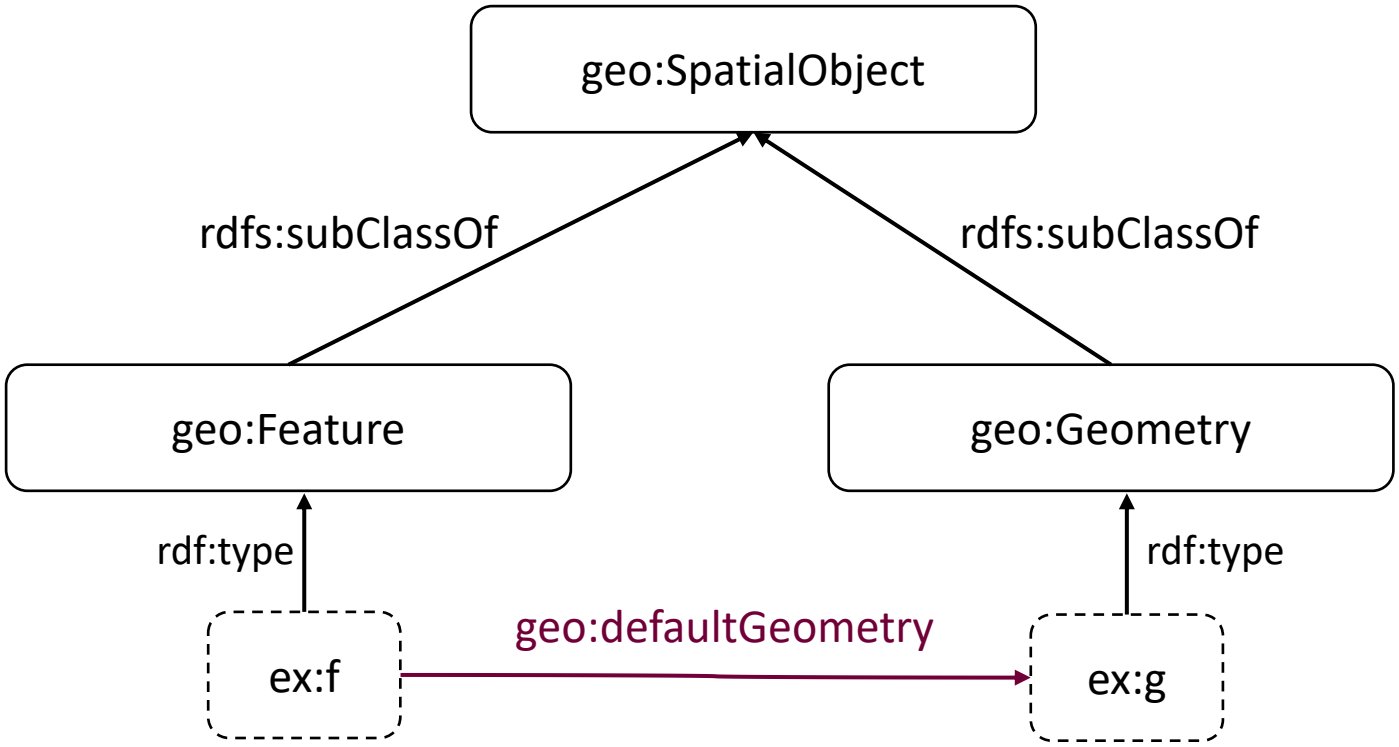
*geo: <http://www.opengis.net/ont/geosparql#>*




**geo:hasGeometry:** A spatial representation for a given feature

# 1. GeoSPARQL Vocabulary

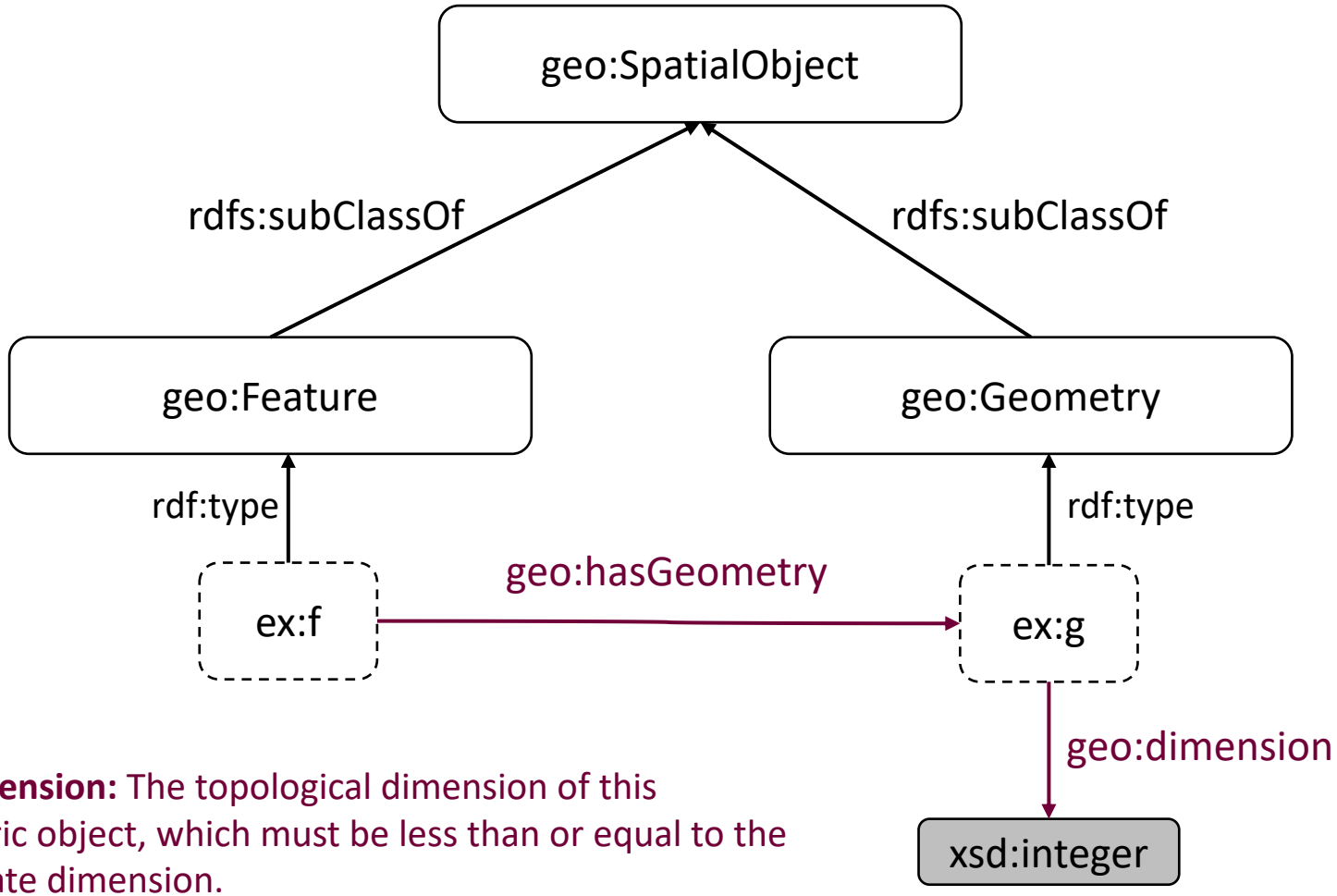
*geo: <http://www.opengis.net/ont/geosparql#>*




**geo:defaultGeometry:** The default geometry to be used in spatial calculations. It is Usually the most detailed geometry.

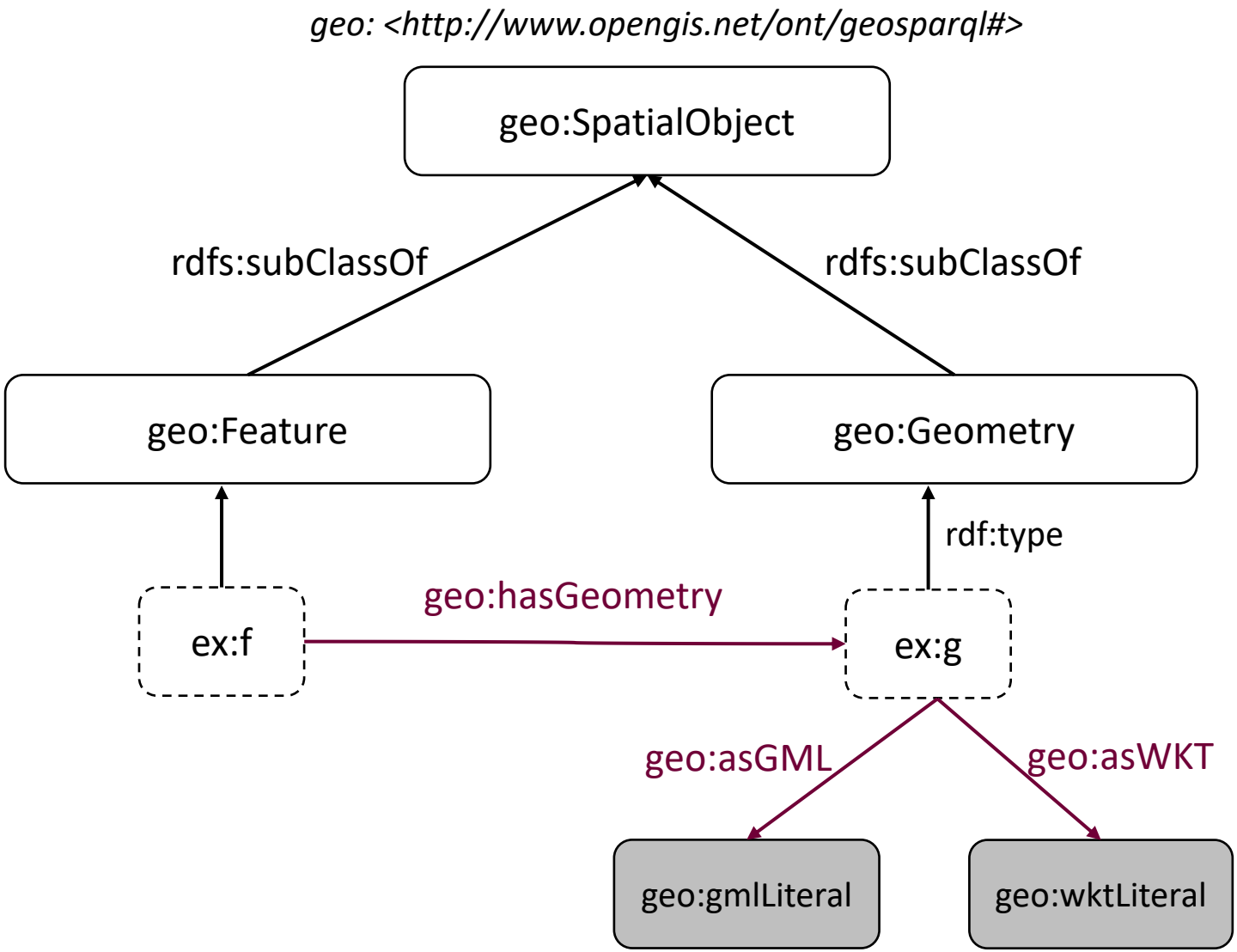
# 1. GeoSPARQL Vocabulary

*geo: <http://www.opengis.net/ont/geosparql#>*




**geo:dimension:** The topological dimension of this geometric object, which must be less than or equal to the coordinate dimension.

# 1. GeoSPARQL Vocabulary



# 1. GeoSPARQL Vocabulary

## Example

***geo:*** *<http://www.opengis.net/ont/geosparql#>*
***wod:*** *<https://paris-saclay.fr/courses/wod/>*

```
iut:city1       rdf:type            iut:City ;
                rdfs:label          “Orsay”@en ;
                rdf:type            geo:Feature ;
                geo:hasGeometry     iut:shape1 .
iut:shape1      geo:asWKT           “Point(2.18737051177 48.7004093953)”^^geo:wktLiteral
```

# GeoSPARQL

- Standard by the Open Geospatial Consortium (OGC) since 2012
- Goal: representing and querying geospatial data in the Web of Data
- GeoSPARQL are aligned to other Geo standards (outside RDF)
- Contributions

1. Vocabulary for representing geospatial data in RDF
2. A set of SPARQL extension functions for spatial computations
3. A set of RIF rules for spatial reasoning (not covered today)

# GeoSPARQL Functions

***geof:*** *<http://www.opengis.net/def/function/geosparql/>*

- geof:contains

  A query function that returns true if the first geometry argument spatially contains the second geometry argument

- geof:distance

  A query function that returns the distance between the two closest points of the input geometries

- geof:intersection

  A query function that returns a geometry consisting of all points that are part of both input geometries.

- …

# Plan for this session

1. Introduce the GeoSPARQL standard
2. Create a simple RDF graph of French cities
3. Have fun with GeoSPARQL ☺

# Exercice 1

**Tool:**
https://yasgui.triply.cc

**Graph:**
https://druid.datalegend.net/nlgis/gemeentegeschiedenis/

**SPARQL endpoint:**
https://api.druid.datalegend.net/datasets/nlgis/gemeentegeschiedenis/services/gemeentegeschiedenis/sparql

**Complete the following query :**

```
prefix geo: <http://www.opengis.net/ont/geosparql#>

SELECT ?feature ?shape {
  This query shows 5 random cities and their geographical shape
}
LIMIT 5
```

# Exercice 2

**Tool:**



**Data:**

https://github.com/raadjoe/wod-2023-24/tree/main/session-2

**Task:**

## Create the dataset

1. Create a simple ontology that can model the data (using Protégé)
2. Convert the CSV dataset to an RDF graph (using Ontorefine)
3. Upload the ontology and the RDF graph to the same repository (using GraphDB)

## Query the dataset

1. Show the number of cities for each French region
2. Show the 10 closest cities to “Orsay” (show the name of the city and the distance)
3. Show the two cities that are the furthest from each other in the “Essonne” dept.

**This slightly modified dataset was downloaded from:**

https://www.data.gouv.fr/fr/datasets/communes-de-france-base-des-codes-postaux/