Why where matters. Geospatial SQL

Koen Decorte

CD INVEST
BI · AI · IBMi

CD-Invest - Introduction

- International IBM i ISV and IBM business partner located in Antwerp, Belgium and Madrid, Spain
- Working with IBM i and its predecessors for more than 40 years
- Applications : CDQuery, CDSecure, CDView, CDLightning, CDReport, CDAccount and CDERP
- Expertise in RPG, SQL, Python, PHP, HTML, Unity, nodejs, linux, ...
- Website : www.cdinvest.eu
- Member of CEAC since 2018
- 6 IBM Champions in the company
- What others talk about, we do.

CD-Invest - Some of our customers

CD-Invest - IBM i Client Stories

Deknudt Frames
Building the framework for a thriving e-commerce operation with IBM i

JORI
Increasing Manufacturing Efficiency During COVID-19 With IBM I and advanced 3D-configurator

Wijnen Van Maele
Tracking wine production with blockchain on IBM i

ID-Logistics
Meeting the Challenges of a Pandemic with IBM i in the Cloud

Diners Club Spain
Streamlining Customer Support with a Hybrid Cloud Application and IBM i

Optimco
Introducing AI and a new customer experience in the car insurance industry on IBM i

CD-Invest - IBM i Client Stories

Fibrocit
Providing a comfortable seat with IBM i

Cras Woodgroup
Modernizing the wood industry with IBM i

Oris
Making vacations easier with IBM i

Steffimmo
Moving to IBM i on POWER9 in the cloud for growth

Stonetales properties
Upgrading and Centralizing on the Cloud with IBM i

Winsol
Digitizing manufacturing on IBM i

CD-Invest - IBM i Client Stories

CSM
Empower more small businesses to access global trade

CSMe
Empowering real-estate since 1978

Bonehill
Adapting IBM i to the modern web

Immo Bonehill

Read more on on https://www.ibm.com/it-infrastructure/us-en/resources/power/ibm-i-customer-stories/

CD INVEST
BI · AI · IBMi

Agenda

- What is a Geospatial database
- Why does it matter
- Geospatial data types
- Geospatial SQL
- Sample cookbook recipes
- Next steps

What is a geospatial database

What is a Spatial Database?

Database that:

- Stores spatial objects
- Manipulates spatial objects just like other objects in the database

What is a Spatial Database?

Vector (streets, political/administrative boundaries, parcels)

Network (road transportation, telco, utility, energy)

Raster (elevation, and gridded data for land usage)

Real World

What is Spatial data ?

Spatial data, also known as geospatial data or geographic information, is data that identifies the geographic location of features and boundaries on Earth. We usually use spatial data to store coordinates, topology, or other data that can be mapped.

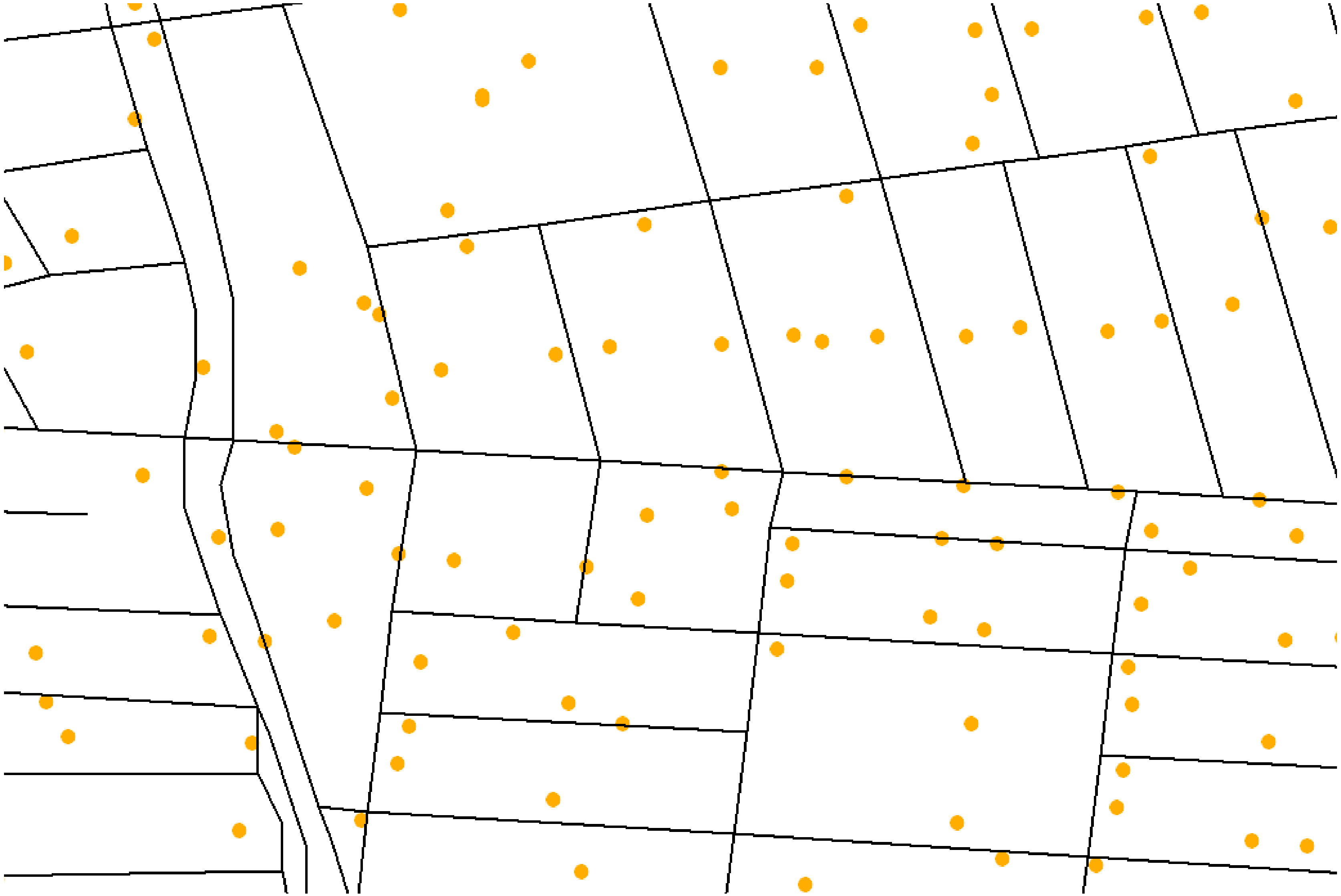
What is Spatial data ?

Data which describes either location or shape

e.g.House or Fire Hydrant location
    Roads, Rivers, Pipelines, Power lines
    Forests, Parks, Municipalities, Lakes

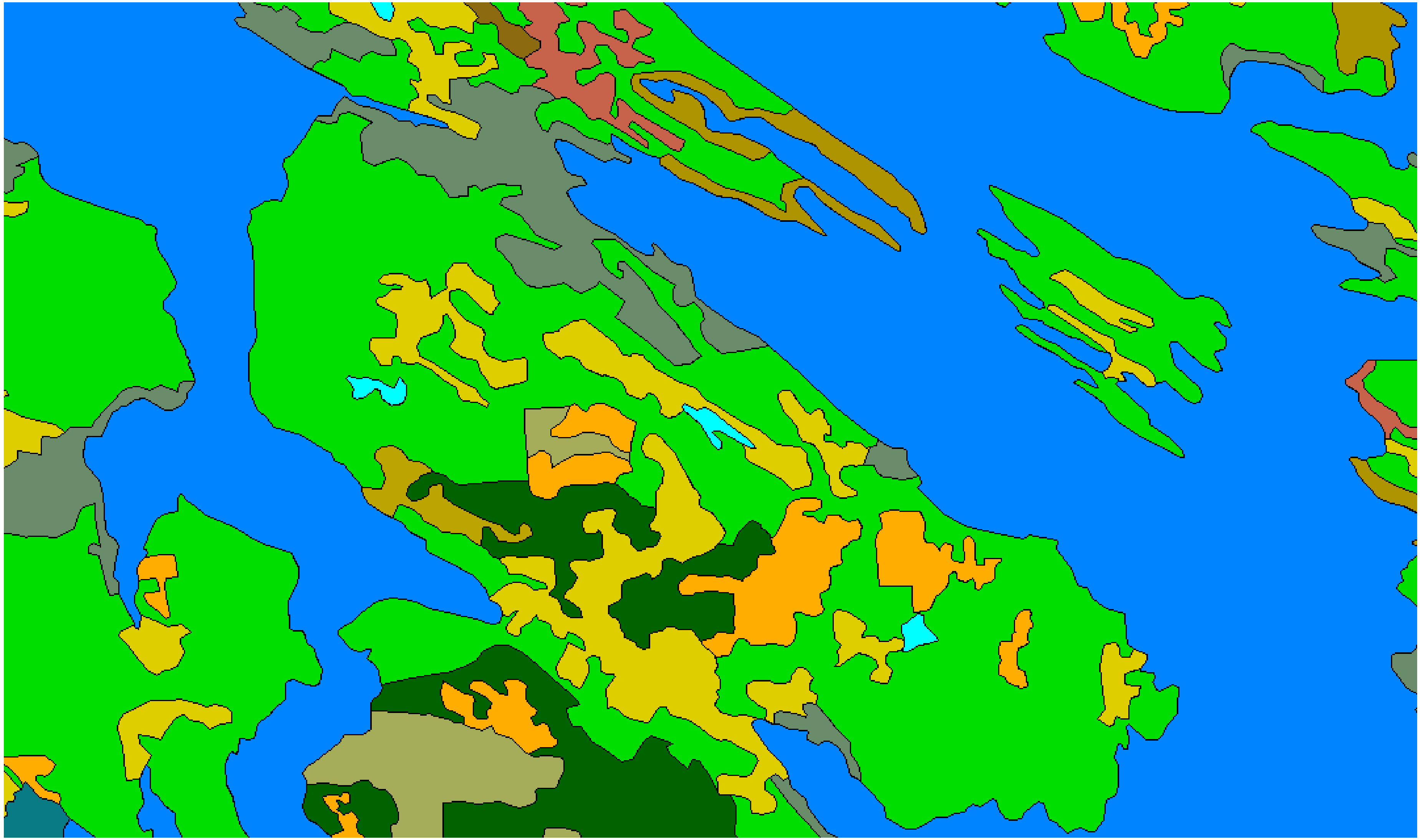
What is Spatial data ?

In the abstract, reductionist view of the computer, these entities are represented as Points, Lines, and Polygons.

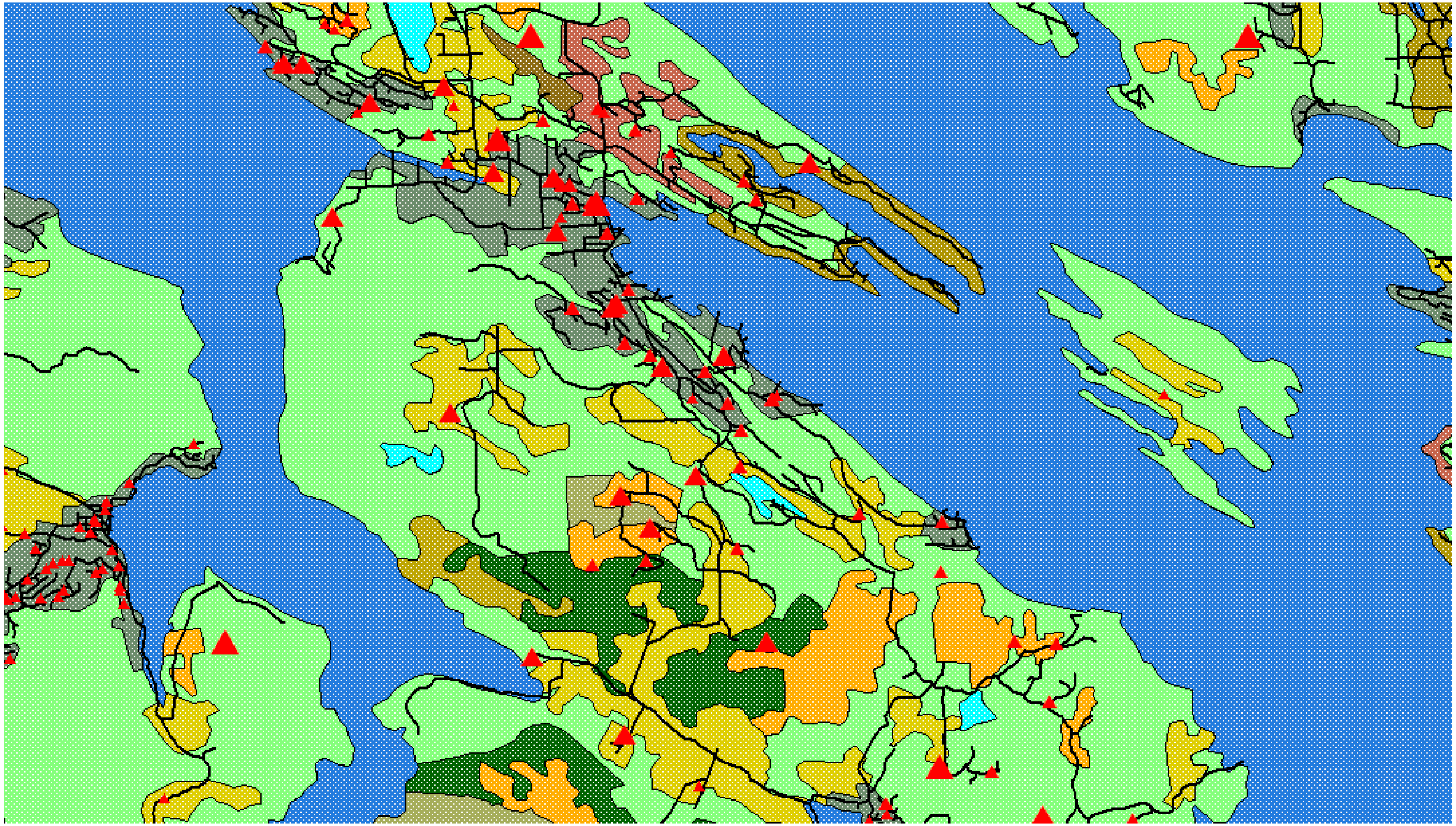


Roads are represented as Lines

Mail Boxes are represented as Points

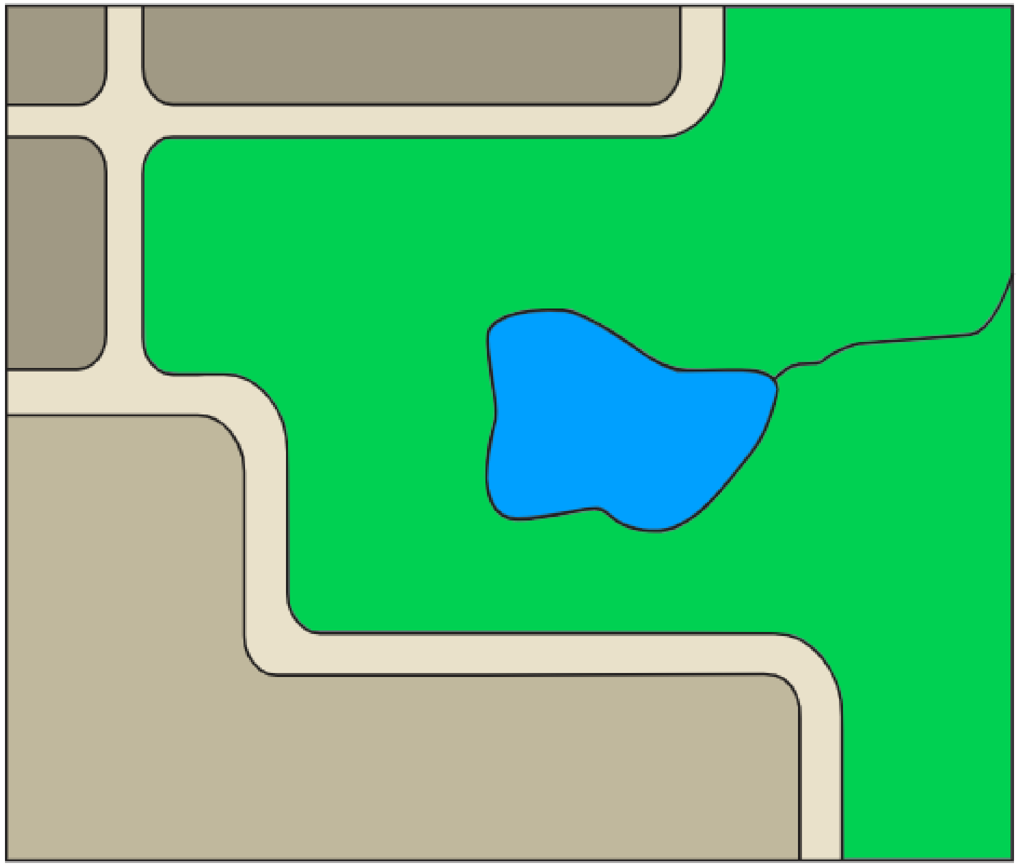


Land Use Classifications are represented as Polygons

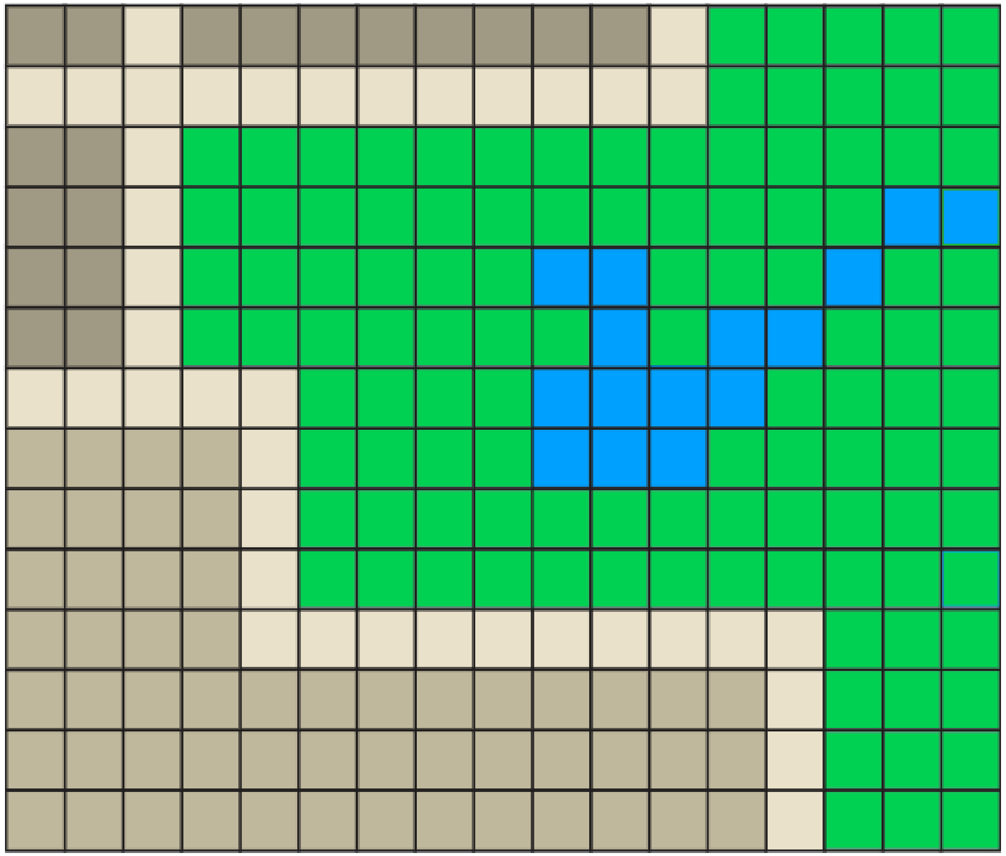


Combination of all the previous data

Vector data representation

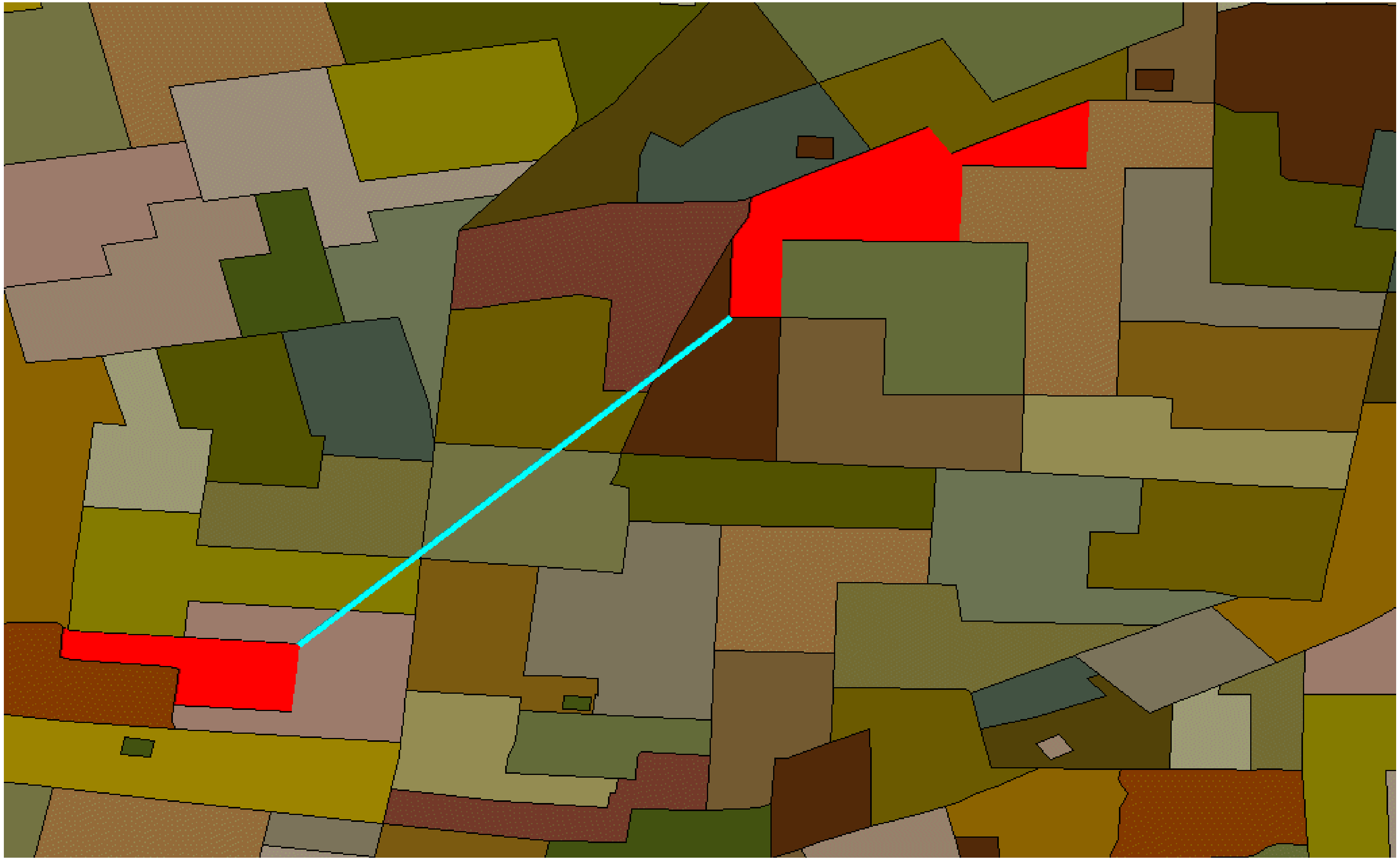


Raster data representation

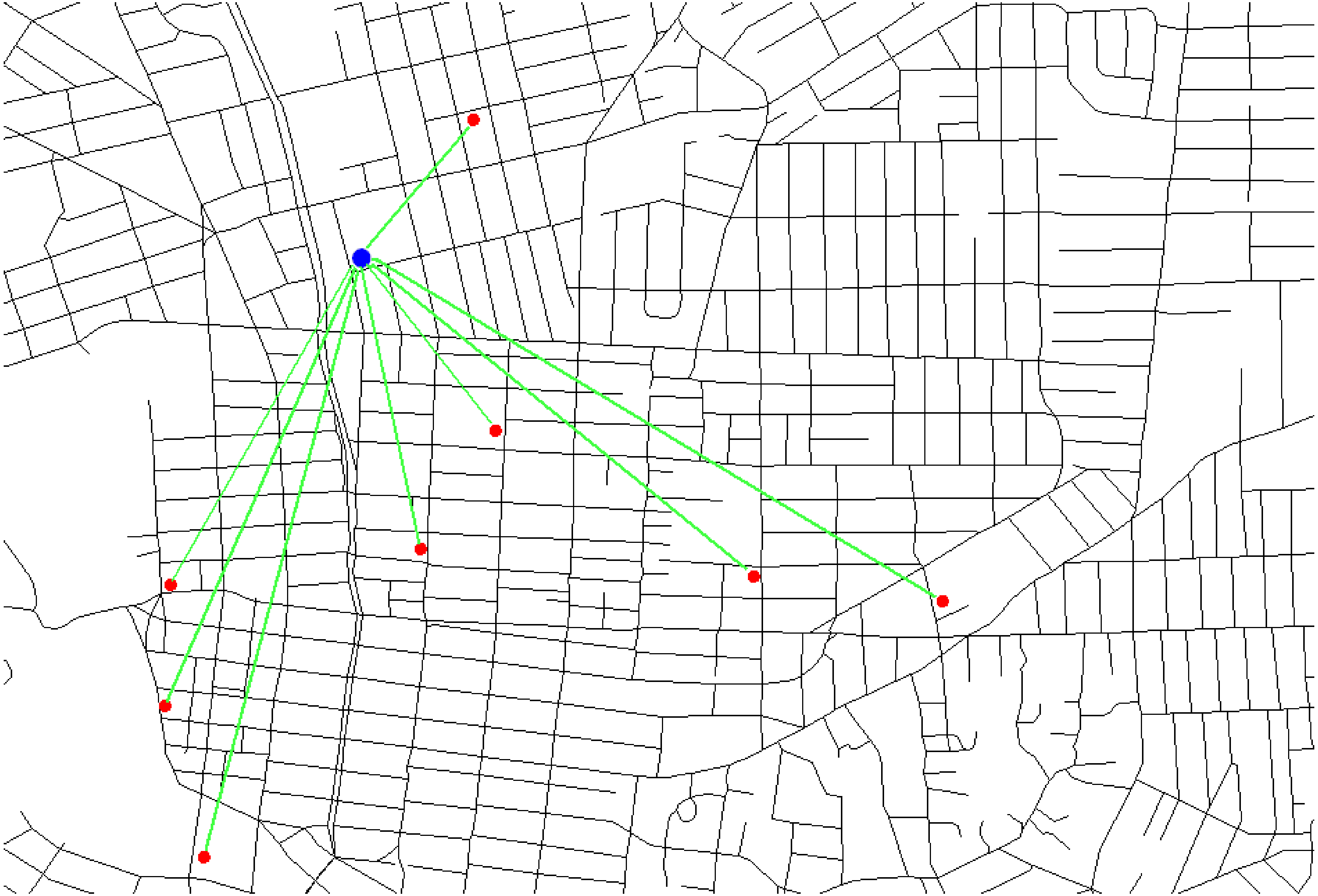


Spatial relationships

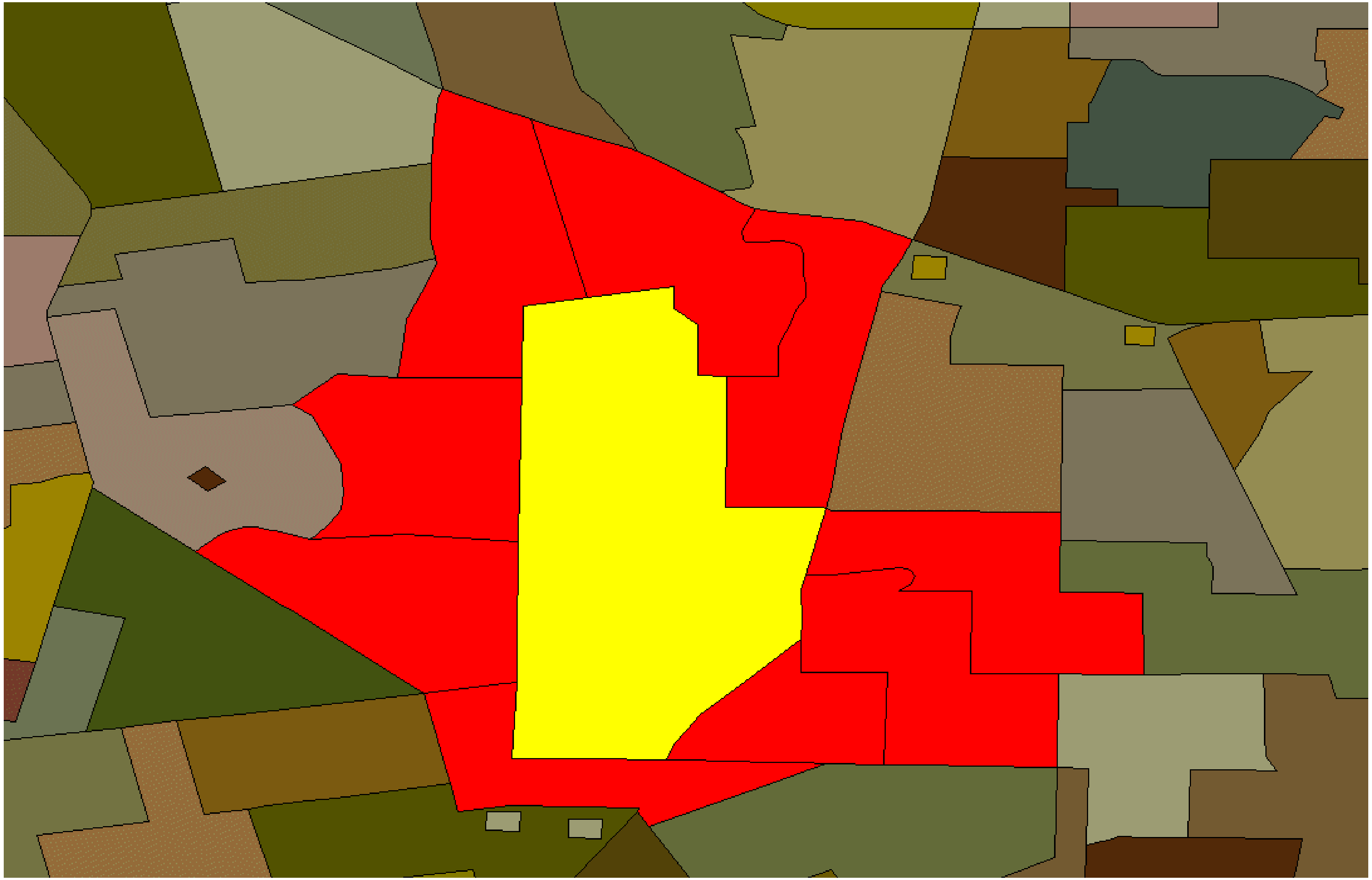
- Not just interested in location, also interested in "Relationships" between objects that are very hard to model outside the spatial domain.
- The most common relationships are
    - Proximity : distance
    - Adjacency : "touching" and "connectivity"
    - Containment : inside/overlapping



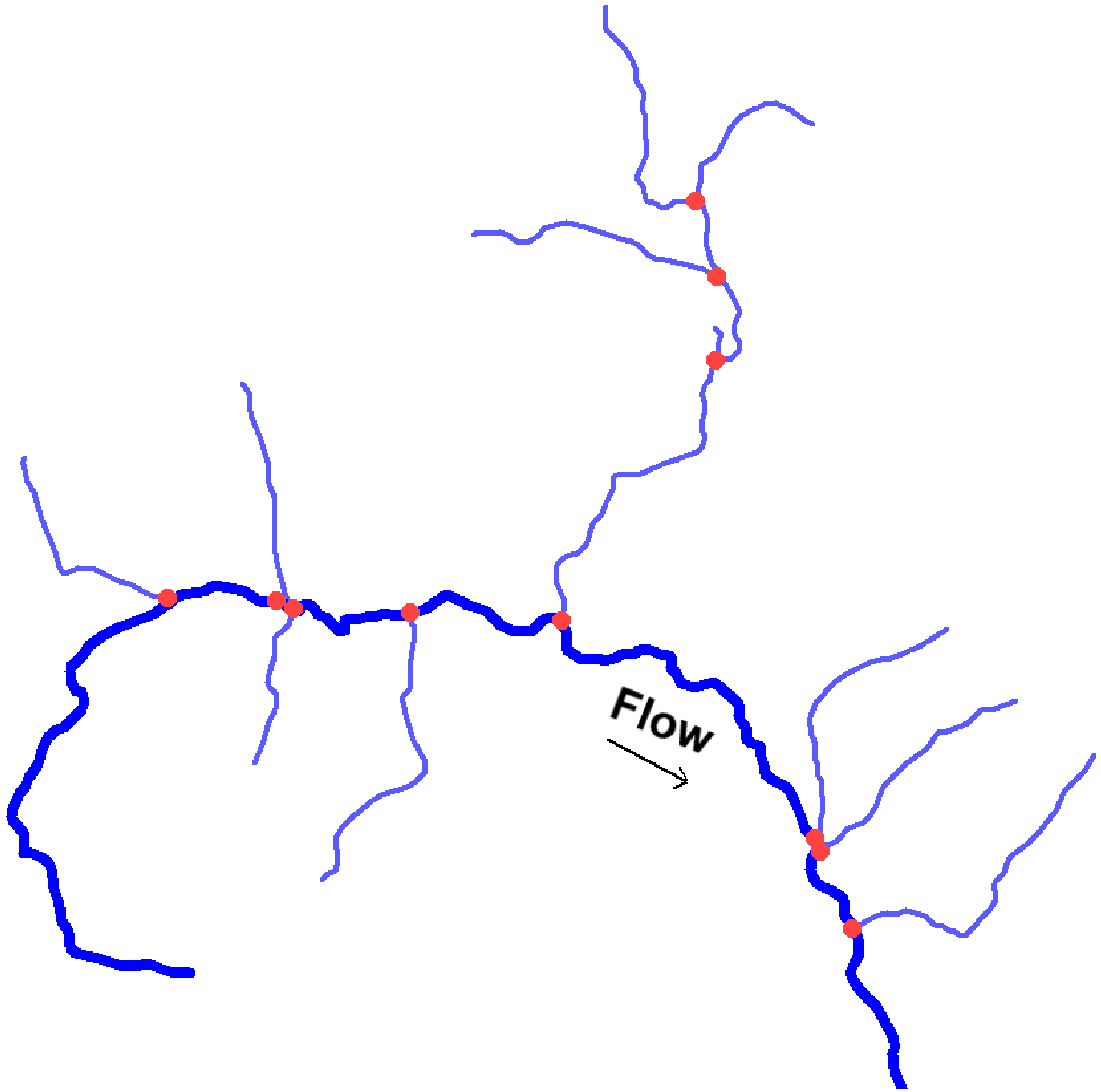
Distance between a toxic waste dump and a piece of property you were considering buying.



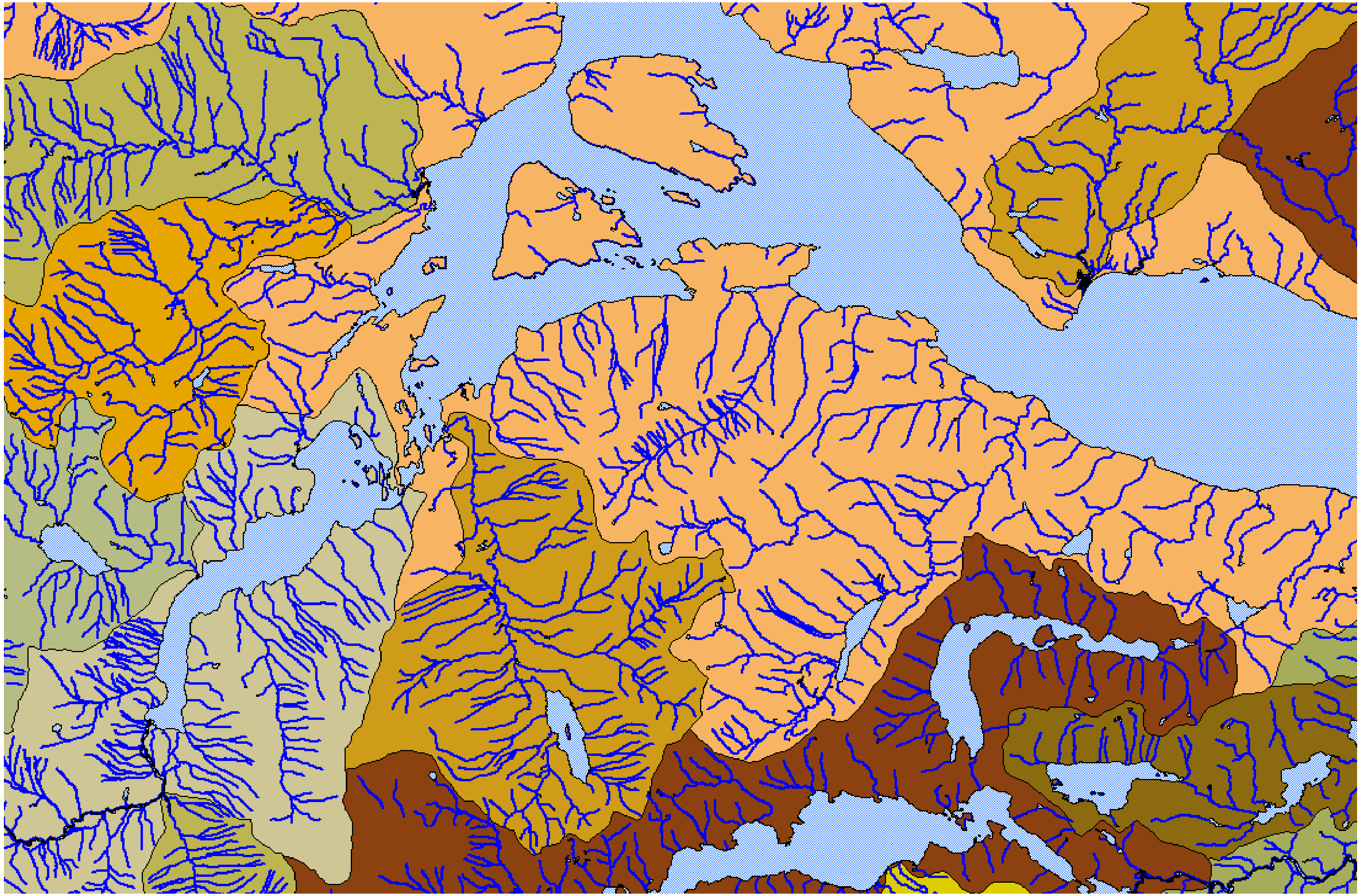
Distance to various pubs



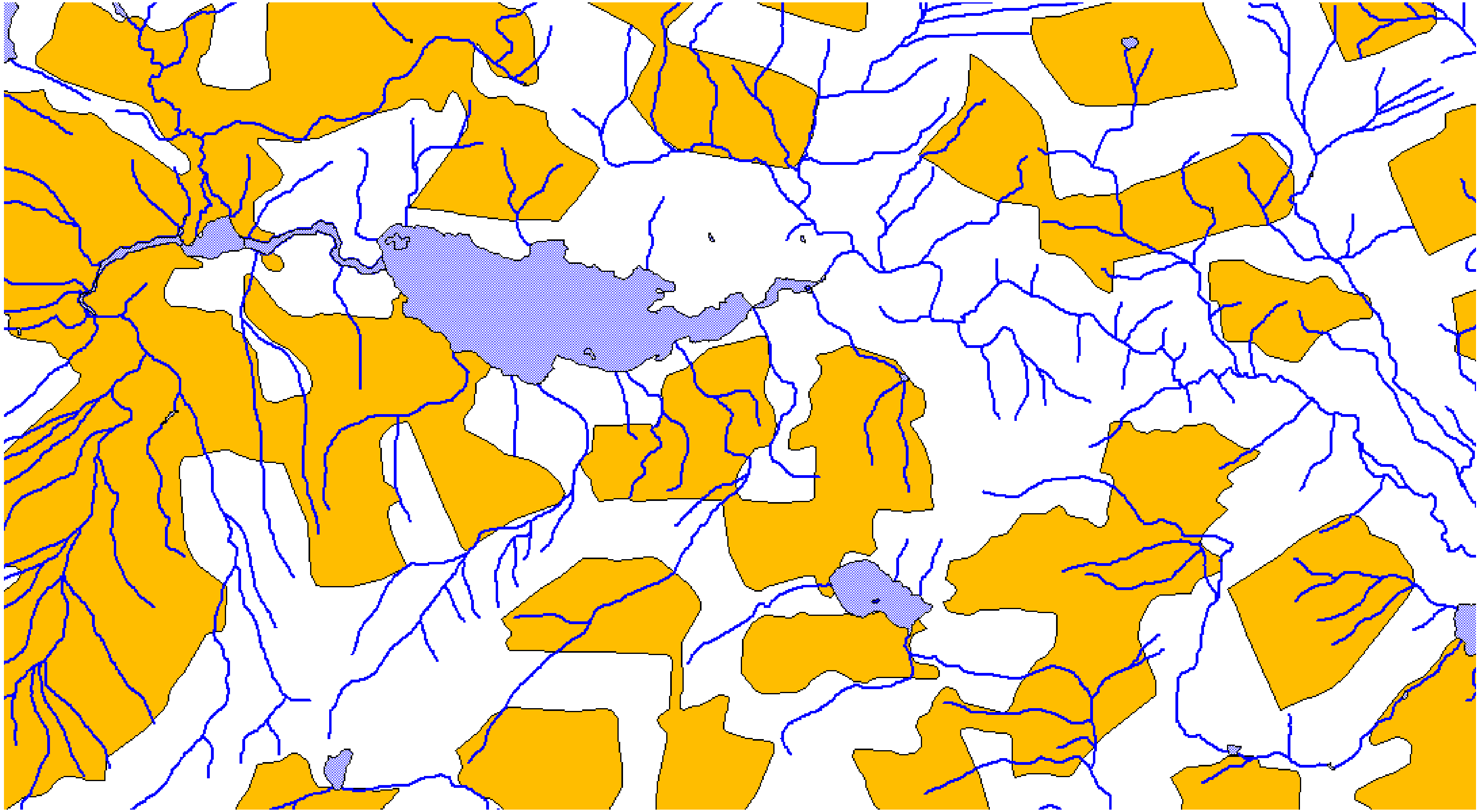
Adjacency: All the lots which share an edge



Connectivity: Tributary relationships in river networks



Containment: Rivers inside watersheds and land (islands) inside lakes



Stream side logging - adjacency and containment.

Most Organizations have Spatial Data

- Geocodable addresses
- Customer location
- Store locations
- Transportation tracking
- Statistical/Demographic
- Cartography
- Epidemiology
- Crime patterns
- Weather Information
- Land holdings
- Natural resources
- City Planning
- Environmental planning
- Information Visualization
- Hazard detection

Why put spatial data in a RDBMS?

Spatial data is usually related to other types of data. Allows one to encode more complex spatial relationships.

- Fire Hydrant: number of uses, service area, last maintenance date.
- River: flow, temperature, fish presence, chemical concentrations
- Forested Area: monetary value, types of trees, ownership

Why put spatial data in a RDBMS?

- Query and manage spatial datasets
- Manage projections and re-project data
- Store spatial data in multiple formats (WKT, GeoJSON, etc.)
- Analyze and understand many types spatial relationships
- Perform spatial clustering and perform nearest neighbor analysis
- Transform geometries using buffers, Voronoi polygons, and convex/concave hulls
- Calculate distances – straight line and using the curve of the earth
- Query and manage 3D data

Why put spatial data in a RDBMS?

- Create triggers to change and update data based on different events such as an INSERT into a specific table
- Connect to APIs and other external services
- Create user-defined functions in other languages such as Python, Javascript, and SQL
- Perform statistical analysis and machine learning using functions and tools like BigQuery ML
- Store and manage unstructured data like JSON
- Return data in many formats such as GeoJSON or Shapefiles
- Produce and generate map tiles for frontend apps

Historically?

- In early GIS implementations, spatial data and related attribute information were stored separately. The attribute information was in a database (or flat file), while the spatial information was in a separate, proprietary, GIS file structure.

  For example, municipalities often would store property line information in a GIS file and ownership information in a database.

- Spatial databases were born when people started to treat spatial information as first class database objects.

Advantages of Spatial Databases

Able to treat your spatial data like anything else in the DB

- transactions
- backups
- integrity checks
- less data redundancy
- fundamental organization and operations handled by the DB
- multi-user support
- security/access control
- locking

Advantages of Spatial Databases

Offset complicated tasks to the DB

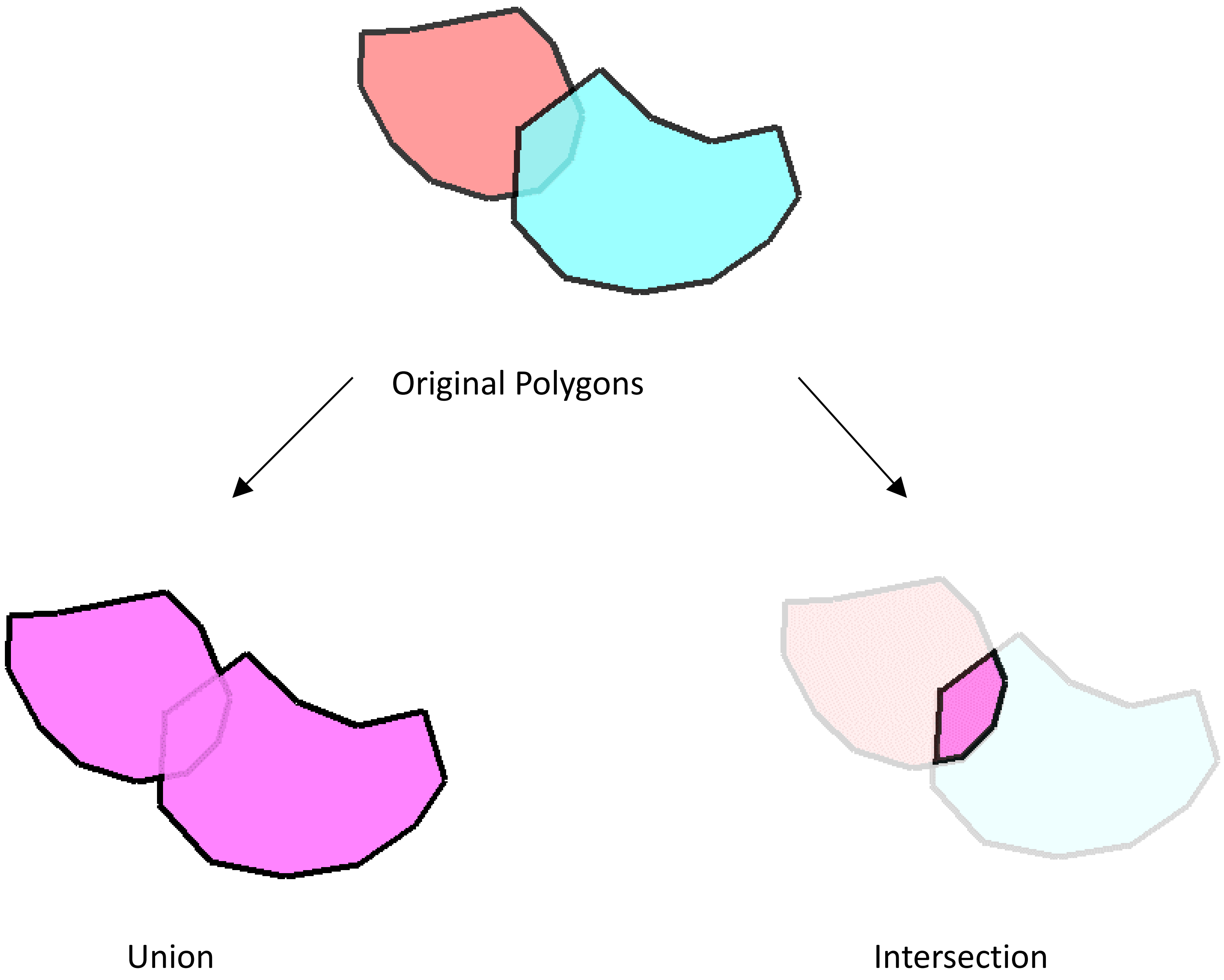
- organization and indexing done for you
- do not have to re-implement operators
- do not have to re-implement functions

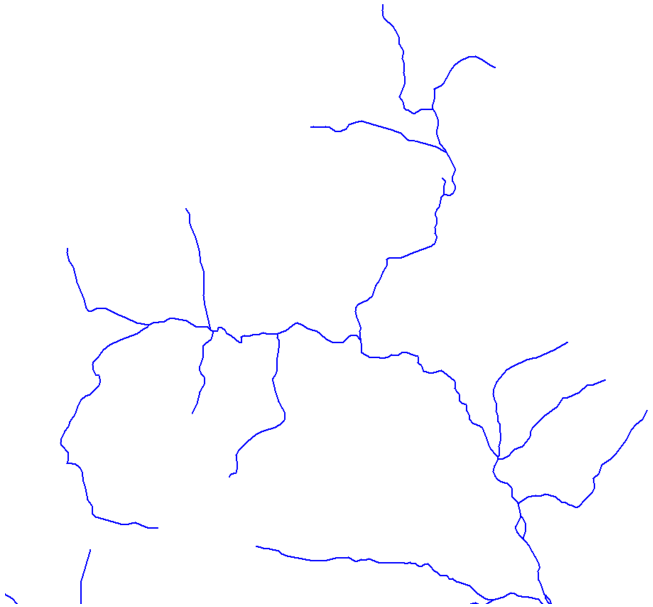
Significantly lowers the development time of client applications

Advantages of Spatial Databases

Spatial querying using SQL

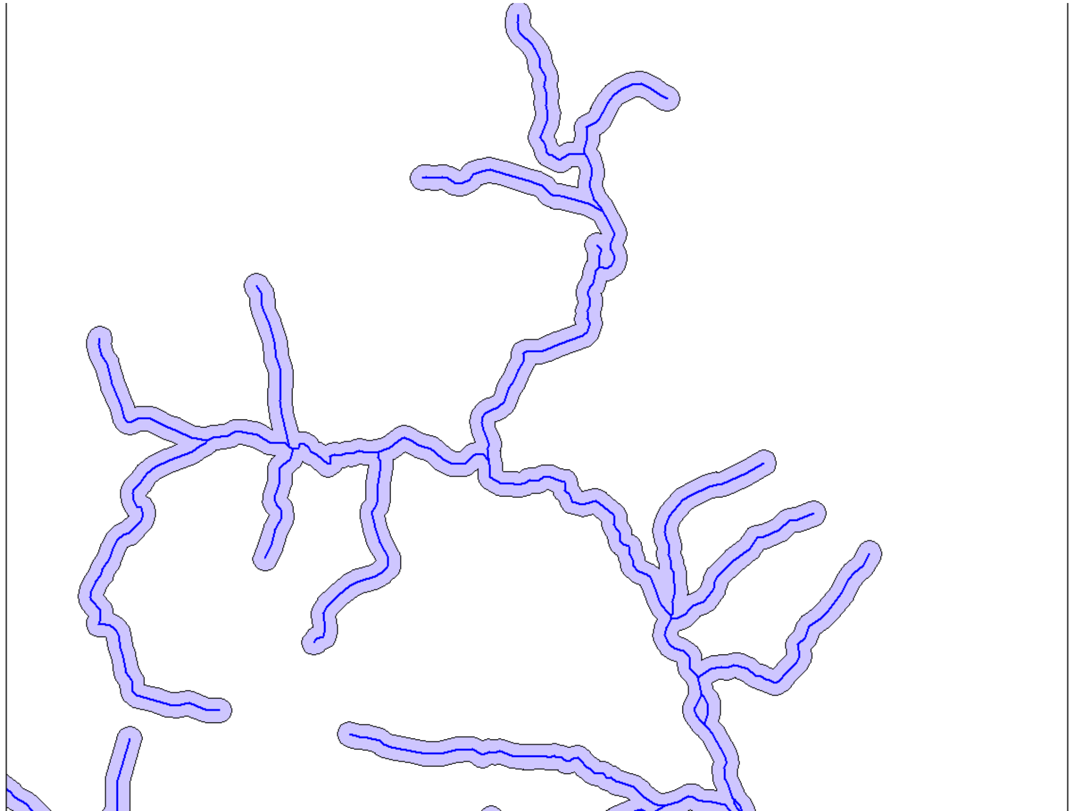
- use simple SQL expressions to determine spatial *relationships*
  - distance
  - adjacency
  - containment
- use simple SQL expressions to perform spatial *operations*
  - area
  - length
  - intersection
  - union
  - buffer


Original Polygons
Union
Intersection



Original river network

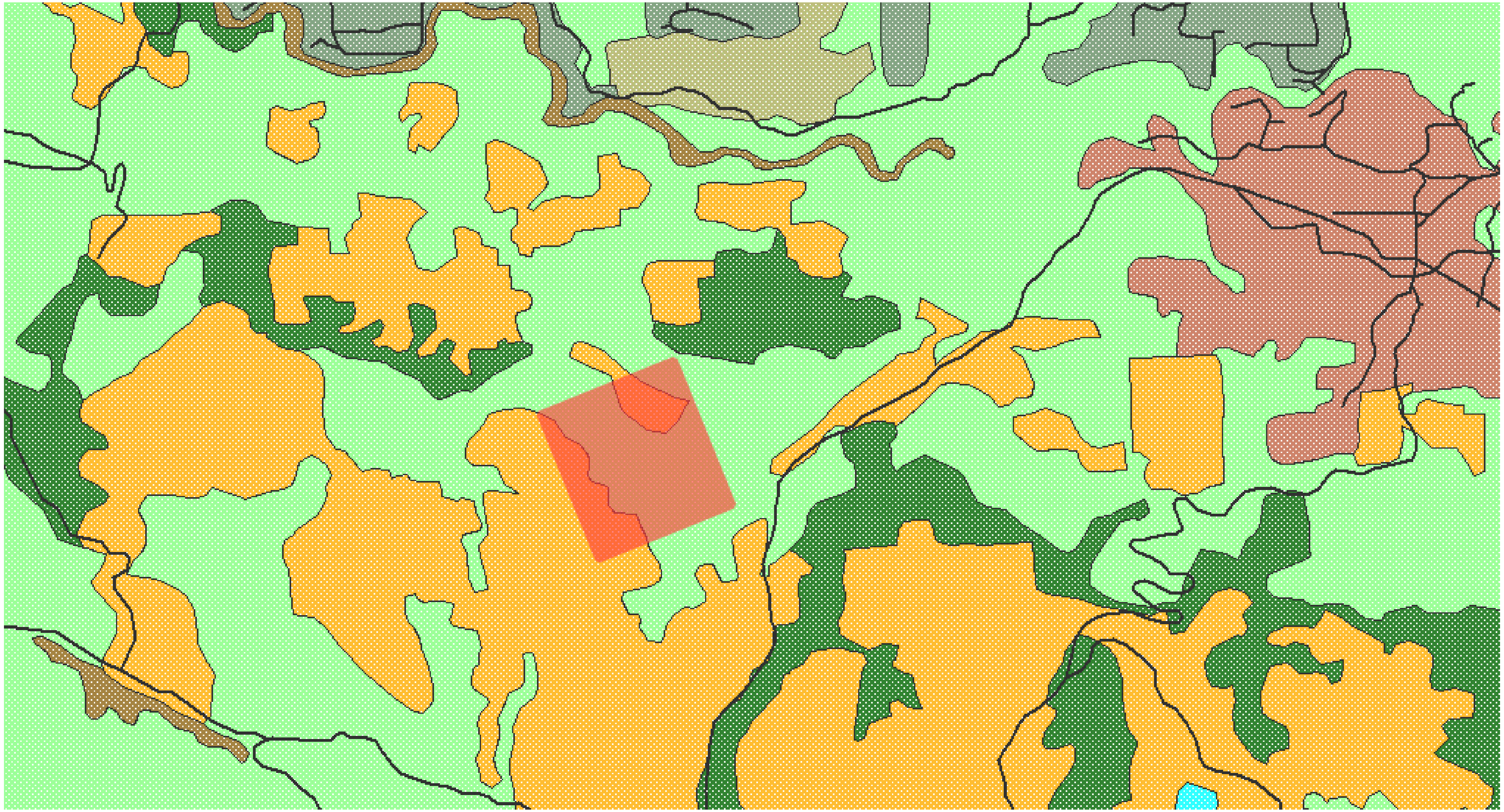
Buffered rivers



... WHERE distance(<me>,pub_loc) < 1000

SELECT distance(<me>,pub_loc)*0.01 + beer_cost ...
    ... WHERE touches(pub_loc, street)

... WHERE inside(pub_loc,city_area) and city_name = ...



Simple value of the proposed lot

Area(<my lot>) * <price per acre>
+ area(intersect(<my log>,<forested area>)) * <wood value per acre>
- distance(<my lot>, <power lines>) * <cost of power line laying>

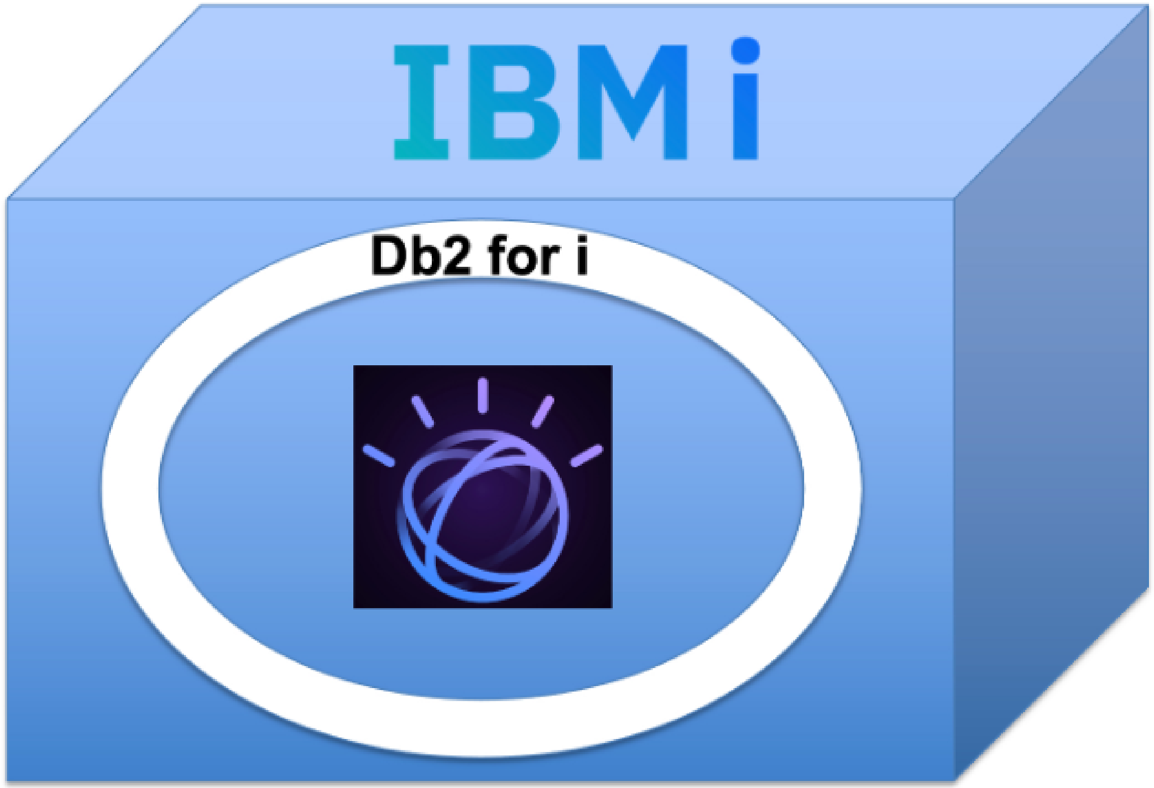
Disadvantages of Spatial Databases

- Cost to implement can be high
- Some inflexibility
- Incompatibilities with some GIS software
- Slower than local, specialized data structures
- User/managerial inexperience and caution

Spatial Database Offerings

- ESRI ArcSDE (on top of several different DBs)
- Oracle Spatial
- IBM DB2 Spatial Extender
- Informix Spatial DataBlade
- MS SQL Server (with ESRI SDE)
- Geomedia on MS Access
- PostGIS / PostgreSQL

Spatial Database Offerings – DB2 for i

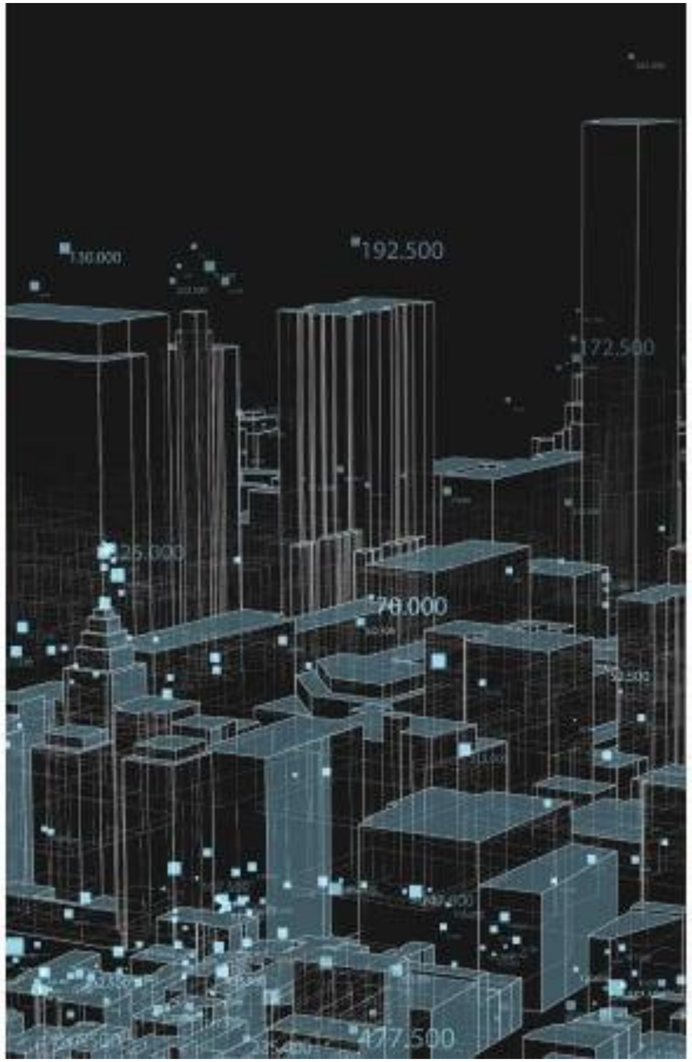


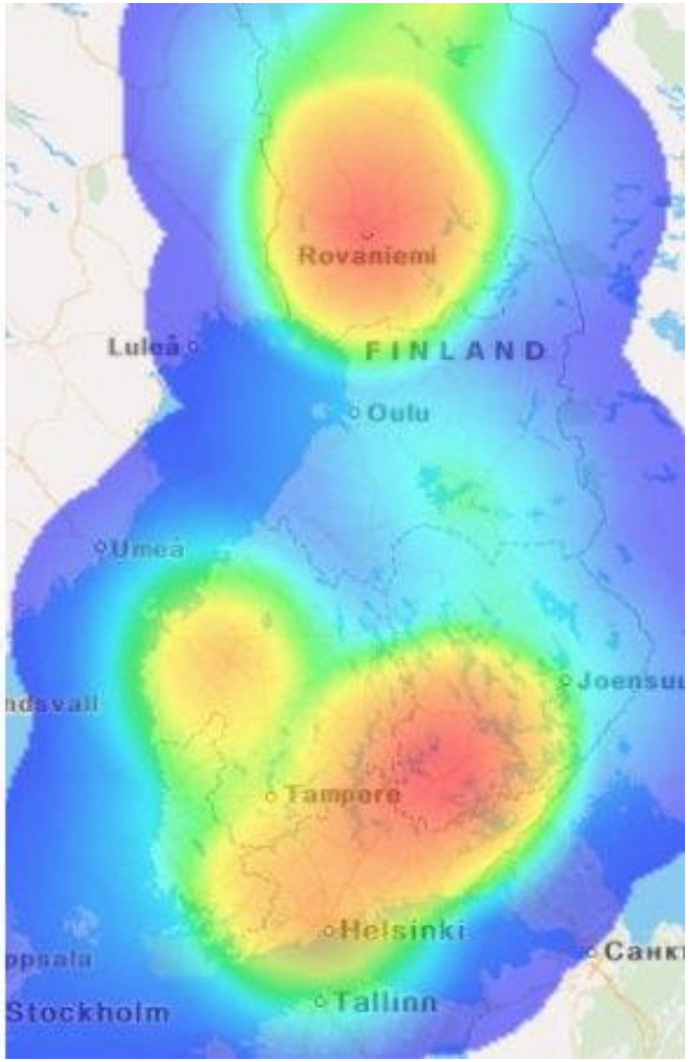
IBM i 7.4 and up

- **IBM Watson features are integrated into Db2 for i:**
  – no reaching outside of IBM i
  – no data movement
  – no extra charge
  – no additional risk
- **Inner Source**
- **Stage 1 – Geospatial Analytics**

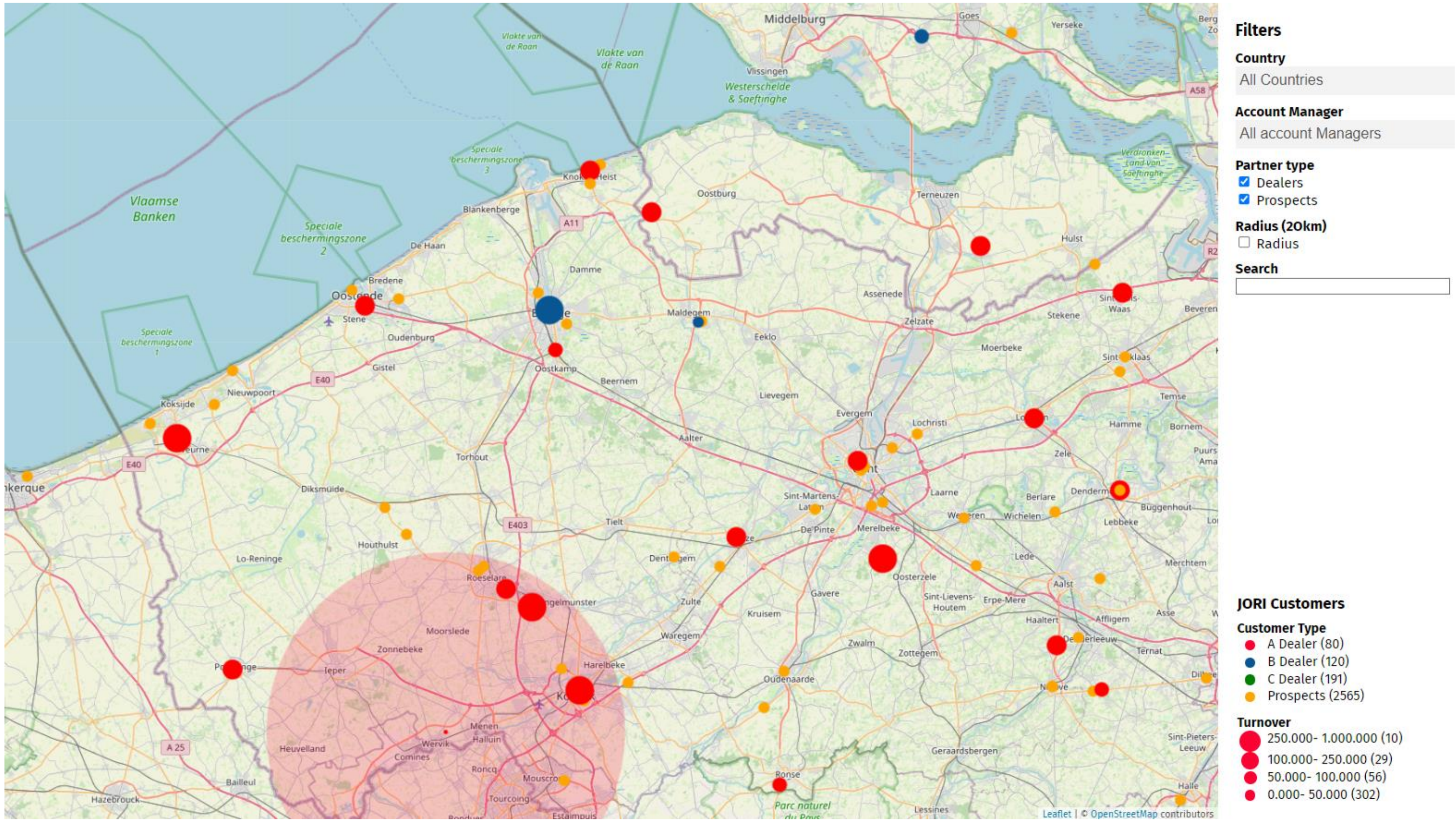
Why where matters

Recent use cases of geospatial

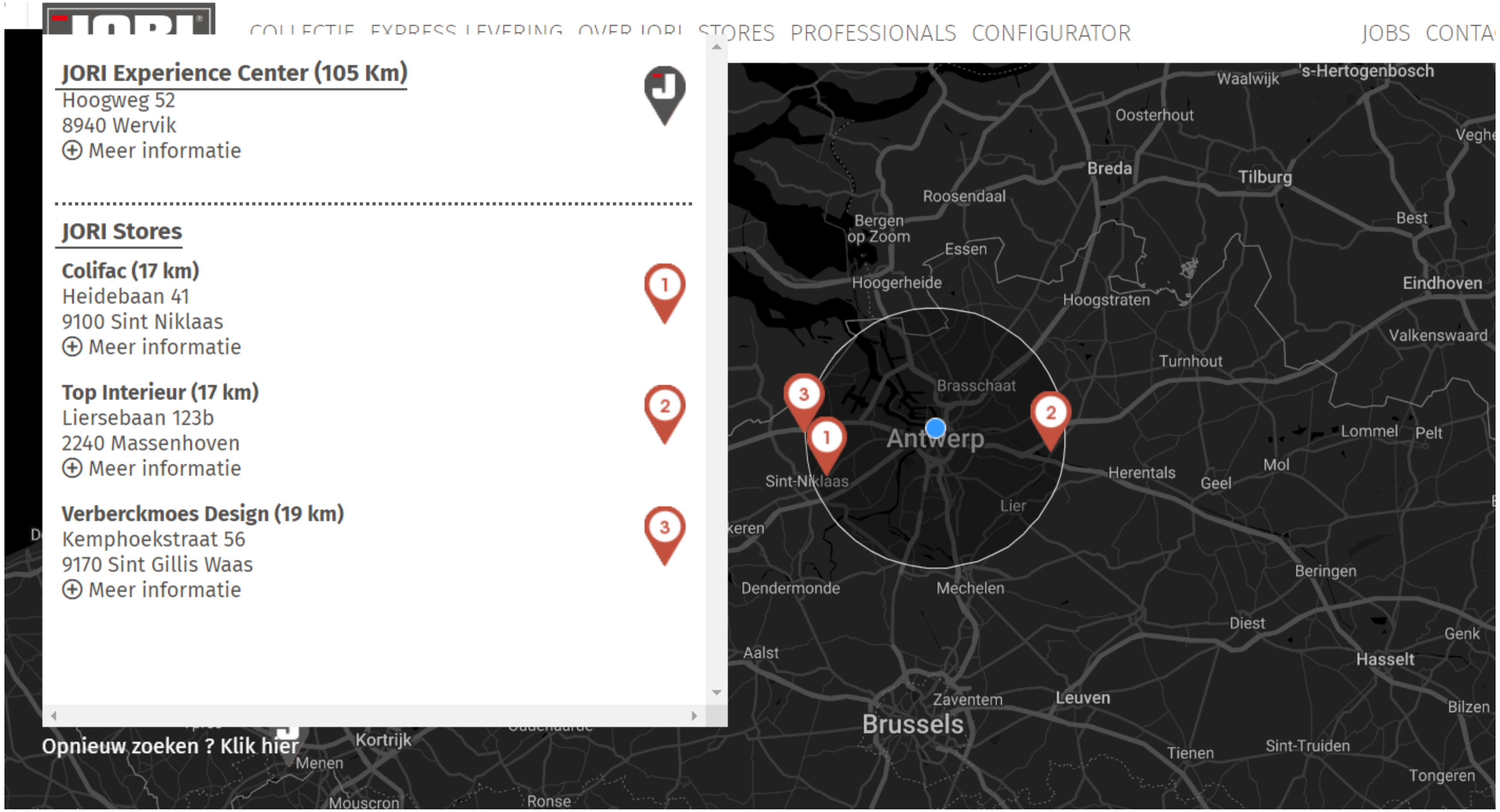




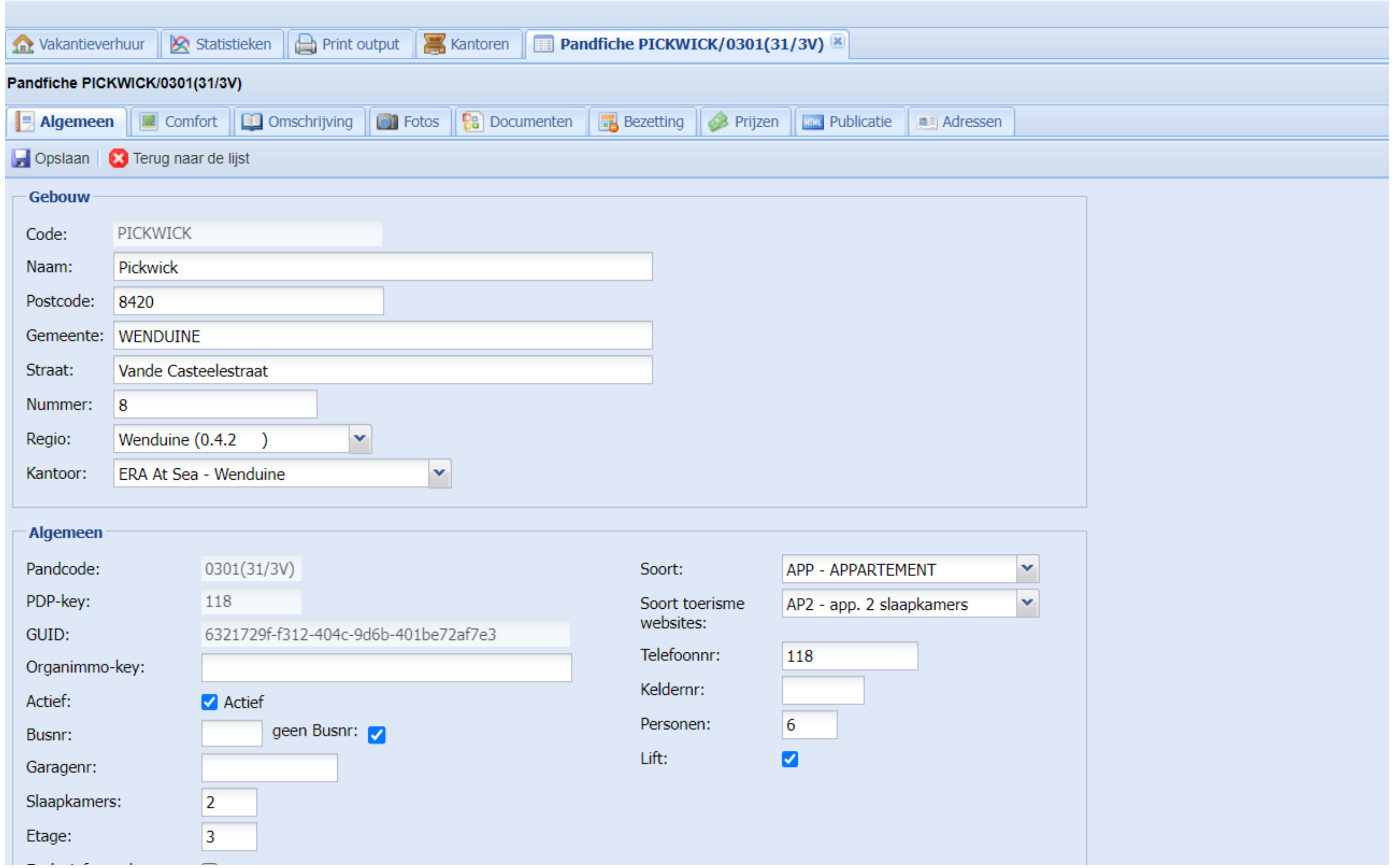
Geomarketing



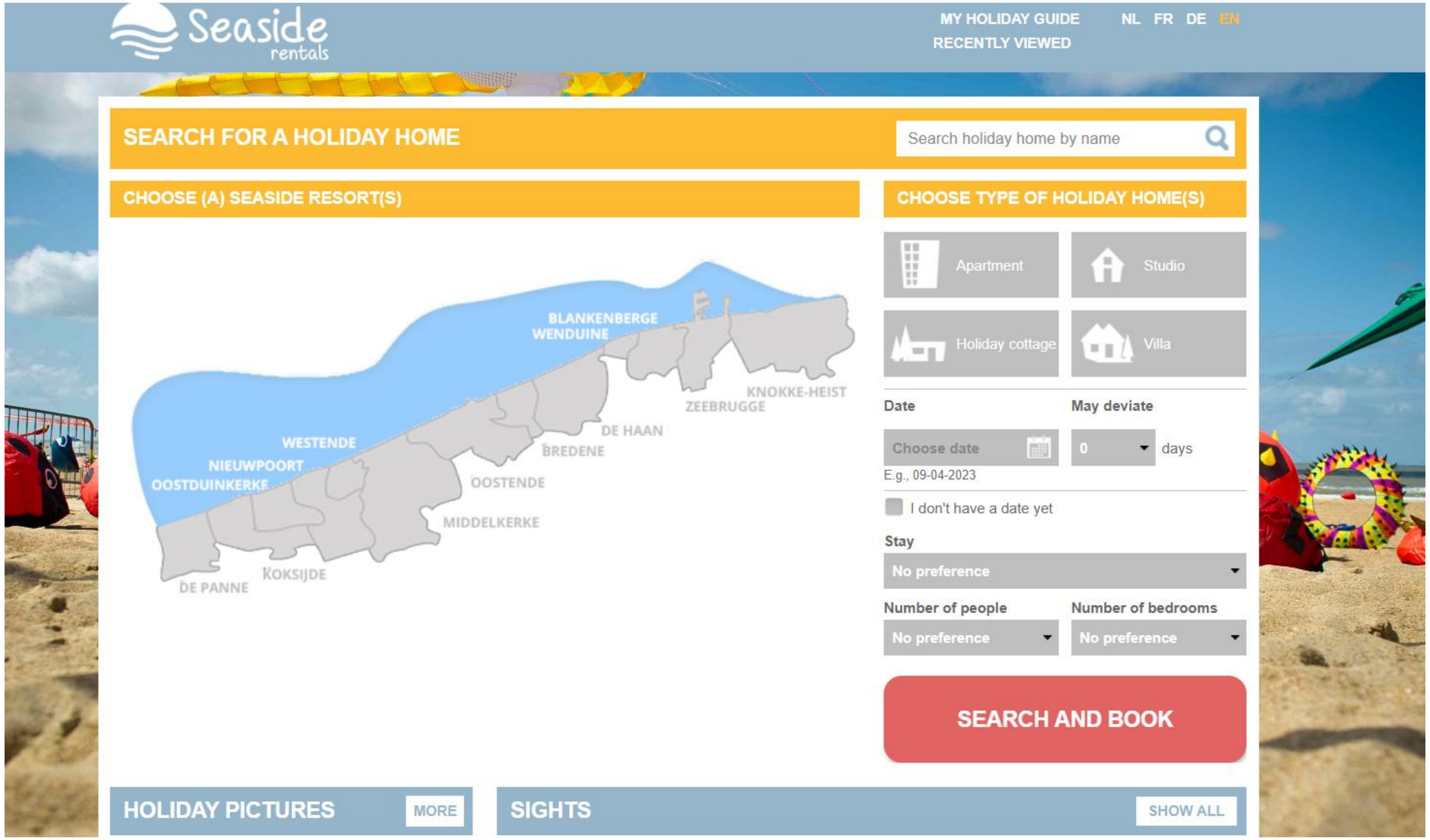
Store locator



Address validation / real estate



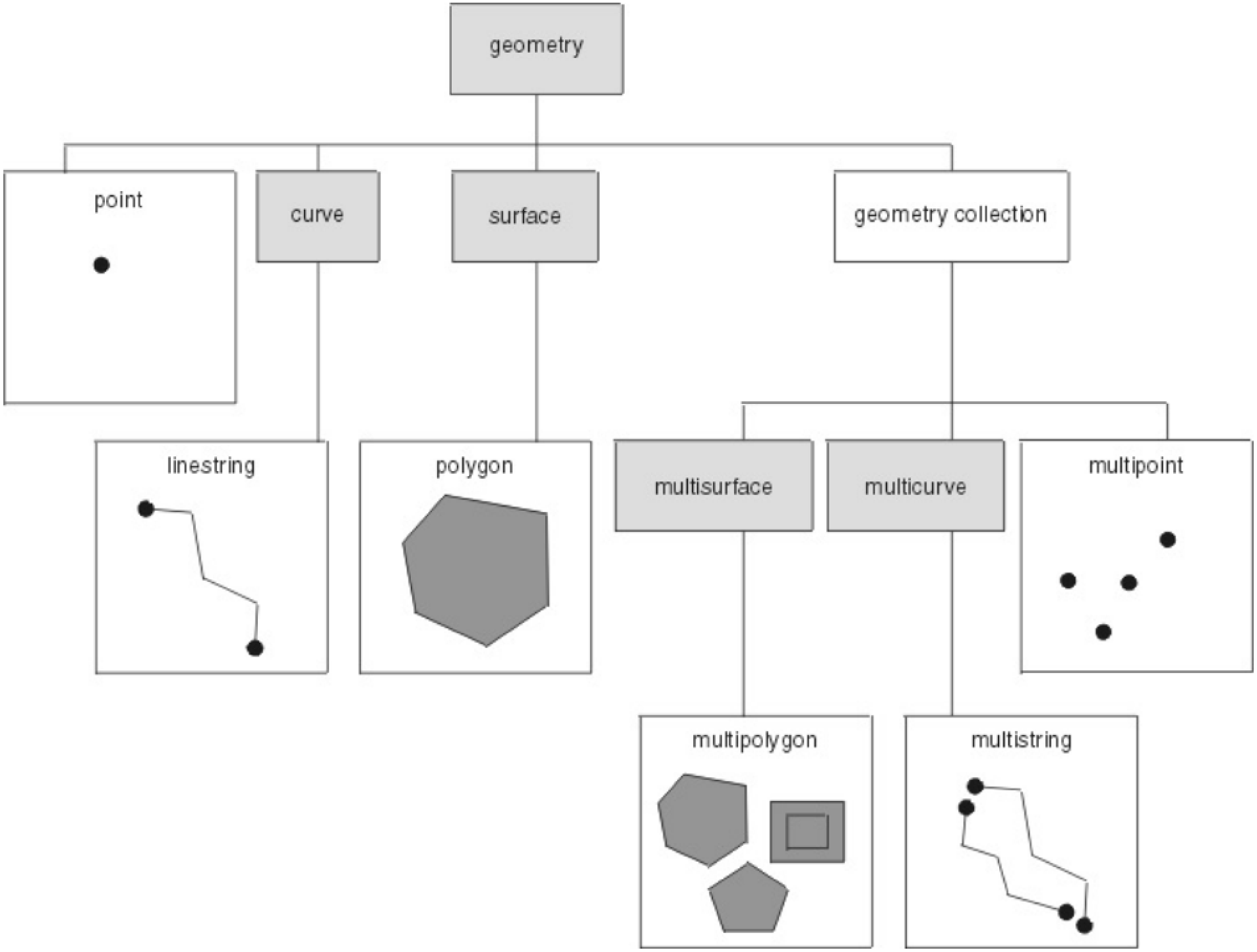
Address validation / real estate



Geospatial Data Types

Geometries

- Spatial data describes the physical location and shape of geometric objects. These objects can be point locations or complex objects such as countries, roads, or lakes.
- For Geospatial Analytics, the operational definition of geometry is "a model of a geographic feature.
- The geometry hierarchy is defined by the OpenGIS Consortium, Inc. (OGC) document "OpenGIS Simple Features Specification for SQL".


geometry
point
curve
surface
geometry collection
linestring
polygon
multisurface
multicurve
multipoint
multipolygon
multistring

Geometries

- **Points (ST_Point)**

A single point. Points represent discrete features that are perceived as occupying the locus where an east-west coordinate line (such as a parallel) intersects a north-south coordinate line (such as ameridian). For example, suppose that the notation on a world map shows that each city on the map islocated at the intersection of a parallel and a meridian. A point could represent each city.

- **Linestrings (ST_LineString)**

A line between two or more points. It does not have to be a straight line. Linestrings represent linear geographic features such as streets, canals, and pipelines.

- **Polygons (ST_Polygon)**

A polygon or surface within a polygon. Polygons represent multisided geographic features such asdistricts, forests, and wildlife habitats.

Geometries

- **Geometry Collection (ST_GeomCollection)**

A geometry collection containing one or more geometry types. Geometry collections represent multipart features with a variety of components such as a group of lakes (polygons) and rivers (linestrings) that form a watershed.

The homogeneous collections are:

- **Multipoints (ST_MultiPoint)**

A geometry collection containing multiple points. Multipoints represent multipart features whose components are each located at the intersection of an east-west coordinate line and a north-south coordinate line. An example is an island chain whose members are each situated at an intersection of a parallel and a meridian.

Geometries

- **Multilinestrings (ST_MultiLinestring)**

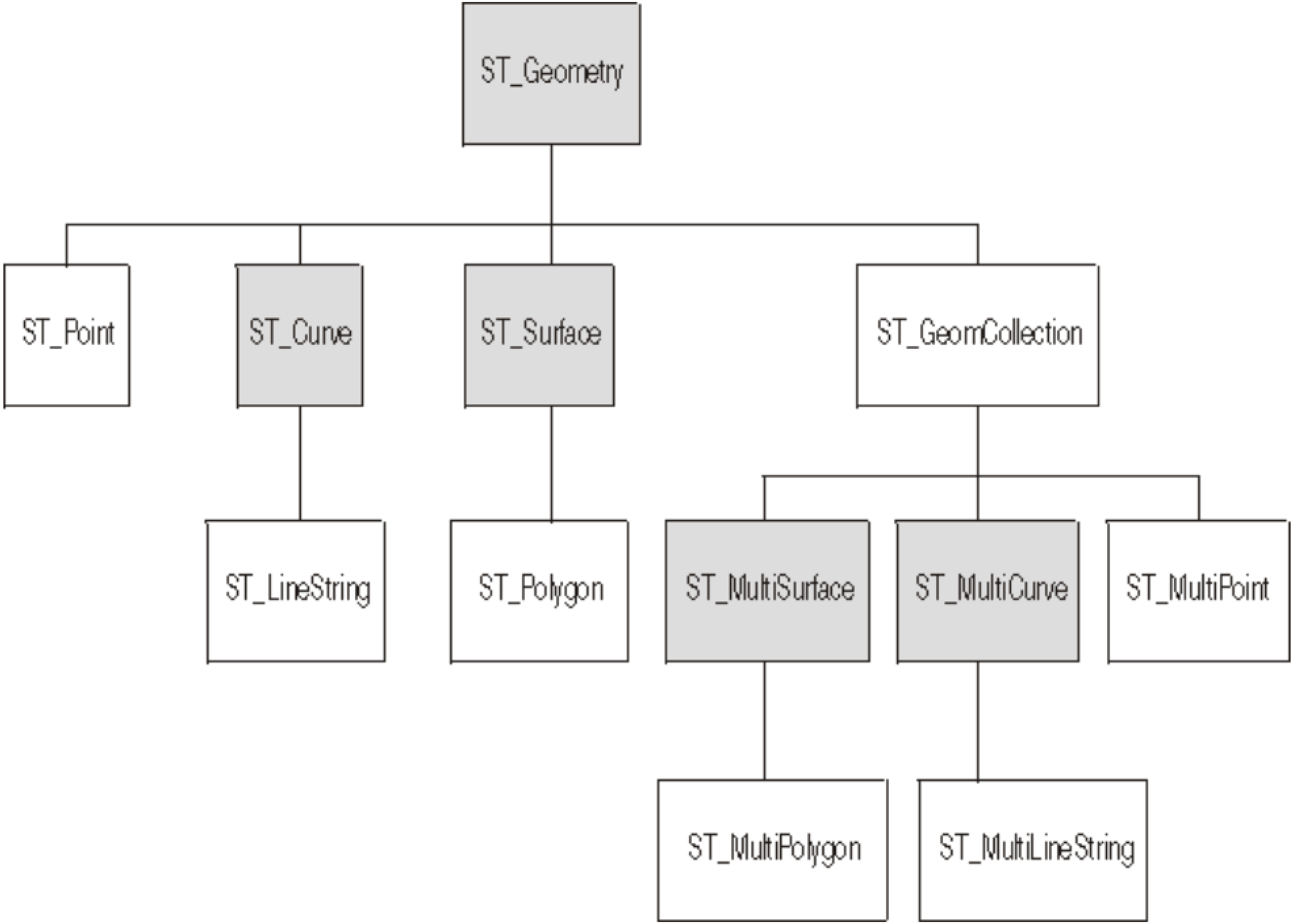
A geometry collection containing multiple linestrings. Multilinestrings represent multipart features made up of more than one linear component such as river systems and highway systems.

- **Multipolygons (ST_MultiPolygon)**

A geometry collection containing multiple polygons. Multipolygons represent multipart features made up of multisided units or components such as the collective farmlands in a specific region or a system of lakes

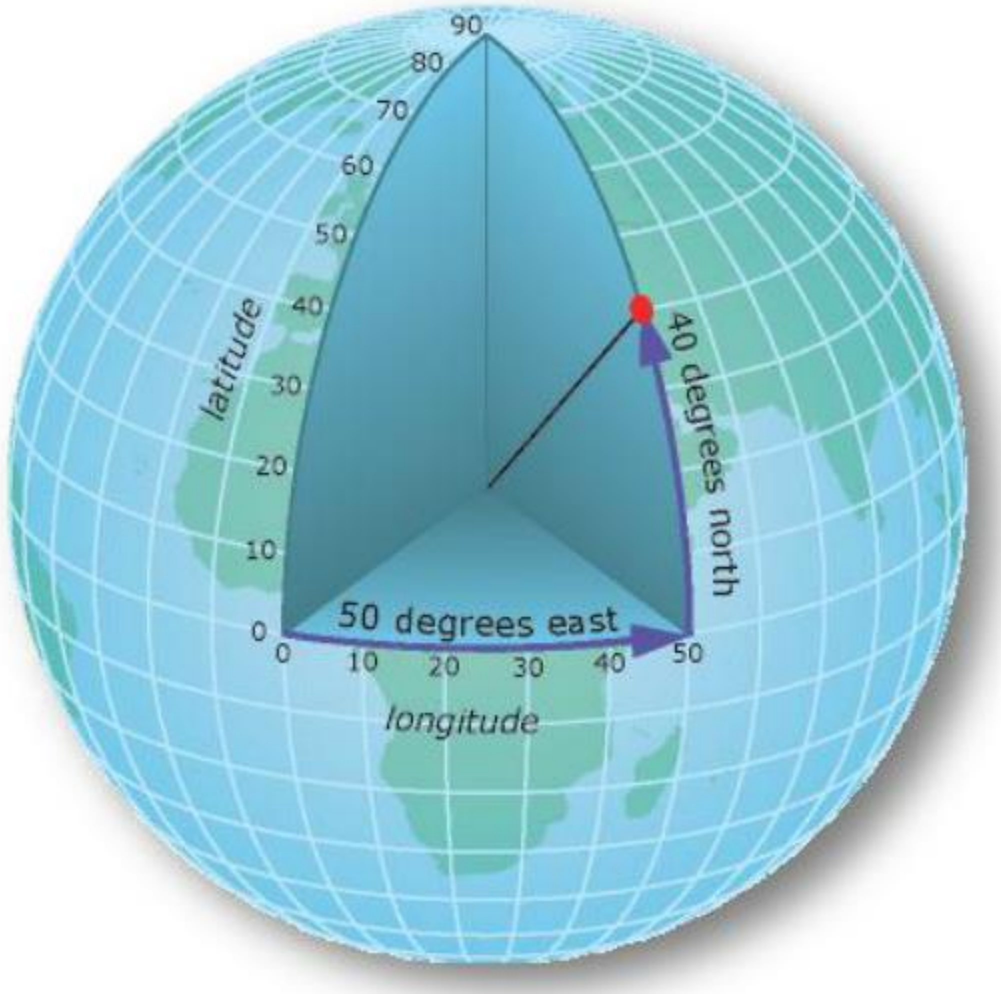
- **Empty geometries (ST_Geometry)**

A geometry is empty if it does not contain any points. An empty geometry is considered a simple geometry.


ST_Geometry
ST_Point
ST_Curve
ST_Surface
ST_GeomCollection
ST_LineString
ST_Polygon
ST_MultiSurface
ST_MultiCurve
ST_MultiPoint
ST_MultiPolygon
ST_MultiLineString

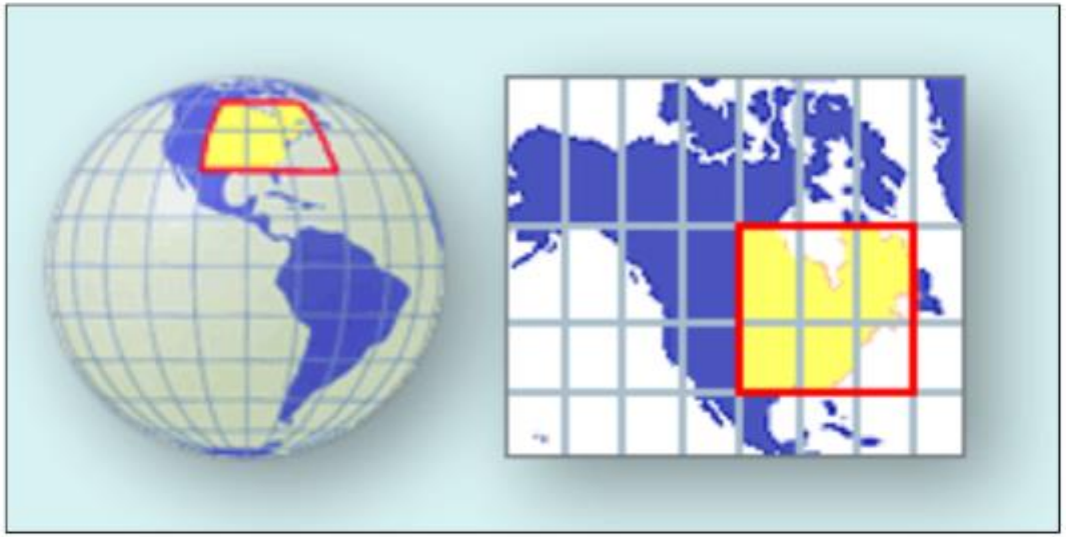
Geometry coordinates

- latitude and longitude are components of the *Geographic* reference frame and are the most common ways to encode geospatial information



Map projections

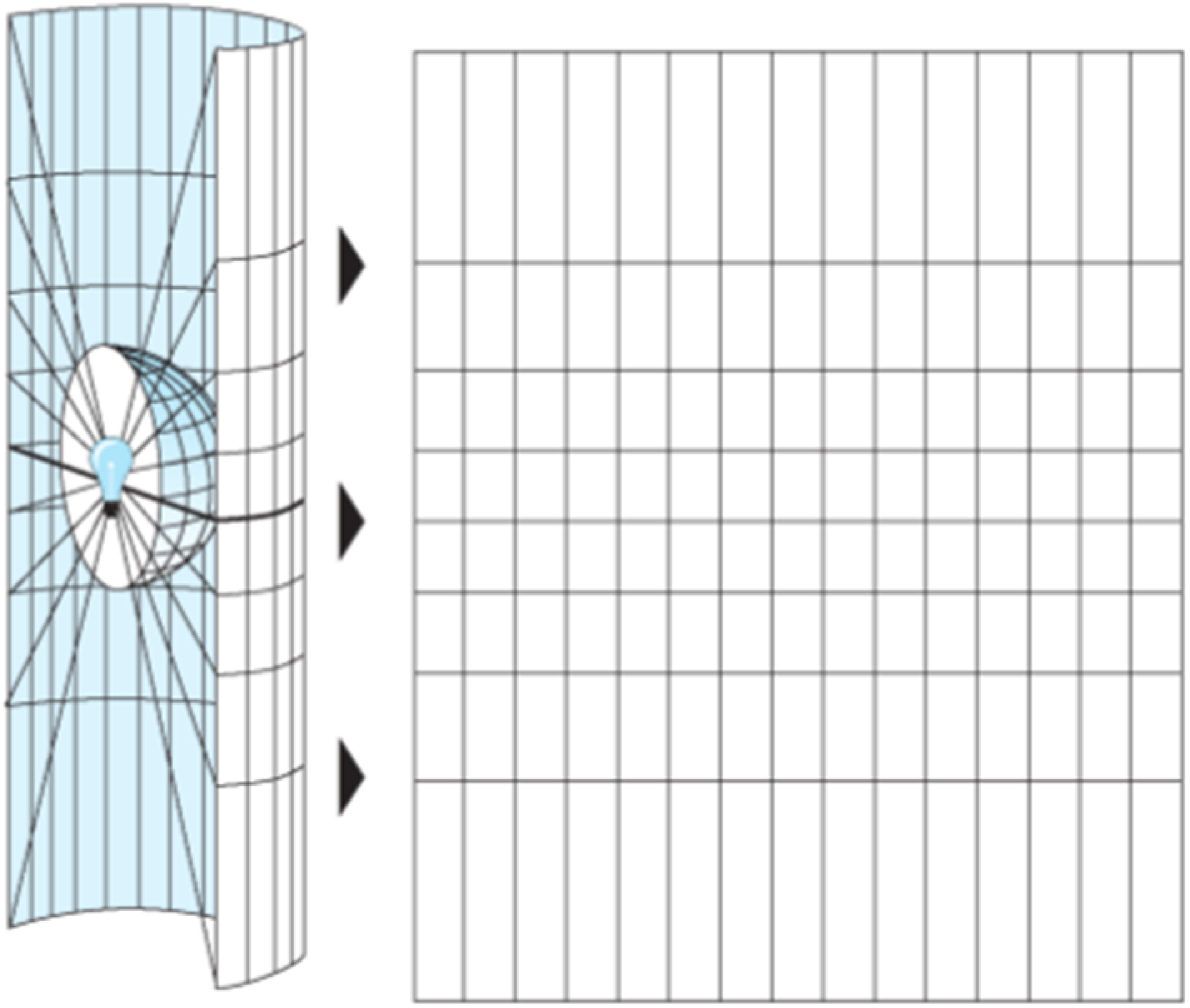
- often we need to convert a 3D earth to a 2D plane
- this is necessary for accurate calculation of distances, bearing and area calculations



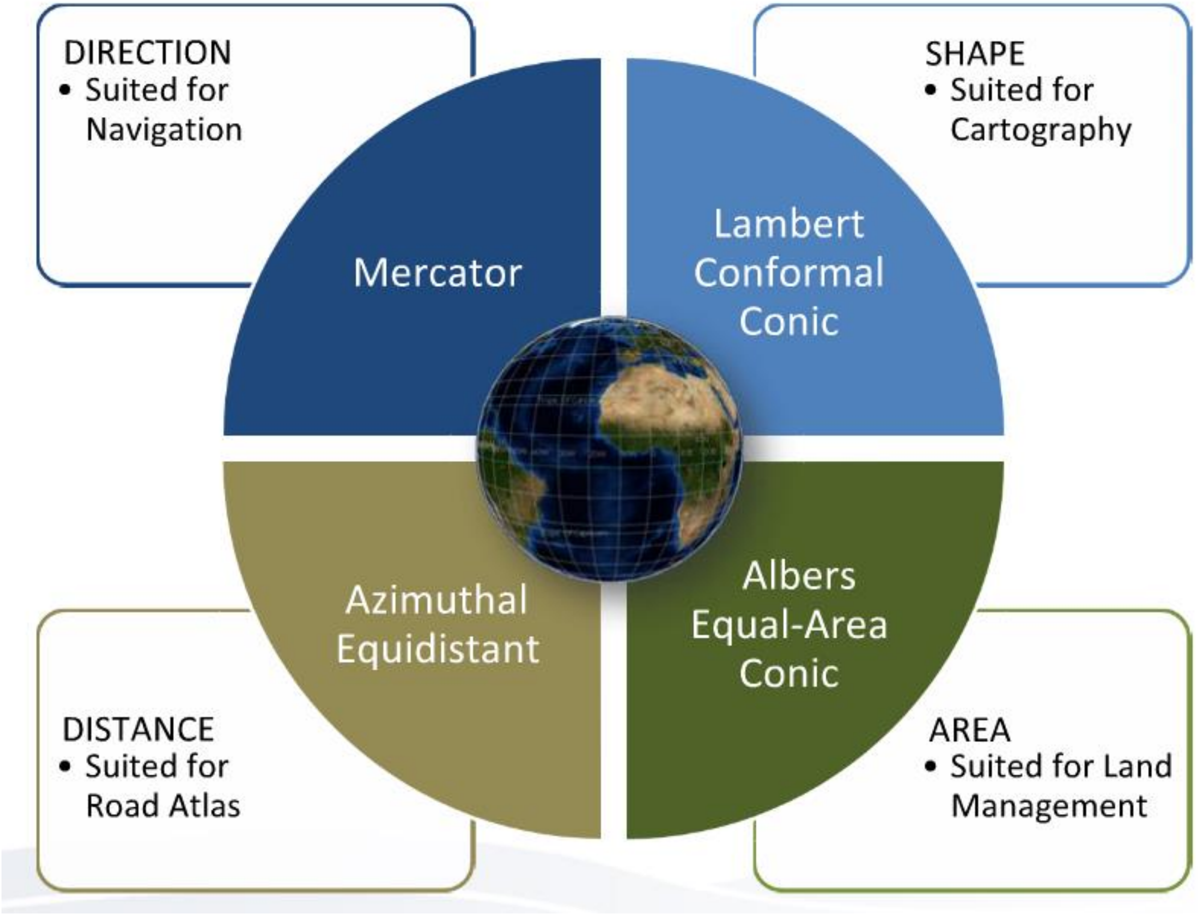
3 dimensional → 2 dimensional

Map projections

in a general sense, projection is similar to shining a light through a transparent sphere and tracing the lines of lat/long



Map projections



DB2 for I Coordinate systems

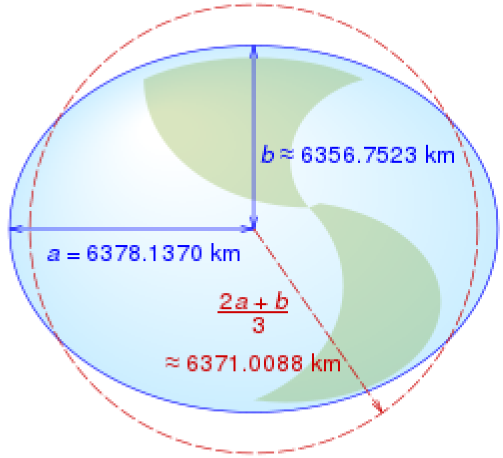
A coordinate system is a framework for defining the relative locations of things in a specified area; for example, an area on the Earth's surface or the Earth's surface as a whole. DB2 for i Geospatial Analytics supports the Geographic Coordinate System using WGS_1984 datum (GCS_WGS_1984). Information about the coordinate system can be accessed through the QSYS2.ST_COORDINATE_SYSTEMS catalog view.

```
1 select * from QSYS2.ST_COORDINATE_SYSTEMS
```

COORDSYS_NAME	COORDSYS_TYPE	DEFINITION	ORGANIZATION	ORGANIZATION_COORDSYS_ID
GCS WGS 1984	GEOGRAPHIC	GEOGCS[WGS 84,DATUM[WGS 1984,SPHEROID[WGS 84,6378137,298.257223563,AUTHORITY[EPSG,7030]],AUTHORITY[EPS	EPSG	4326

WGS_1984 datum

- In the early 1980s, the need for a new world geodetic system was generally recognized by the geodetic community as well as within the US Department of Defense.
- The new world geodetic system was called WGS 84. It is the reference system used by the Global Positioning System (GPS).
- https://en.wikipedia.org/wiki/World_Geodetic_System#WGS84



Geospatial SQL

Working with geometries

Create a point geometry from a latitude/longitude pair

```
CREATE TABLE GEODATA/DLRGPSNEW (
        DLRID CHAR(8) CCSID 37 NOT NULL DEFAULT '' ,
        LOCATION QSYS2.ST_POINT)
        RCDFMT DLRGPSR    ;

INSERT INTO  GEODATA/DLRGPSNEW (DLRID, LOCATION) SELECT ID,
QSYS2.ST_POINT(GPSLON, GPSLAT) FROM GEODATA/DLRGPS2;
```

Create a geometry from Well Known Text

Well-known text (**WKT**) is a text markup language for representing vector geometry objects.

A binary equivalent, known as **well-known binary** (**WKB**), is used to transfer and store the same information.

The formats were originally defined by the Open Geospatial Consortium (OGC)

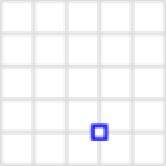
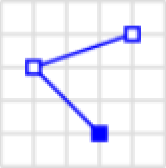
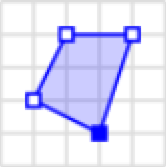
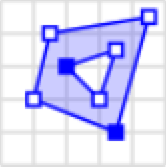
Create a geometry from Well Known Text

INSERT INTO sample_points(id, geometry) VALUES(10, QSYS2.ST_POINT(10, 20)), (20, QSYS2.ST_POINT('point (30 40)’));

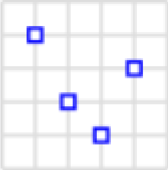
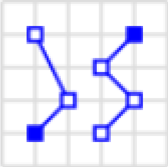
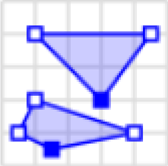
INSERT INTO sample_geometries VALUES (10, QSYS2.ST_WKTTOSQL('point (44 14)')), (11, QSYS2.ST_WKTTOSQL('point (24 13)')), (12, QSYS2.ST_WKTTOSQL('polygon ((50 20, 50 40, 70 30, 50 20))'));

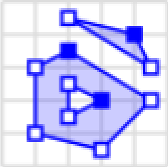
Create a geometry from Well Known Text

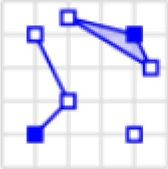
Geometry primitives (2D)

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))`

Multipart geometries (2D)

Type	Examples	
MultiPoint		`MULTIPOINT ((10 40), (40 30), (20 20), (30 10))`
		`MULTIPOINT (10 40, 40 30, 20 20, 30 10)`
MultiLineString		`MULTILINESTRING ((10 10, 20 20, 10 40),` `(40 40, 30 30, 40 20, 30 10))`
MultiPolygon		`MULTIPOLYGON (((30 20, 45 40, 10 40, 30 20)),` `((15 5, 40 10, 10 20, 5 10, 15 5)))`

		`MULTIPOLYGON (((40 40, 20 45, 45 30, 40 40)),`
		`((20 35, 10 30, 10 10, 30 5, 45 20, 20 35),`
		`(30 20, 20 15, 20 25, 30 20)))`

GeometryCollection		`GEOMETRYCOLLECTION (POINT (40 10),`
		`LINESTRING (10 10, 20 20, 10 40),`
		`POLYGON ((40 40, 20 45, 45 30, 40 40)))`

Get a latitude and longitude from a geometry

```
SELECT
ST_MINX(LOCATION) as longitude,
ST_MINY(LOCATION) as latitude
FROM  GEODTA/DLRGPSNEW;
```

LONGITUDE	LATITUDE
5.1208922	51.5775033
4.939777	51.140887
-16.004154	28.049509
121.577406	25.059627
121.488462	25.04755
121.541077	25.046594
121.552972	25.080458
4.464186	50.872804
4.9469	52.3026707
5.433568	51.418669
5.7030227	50.8420631
3.540088	51.464231
5.454767	51.560976

Measurements

Find the area of a polygon

```
SELECT park_name,
      QSYS2.ST_AREA(geometry) as area_square_meters
 FROM geodata.sample_parks;
```

PARK_NAME	AREA_SQUARE_METERS
Central Park	3427480.9038014133
Washington Square Park	40003.49153390578

Find the distance between to geometries

SELECT location_name, QSYS2.ST_DISTANCE(location_point, geometry)
 as distance_meters
  FROM sample_points, sample_parks
  WHERE park_name = 'Washington Square Park';

LOCATION_NAME	DISTANCE_METERS
Empire State Building	2166.918672580893
Chrysler Building	2883.6870923750703
Rockefeller Center	3403.6531541977865

Possible improvements

- ST_PERIMETER
- ST_LENGTH

Transform geometries

Create a buffer around a geometry

Create a buffer around a geometry, which returns a new geometry. The number in the function below is the distance in meters for the buffer.

SELECT

ST_BUFFER(location, 100) as closest_point

FROM GEODATA/DLRGPSNEW;

CLOSEST_POINT
01000000E61000000100000000900000004684A82ECEC94940E98DE60D09CA4940EE7DF92D507A1440D54A990B467D1440000000
01000000E6100000010000000090000000B51C3026EB91494066196E052692494031FC5D90DDC013405E47663DCCC3134000000
01000000E61000000100000000900000007653F2BF710C3C40DB9F6A7EE70C3C4015A2B0F1520130C07AA96B87CD0030C0000000
01000000E6100000010000000090000000F6C3D1D7080F3940B2B949967E0F3940BA55F2F8E3645E4015ECA47704655E40000000
01000000E6100000010000000090000000A8BB615DF10B39400AB1D91B670C3940A8732EB7325F5E40B5370F35535F5E40000000
01000000E610000001000000009000000087FF5CB6B20B3940DEF4D474280C3940BED284C290625E4053FC5440B1625E40000000
01000000E6100000010000000090000000948101065E143940E77779C4D3143940087E9DA453635E403E4FBA2474635E40000000
01000000E61000000100000000900000000656FD9A9A6F4940DD4A3B7AD56F4940EE325C66DED91140B7E3EDBFC8DC1140000000
01000000E6100000010000000090000000AB7BBE7CA0264A406D9BFC5BDB264A40460B980B1FC813407A9D0C4321CB1340000000
01000000E6100000010000000090000000ECC97E8279B54940CECEBC61B4B54940235E9BA27FBA15408B5E49DD72BD1540000000
01000000E61000000100000000900000003084CA48AB6B494022780828E66B49402813115570CE1640949CB9305AD11640000000
01000000E6100000010000000090000000ECC0407C4EBB494032C77E5B89BB4940DD979FAC254F0C40AECCEFA30D550C40000000
01000000E6100000010000000090000000A04422A0B0C74940C74D607FEBC74940DAA933A533D015409DBC4E3C29D31540000000
01000000E6100000010000000090000000D21584B9CB8D4540DB47C198068E4540EC7A0E9F296C1740300CECBEAE6E1740000000
01000000E61000000100000000900000009AF6D6A01D3B4840AEA81480583B48400508242E4986414071A082F6A1864140000000

Create a buffer around a geometry

SELECT

ST_ASTEXT(ST_Buffer(location, 100)) as closest_point

FROM QS36F/DLRGPSNEW;

CLOSEST_POINT
POLYGON ((5.120293 51.578401, 5.119446 51.577875, 5.119447 51.577130999999994, 5.120293 51.576605, 5.121491 51.576605, 5.122337 51.577130999999994, 5.122338 51.577875, 5.121491 51.578401, 5.120293 51.578401))
POLYGON ((4.939184 51.141785, 4.938345 51.141259, 4.938345 51.140515, 4.939184 51.139989, 4.94037 51.139989, 4.941209 51.140515, 4.941209 51.141259, 4.94037 51.141785, 4.939184 51.141785))
POLYGON ((-16.004576 28.050407, -16.005171999999998 28.049881, -16.005171999999998 28.049136999999998, -16.004576 28.048610999999998, -16.003732 28.048610999999998, -16.003135999999998 28.049136999999998, -16.003135999999998 28.049881, -16.003732 28.050407, -16.0045
POLYGON ((121.576995 25.060525, 121.576414 25.059998999999998, 121.576414 25.059255, 121.576995 25.058729, 121.577817 25.058729, 121.57839799999999 25.059255, 121.57839799999999 25.059998999999998, 121.577817 25.060525, 121.576995 25.060525))
POLYGON ((121.488051 25.048448, 121.48746999999999 25.047922, 121.48746999999999 25.047178, 121.488051 25.046651999999998, 121.488873 25.046651999999998, 121.489454 25.047178, 121.489454 25.047922, 121.488873 25.048448, 121.488051 25.048448))
POLYGON ((121.54066599999999 25.047492, 121.54008499999999 25.046965999999998, 121.54008499999999 25.046222, 121.54066599999999 25.045696, 121.541488 25.045696, 121.542069 25.046222, 121.542069 25.046965999999998, 121.541488 25.047492, 121.54066599999999 25.047492))
POLYGON ((121.552561 25.081356, 121.55198 25.08083, 121.55198 25.080085999999998, 121.552561 25.079559999999997, 121.553383 25.079559999999997, 121.553964 25.080085999999998, 121.553964 25.08083, 121.553383 25.081356, 121.552561 25.081356))
POLYGON ((4.463596 50.873701999999994, 4.462762 50.873176, 4.462762 50.872431999999996, 4.463596 50.871905999999996, 4.464776 50.871905999999996, 4.46561 50.872431999999996, 4.46561 50.873176, 4.464776 50.873701999999994, 4.463596 50.873701999999994))
POLYGON ((4.9462909999999995 52.303568999999996, 4.945431 52.303042999999995, 4.945431 52.302299, 4.946292 52.301773, 4.947508 52.301773, 4.948369 52.302299, 4.948369 52.303042999999995, 4.947509 52.303568999999996, 4.9462909999999995 52.303568999999996))
POLYGON ((5.432970999999999 51.419567, 5.432128 51.419041, 5.432128 51.418296999999995, 5.432970999999999 51.417770999999995, 5.434165 51.417770999999995, 5.435008 51.418296999999995, 5.435008 51.419041, 5.434165 51.419567, 5.432970999999999 51.419567))
POLYGON ((5.702433999999999 50.842960999999995, 5.7016 50.842434999999995, 5.7016 50.841691, 5.702433999999999 50.841165, 5.703612 50.841165, 5.704446 50.841691, 5.704446 50.842434999999995, 5.703612 50.842960999999995, 5.702433999999999 50.842960999999995))
POLYGON ((3.539491 51.465129, 3.538646 51.464603, 3.538646 51.463859, 3.539491 51.463333, 3.540685 51.463333, 3.54153 51.463859, 3.54153 51.464603, 3.540685 51.465129, 3.539491 51.465129))
POLYGON ((5.454168 51.561873999999996, 5.453322 51.561347999999995, 5.453322 51.560604, 5.454168 51.560078, 5.455366 51.560078, 5.456212 51.560604, 5.456212 51.561347999999995, 5.455366 51.561873999999996, 5.454168 51.561873999999996))

Union geometries together

If you want to union your geometries, use ST_Union to return a new geometry:

```
VALUES QSYS2.ST_ASTEXT(
  QSYS2.ST_UNION(QSYS2.ST_POLYGON('polygon((30 30, 50 30, 50 50, 30 50, 30 30))'),
                 QSYS2.ST_POLYGON('polygon((40 40, 60 40, 60 60, 40 60, 40 40))')));
```

Find the resulting polygon of an intersection

Commonly know as a clip in GIS, use this function to return the resulting intersecting area from two geometries:

```
VALUES QSYS2.ST_ASTEXT(
  QSYS2.ST_INTERSECTION(QSYS2.ST_POLYGON('polygo
n((30 30, 50 30, 50 50, 30 50, 30 30))'),
                        QSYS2.ST_POLYGON('polygo
n((40 40, 60 40, 60 60, 40 60, 40 40))')));
```

Find the resulting polygon of an intersection

Or if you want to find the leftover parts of a geometry relations ship you can use ST_Difference:

```
VALUES QSYS2.ST_ASTEXT(
   QSYS2.ST_DIFFERENCE(QSYS2.ST_POLYGON('polygon
((10 10, 20 10, 20 20, 10 20, 10 10))'),
                       QSYS2.ST_POLYGON('polygon
((30 30, 50 30, 50 50, 30 50, 30 30))')));
```

Spatial Relationships

Spatial relationships

See if two geometries overlap, touch, cross, intersect, contain, etc. (or evaluate spatial relationships)

This is a more complicated one as there are several different functions, each with slight differences between them, to evaluate spatial relationships.

Make sure to check the docs of the database.

https://www.ibm.com/support/pages/node/6828077

Spatial relationships

- ST_Equals – returns 1 if the two geometries are exactly equal
- ST_Intersects – returns 1 if the two geometries share any space in common
- ST_Disjoint – returns 1 is the two geometries do not share any space, or the opposite of ST_intersects
- ST_Crosses – returns 1 if the geometries have some, but not all, interior points in common
- ST_Overlaps – returns 1 if the two geometries overlap spatially, or intersect but one does not completely contain the other

Spatial relationships

- ST_Touches – returns 1 if one geometry touches another, but does not intersect the interior
- ST_Within – returns 1 if the first geometry is completely within the second geometry
- ST_Contains – returns 1 if the second geometry is completely contained by the first geometry (opposite of ST_Within)
- ST_Covers - returns 1 if if the first geometry completely covers the second geometry

How to use spatial functions

Single Geometry

```
SELECT
ST_Distance(
        location,
        QSYS2.ST_POINT(4.40647, 51.23774)) as distance
FROM QS36F/DLRGPSNEW;
```

DISTANCE
62381.45528921901
38738.98043446731
3095438.6156673958
9563717.557481514
9559821.684618128
9562874.81835397
9560418.035146102
40824.72818795391 5
124254.71019740666

GEOHASH

Geohash is a public domain geocode system invented in 2008 by Gustavo Niemeyer which encodes a geographic location into a short string of letters and digits.

The 6g cell and its sub-grid

Other complex or specific functions

ST_GEOHASH

A geohash is a number that uniquely identifies a specific region. The geohash algorithm divides the Earth into regions, called cells, and converts the latitude and longitude of the center of each cell into a number that uniquely identifies it. The size of each cell is determined by the depth value. The smaller the depth value, the larger the cell size

Other complex or specific functions

ST_GEOHASH

Geohash Depth	Approximate Cell Size	Description	Examples
45	.1 km^2	Single point or address	GPS-location or house
28	3 km^2	Small region	city block
23	100 km^2	Medium-sized region	forest or lake
18	3,000 km^2	Large region	county or postal code area
13	100,000 km^2	Very large region	state or country

Geohash functions

Filtering using a geohash cover

You can use a geohash cover to filter geometry objects when querying geospatial data, which will greatly improving query performance.

A geohash cover is the set of geohash cells that are needed to completely cover a particular geometry.

The ST_GEOHASH, ST_GEOHASHCOVER, and ST_GEOHASHCOVEREXTEND table functions can be used to generate a geohash, a geohash cover, or extend a geohash cover by a given distance.

Sample Cookbook Recipes

Simple nearest neighbor

```
/* test antwerp */
WITH p(LATPOINT, LONGPOINT, RADIUS, DISTANCE_UNIT) AS
 (VALUES(51.23774, 4.40647, 200, 111.045) )
select id, gpslat, gpslon, latpoint, longpoint,radius, distance_unit,
 DECIMAL((P.DISTANCE_UNIT *  DEGREES(ACOS(COS(RADIANS(latpoint))
 * COS(RADIANS(gpslat))
 * COS(RADIANS(longpoint - gpslon))
 + SIN(RADIANS(latpoint))
 * SIN(RADIANS(gpslat))))), 13, 7)
 as distance
 from joricfg.dlrgps, p  WHERE gpslat
 BETWEEN latpoint  -
 (radius / distance_unit)
  AND latpoint  +
  (radius / distance_unit)
 and gpslon BETWEEN longpoint  -
  (radius / distance_unit *
  COS(RADIANS(p.latpoint)))
 AND longpoint  +
 (radius / distance_unit *
 COS(RADIANS(p.latpoint)))
 ORDER by distance;
```

Simple nearest neighbor

ID	GPSLAT	GPSLON	LATPOINT	LONGPOINT	RADIUS	DISTANCE_UNIT	DISTANCE
TOPINE	51.2030455	4.6467983	51.23774	4.40647	200	111.045	17.1531803
WOONAR	51.1699430	4.1792540	51.23774	4.40647	200	111.045	17.5097565
VBERCK	51.2283508	4.1337293	51.23774	4.40647	200	111.045	18.9926428
VECO	51.3912642	4.5931434	51.23774	4.40647	200	111.045	21.4128865
DSCHOE	51.0413713	4.4357621	51.23774	4.40647	200	111.045	21.9010576
LP	51.0320171	4.1291615	51.23774	4.40647	200	111.045	29.9204991
COCKAR	50.9586037	4.3181057	51.23774	4.40647	200	111.045	31.6032527
SUBLIM	51.1034001	3.9931346	51.23774	4.40647	200	111.045	32.4153080
BERDEP	51.2743788	3.9088703	51.23774	4.40647	200	111.045	34.8198880
MASTER	51.2790459	4.9545805	51.23774	4.40647	200	111.045	38.3649227
ABITAG	51.1408870	4.9397770	51.23774	4.40647	200	111.045	38.6434581
WEYTS	51.5499570	4.1453111	51.23774	4.40647	200	111.045	39.1081721
STOLZ	50.8840061	4.5286189	51.23774	4.40647	200	111.045	40.1947902
AHRENB	50.8728040	4.4641860	51.23774	4.40647	200	111.045	40.7240628
PEETER	51.0725332	4.9641264	51.23774	4.40647	200	111.045	42.9546332
REPOS	50.8410291	4.3510150	51.23774	4.40647	200	111.045	44.2226003
RIANT B	51.5814475	4.7312799	51.23774	4.40647	200	111.045	44.3041355
SATELL	51.5794800	4.7532900	51.23774	4.40647	200	111.045	44.9129181
LONCIN L	50.8821044	4.7134029	51.23774	4.40647	200	111.045	44.9272832
PONSAE	50.9759220	4.9149330	51.23774	4.40647	200	111.045	45.8478786
2BWORK	51.5626305	4.8177798	51.23774	4.40647	200	111.045	45.9731239

Simple nearest neighbor

```
/* NEW GEOSPATIAL QUERY */
CREATE VARIABLE ANTWERP QSYS2.ST_POINT;
SET ANTWERP = QSYS2.ST_POINT(4.40647, 51.23774);

SELECT DLRID,
ROUND(QSYS2.ST_DISTANCE(ANTWERP, LOCATION)/1000, 4) AS DISTANCE,
QSYS2.ST_ASTEXT(LOCATION)
FROM QS36F/DLRGPSNEW
ORDER BY DISTANCE ASC;
```

Simple nearest neighbor

DLRID	DISTANCE	00003
TOPINE	17.1955	POINT (4.6467979999999995 51.203046)
WOONAR	17.553	POINT (4.179254 51.169942999999996)
VBERCK	19.0396	POINT (4.133729 51.228350999999996)
VECO	21.4658	POINT (4.5931429999999995 51.391264)
DSCHOE	21.9552	POINT (4.4357619999999995 51.041371)
LP	29.9944	POINT (4.129162 51.032016999999996)
COCKAR	31.6813	POINT (4.318106 50.958604)
SUBLIM	32.4954	POINT (3.9931349999999997 51.1034)
BERDEP	34.906	POINT (3.90887 51.274378999999996)
MASTER	38.4598	POINT (4.954581 51.279046)
ABITAG	38.739	POINT (4.939776999999999 51.140887)
WEYTS	39.2048	POINT (4.1453109999999995 51.549957)
STOLZ	40.2942	POINT (4.528619 50.884006)

Calculate the percentage overlap between polygons

Calculate the percentage overlap of polygons with ST_Intersection and ST_Area. You can do this for one polygon for every row in a table or in a where clause to find overlapping areas that intersect a certain amount.

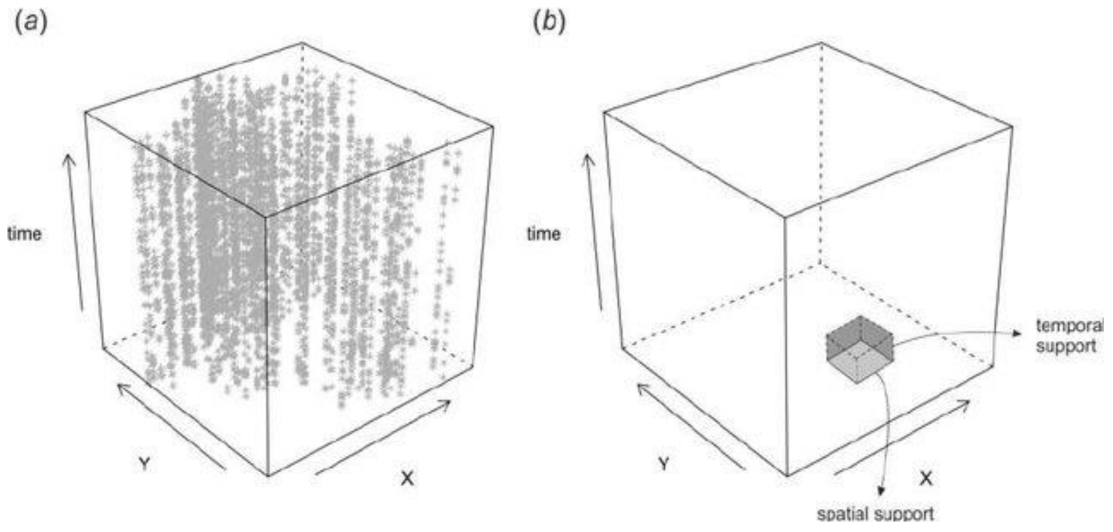
```
WITH a AS (SELECT geom FROM table WHERE id = 1)
SELECT
ST_Area(ST_Intersection(geom, (SELECT geom FROM a))/ST_Area(geom) AS
overlap
FROM table
ORDER BY overlap
```

Calculate the percentage overlap between polygons

```
SELECT
COUNT(a.column), b.id, b.geom
FROM table b
LEFT JOIN table_2 a
ON ST_Intersects(a.geom, b.geom)
WHERE ST_Area(ST_Intersection(a.geom, b.geom)/ST_Area(geom) > .5
```

Spatio-temporal data analysis

- 2 aspects of data:
  - Space – data in spatial context
  - Time – data over a time period
- Collection nature of data: discrete VS continuous observations
- 3 distinct types of attributes: non-spatiotemporal, spatial and temporal attributes
- 2D + t

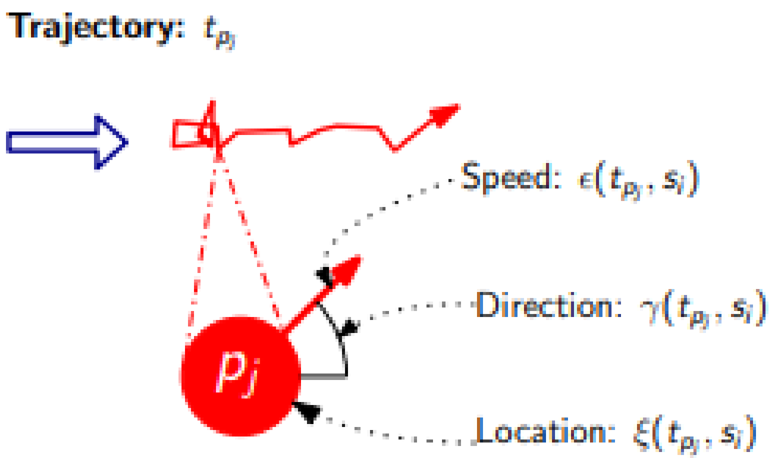


What makes spatio-temporal data difficult?

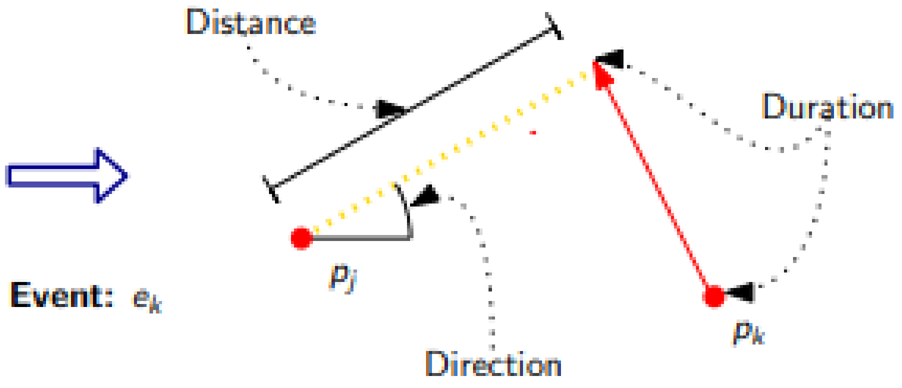
- Combination of space and time
- Spatial ➔ spatio-temporal is difficult
- Very difficult to obtain good prediction (especially further in the future)
- Correlation structure
- Spatiotemporal objects relationships that are complex and implicit
- Non-independent and non-identical distribution across space (spatial heterogeneity) and over time (temporal non-stationarity)

Spatio-temporal data analysis

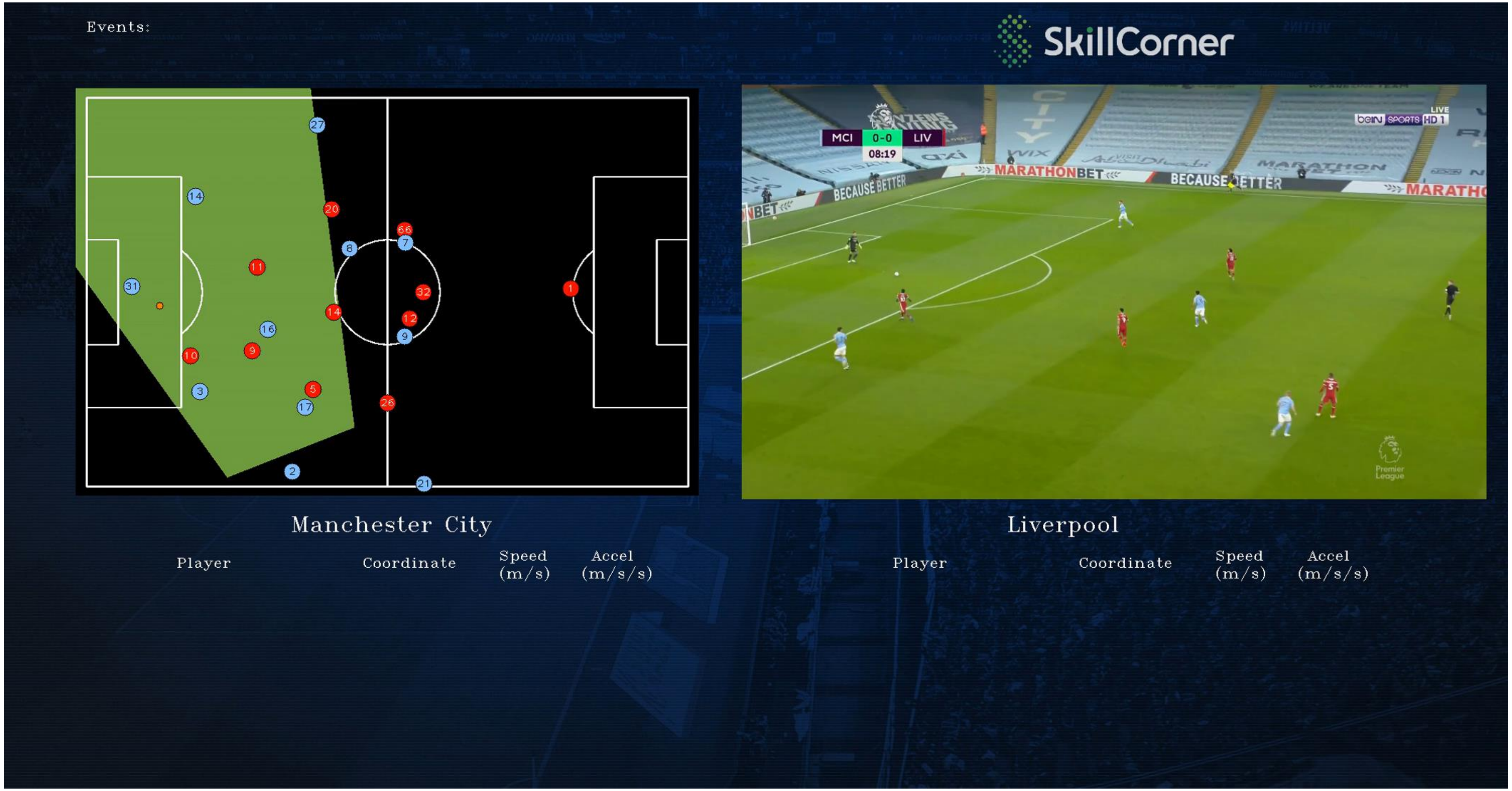
p	s_i	x_1	x_2
...			
p_j	10.1	3940	-2362
p_j	10.2	3948	-2346
p_j	10.3	3956	-2331
p_j	10.4	3923	-2392
p_j	10.5	3977	-2309
...			



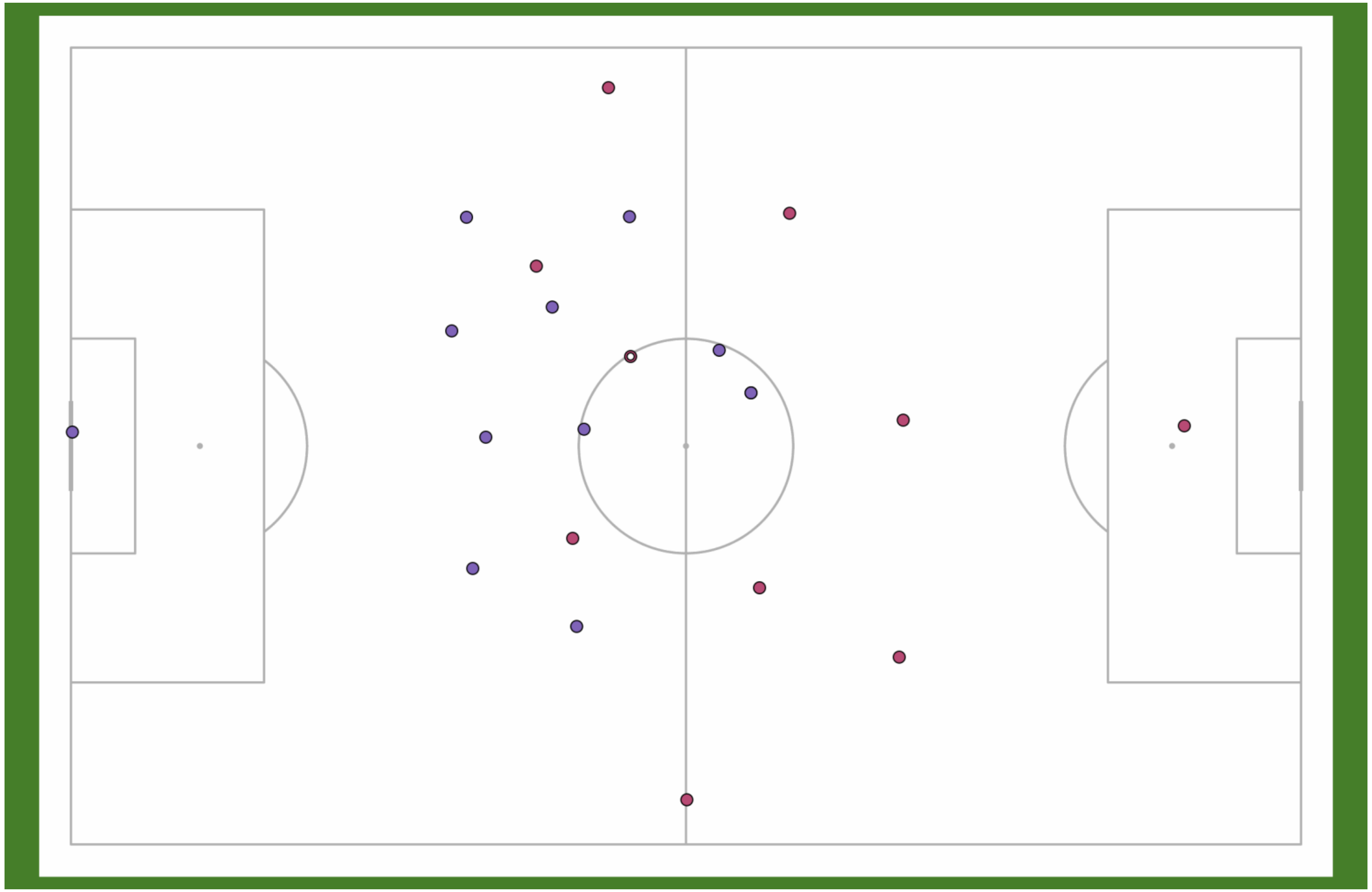
s_i	v	P
...		
10.1	*Touch*	$\{p_j\}$
10.2	*Pass*	$\{p_j\}$
10.6	*Touch*	$\{p_k\}$
11.0	*Tackle*	$\{p_l, p_k\}$
...		



How is spatio-temporal data captured?

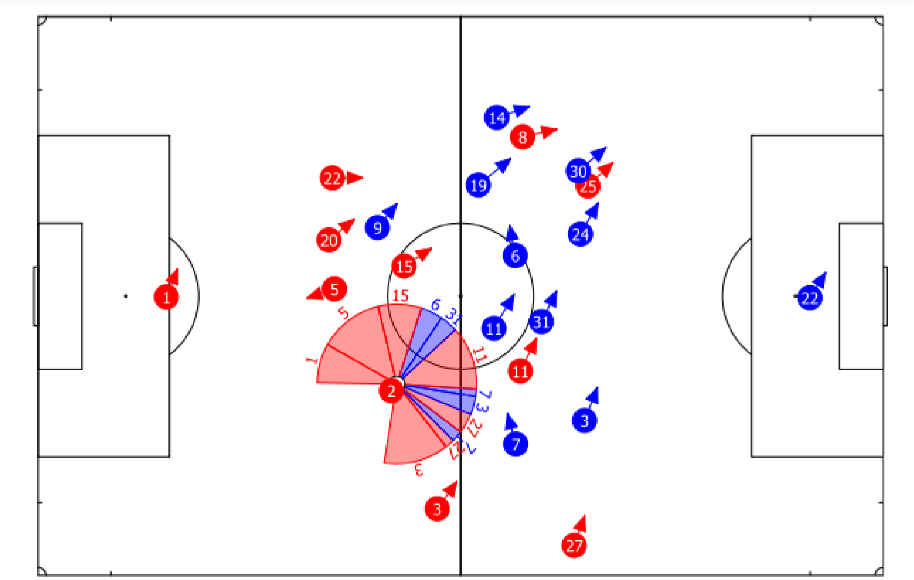


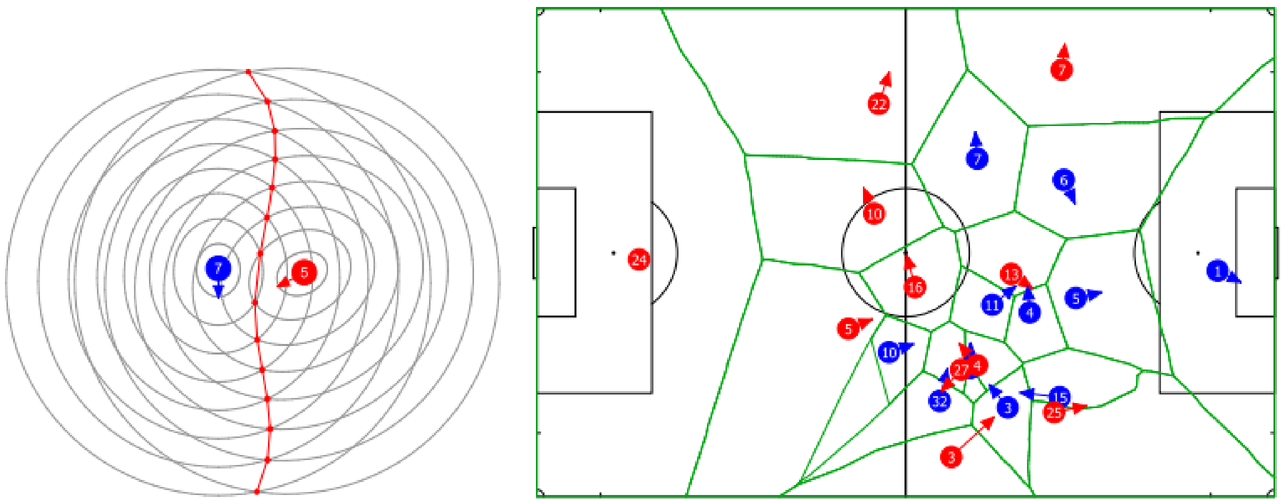
Spatio-temporal data analysis



Spatio-temporal data analysis

$$m\frac{d}{dt}v = F - kv$$





(a) (b)

What about route calculations ?

Our technical setup

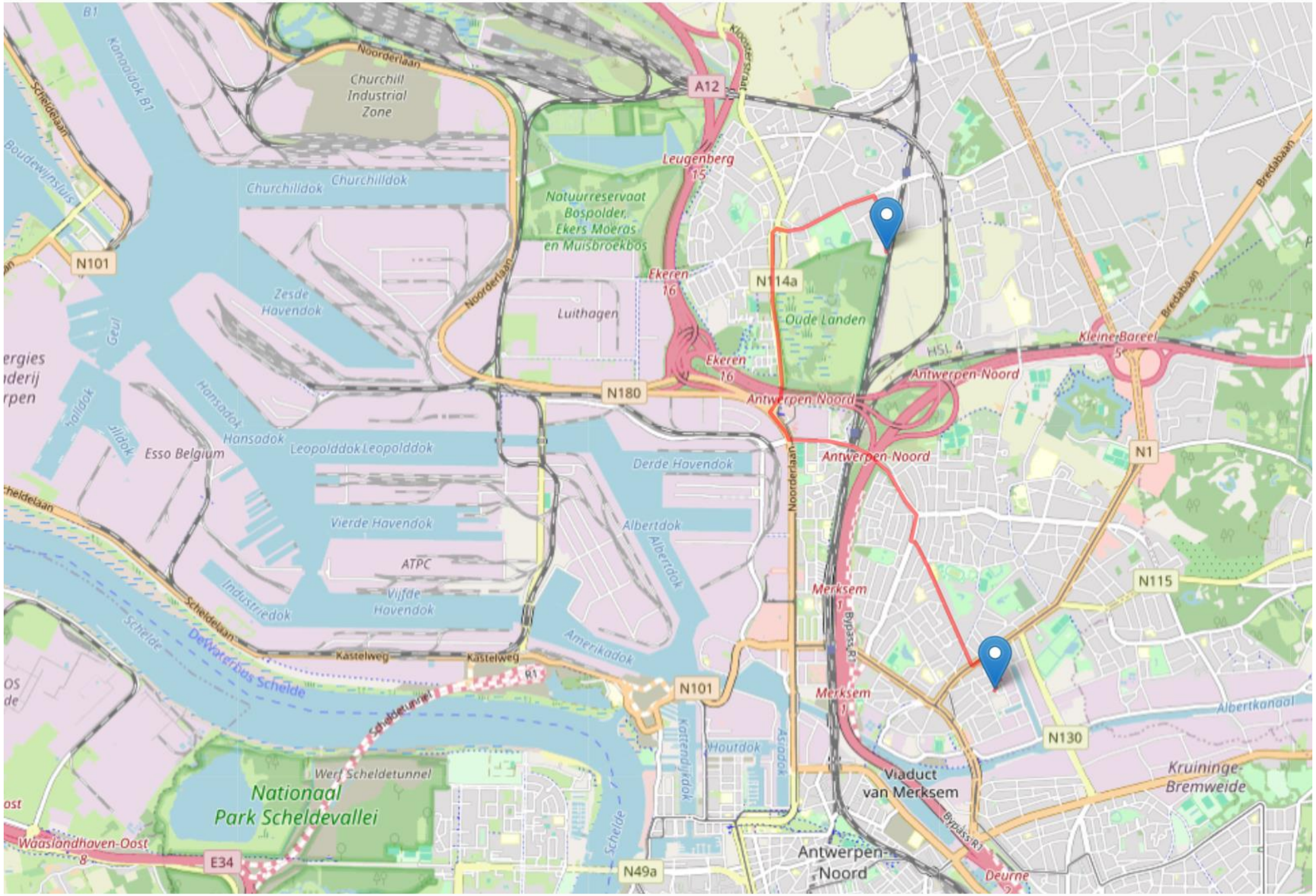
- Download of the roads and metadata for Belgium from EU website
- Upload to a geospatial database (files)
- Create a service to do several calculations (in our case RPG / Python)
- AI Models used for more difficult calculations

Be aware it takes a lot of disk the database. Belgium was already 11 GB

Directions

- Consume rich route instructions in your applications for cars, trucks, different bike profiles, walking, hiking or wheelchair. Make use of plenty of options, ranging from different kinds of road restrictions to vehicle dimensions.

IBM i GEOSPATIAL ROUTE CALCULATOR



```
},
{
    "distance": 257.2,
    "duration": 37.8,
    "type": 1,
    "instruction": "Turn right onto Madrasstraat",
    "name": "Madrasstraat",
    "way_points": [
        1,
        4
    ]
},
{
    "distance": 499.8,
    "duration": 60,
    "type": 0,
    "instruction": "Turn left onto Kattendijkdok-Oostkaai",
    "name": "Kattendijkdok-Oostkaai",
    "way_points": [
        4,
        13
    ]
},
{
    "distance": 439.4,
    "duration": 58.5,
    "type": 1,
    "instruction": "Turn right onto Londenstraat",
    "name": "Londenstraat",
    "way_points": [
        13,
        23
    ]
},
{
    "distance": 3359,
    "duration": 450.3,
    "type": 0,
    "instruction": "Turn left onto Rijnkaai",
    "name": "Rijnkaai",
    "way_points": [
        23,
        91
    ]
},
{
```


Instruction: Head northeast on Groot Hagelkruis

Distance: 116.8 meters

Duration: 28 seconds

Instruction: Turn right onto Steenstraat

Distance: 117.7 meters

Duration: 28.2 seconds

Instruction: Turn right onto Steenstraat, N114

Distance: 1593.2 meters

Duration: 188.5 seconds

Instruction: Turn left onto Noorderlaan, N180

Distance: 302.9 meters

Duration: 44.3 seconds

Instruction: Turn left

Distance: 1403.7 meters

Duration: 195.7 seconds

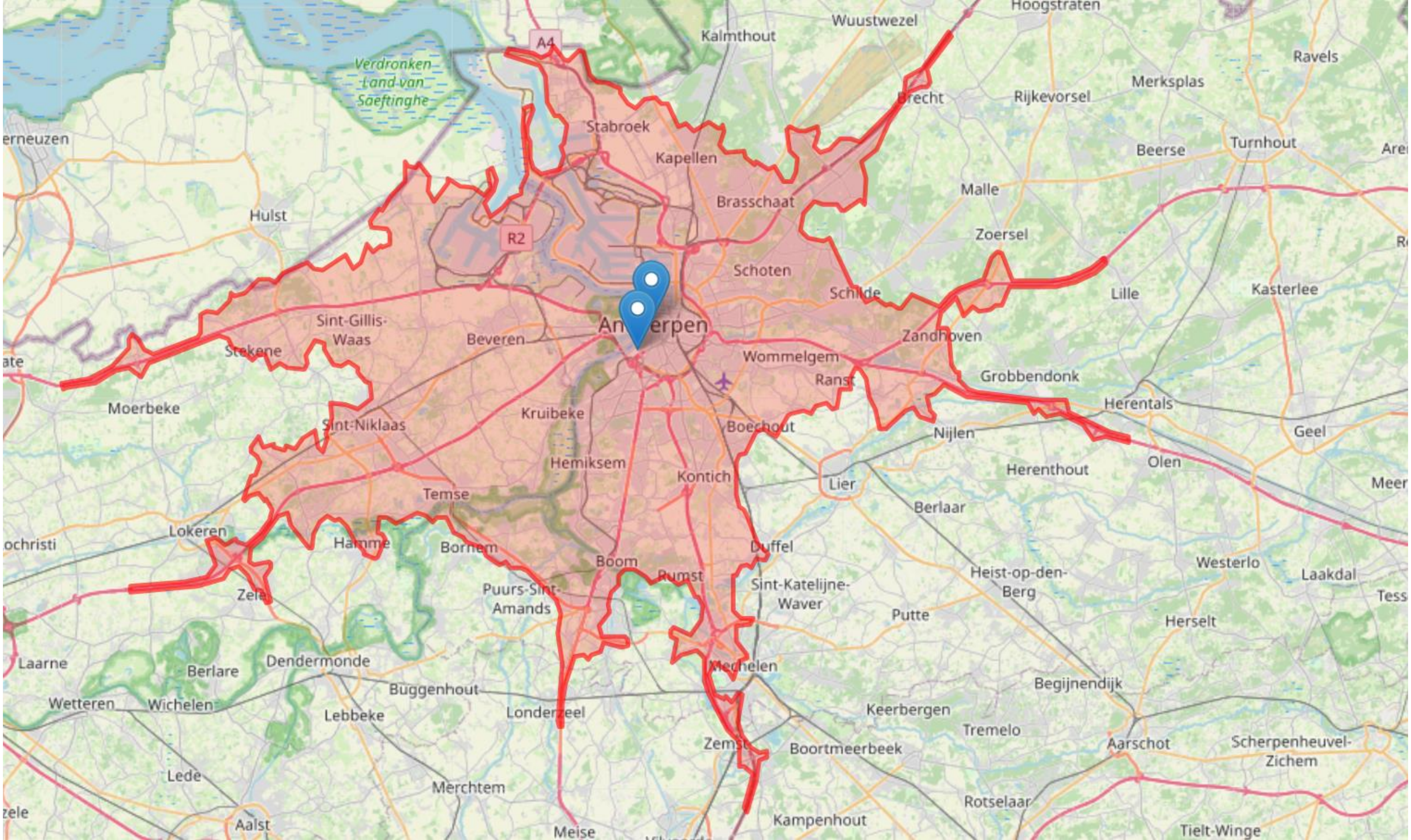
Instruction: Turn left onto Maantjessteenweg

Distance: 834.1 meters

Duration: 70.5 seconds

Isochrones

- Reachability has become a crucial component for many organizations from all different kinds of domains. Isochrones which will help you determine which areas objects are able reach in given times or distances.


Hoogstraten
Wuustwezel
Kalmthout
A4
Verdronken Land van Saeftinghe
Ravels
Merksplas
Brecht
Rijkevorsel
Stabroek
erneuzen
Kapellen
Beerse
Turnhout
Malle
Brasschaat
Hulst
R2
Zoersel
Schoten
Schilde
Lille
Kasterlee
Sint-Gillis-Waas
Beveren
Zandhoven
Stekene
Wommelgem
Grobbendonk
Ranst
Herentals
Moerbeke
Kruibeke
Sint-Niklaas
Boechout
Nijlen
Geel
Hemiksem
Herenthout
Olen
Kontich
Lier
Temse
Berlaar
Lokeren
Hamme
Bornem
Duffel
Boom
Westerlo
Rumst
Heist-op-den-Berg
Laakdal
Sint-Katelijne-Waver
Zele
Puurs-Sint-Amands
Putte
Herselt
Laarne
Dendermonde
Mechelen
Berlare
Begijnendijk
Buggenhout
Wetteren
Wichelen
Keerbergen
Lebbeke
Londerzeel
Tremelo
Zemst
Aarschot
Boortmeerbeek
Scherpenheuvel-Zichem
Lede
Merchtem
Rotselaar
Aalst
Kampenhout
Meise
Tielt-Winge

```
{
    "type": "FeatureCollection",
    "bbox": [
        3.865827,
        50.852759,
        4.985571,
        51.476335
    ],
    "features": [
        {
            "type": "Feature",
            "properties": {
                "group_index": 0,
                "value": 1800,
                "center": [
                    4.38441172415126,
                    51.206450821368946
                ]
            },
            "geometry": {
                "coordinates": [
                    [
                        [
                            3.867398,
                            51.184704
                        ],
                        [
                            3.871146,
                            51.185088
                        ],
                        [
                            3.87484,
                            51.185468
                        ],
                        [
```

POIs

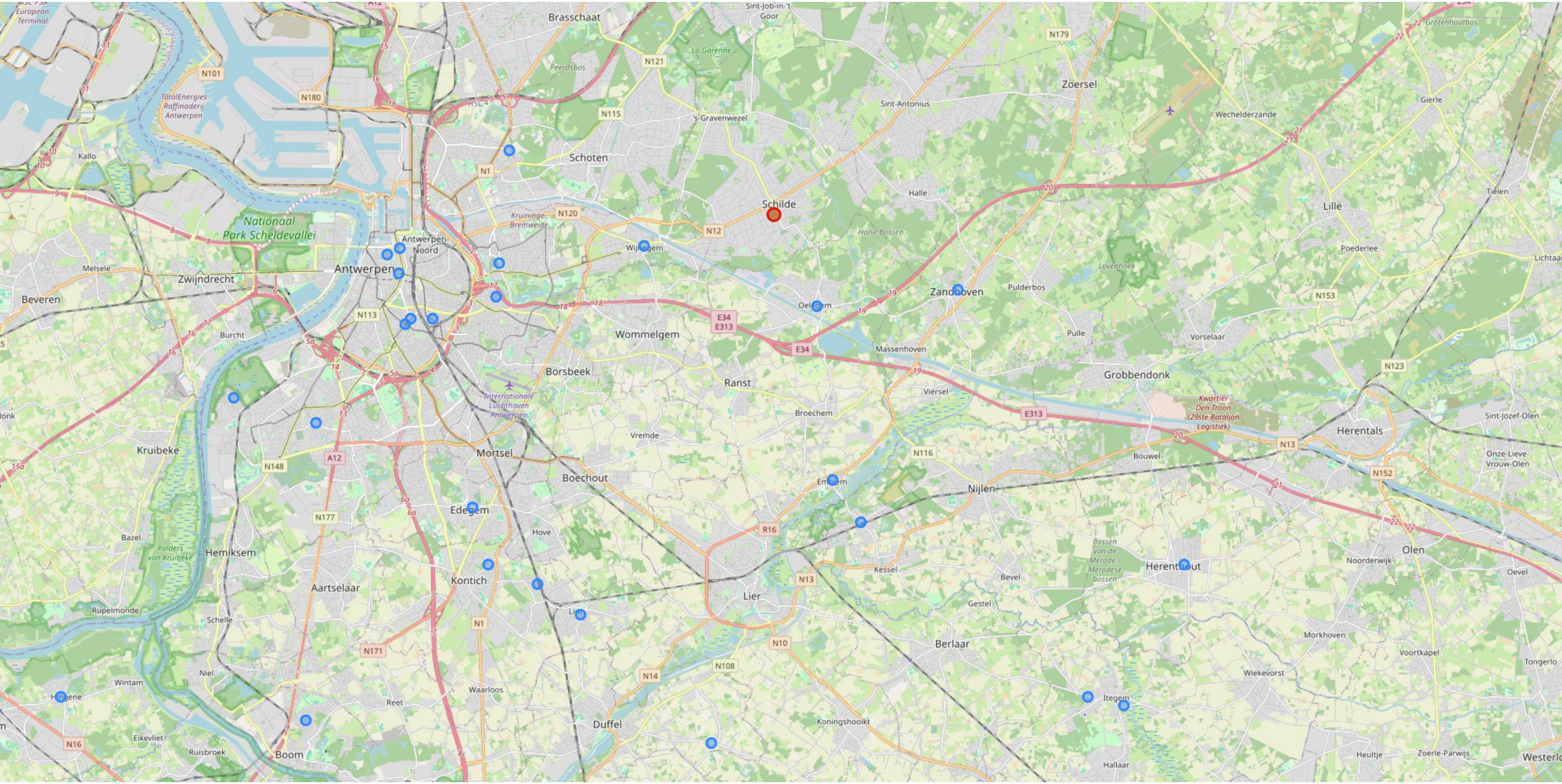
- You can search for categories of points of interest around a point, path or even within a given polygon and consume the rich meta information returned for your needs.

- https://github.com/GIScience/openpoiservice

```
</node>
<node id="10186704308" visible="true" version="2" changeset="133087338" timestamp="2023-02-27T13:15:51Z" user="cEvLGWiQ" uid="5432507" lat="51.2335440" lon="4.5615503">
 <tag k="addr:housenumber" v="104"/>
 <tag k="addr:street" v="Turnhoutsebaan"/>
 <tag k="branch" v="Schilde"/>
 <tag k="name" v="Delitraiteur"/>
 <tag k="opening_hours" v="Mo-Su,PH 07:00-22:00"/>
 <tag k="operator" v="Delitraiteur"/>
 <tag k="operator:wikidata" v="Q115222326"/>
 <tag k="phone" v="+32 3 361 92 59"/>
 <tag k="shop" v="deli"/>
</node>
<node id="10858959006" visible="true" version="1" changeset="135555205" timestamp="2023-04-30T22:40:28Z" user="pi11" uid="12066190" lat="51.2357383" lon="4.5557683">
 <tag k="denomination" v="catholic"/>
 <tag k="description" v="Verwerkt in toegangspoort Sint-Lutgardisschool. Kleine afgesloten kastje met Mariabeeldje."/>
 <tag k="historic" v="wayside_shrine"/>
 <tag k="religion" v="christian"/>
</node>
```


- Schools
- 278795007
- 278883149
- 280282697
- 280287384
- 280395082
- Apotheker H. Keersmaekers
- 291966325
- 291967367
- 292082330
- Apotheek Pharmamax
- Apotheek De Statie
- 305753470
- W. Keyenberg
- Ameloot
- Upharma
- Kathleen Verbeeck
- Apotheek Bossers
- Apotheek Van Daele
- 351493998
- Verhamme
- 355251206
- 355251301
- 362658945
- Gorissen
- Apotheek Keymoken
- Apotheek Vandeneede
- Alphega Apotheek Heikant


```
<node id="5609564707" visible="true" version="1" changeset="58851937" timestamp="2018-05-10T14:09:51Z" user="lodde1949" uid="138772" lat="51.2327681" lon="4.5600899"/>
<node id="5609564708" visible="true" version="1" changeset="58851937" timestamp="2018-05-10T14:09:51Z" user="lodde1949" uid="138772" lat="51.2327584" lon="4.5600181"/>
<node id="5609564709" visible="true" version="1" changeset="58851937" timestamp="2018-05-10T14:09:51Z" user="lodde1949" uid="138772" lat="51.2327271" lon="4.5600289"/>
<node id="5609564710" visible="true" version="1" changeset="58851937" timestamp="2018-05-10T14:09:51Z" user="lodde1949" uid="138772" lat="51.2327367" lon="4.5601007"/>
<node id="5609564711" visible="true" version="1" changeset="58851937" timestamp="2018-05-10T14:09:51Z" user="lodde1949" uid="138772" lat="51.2328206" lon="4.5601155"/>
<node id="5609564712" visible="true" version="1" changeset="58851937" timestamp="2018-05-10T14:09:51Z" user="lodde1949" uid="138772" lat="51.2328600" lon="4.5600880"/>
<node id="5609564713" visible="true" version="1" changeset="58851937" timestamp="2018-05-10T14:09:51Z" user="lodde1949" uid="138772" lat="51.2328748" lon="4.5601422"/>
<node id="5609564714" visible="true" version="1" changeset="58851937" timestamp="2018-05-10T14:09:51Z" user="lodde1949" uid="138772" lat="51.2328354" lon="4.5601697"/>
<node id="5609564715" visible="true" version="1" changeset="58851937" timestamp="2018-05-10T14:09:51Z" user="lodde1949" uid="138772" lat="51.2323147" lon="4.5592980"/>
<node id="5609564716" visible="true" version="1" changeset="58851937" timestamp="2018-05-10T14:09:51Z" user="lodde1949" uid="138772" lat="51.2325534" lon="4.5597041"/>
<node id="5609564717" visible="true" version="1" changeset="58851937" timestamp="2018-05-10T14:09:51Z" user="lodde1949" uid="138772" lat="51.2325472" lon="4.5596388"/>
<node id="5609564718" visible="true" version="1" changeset="58851937" timestamp="2018-05-10T14:09:51Z" user="lodde1949" uid="138772" lat="51.2328196" lon="4.5591388"/>
<node id="5609564719" visible="true" version="2" changeset="111877014" timestamp="2021-09-29T16:54:00Z" user="pi11" uid="12066190" lat="51.2326926" lon="4.5591749">
 <tag k="name" v="Schilde"/>
 <tag k="traffic_sign" v="city_limit"/>
 <tag k="traffic_sign:direction" v="forward"/>
</node>
<node id="5609564720" visible="true" version="1" changeset="58851937" timestamp="2018-05-10T14:09:51Z" user="lodde1949" uid="138772" lat="51.2322781" lon="4.5597524"/>
```


Brasschaat
Schoten
Schilde
Zoersel
Sint-Antonius
Halle
Wechelderzande
Gierle
Lille
Tielen
Poederlee
Lichtaart
Antwerpen
Antwerpen-Noord
Zwijndrecht
Beveren
Melsele
Kallo
Burcht
Nationaal Park Scheldevallei
Wijnegem
Wommelgem
Borsbeek
Ranst
Oelegem
Zandhoven
Pulderbos
Pulle
Massenhoven
Vorselaar
Grobbendonk
Viersel
Broechem
Vremde
Mortsel
Boechout
Edegem
Hove
Kontich
Lier
Emblem
Kessel
Nijlen
Bouwel
Herentals
Olen
Noorderwijk
Herenthout
Bevel
Gestel
Berlaar
Wiekevorst
Morkhoven
Itegem
Hallaar
Koningshooikt
Duffel
Waarloos
Reet
Boom
Niel
Schelle
Aartselaar
Hemiksem
Kruibeke
Bazel
Rupelmonde
Wintam
Eikevliet
Ruisbroek
Heultje
Zoerle-Parwijs
Voortkapel
Tongerlo
Oevel
Onze-Lieve-Vrouw-Olen
Sint-Jozef-Olen
's-Gravenwezel
Sint-Job-in-'t-Goor
Internationale Luchthaven Antwerpen
Kwartier Den Troon (29ste Bataljon Logistiek)
Halse Bossen
Lovenhoek
La Garenne
Peerdsbos
Kruiningen Bremweide
Polders van Kruibeke
Bossen van de Merode / Merodese bossen
TotalEnergies Raffinaderij Antwerpen
MSC PSA European Terminal
E34
E313
N12
N120
N121
N115
N1
N113
N148
N177
N171
N14
N108
N10
N13
N16
N116
N153
N123
N152
N179
N180
N101
A12
R16
Westerlo

```sql
SELECT
  NODEID,
  NODEVISIBLE,
  NODEVERSION,
  QSYS2.ST_ASTEXT(NODEPOINT)
FROM CDLIGHT.OSMNODES;
```

	VISIBLE	VERSION	
NODEID	NODEVISIBLE	NODEVERSION	00004
127977864	true	4	POINT (4.553736 51.231606)
127977871	true	4	POINT (4.567771 51.235670999999996)
127977873	true	10	POINT (4.574221 51.237519)
243670933	true	7	POINT (4.573567 51.234032)
246834665	true	5	POINT (4.556564 51.232423)
246834718	true	9	POINT (4.548624 51.23976)
246834719	true	11	POINT (4.551877 51.238606999999995)
246834721	true	5	POINT (4.552597 51.237164)
246834722	true	4	POINT (4.553826 51.234783)
246834723	true	7	POINT (4.554424 51.233844999999995)
246834724	true	2	POINT (4.554666 51.233756)
246834725	true	3	POINT (4.555461 51.233934999999995)
246834726	true	3	POINT (4.555664 51.233933)
246834727	true	2	POINT (4.5557739999999995 51.233914999999996)
246834728	true	2	POINT (4.556007 51.23379)

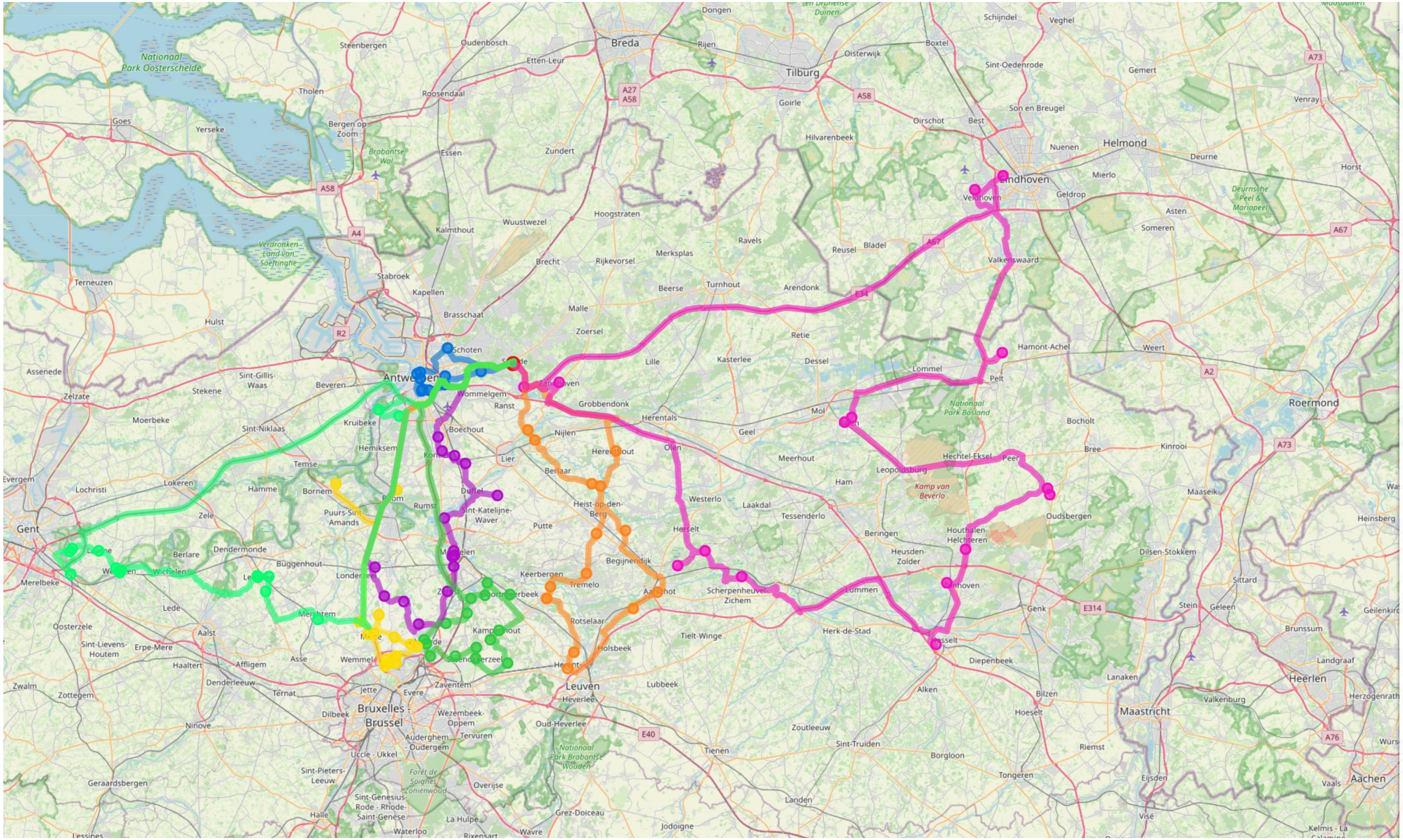
	KEY	VALUE
NODEID	TAGKEY	TAGVALUE
243670933	direction	both
243670933	surface	paving stones
243670933	traffic calming	table
246834665	highway	traffic signals
246834665	traffic signals	blink mode
246834665	traffic signals:direction	forward
246834719	network:type	node network
246834719	rcn ref	80
246834723	network:type	node network
246834723	rwn ref	48
246834833	direction	backward
246834833	highway	stop
255406352	direction	both
255406352	network:type	node network
255406352	rwn ref	57
255406352	surface	paving stones
255406352	traffic calming	table
255602843	highway	traffic signals
255602843	traffic signals:direction	backward
369988615	amenity	fuel
369988615	brand	Esso
369988615	brand:wikidata	Q867662
369988615	brand:wikipedia	en:Esso
369988615	check date	2022-09-01
369988615	compressed air	yes
369988615	name	Express
369988615	operator	Esso

Time-Distance Matrix

- Matrices allow you to compute many-to-many distances and the times of routes much faster than consuming the directions api over and over again. This calculation is frequently used by logistics organizations trying to figure out the most optimal route for deliveries.

Optimization

- Traveling Salesmen and other Vehicle Routing Problems are no problem for our optimization endpoint. Based on the excellent Vroom project this service provides you with optimal routes while considering your specific vehicle and time constraints.



Vehicle 1

Clone >>

- ⊗ 15, Nieuwstraat, Schilde
- ⊗ 15, Nieuwstraat, Schilde

Vehicle 2

Clone >>

- ⊗ 15, Nieuwstraat, Schilde
- ⊗ 15, Nieuwstraat, Schilde

Vehicle 3

Clone >>

- ⊗ 15, Nieuwstraat, Schilde
- ⊗ 15, Nieuwstraat, Schilde

Vehicle 4

Clone >>

- ⊗ 15, Nieuwstraat, Schilde
- ⊗ 15, Nieuwstraat, Schilde

Vehicle 5

Clone >>

- ⊗ 15, Nieuwstraat, Schilde
- ⊗ 15, Nieuwstraat, Schilde

Vehicle 6

Clone >>

- ⊗ 15, Nieuwstraat, Schilde
- ⊗ 15, Nieuwstraat, Schilde

Vehicle 7

Clone >>

- ⊗ 15, Nieuwstraat, Schilde
- ⊗ 15, Nieuwstraat, Schilde

Vehicle 8

Clone >>

- ⊗ 15, Nieuwstraat, Schilde
- ⊗ 15, Nieuwstraat, Schilde

Add locations

- by clicking on the map

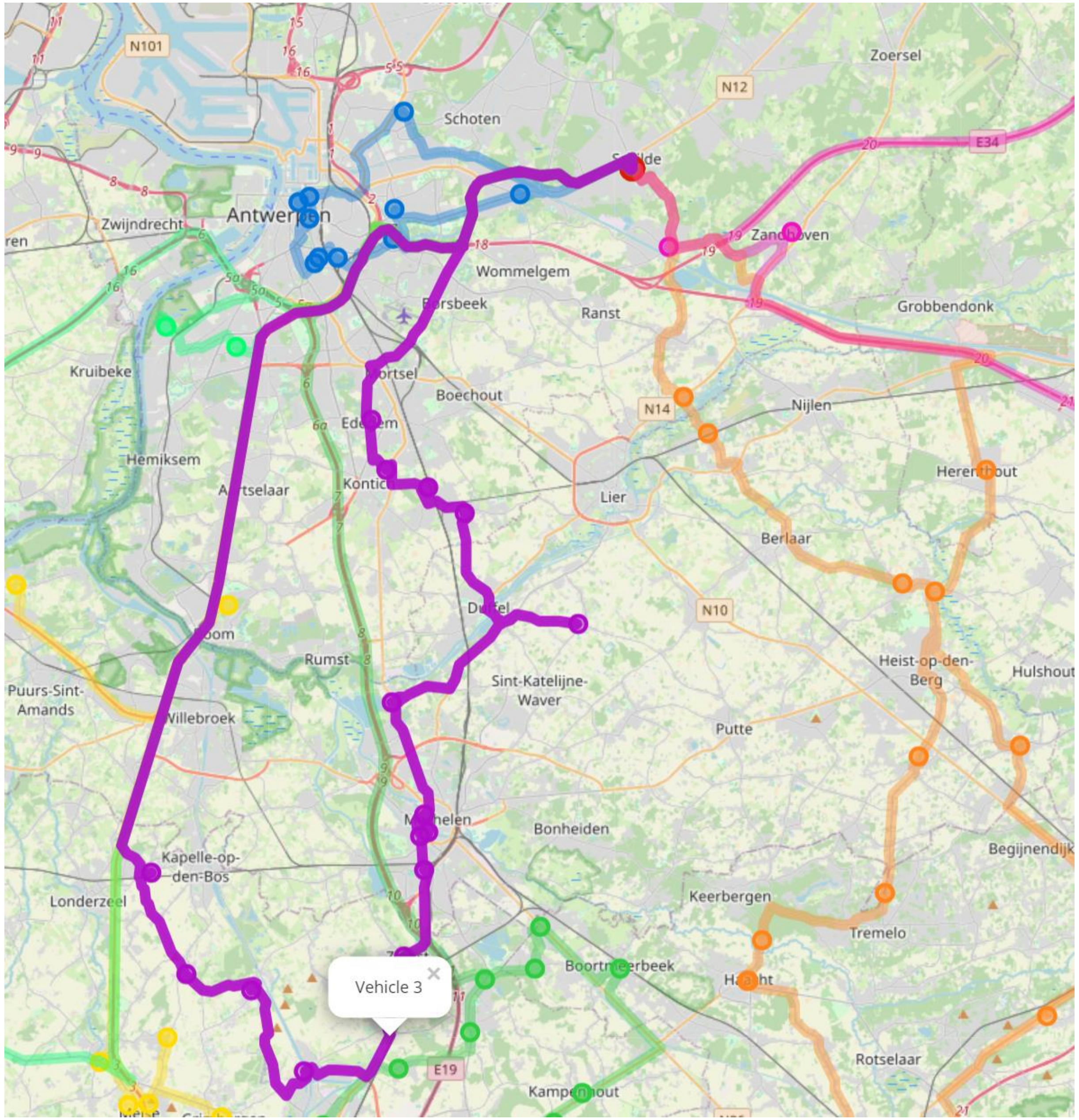
- using OpenStreetMap tag

amenity | pharmacy | More values for amenity.

Add

Vehicle 1

1 Apotheek Meeussen
2 De Beul
3 Ter Rivieren
4 Zwitserse Apotheek
5 Apotheek Bek
6 Kruispunt Apotheek

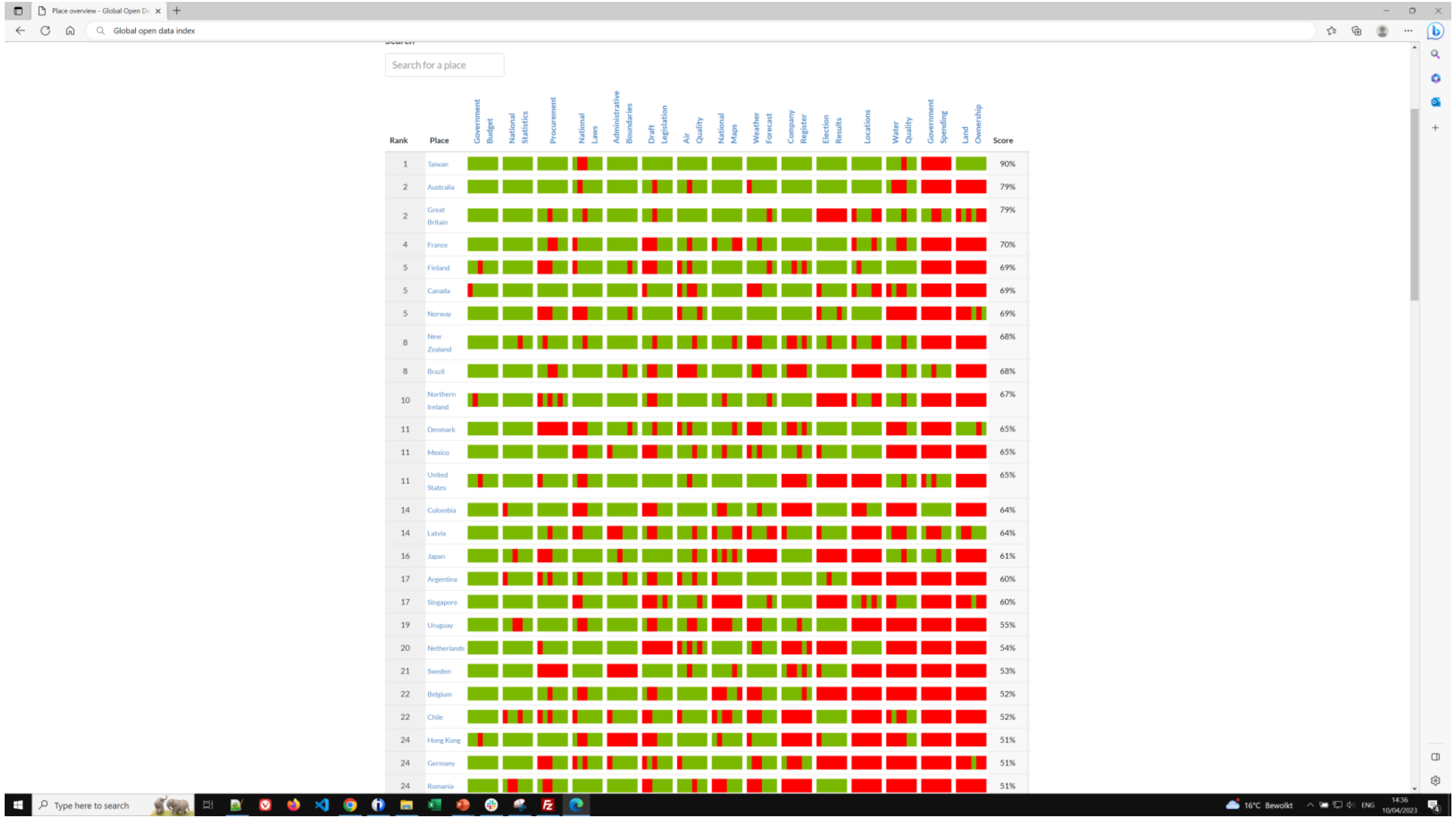


Vehicle 3
Schoten
Zoersel
Antwerpen
Zwijndrecht
Wommelgem
Borsbeek
Ranst
Zandhoven
Grobbendonk
Kruibeke
Boechout
Nijlen
Hemiksem
Lier
Herenthout
Berlaar
Boom
Rumst
Sint-Katelijne-Waver
Heist-op-den-Berg
Hulshout
Puurs-Sint-Amands
Willebroek
Putte
Bonheiden
Begijnendijk
Kapelle-op-den-Bos
Londerzeel
Keerbergen
Tremelo
Rotselaar
Kampenhout
N101
N12
E34
N14
N10
E19

Next Steps

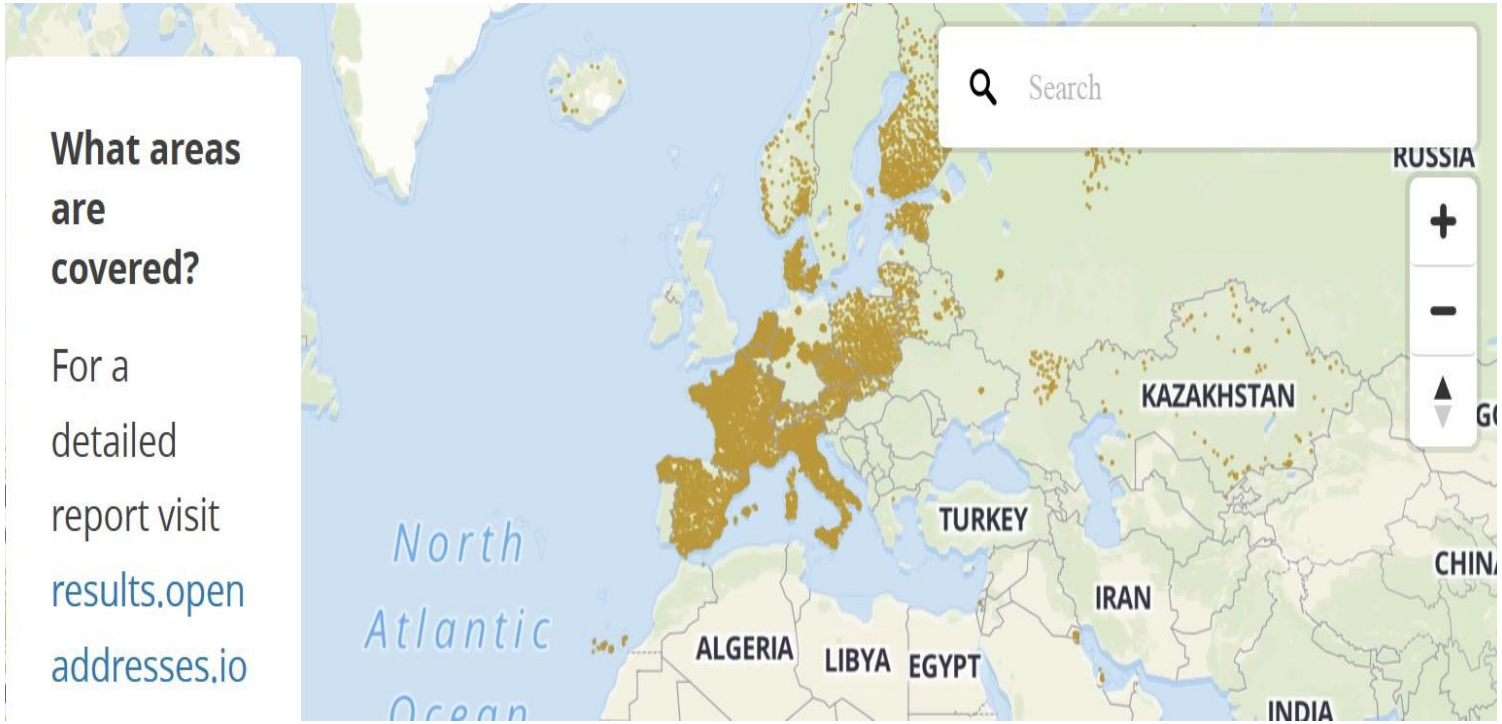
Global open data index

[Place overview - Global Open Data Index (okfn.org)]

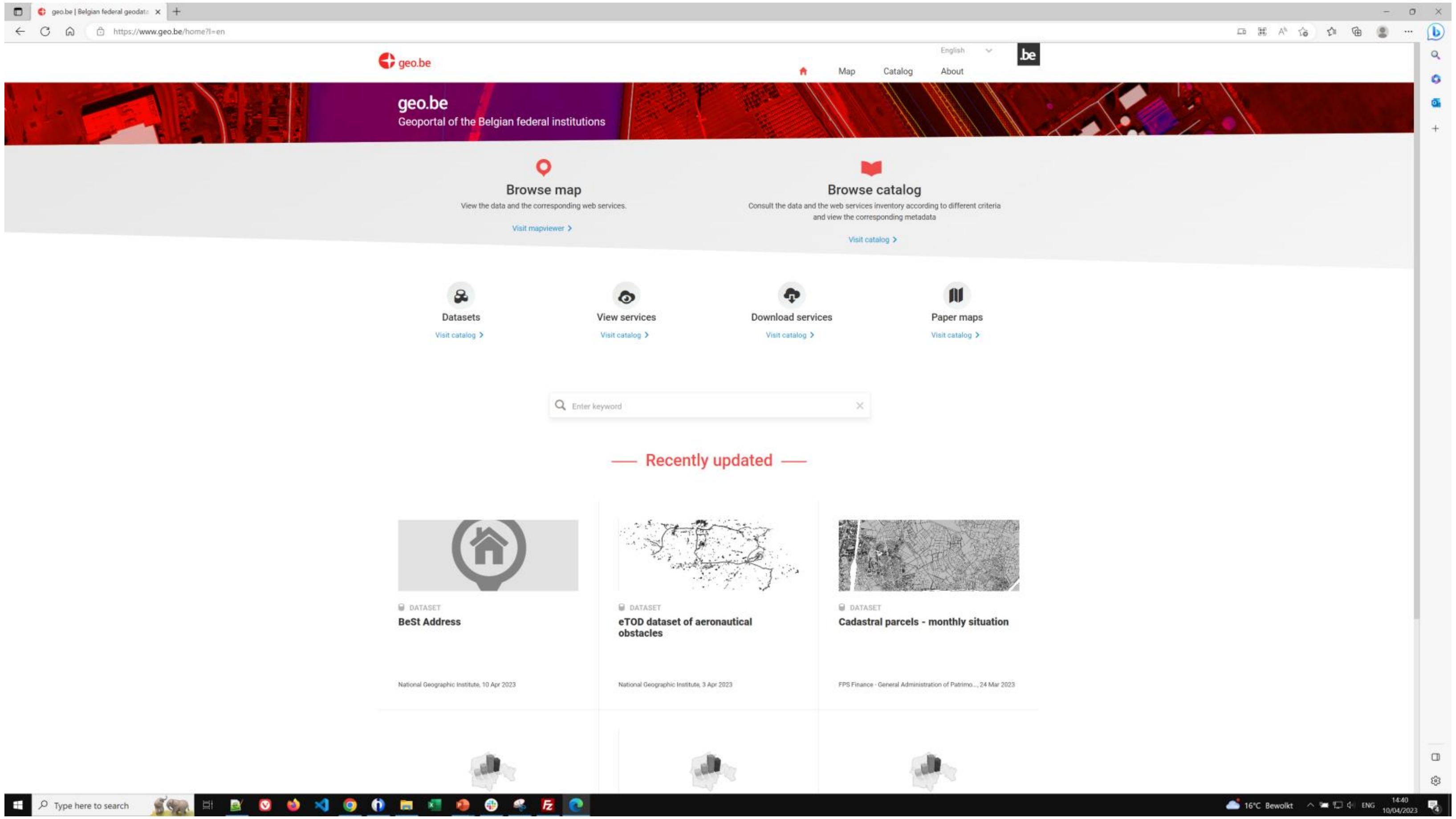


Addressess

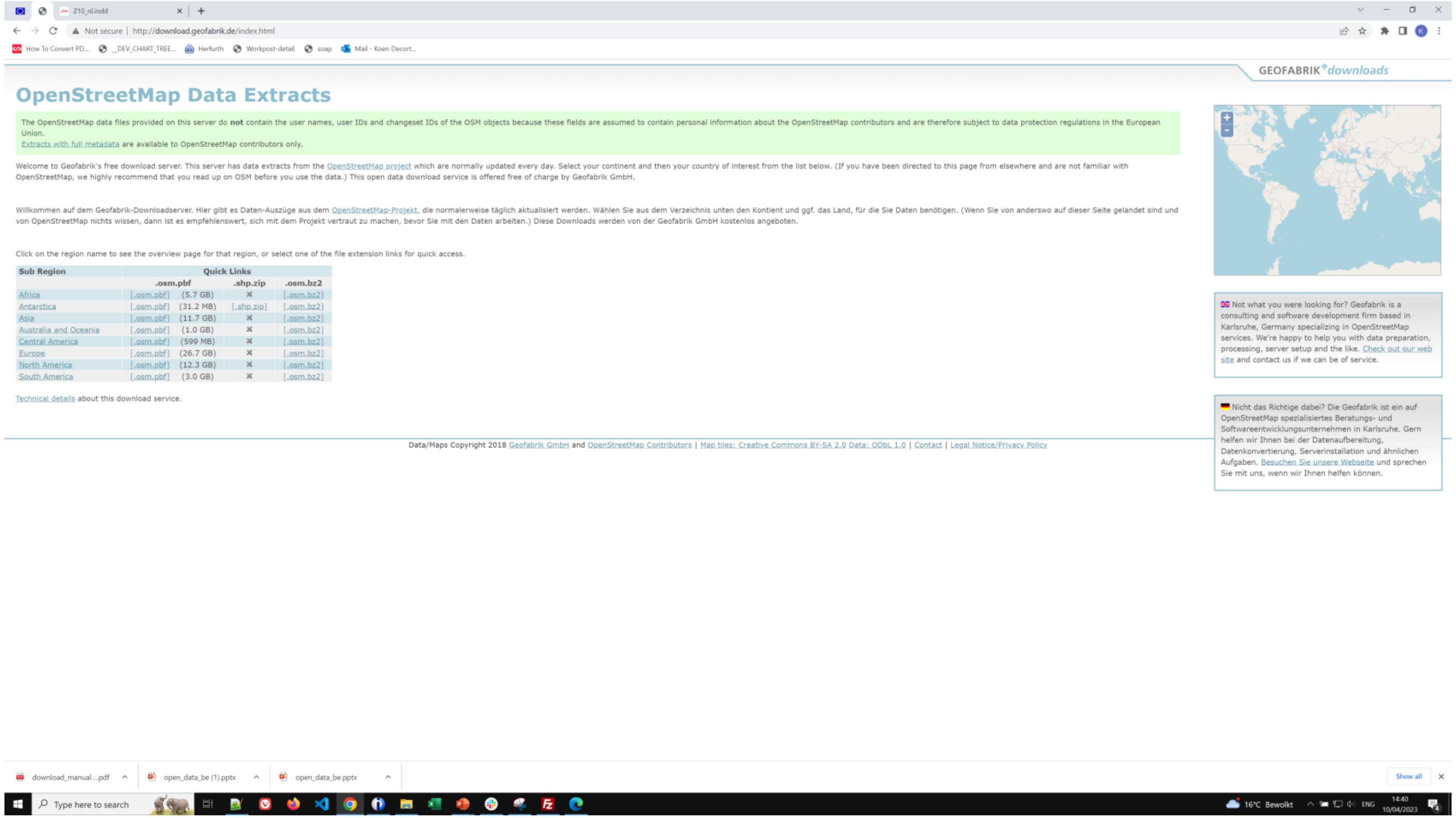
Openaddresses: https://openaddresses.io/



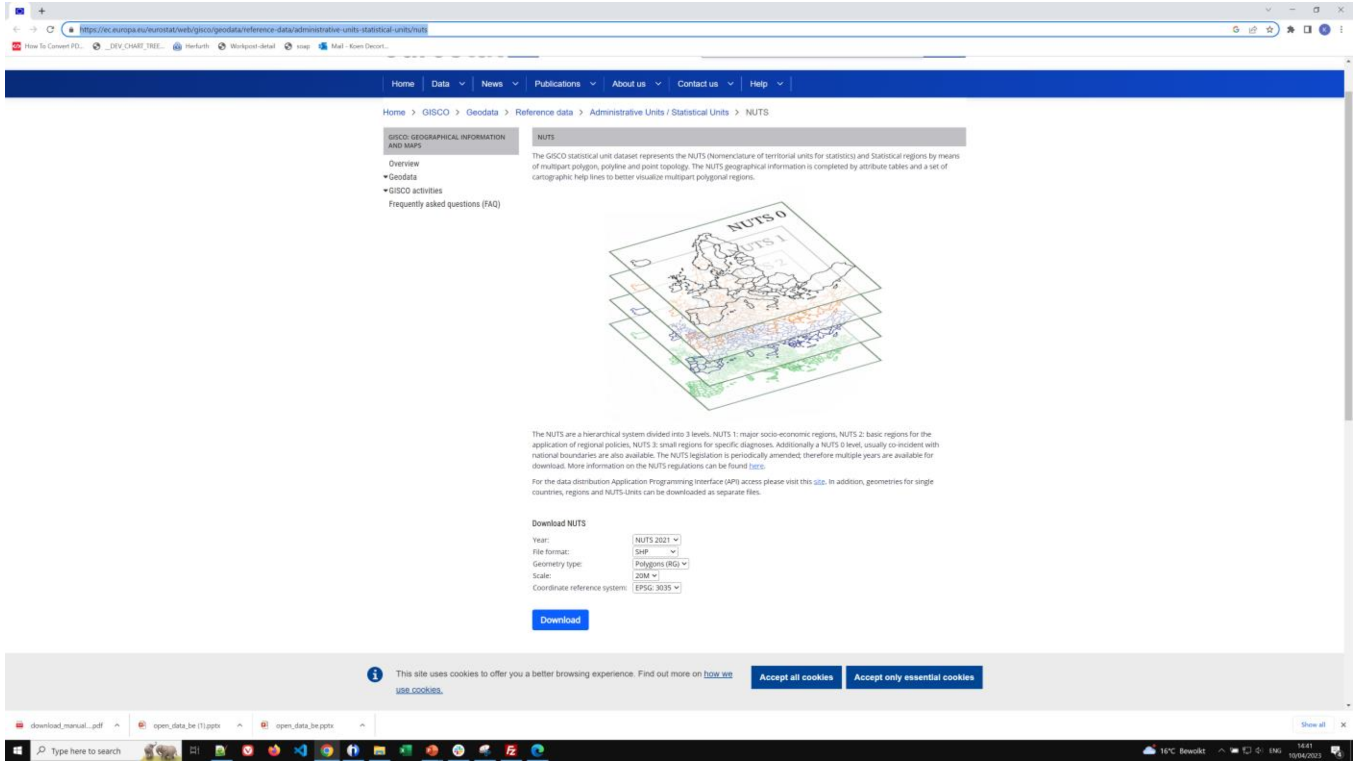
geo.be | Belgian federal geodata portal



http://download.geofabrik.de/index.html



https://ec.europa.eu/eurostat/web/gisco/geodata/reference-data/administrative-units-statistical-units/nuts



http://www.eurogeographics.org/products-and-services/euroglobalmap

GeoExt

- https://geoext.github.io/geoext/
- https://github.com/geoext
- https://geoext.org/

GeoExt is Open Source and enables building desktop-like GIS applications through the web. It is a JavaScript framework that combines the GIS functionality of OpenLayers with the user interface of the ExtJS library provided by Sencha.

GeoExt 3 — JavaScript Toolkit for Rich Web Mapping Applications

GeoExt is Open Source and enables building desktop-like GIS applications through the web. It is a JavaScript framework that combines the GIS functionality of OpenLayers with the user interface of the ExtJS library provided by Sencha.

Version 3 of GeoExt is the successor to the GeoExt 2-series and is built atop the following official installments of it's base libraries; OpenLayers 4.6.5 and ExtJS 6.2.0. The versions of GeoExt that support these libraries are version 3.3.2 and the older version 3.2.0.

GeoExt is an OSGeo Community project - a member of the Open Source Geospatial Foundation (OSGeo) family of projects. Everybody is invited to help us create the next version of GeoExt.

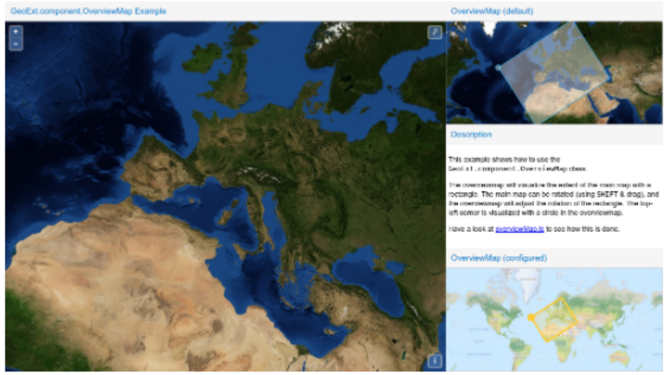
About GeoExt

Since version 3, GeoExt is based upon Ext JS 6.2.0.

This means GeoExt can be used just like any other Ext component, and applications making use of GeoExt also profit from Ext JS enhancements like charting, a harmonized API with Sencha Touch and a sophisticated single-file build tool.

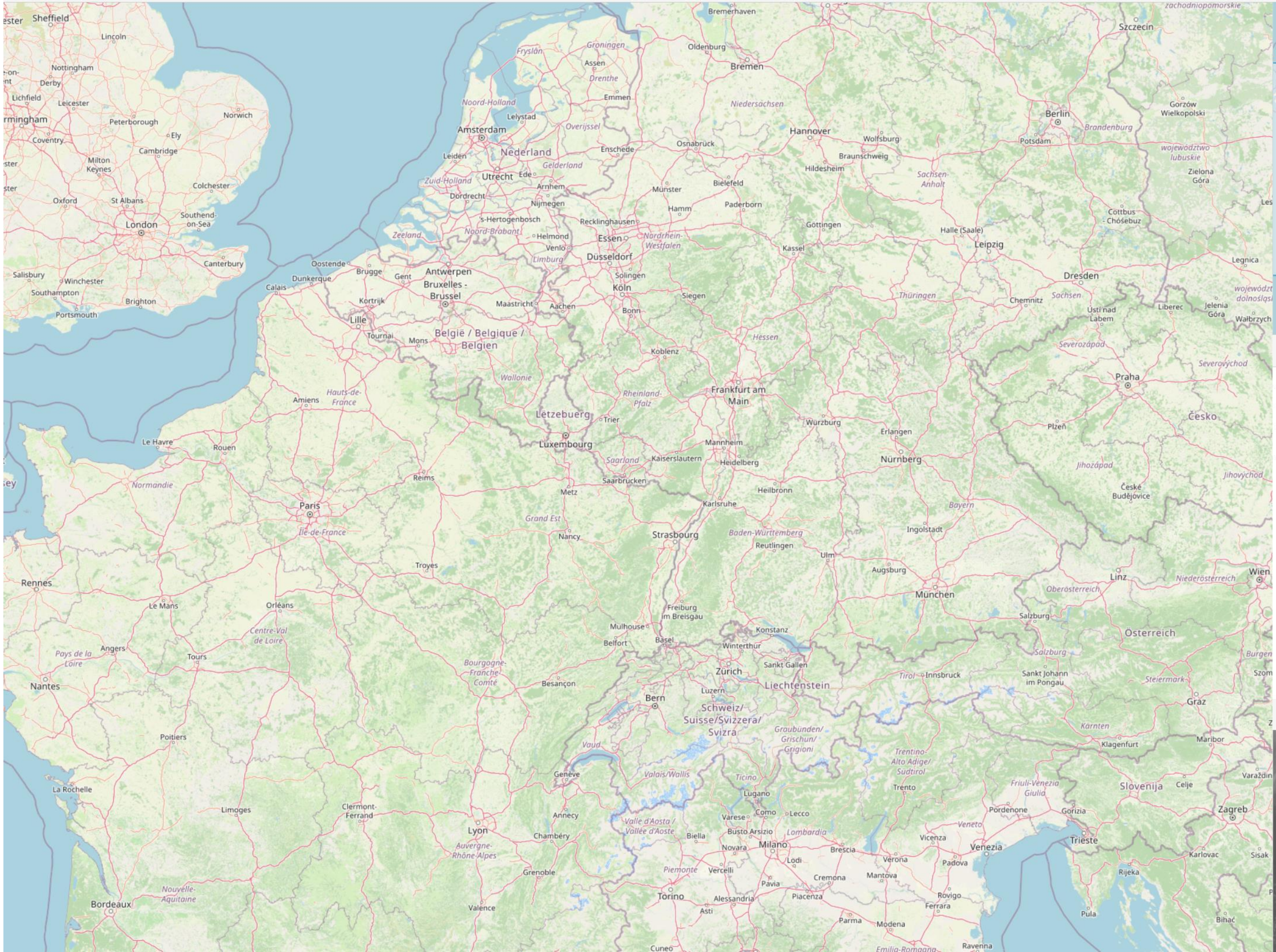
GeoExt 3 is a rather young project, a lot of the code and structural decisions come from a code sprint in Bonn. 9 developers gathered there from 2015-06-17 to 2015-06-19. We are deeply grateful that our sponsors helped to start GeoExt 3.

Now, have a look at the examples below, read the API documentation, the API documentation (including ExtJS classes) or checkout the code

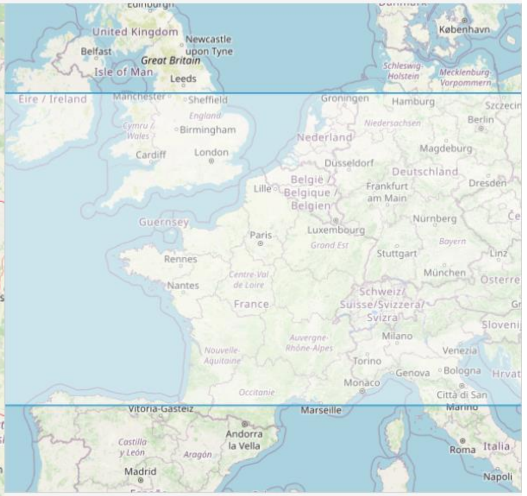


v3.3.2	API documentation	API documentation (including ExtJS classes)
v3.2.0	API documentation	API documentation (including ExtJS classes)
v3.1.0	API documentation	API documentation (including ExtJS classes)
v3.0.0	API documentation	API documentation (including ExtJS classes)
master	API documentation	API documentation (including ExtJS classes)

OSGeo Community



OverviewMap (default)



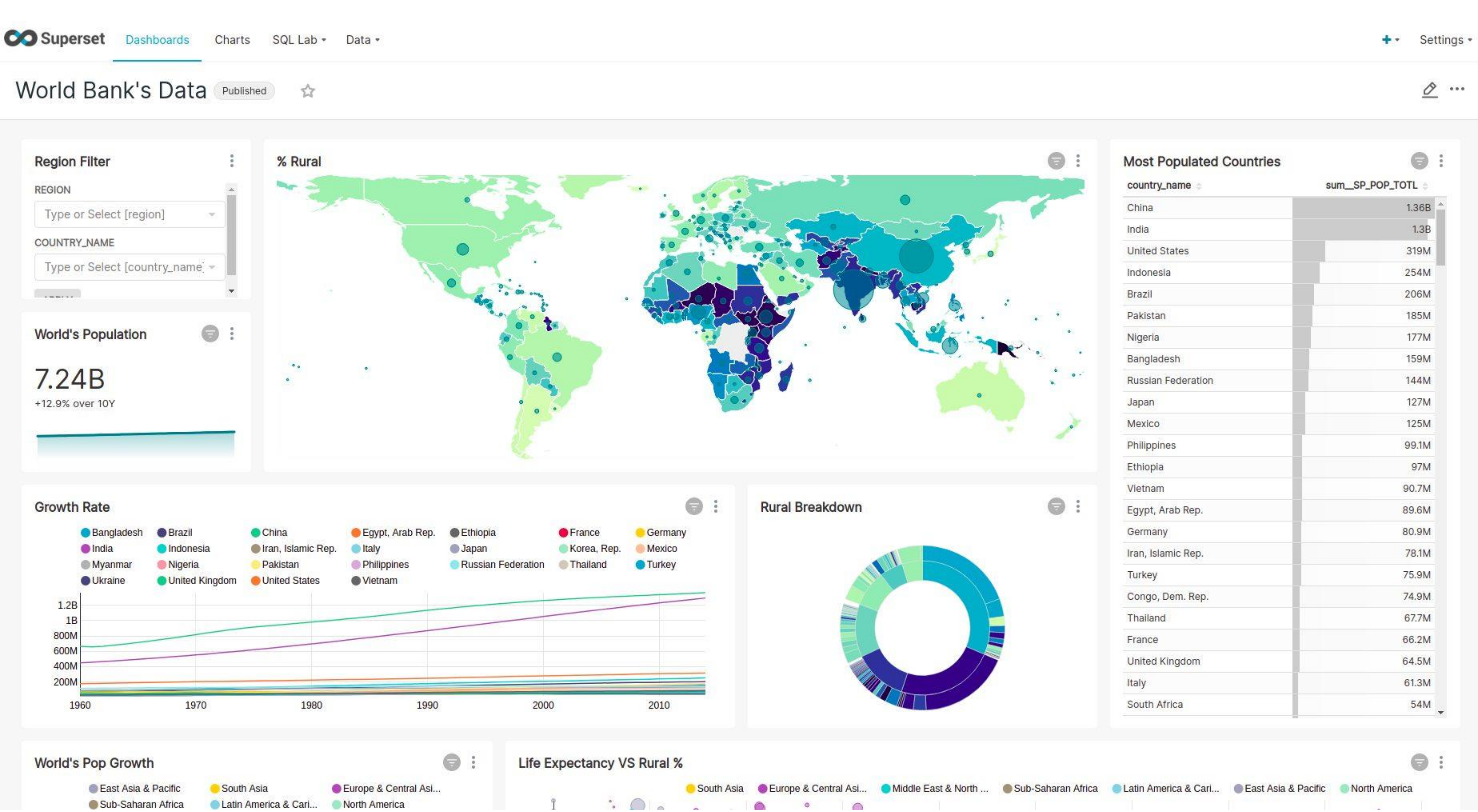
Description

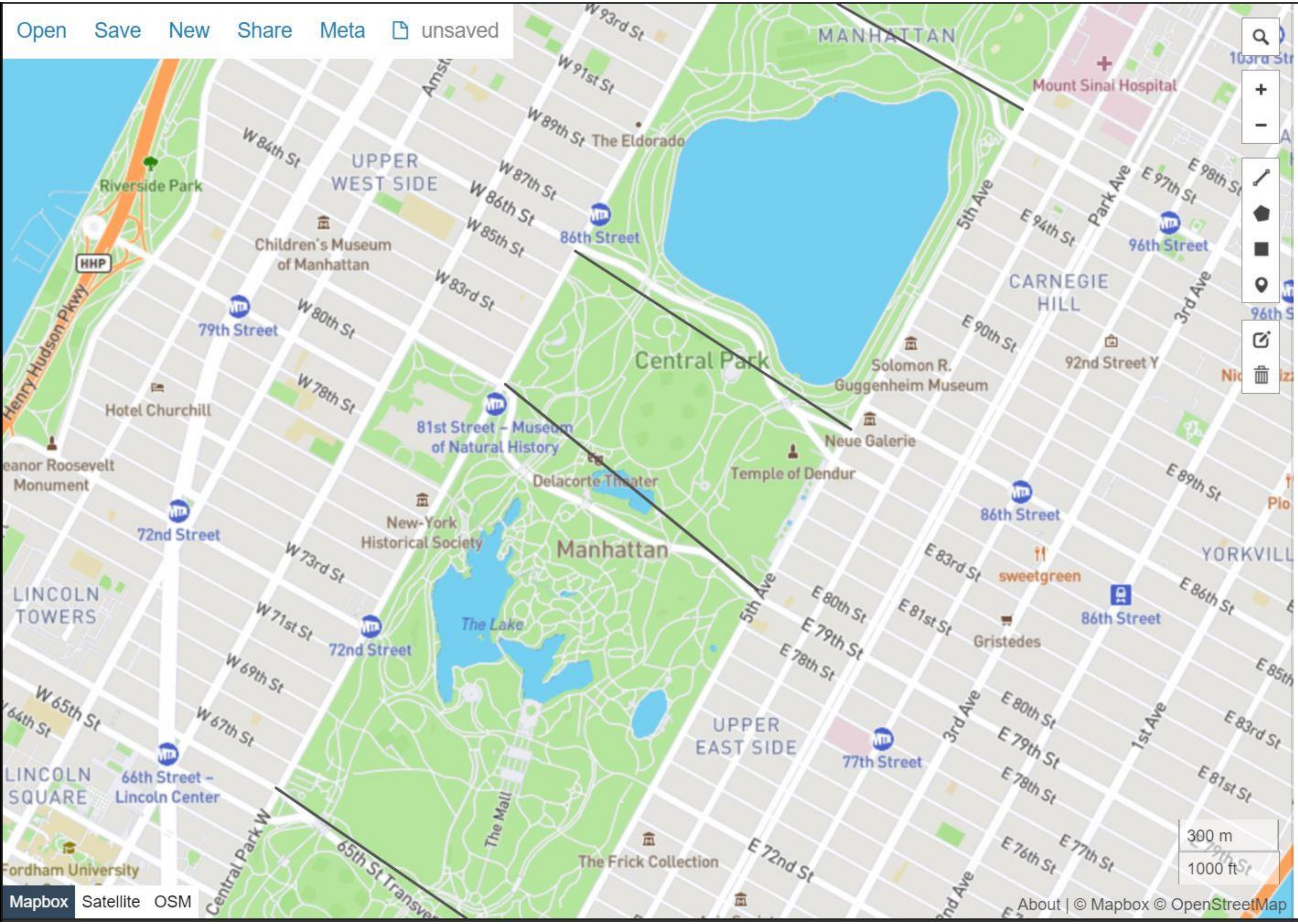
This example shows how to use the `GeoExt.component.OverviewMap` class.

The overviewmap will visualize the extent of the main map with a rectangle. The main map can be rotated (using `SHIFT` & drag), and the overviewmap will adjust the rotation of the rectangle. The top-left corner is visualized with a circle in the overviewmap.

Have a look at [overviewMap.js] to see how this is done.

OverviewMap (configured)


Superset
Dashboards
Charts
SQL Lab
Data
Settings
World Bank's Data
Published
Region Filter
REGION
Type or Select [region]
COUNTRY_NAME
Type or Select [country_name]
% Rural
Most Populated Countries
country_name
sum__SP_POP_TOTL
China 1.36B
India 1.3B
United States 319M
Indonesia 254M
Brazil 206M
Pakistan 185M
Nigeria 177M
Bangladesh 159M
Russian Federation 144M
Japan 127M
Mexico 125M
Philippines 99.1M
Ethiopia 97M
Vietnam 90.7M
Egypt, Arab Rep. 89.6M
Germany 80.9M
Iran, Islamic Rep. 78.1M
Turkey 75.9M
Congo, Dem. Rep. 74.9M
Thailand 67.7M
France 66.2M
United Kingdom 64.5M
Italy 61.3M
South Africa 54M
World's Population
7.24B
+12.9% over 10Y
Growth Rate
Bangladesh
Brazil
China
Egypt, Arab Rep.
Ethiopia
France
Germany
India
Indonesia
Iran, Islamic Rep.
Italy
Japan
Korea, Rep.
Mexico
Myanmar
Nigeria
Pakistan
Philippines
Russian Federation
Thailand
Turkey
Ukraine
United Kingdom
United States
Vietnam
1.2B
1B
800M
600M
400M
200M
1960
1970
1980
1990
2000
2010
Rural Breakdown
World's Pop Growth
East Asia & Pacific
South Asia
Europe & Central Asi...
Sub-Saharan Africa
Latin America & Cari...
North America
Life Expectancy VS Rural %
South Asia
Europe & Central Asi...
Middle East & North ...
Sub-Saharan Africa
Latin America & Cari...
East Asia & Pacific
North America



JSON Table ? Help

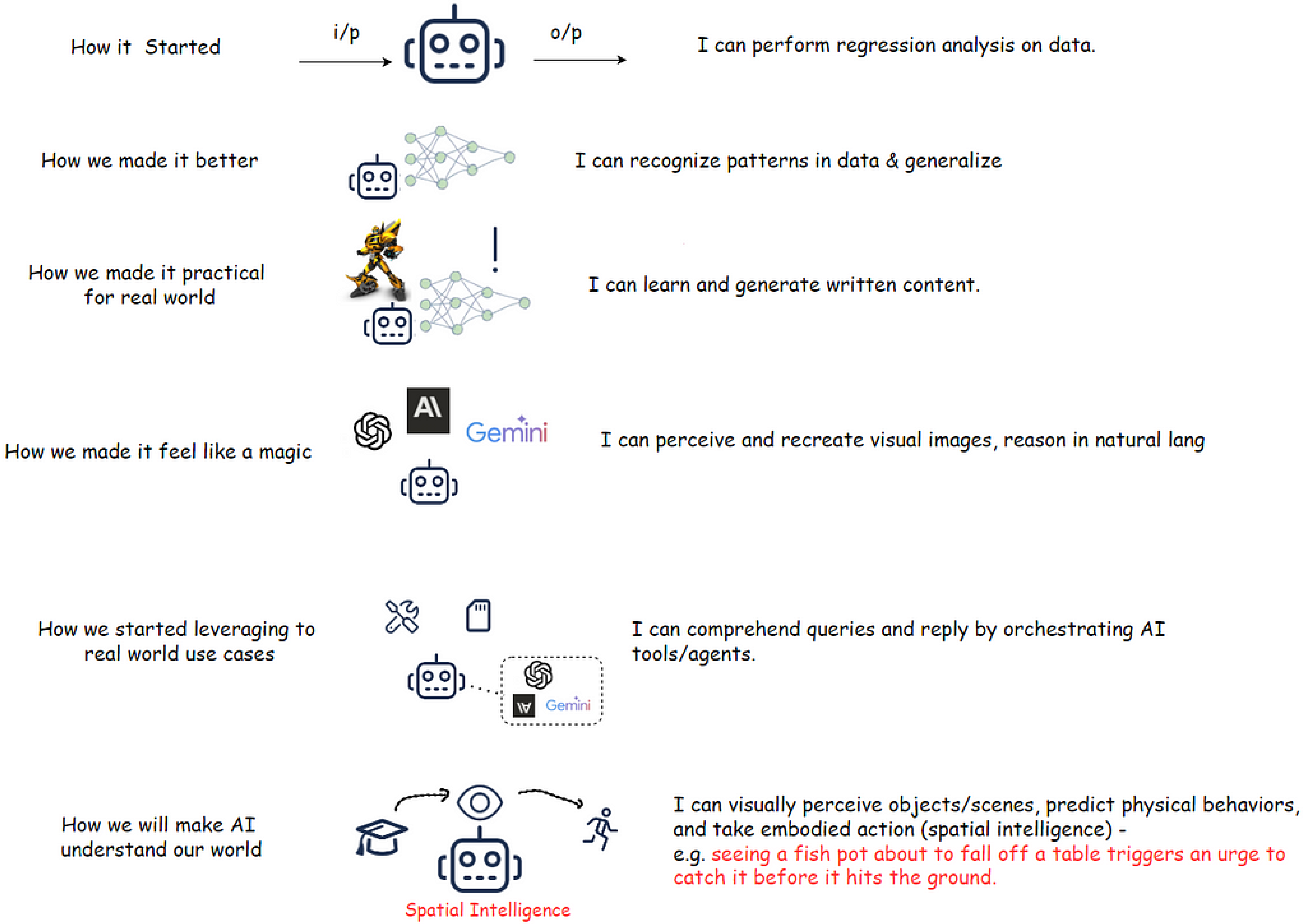

```
{
  "type": "MultiLineString",
  "coordinates": [
    [
      [
        -73.96943,
        40.78519
      ],
      [
        -73.96082,
        40.78095
      ]
    ],
    [
      [
        -73.96415,
        40.79229
      ],
      [
        -73.95544,
        40.78854
      ]
    ],
    [
      [
        -73.97162,
        40.78205
      ],
```

Some other references

- https://github.com/alombarte/utilities/blob/master/sql/NUTS.sql
- https://ec.europa.eu/eurostat/web/gisco/geodata/reference-data/administrative-units-statistical-units/nuts
- https://overheid.vlaanderen.be/crab-de-crab-databank
- https://www.vlaanderen.be/digitaal-vlaanderen/onze-oplossingen/centraal-referentieadressenbestand-crab
- https://ec.europa.eu/eurostat/web/main/data/database

Applications

- Geopandas - https://geopandas.org/en/stable/
- Superset – https://superset.apache.org/
- PostGis – https://postgis.net/
- Carto - https://carto.com/
- …..



How it Started
i/p
o/p
I can perform regression analysis on data.
How we made it better
I can recognize patterns in data & generalize
How we made it practical for real world
I can learn and generate written content.
How we made it feel like a magic
Gemini
I can perceive and recreate visual images, reason in natural lang
How we started leveraging to real world use cases
Gemini
I can comprehend queries and reply by orchestrating AI tools/agents.
How we will make AI understand our world
Spatial Intelligence
I can visually perceive objects/scenes, predict physical behaviors, and take embodied action (spatial intelligence) -
e.g. seeing a fish pot about to fall off a table triggers an urge to catch it before it hits the ground.

Conclusions

- Geospatial analytics spatially enables DB2 for i by adding spatial objects, functions, ….
- Geospatial analytics is freely included with DB2 for I
- Geospatial analytics is an important building block for open source spatial projects.

Thank you for your attention

Appendix 1

Creating a table with a geospatial column

```
CREATE TABLE customer
   (id INT,
    lastname     VARCHAR(30),
    firstname    VARCHAR(30),
    address      VARCHAR(100),
    city         VARCHAR(50),
    postal_code VARCHAR(5),
    state        CHAR(2),
    geopoint     QSYS2.ST_POINT);
```

ID	LASTNAME	FIRSTNAME	ADDRESS	CITY	POSTAL_CODE	STATE	GEOPOINT
101	Kriner	Endela	9 Concourt Circle	San Jose	95141	CA	<Spatial Data BLOB>

QSYS2-based Geospatial Functions

Constructor functions

- ST_Geometry
- ST_Point
- ST_LineString
- ST_Polygon
- ST_GeomCollection
- ST_MultiPoint
- ST_MultiLinestring
- ST_MultiPolygon
- ST_WKTToSQL
- ST_WKBToSQL

Geometric Properties

- ST_Area
- ST_GeometryType
- ST_IsSimple
- ST_IsValid
- ST_MaxX
- ST_MaxY
- ST_MinX
- ST_MinY
- ST_SrsID
- ST_SrsName

Hash a geometry

- ST_FuzzyGeohashCover
- ST_ FuzzyGeohashCoverExtend
- ST_Geohash
- ST_GeohashCover
- ST_GeohashCoverExtend

QSYS2-based Geospatial Functions

Let's Create

Comparing Geometries
- ST_Contains
- ST_Covers
- ST_Crosses
- ST_Difference
- ST_Disjoint
- ST_Distance
- ST_Equals
- ST_Intersects
- ST_Overlaps
- ST_Touches
- ST_Within

Construct a new geometry
- ST_Buffer
- ST_Difference
- ST_Intersection
- ST_SymDifference
- ST_Union

Converting Geometries
- ST_AsText
- ST_AsBinary
- ST_ToPoint
- ST_ToLineString
- ST_ToPolygon
- ST_ToMultiPoint
- ST_ToMultiLine
- ST_ToMultiPolygon

Converting Geometries

- Convert a geometry of the ST_Geometry type or one of its subtypes into a data exchange format

Function Name	Function Use
ST_AsText	Convert a geometry into a WKT object
ST_AsBinary	Convert a geometry into a WKB object

```
-- Convert geometry data back into WKT.

CREATE TABLE sample_points (id INT, geom QSYS2.ST_Point);
INSERT INTO sample_points VALUES
  (100, QSYS2.ST_Point('point (-92.503 44.058)'));

SELECT QSYS2.ST_AsText(geom) AS points FROM sample_points;
```

POINTS
POINT (-92.503 44.058)

Constructing Geometries

- Modify properties of a geometry of type ST_Geometry or one of its subtypes to construct a new geometry

Function Name	Function Use
ST_Buffer ST_Difference ST_Intersection ST_SymDifference	Create new geometries with different space configurations
ST_Union	Create a new geometry by combining multiple geometries

```
-- Update the sales area for a store.
-- Sales area is 10,000 meter buffer around the location.

UPDATE stores
  SET sales_area = QSYS2.ST_Buffer(location, 10000)
  WHERE id = 10
```

Comparing Geometries

IBM | Let's Create

- Return information that is the result of a comparison between geometries
  - How geometries relate to one another or compare with one another

Function Name	Function Use
ST_Equals	Check whether two geometries are identical
ST_Distance	Determine the distance between geometries
ST_Crosses ST_Disjoint ST_Intersects ST_Overlaps ST_Touches	Determine whether geometries intersect
ST_Contains ST_Within	Determine whether a geometry contains another one

```
-- Find all the residences that are within a sales area
SELECT c.first_name, c.last_name, QSYS2.ST_Within(c.location, s.sales_area)
  FROM customers as c, stores AS s
  WHERE s.id = 10
```

Geometric Properties

- Return information about geometric properties such as coordinates, measures, and boundaries.

Function Name	Function Use
ST_GeometryType	Return information about geometry types
ST_Area	Return information about geometry dimensions
ST_IsValid ST_MaxX ST_MaxY ST_MinX ST_MinY	Return information about coordinates and measures
ST_IsSimple	Return information to indicate whether a geometry is simple

```
-- Returns a numeric value that represents the sales area of store 10.
SELECT QSYS2.ST_Area(sales_area)
  FROM stores
  WHERE id = 10;
```