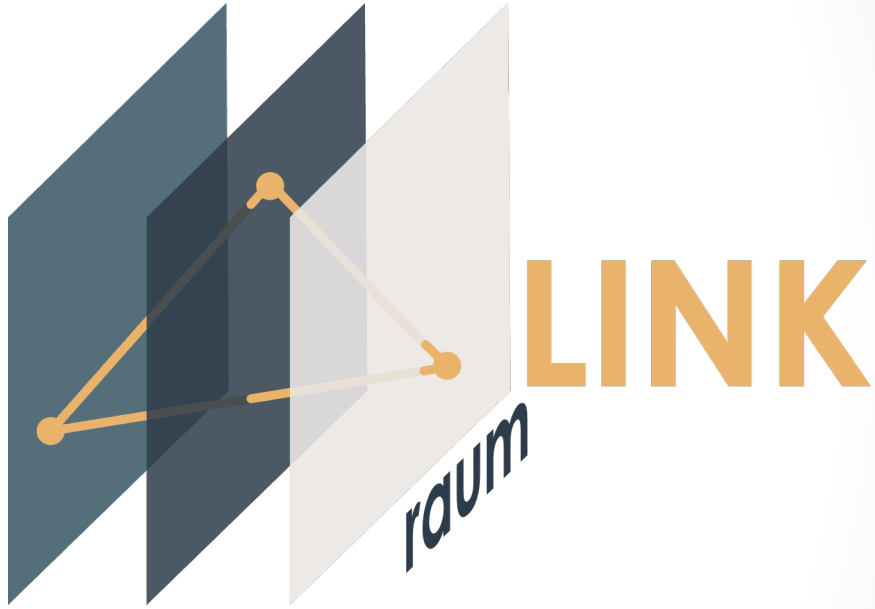
LBD and Spatial Querying

Or: How to bend SPARQL to your will

LDAC2024 Summer School Lecture

Oliver Schulz



About me

- Background in Architecture
  - (Focus: Existing Buildings)
- Several years of experience in BIM VR Development and BIM Coordination
- Research Assistant and PhD Candidate at Design Computation, RWTH Aachen
- Research focus on Common Data Environments

cc Formitas AG

Content

- Introduction
- Basics
- Geospatial Queries
- Spatial 3D Queries
- Challenge
- Conclusion

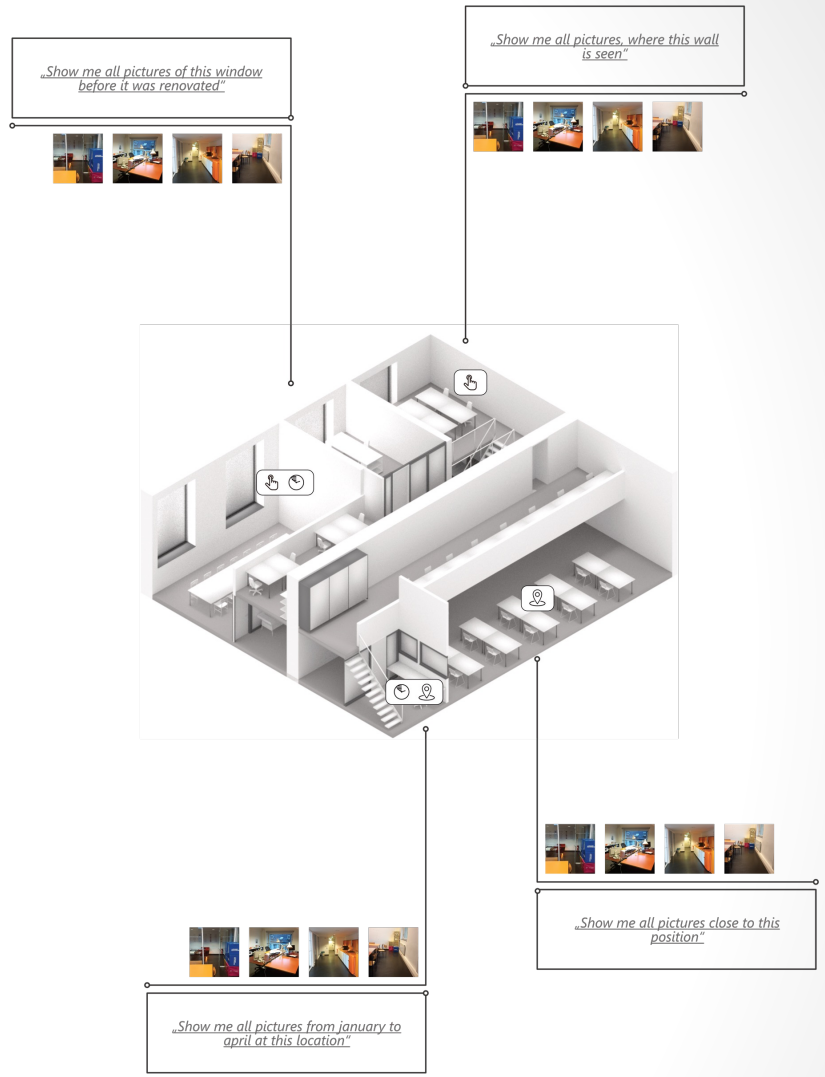
Spatial Queries?

Definition (for today):

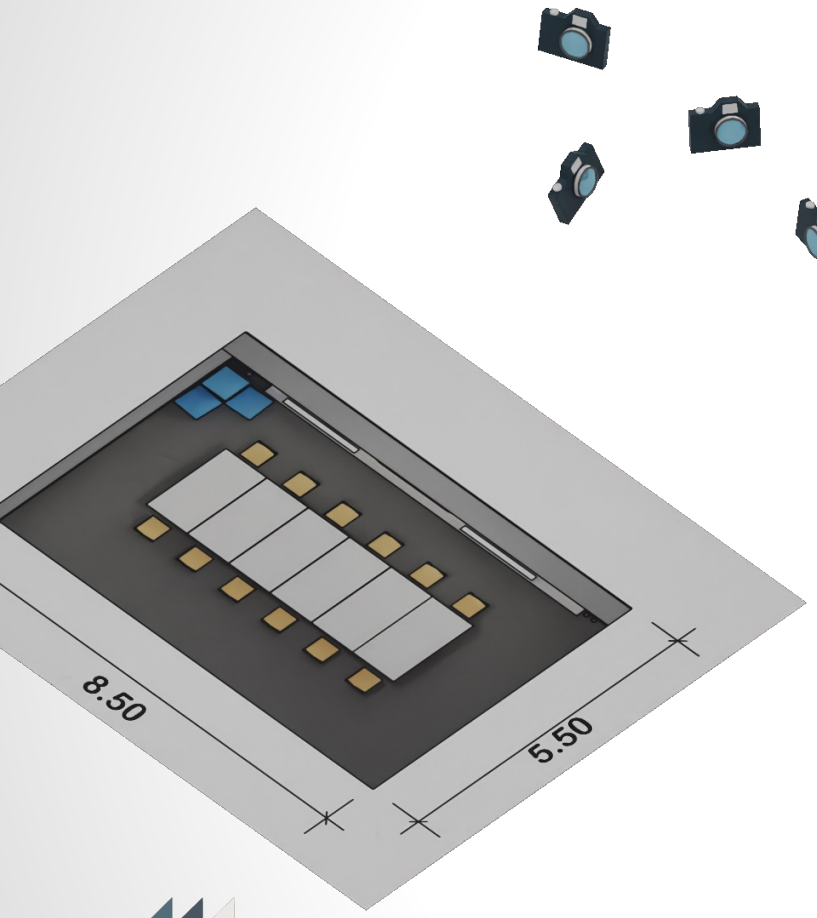
- Spatial queries involve querying objects based on distances, metrics, and spatial relationships within a 3D space.

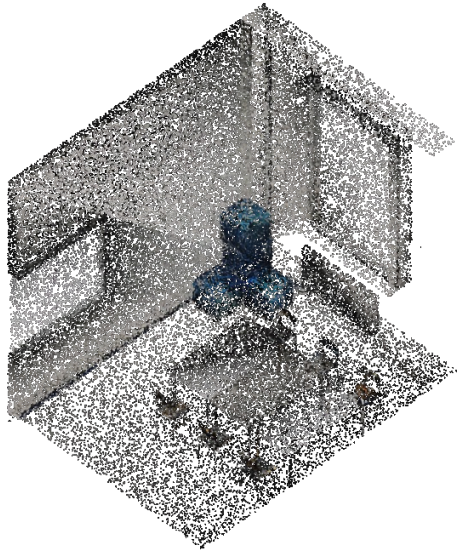
Not a Focus Today:

- Spatial queries that utilise natural language concepts like "above," "beside," "left," or "right."



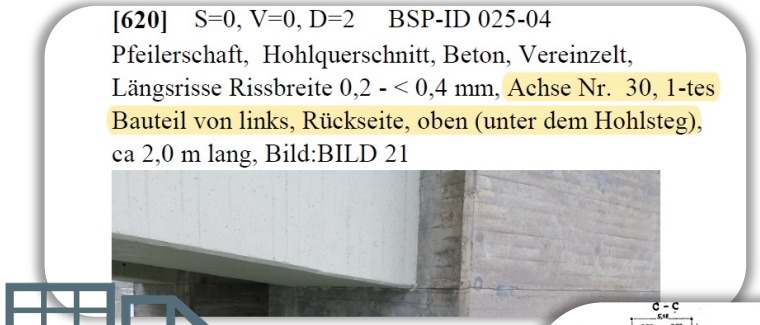
Container Environments

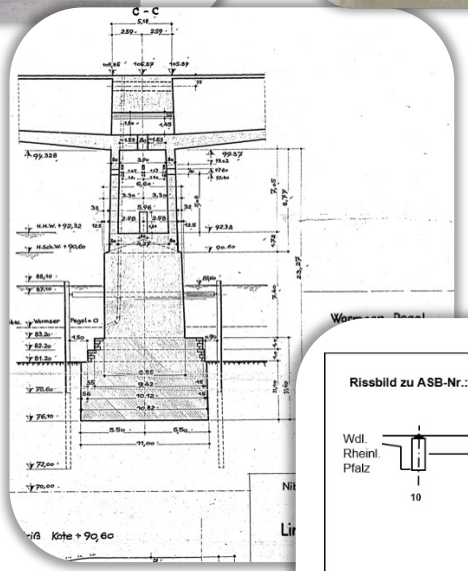


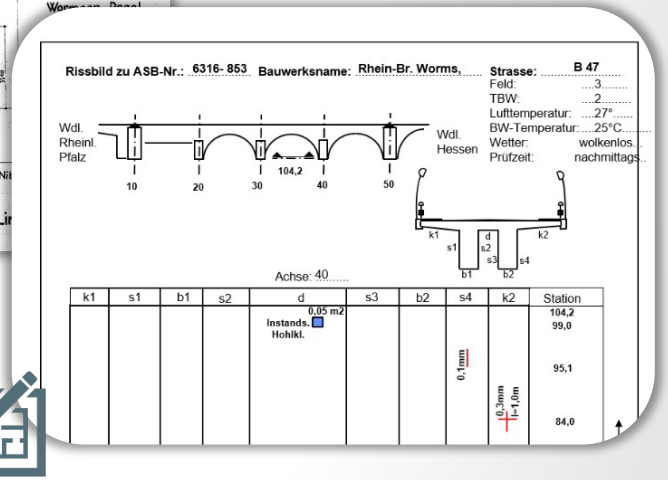


Raumlink

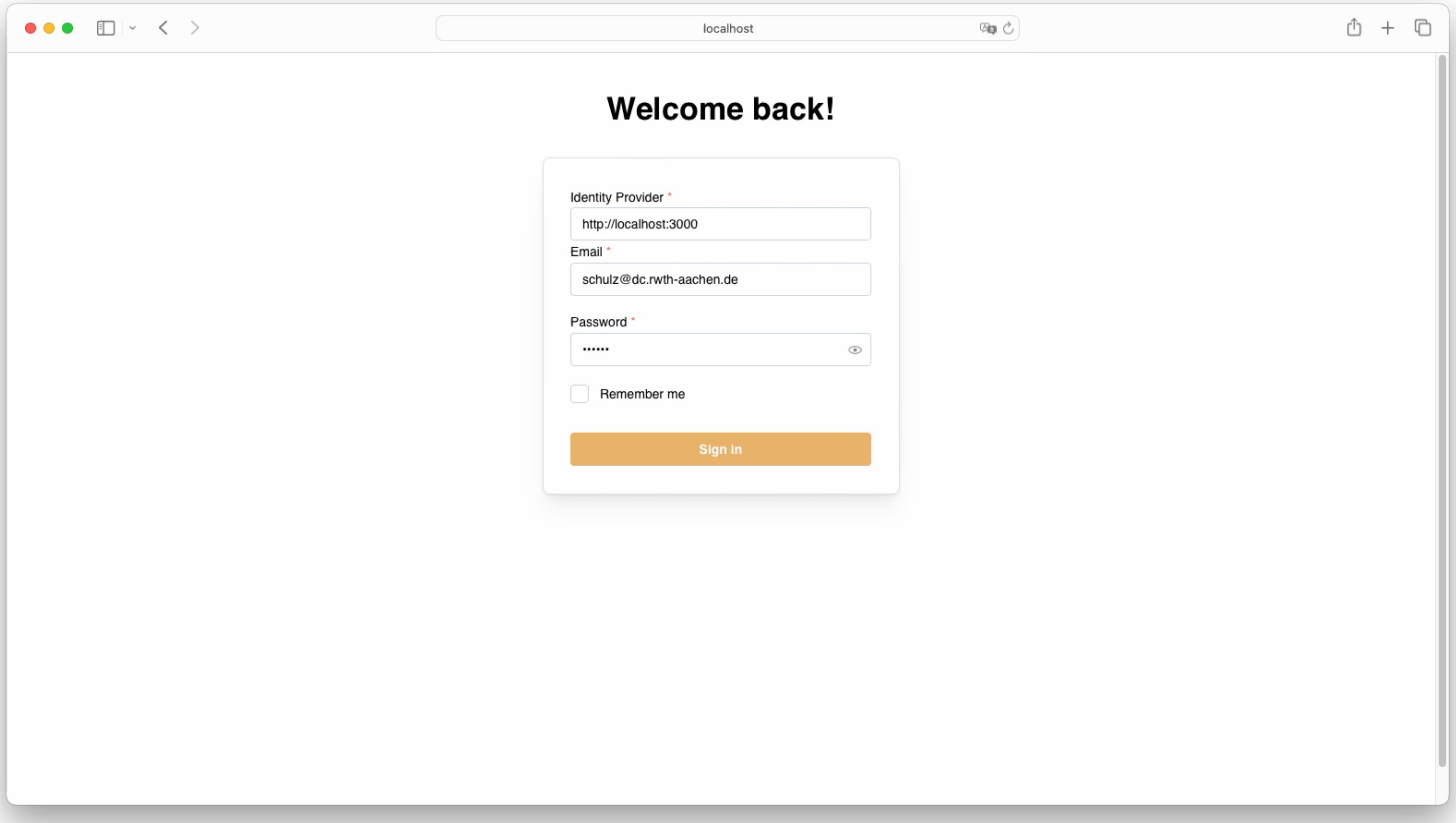
- (Spatial) Knowledge in existing data sources only implicit, not machine-readable
- Spatio-temporal classification only possible for individual experts → not scalable, knowledge is lost
- No cross-sectional views/analyses possible
- Superposition of heterogeneous information with existing methods, models and tools not possible.





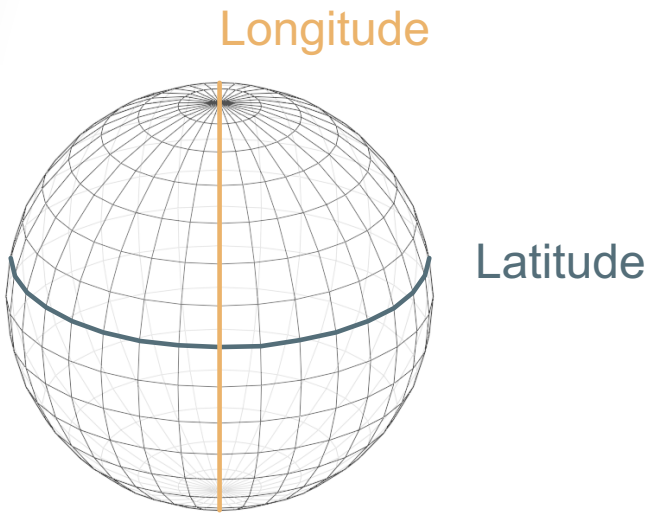


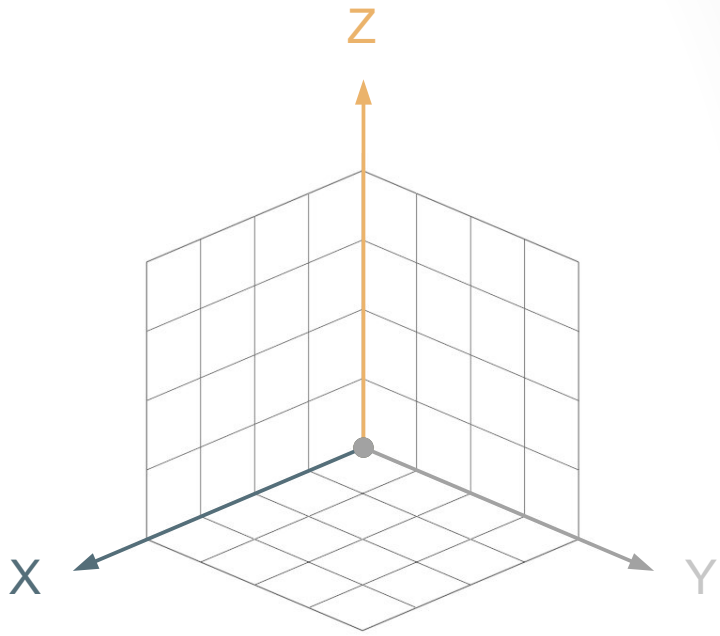
Use Case: Spatial CDE



Basics

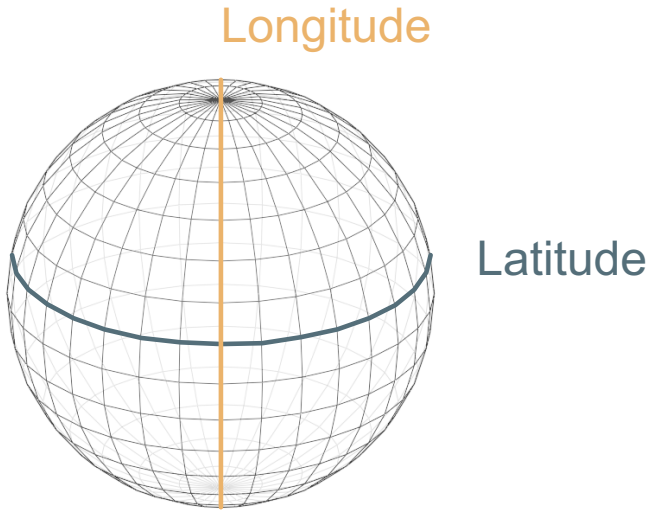
Types of Space





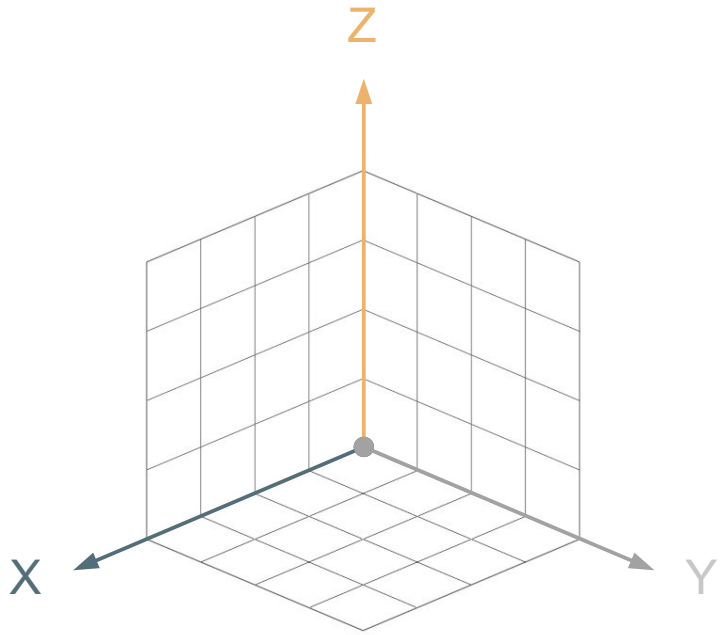
Geographic Coordinate System

- Earth is simplified by a datum, an ellipsoid that approximates the irregular shape of the Earth.
- Multiple datums exist, each optimized for accuracy in specific regions.
- Longitude and latitude lines divide the Earth into a grid system.
- They can be used to describe locations on the Earth's surface.
- Coordinates are usually given in degrees of latitude and longitude.
- The z-coordinate (height) is less commonly used but important in applications like aviation and surveying.



Cartesian Coordinate System

- Uses a grid of perpendicular lines
- Consists of two or three axis with an origin at $\begin{pmatrix}0\\0\end{pmatrix}$ or $\begin{pmatrix}0\\0\\0\end{pmatrix}$
- Axis are named, and coordinates described using $\begin{pmatrix}X\\Y\\Z\end{pmatrix}$ vectors
- Units are usually descibed in length
- Commonly used in computer graphics



Serialisation of Coordinates in Linked Data

Geometry primitives (2D)

Type	Examples	
Point		`POINT (30 10)`
LineString		`LINESTRING (30 10, 10 30, 40 40)`
Polygon		`POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))`
		`POLYGON ((35 10, 45 45, 15 40, 10 20, 35 10), (20 30, 35 35, 30 20, 20 30))`

Source: Well-known text representation of geometry, Wikipedia

* POINTZ (30 10 20)

gom:has TransformationMatrix → "[m_{11},m_{12},m_{13},m_{14}, m_{21},m_{22},m_{23},m_{24}, m_{31},m_{32},m_{33},m_{34}, 0,0,0,1]" ^^gom:rowMajorArray

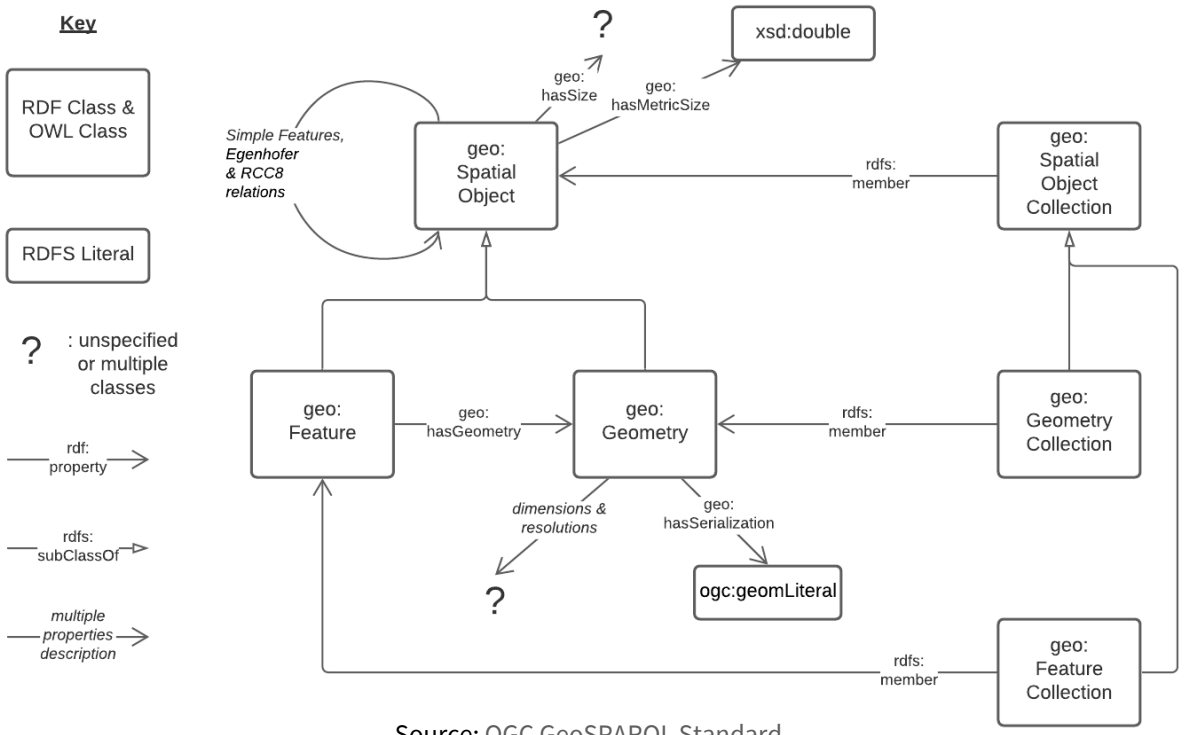
Source: Wagner & Bonduel, 2023

```
top:1edf15de-85be-5120-9ac7-187656f4f0f7 a bot:Space ;
    rdfs:label "1edf15de-85be-5120-9ac7-187656f4f0f7" ;
    top:hasVolume "18.0"^^xsd:float ;
    top:hasX "6.5"^^xsd:float ;
    top:hasY "5.0"^^xsd:float ;
    top:hasZ "4.5"^^xsd:float ;
    [...]
```

Source: topologicpy, GitHub

GeoSPARQL

GeoSPARQL Overview



Source: OGC GeoSPARQL Standard

GeoSPARQL Feature

```
geo:Feature
    a rdfs:Class, owl:Class ;
    rdfs:isDefinedBy geo: ;
    skos:prefLabel "Feature"@en ;
    rdfs:subClassOf geo:SpatialObject ;
    owl:disjointWith geo:Geometry ;
    skos:definition "A discrete spatial phenomenon in a universe of discourse."@en ;
    skos:note "A Feature represents a uniquely identifiable phenomenon, for example
              a river or an apple. While such phenomena (and therefore the Features
              used to represent them) are bounded, their boundaries may be crisp
              (e.g., the declared boundaries of a state), vague (e.g., the
              delineation of a valley versus its neighboring mountains), and change
              with time (e.g., a storm front). While discrete in nature, Features
              may be created from continuous observations, such as an isochrone
              that determines the region that can be reached by ambulance within
              5 minutes."@en ;
.
```

Source: OGC GeoSPARQL Standard

GeoSPARQL Geometry

```
geo:Geometry
    a rdfs:Class, owl:Class ;
    rdfs:isDefinedBy geo: ;
    skos:prefLabel "Geometry"@en ;
    rdfs:subClassOf geo:SpatialObject ;
    owl:disjointWith geo:Feature;
    skos:definition "A coherent set of direct positions in space. The positions
                    are held within a Spatial Reference System (SRS)."@en ;
    skos:note "Geometry can be used as a representation of the shape, extent or
              location of a Feature and may exist as a self-contained entity."@en ;
.
```

Source: OGC GeoSPARQL Standard

GeoSPARQL Functions

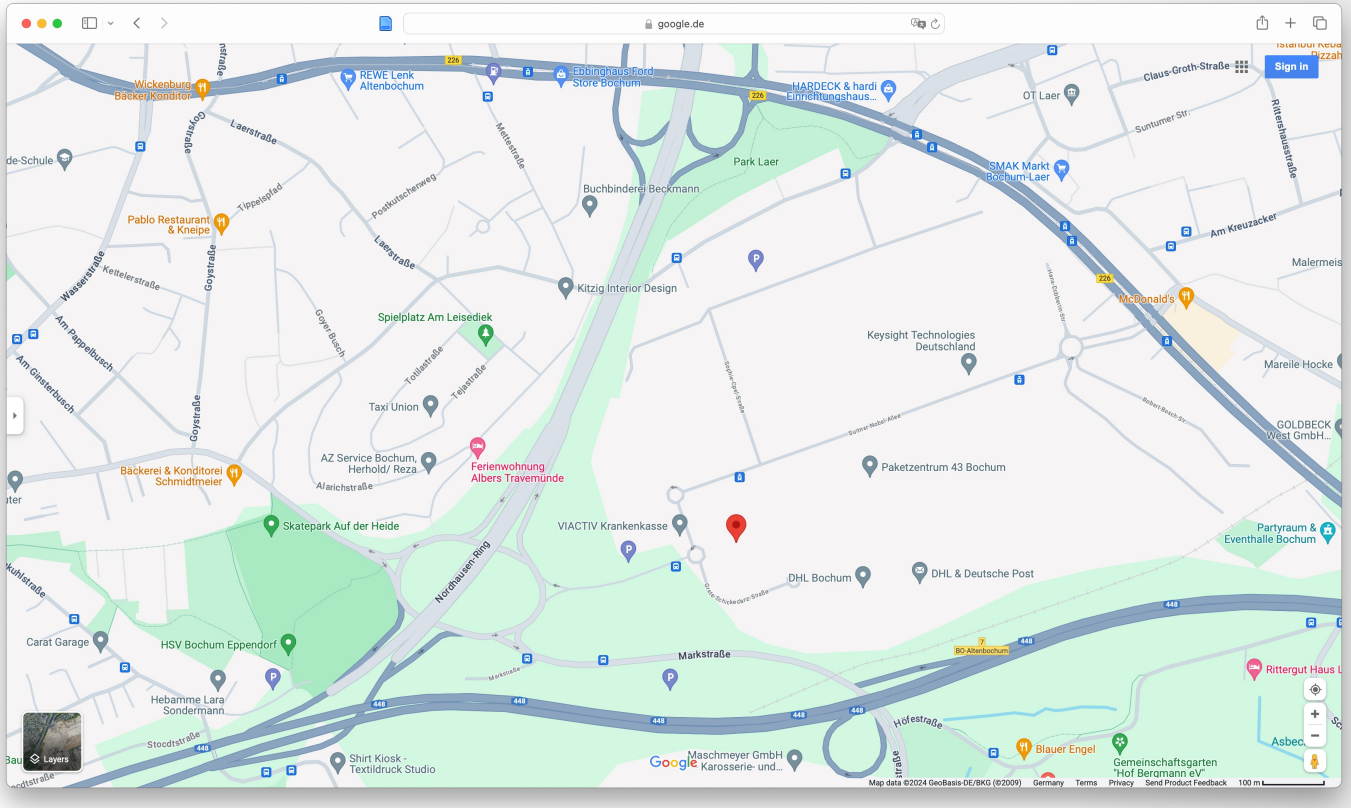
Non-Topological Function:
- Distance
- Union
- Difference
- Etc…

Topological Function:
- Equal
- Disjoint
- Contains
- Etc…

Default CRS is http://www.opengis.net/def/crs/OGC/1.3/CRS84)

GeoSPARQL Example Use Case

- Restaurants in the vicinity of RUB Makerspace



GeoSPARQL Graph

- Graph of "RUB-Makerspace" and five restaurant locations.

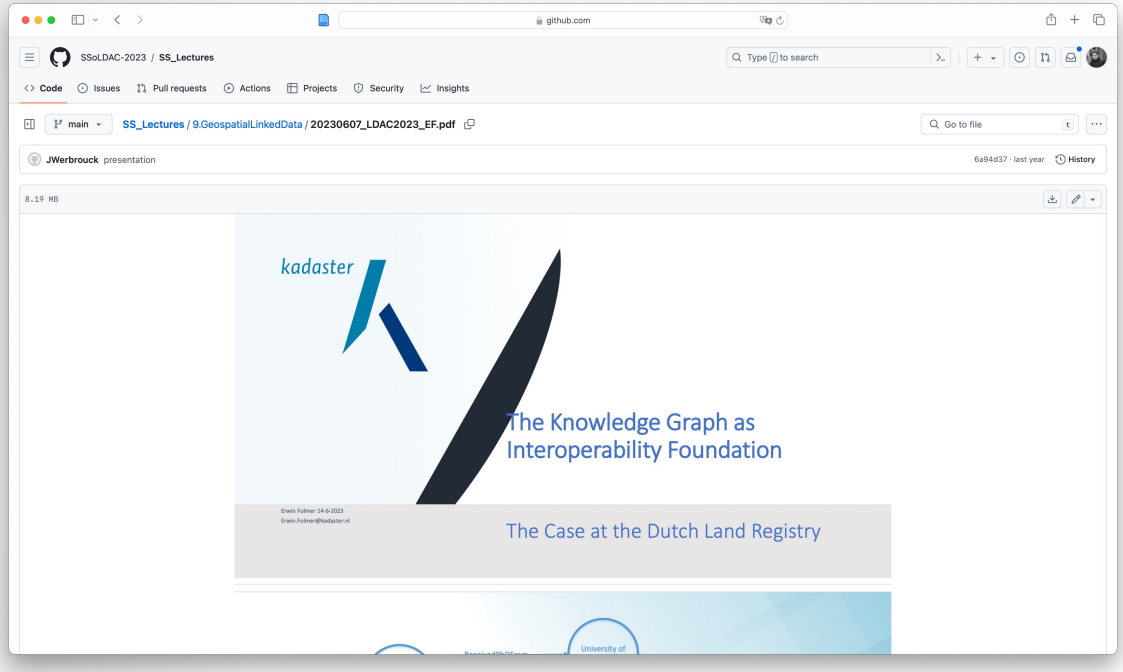
```
@prefix ex: <http://example.org/> .
@prefix geo: <http://www.opengis.net/ont/geosparql#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ex:Makerspace a geo:Feature ;
    ex:name "RUB-Makerspace" ;
    geo:hasGeometry ex:MakerspaceGeom .

ex:MakerspaceGeom a geo:Geometry ;
        geo:asWKT "POINT(7.259804 51.465457)"^^geo:wktLiteral .

ex:Schmidtmeier a ex:Restaurant ;
        ex:name "Schmidtmeier" ;
        geo:hasGeometry ex:Restaurant1Geom .

[…] Link to GitHub Repository
```

GeoSPARQL Query

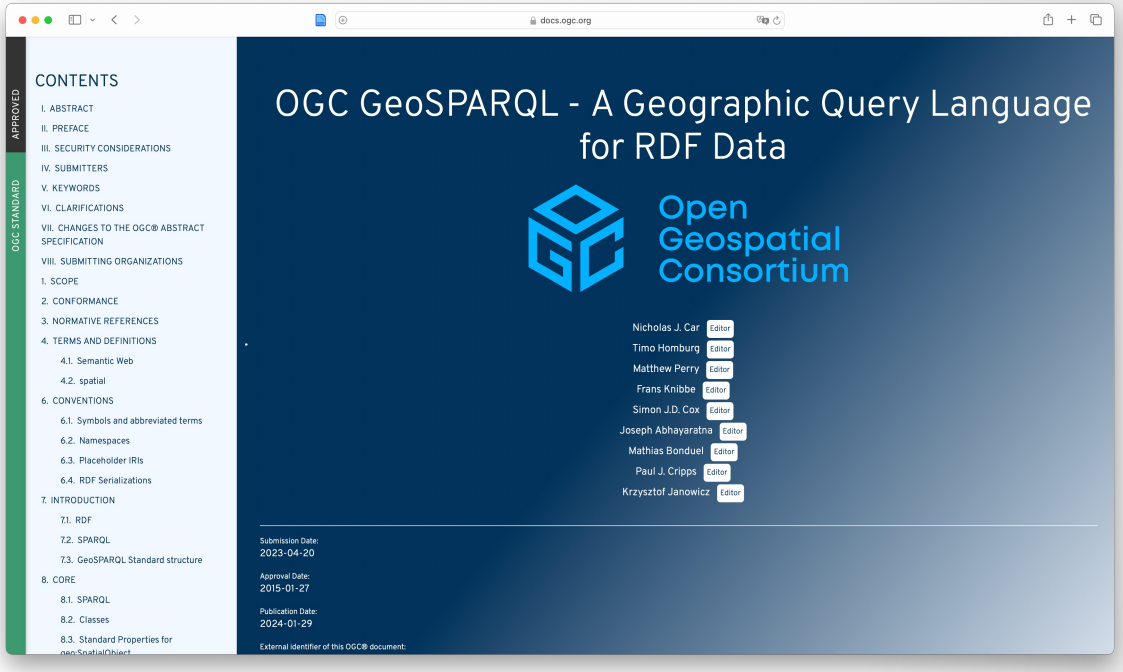
- Query to find distances between "RUB-Makerspace" and nearby restaurants.

```
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX uom: <http://www.opengis.net/def/uom/OGC/1.0/>
PREFIX ex: <http://example.org/>

SELECT ?restaurant ?name ?distance
WHERE {
    ex:Makerspace geo:hasGeometry ?MakerspaceGeom .
    ?MakerspaceGeom geo:asWKT ?makerspaceWKT .

    ?restaurant a ex:Restaurant ;
        ex:name ?name ;
        geo:hasGeometry ?restaurantGeom .
    ?restaurantGeom geo:asWKT ?wkt .

    BIND(geof:distance(?makerspaceWKT, ?wkt, uom:metre) as  ?distance)
}

Link to GitHub Repository
```

GeoSPARQL Further Reading



SSoLDAC 2023 Geospatial Linked Data

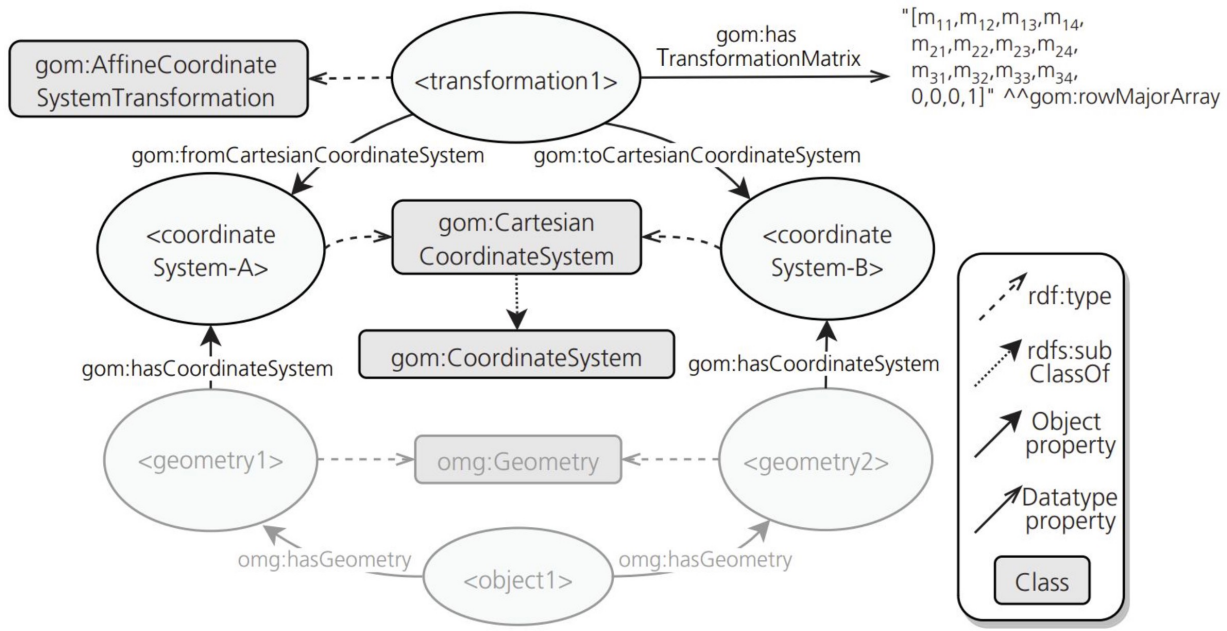


OGC GeoSPARQL Standard

Spatial Queries in SPARQL

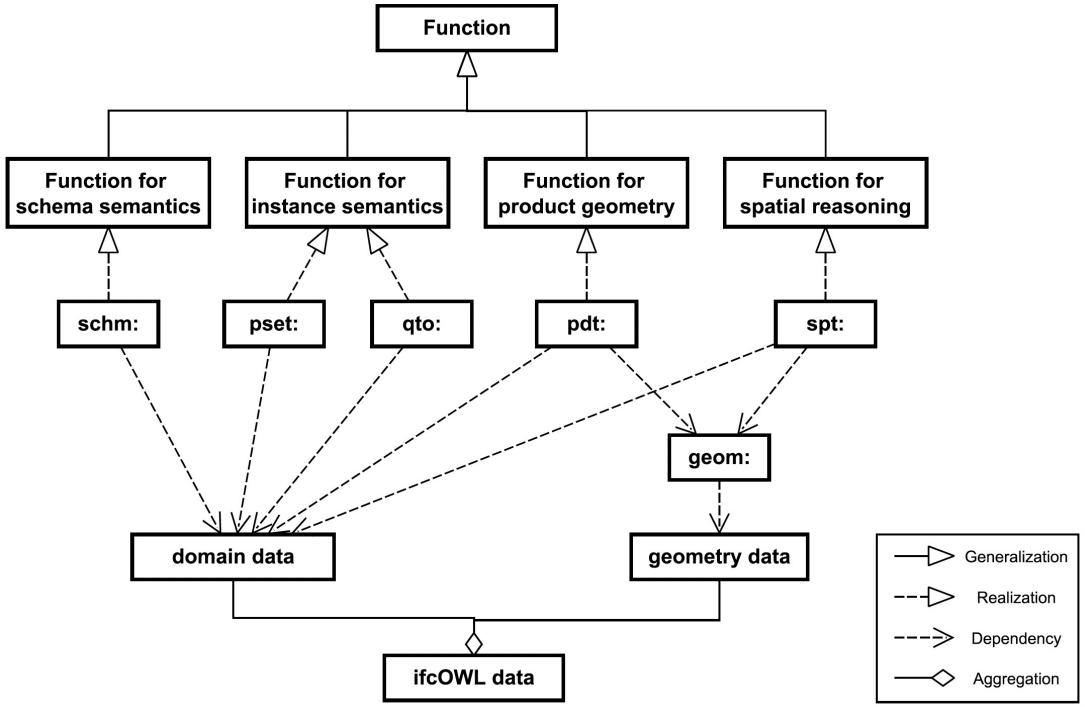
Geometry Metadata Ontology (GOM)

https://www.w3id.org/gom#



Source: Wagner & Bonduel, 2023

BIMSPARQL



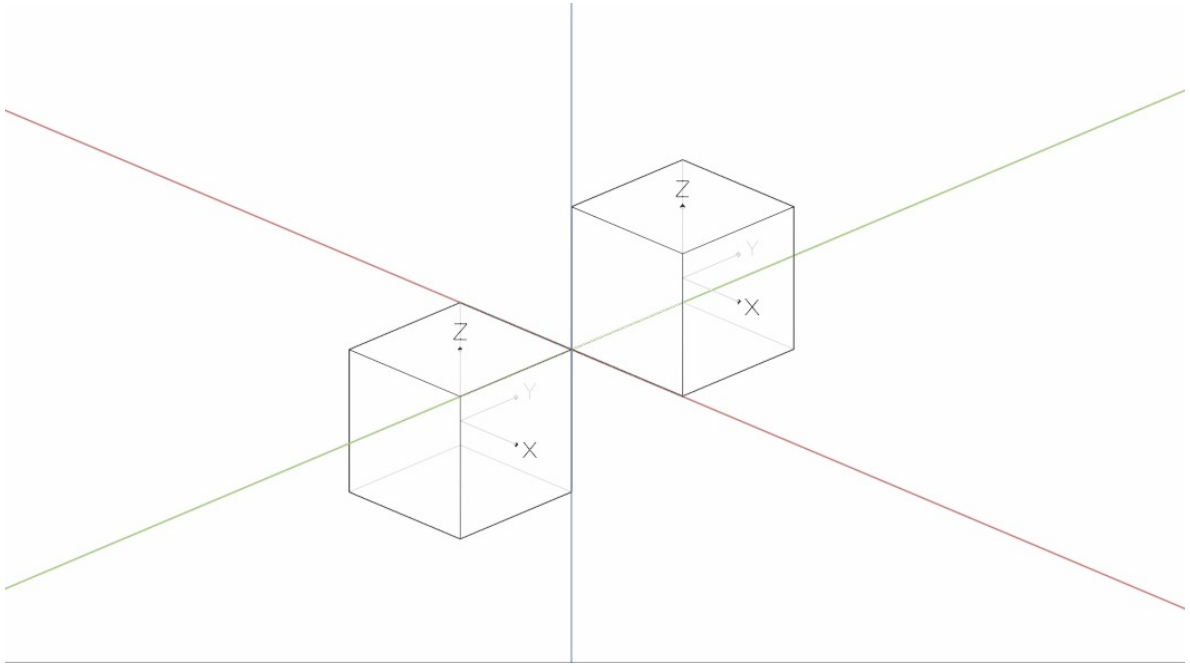

```
SELECT DISTINCT ?ceiling
WHERE {
     ?ceiling a ifc:IfcCovering .
     ?ceiling ifc:predefinedType ifc:CEILING
     .
     ?ceiling schm:isContainedIn ?storey1 .
     ?storey1 spt:hasUpperStorey ?storey2 .
     ?slab schm:isContainedIn ?storey2 .
     ?slab a ifc:IfcSlab .
     ?slab ifc:predefinedType ifc:FLOOR .
     (?slab ?ceiling) spt:distanceZ
     ?distance .
     FILTER (?distance<0.4)
}
```

Source: Zhang et al., 2018

Scenario 1: Absolute Positions

$$Cube_2 = \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix}$$

```
inst:cube_2 a bot:element ;
     spatial:hasX -1 ;
     spatial:hasY -1 ;
     spatial:hasZ -1 .
```



$$Cube_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

```
inst:cube_1 a bot:element ;
     spatial:hasX 1 ;
     spatial:hasY 1 ;
     spatial:hasZ 1 .
```

Scenario 1: Hands-on

- Write a query to select all elements in the range of 0.5 to 1.5 for X, Y and Z

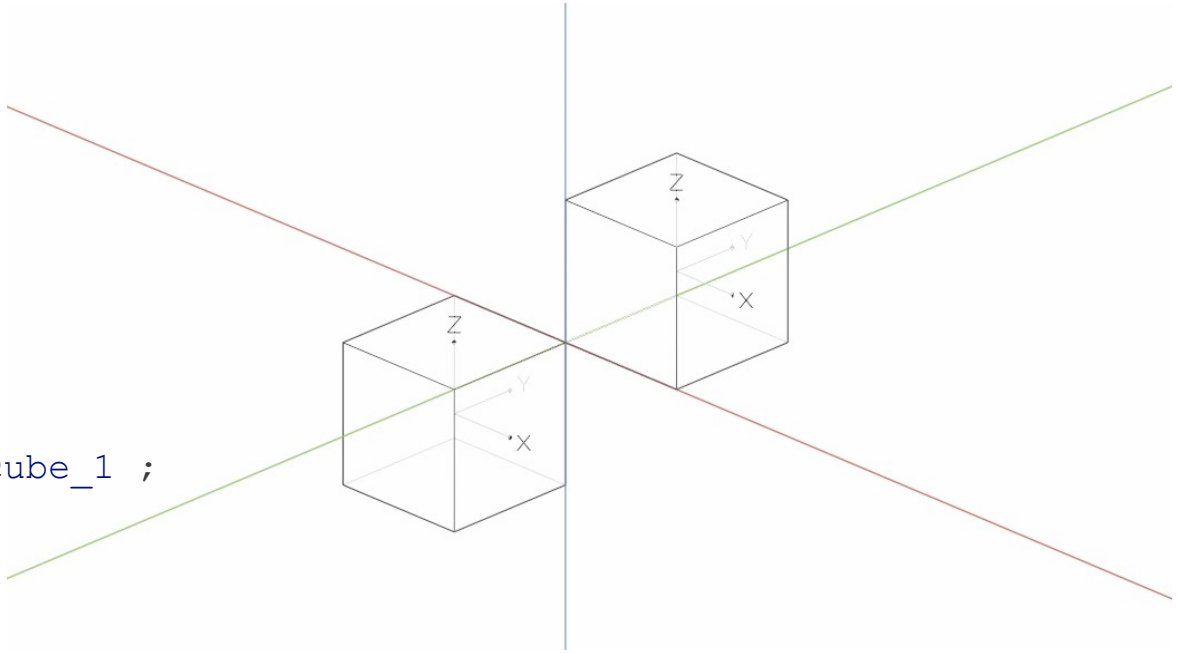
```
SELECT ?element
WHERE {
     ?element a bot:element ;
          spatial:hasX ?x ;
          spatial:hasY ?y ;
          spatial:hasZ ?z .
     FILTER (
          ?x >= 0.5 && ?x <= 1.5 &&
          ?y >= 0.5 && ?y <= 1.5 &&
          ?z >= 0.5 && ?z <= 1.5
     )
}
```

Link to GitHub Repository

Scenario 2: Relative Positions

$$Cube_{2abs} = \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix}$$

$$Cube_{2rel} = \begin{pmatrix} -2 \\ -2 \\ -2 \end{pmatrix}$$



$$Cube_{1} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

```
inst:cube_2 a bot:element ;
    spatial:hasParent inst:cube_1 ;
    spatial:hasX -2 ;
    spatial:hasY -2 ;
    spatial:hasZ -2 .
```

```
inst:cube_1 a bot:element ;
    spatial:hasX 1 ;
    spatial:hasY 1 ;
    spatial:hasZ 1 .
```

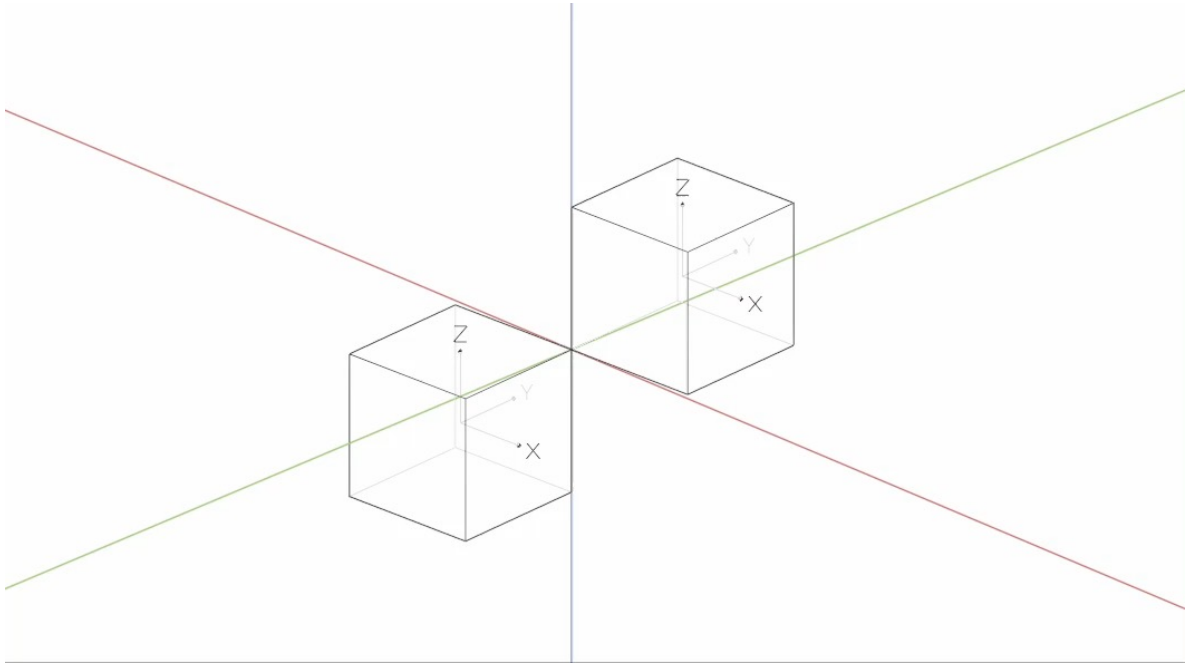
Scenario 2: Hands-on

- Write a query to select all elements in the range of 0.5 to 1.5 for X, Y and Z for their absolute positions

```
SELECT ?s (sum(?_absX) as ?absX) (sum(?_absY) as ?absY) (sum(?_absZ) as ?absZ)
FROM <http://example.org/relative-position>
WHERE {
    ?s a bot:element ;
        spatial:hasX ?relX ;
        spatial:hasY ?relY ;
        spatial:hasZ ?relZ.
    ?s spatial:hasParent* ?sp.
    ?sp spatial:hasX ?_X .
    ?sp spatial:hasY ?_Y .
    ?sp spatial:hasZ ?_Z .

    BIND(?_X as ?_absX)
    BIND(?_Y as ?_absY)
    BIND(?_Z as ?_absZ)

}
GROUP BY ?s
HAVING(
    ?absX >= -0.5 && ?absX <= 1.5 &&
    ?absY >= -0.5 && ?absY <= 1.5 &&
    ?absZ >= -0.5 && ?absZ <= 1.5
)
```

Link to GitHub Repository

Scenario 3: Relative Rotated Positions



Scenario 3: Rotations

- With rotations, calculations get more complex
- Especially when storing them as a 3x1 Vector

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) & 0 \\ 0 & -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_y(\theta) = \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_z(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R(\theta) = R_z(\theta) * (R_y(\theta) * R_x(\theta))$$

$$Cube_{trans} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad Cube_{rot} = \begin{pmatrix} 45° \\ 45° \\ 45° \end{pmatrix}$$

Scenario 3: Transformation Matrix

- We can store all transformations (scaling, rotation, translation, etc.) in one 4x4 matrix.
- Matrix multiplication can efficiently calculate the resulting transformation from relative transformations.

Rotation, Scale, Shear

$$\mathrm{M}_{\mathrm{Trans}} = \begin{pmatrix} a1 & a2 & a3 & a4 \\ a5 & a6 & a7 & a8 \\ a9 & a10 & a11 & a12 \\ a13 & a14 & a15 & a16 \end{pmatrix}$$

Translation

$$M_{abs} = M_{parent} * M_{child}$$

```
inst:cube_1 a bot:element ;
    spatial:m1 -4.371138828673793e-08 ;
    spatial:m2 -1 ;
    spatial:m3 0 ;
    spatial:m4 1 ;
    spatial:m5 1 ;
    spatial:m6 -4.371138828673793e-08 ;
    […]
    spatial:m13 0 ;
    spatial:m14 0 ;
    spatial:m15 0 ;
    spatial:m16 1 .
```

Scenario 3: Hands-on

- Write a query to select an element, it's parent and it's absolute coordinates in a transformation matrix

```
SELECT DISTINCT ?element ?parent ?absM1Test ?absM2Test ?absM3Test ?absM4Test
FROM <http://example.org/rotate-position>
WHERE {
?element a bot:element ;
     spatial:parent ?parent ;
     spatial:m1 ?m1 ;
     spatial:m2 ?m2 ;
     spatial:m3 ?m3 ;
     spatial:m4 ?m4 ;
     spatial:m5 ?m5 ;
     spatial:m6 ?m6 ;
     spatial:m7 ?m7 ;
     spatial:m8 ?m8 ;
     spatial:m9 ?m9 ;
     spatial:m10 ?m10 ;
     spatial:m11 ?m11 ;
     spatial:m12 ?m12 ;
     spatial:m13 ?m13 ;
     spatial:m14 ?m14 ;
     spatial:m15 ?m15 ;
     spatial:m16 ?m16 .

[...]
```

Link to GitHub Repository

Challenge

- In Graph “Absolute”, calculate and return the distance of inst:cube_1 to inst:cube_2
- If no parent is available, calculate the distance to the origin
- Hint 1: Calculate the distance between two points
- Hint 2: GraphDB supports Math functions

$$distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

```
PREFIX ofn: <http://www.ontotext.com/sparql/functions/>

BIND(ofn:sqrt(?sumOfSquares) AS ?distance)
```

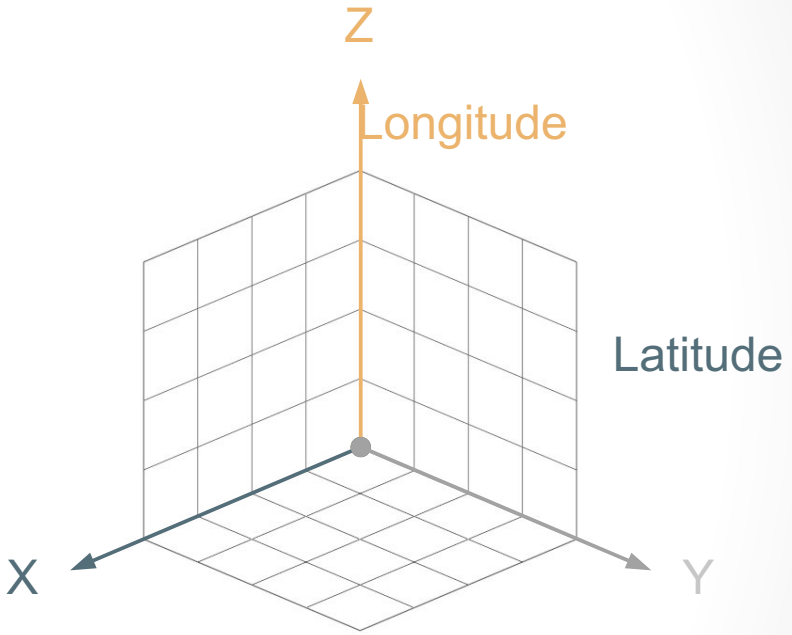
Possible Solution

- Take Math functions with a grain of salt:
  - They are often specific to the provider of the Graph Database

```
PREFIX bot: <http://example.org/bot#>
PREFIX spatial: <http://example.org/spatial#>
PREFIX inst: <http://example.org/instances#>
PREFIX ofn: <http://www.ontotext.com/sparql/functions/>

SELECT ?distance
FROM <http://example.org/absolute-position>
WHERE {

    inst:cube_1 a bot:element ;
        spatial:hasX ?x1 ;
        spatial:hasY ?y1 ;
        spatial:hasZ ?z1 .

    inst:cube_2 a bot:element ;
        spatial:hasX ?x2 ;
        spatial:hasY ?y2 ;
        spatial:hasZ ?z2 ;
    BIND((?x2 - ?x1) AS ?dx)
    BIND((?y2 - ?y1) AS ?dy)
    BIND((?z2 - ?z1) AS ?dz)

    BIND((?dx * ?dx) + (?dy * ?dy) + (?dz * ?dz) AS
    ?sumOfSquares)
    BIND(ofn:sqrt(?sumOfSquares) AS ?distance)
}
```

Spatial Queries Summarize

- Many features for geospatial SPARQL queries
- Cartesian queries are possible in SPARQL
  - But just to a specific degree
  - Mostly limited to calculating the values
  - There are no spatial functions as in GeoSPARQL
- Math functions in SPARQL are quite sparse



Thank you for your attention

This research is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project number 501812634

SPP100+

Funded by DFG Deutsche Forschungsgemeinschaft German Research Foundation

Design Computation | RWTH AACHEN UNIVERSITY