Sebastian Ernst, PhD

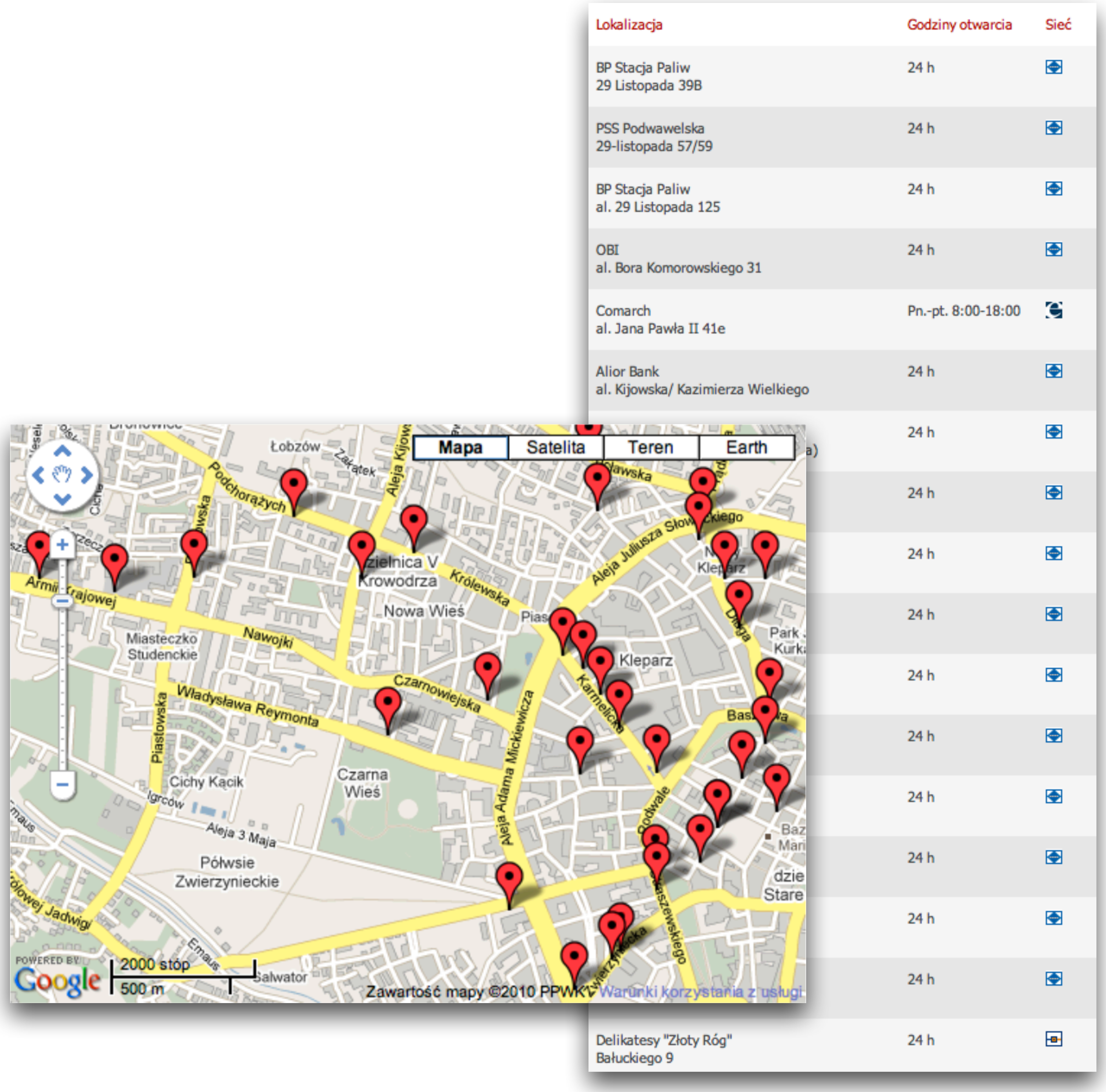
GIS & Co.

Processing Spatial Data in Databases and Beyond

ATM database

What we need to store

- ATM name,
- opening hours,
- location

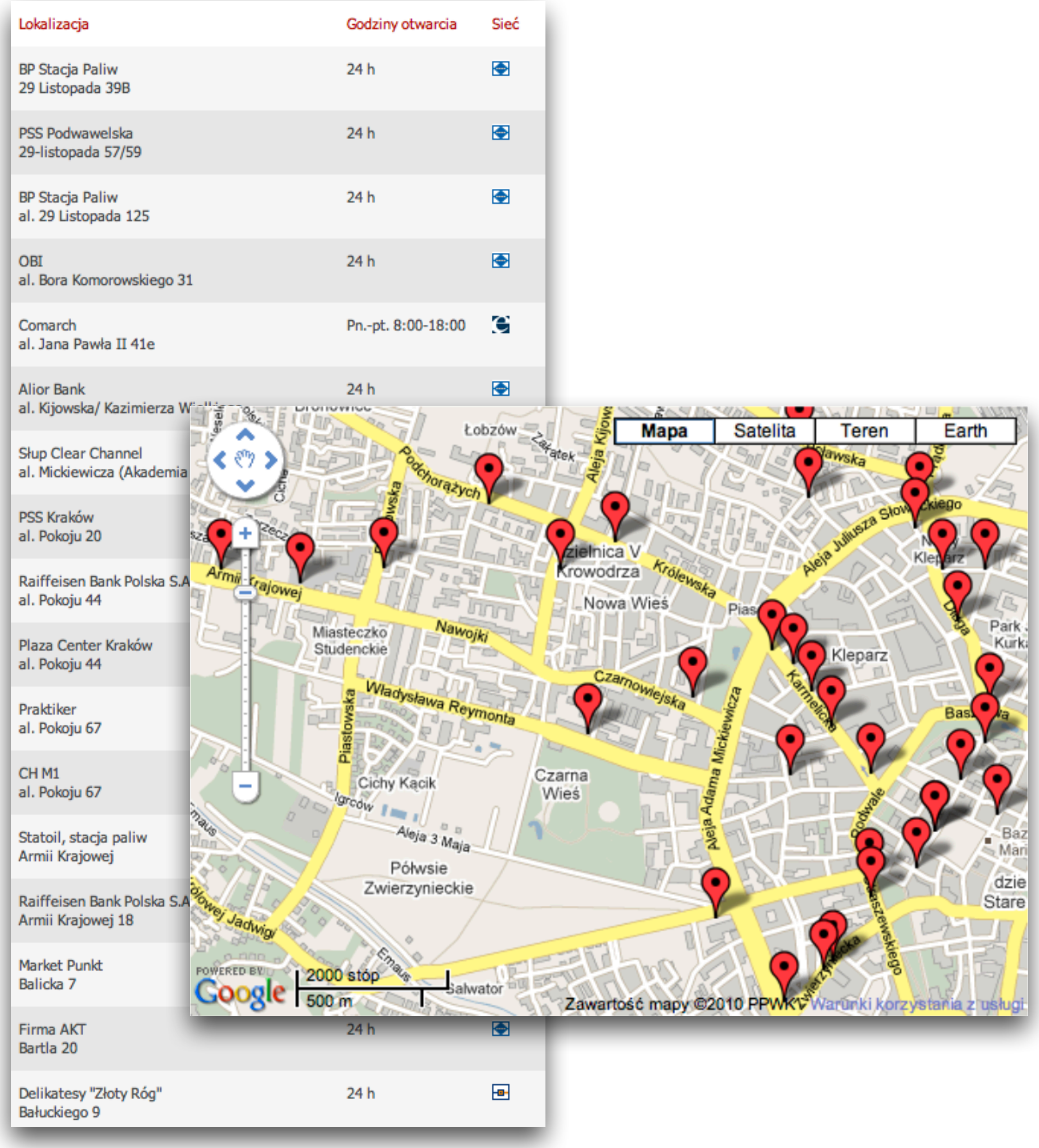


Lokalizacja	Godziny otwarcia	Sieć
BP Stacja Paliw 29 Listopada 39B	24 h	◈
PSS Podwawelska 29-listopada 57/59	24 h	◈
BP Stacja Paliw al. 29 Listopada 125	24 h	◈
OBI al. Bora Komorowskiego 31	24 h	◈
Comarch al. Jana Pawła II 41e	Pn.-pt. 8:00-18:00	◐
Alior Bank al. Kijowska/ Kazimierza Wielkiego	24 h	◈
a)	24 h	◈
	24 h	◈
	24 h	◈
	24 h	◈
	24 h	◈
	24 h	◈
	24 h	◈
	24 h	◈
	24 h	◈
	24 h	◈
Delikatesy "Złoty Róg" Bałuckiego 9	24 h	▣

ATM database

Create statement

```
CREATE TABLE atms (
   id int(11) AUTO_INCREMENT,
   name varchar(30),
   working_hours varchar(10),
   lat float,
   lon float,
)
```

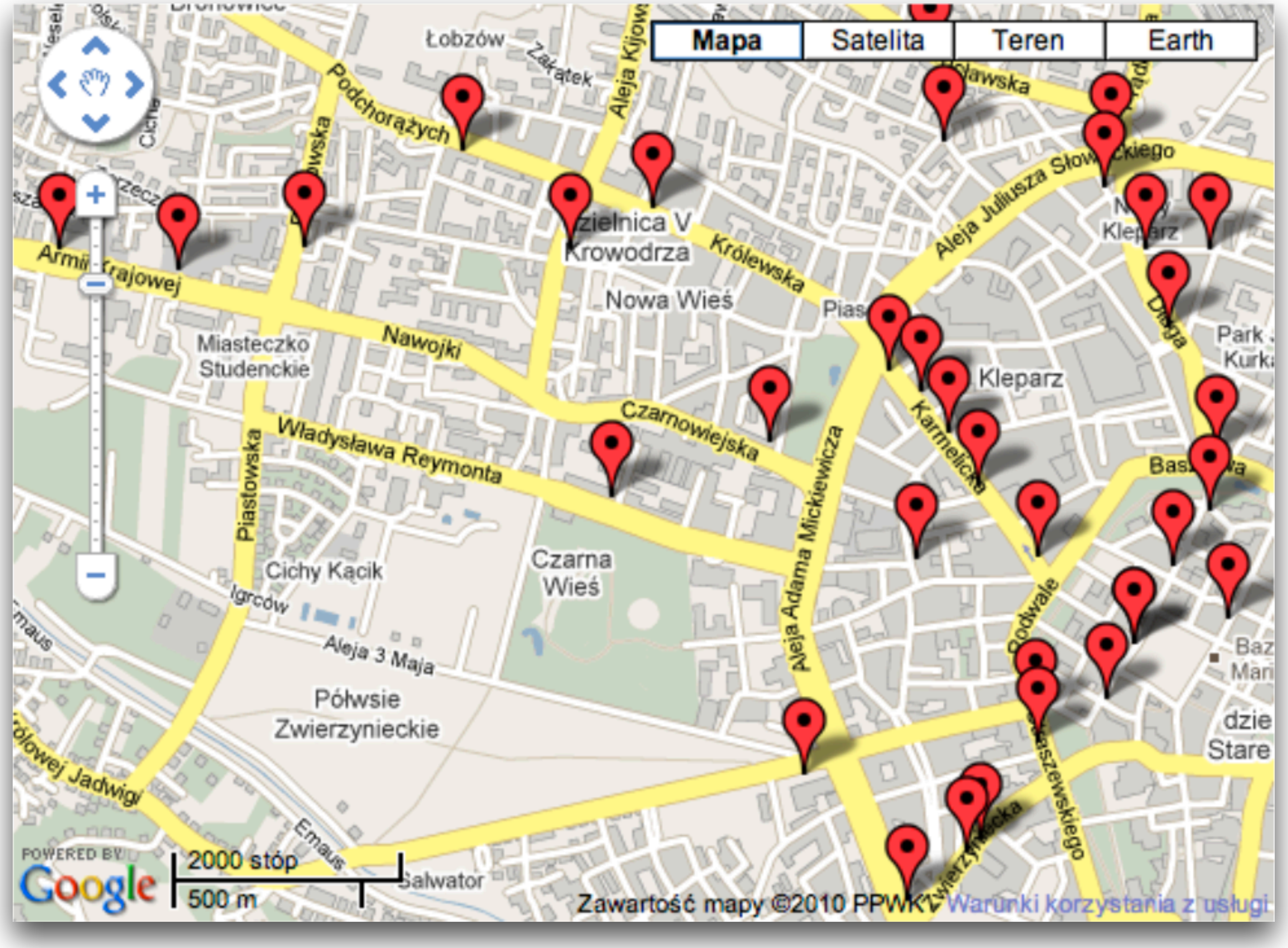


Lokalizacja	Godziny otwarcia	Sieć
BP Stacja Paliw 29 Listopada 39B	24 h	▣
PSS Podwawelska 29-listopada 57/59	24 h	▣
BP Stacja Paliw al. 29 Listopada 125	24 h	▣
OBI al. Bora Komorowskiego 31	24 h	▣
Comarch al. Jana Pawła II 41e	Pn.-pt. 8:00-18:00	▣
Alior Bank al. Kijowska/ Kazimierza W	24 h	▣
Słup Clear Channel al. Mickiewicza (Akademia		
PSS Kraków al. Pokoju 20		
Raiffeisen Bank Polska S.A al. Pokoju 44		
Plaza Center Kraków al. Pokoju 44		
Praktiker al. Pokoju 67		
CH M1 al. Pokoju 67		
Statoil, stacja paliw Armii Krajowej		
Raiffeisen Bank Polska S.A Armii Krajowej 18		
Market Punkt Balicka 7		
Firma AKT Bartla 20	24 h	▣
Delikatesy "Złoty Róg" Bałuckiego 9	24 h	▣

ATM database

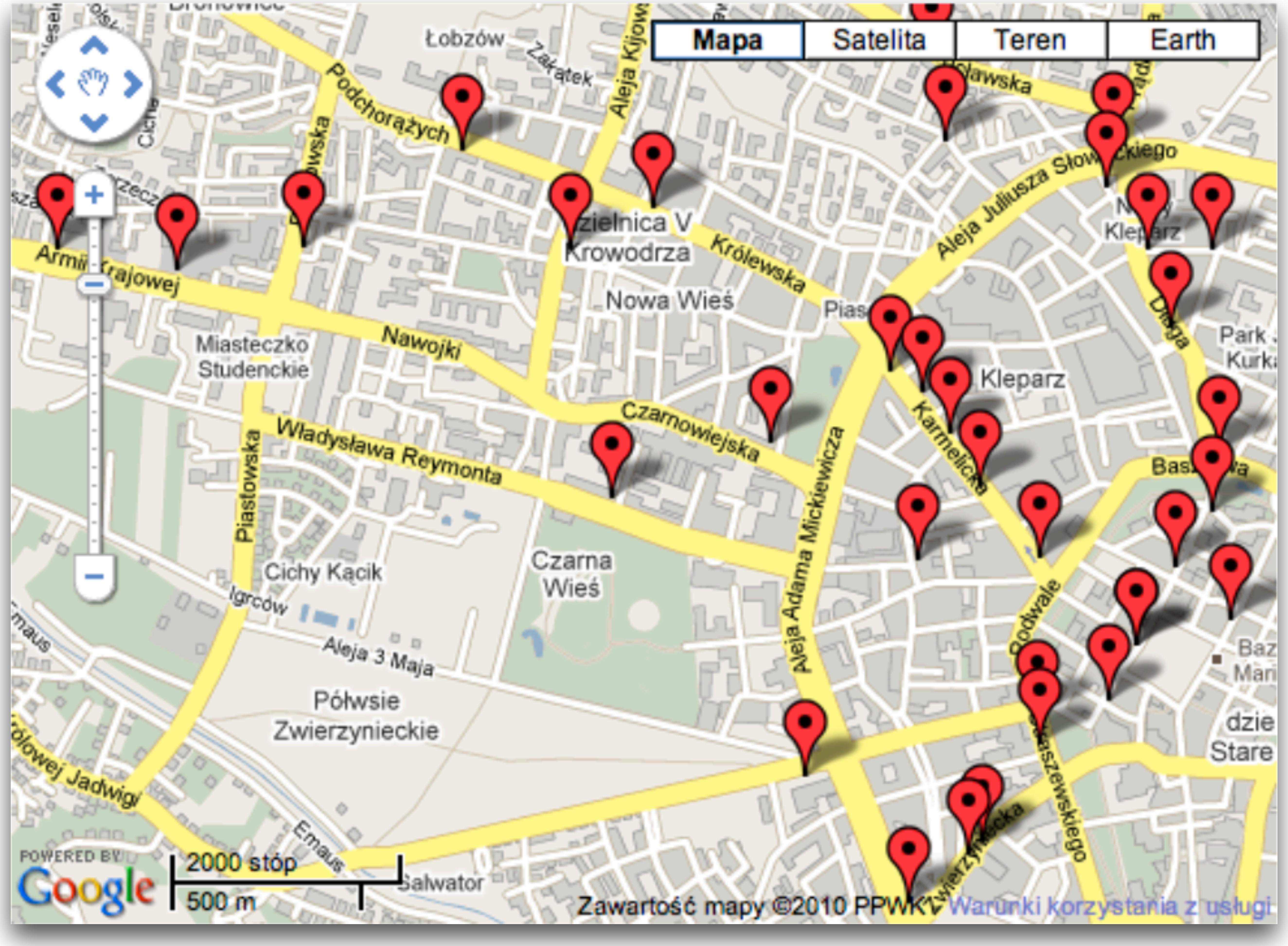
Find all ATMs within area

```
SELECT
  id, name, lat, lon
FROM
  atms
WHERE
  lat > 50.068902 AND
  lat < 50.063255 AND
  lon > 19.913750 AND
  lon < 19.923878
```



ATM database

But how do we handle this?



Spatial data

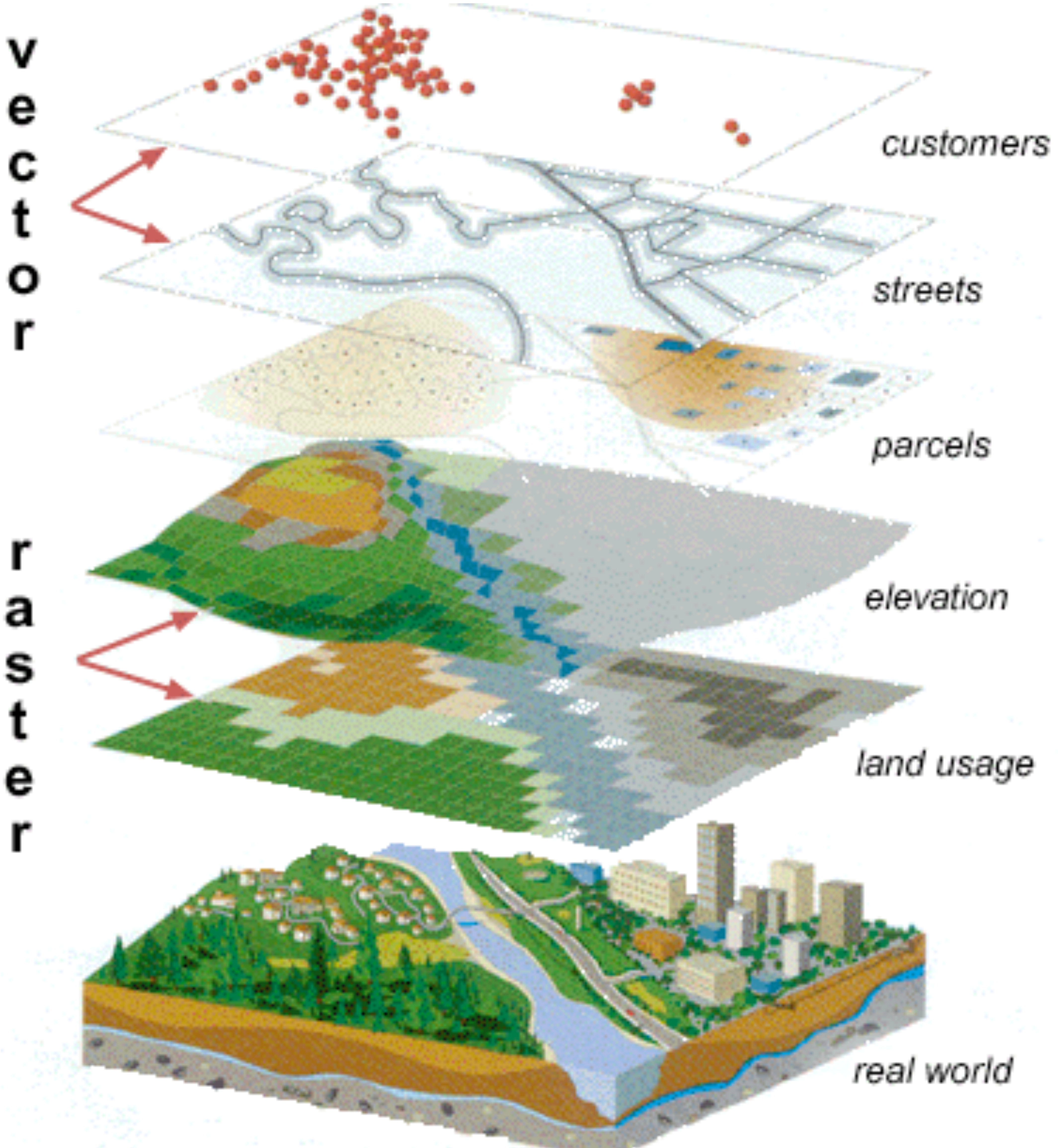
What is spatial data?

- Spatial data usually contains two components:
    - spatial component – geometry: location, shape
    - attribute component
- Also called *geospatial data* or *georeferenced data*.

Spatial data

Types of geometry layers

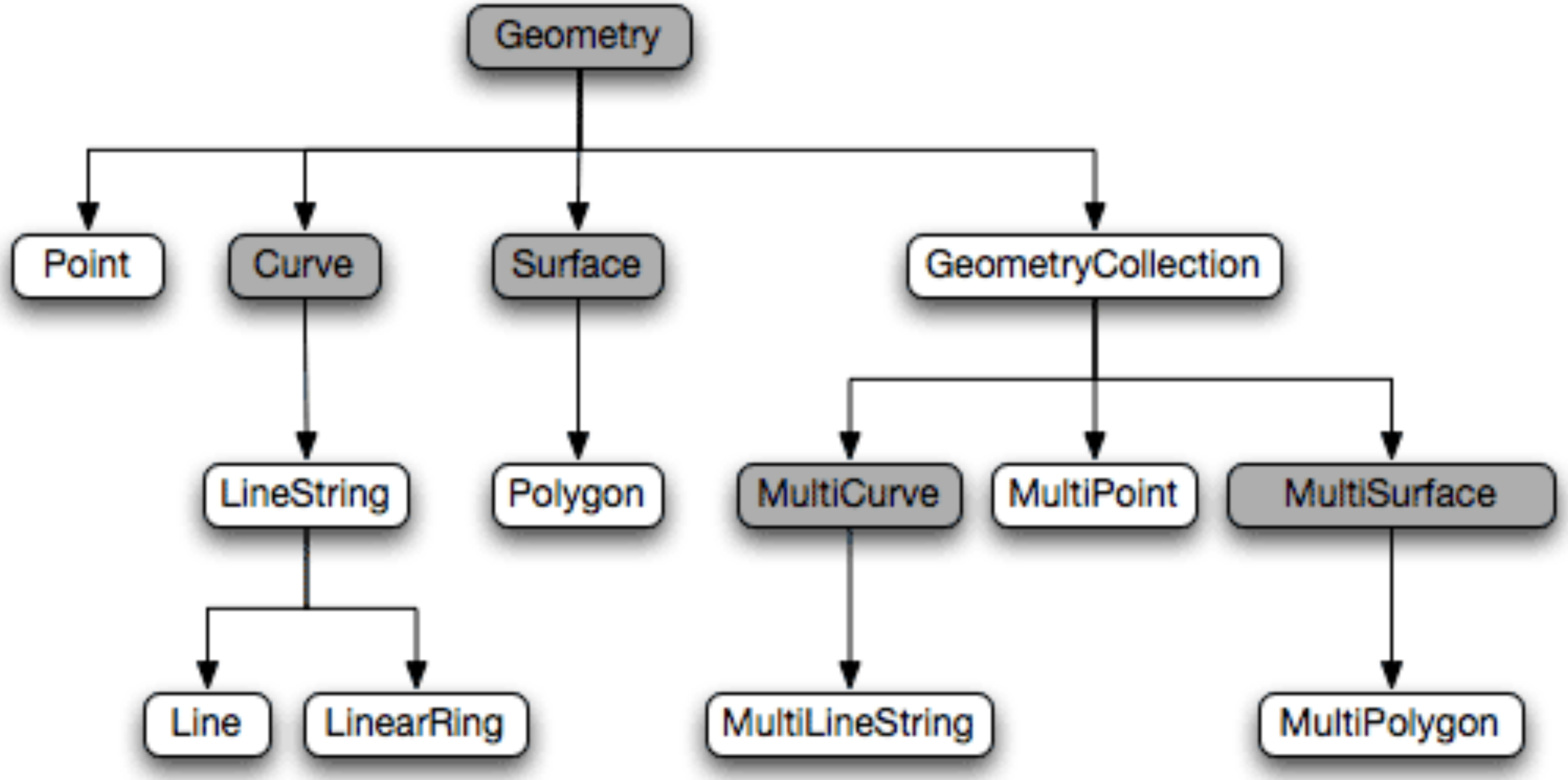
- raster layers:
  - area is divided into a grid of equally-sized ‘pixels’
  - each field is assigned a value
- vector layers:
  - objects are described using basic geometric shapes (point, line, polygon)



source: National Coastal Data Development Center

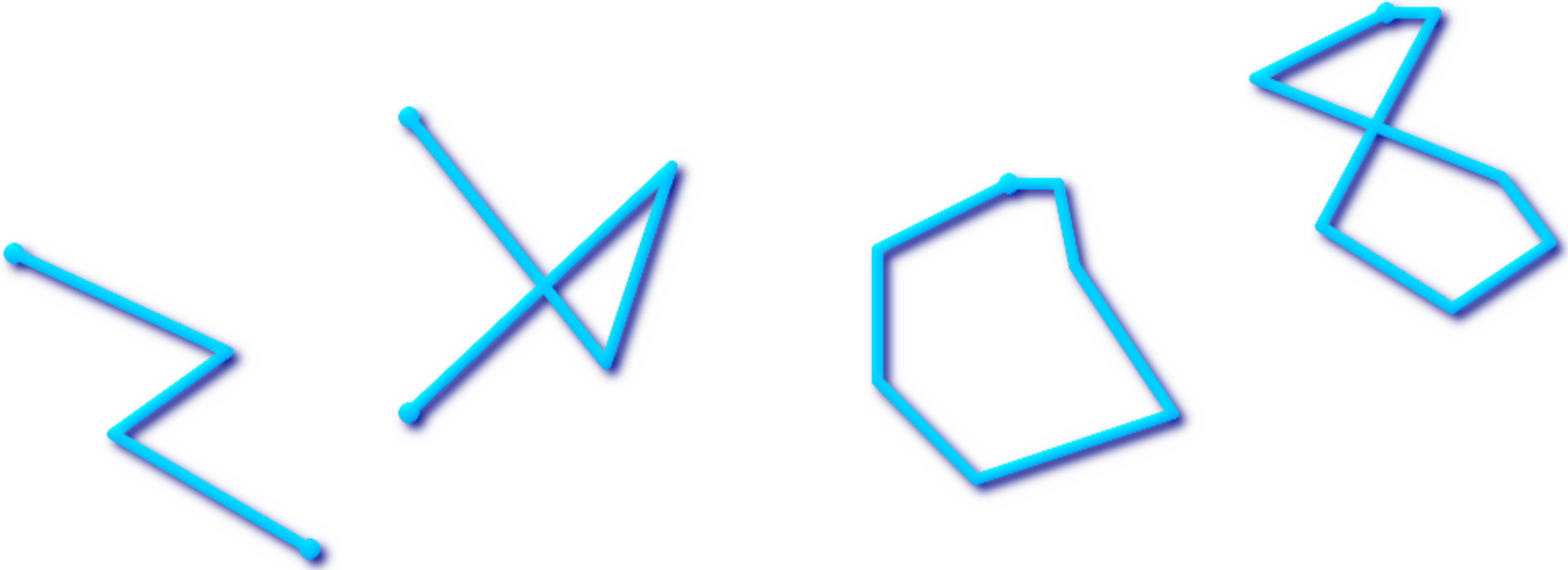
Spatial data

Types of geometric primitives



Spatial data

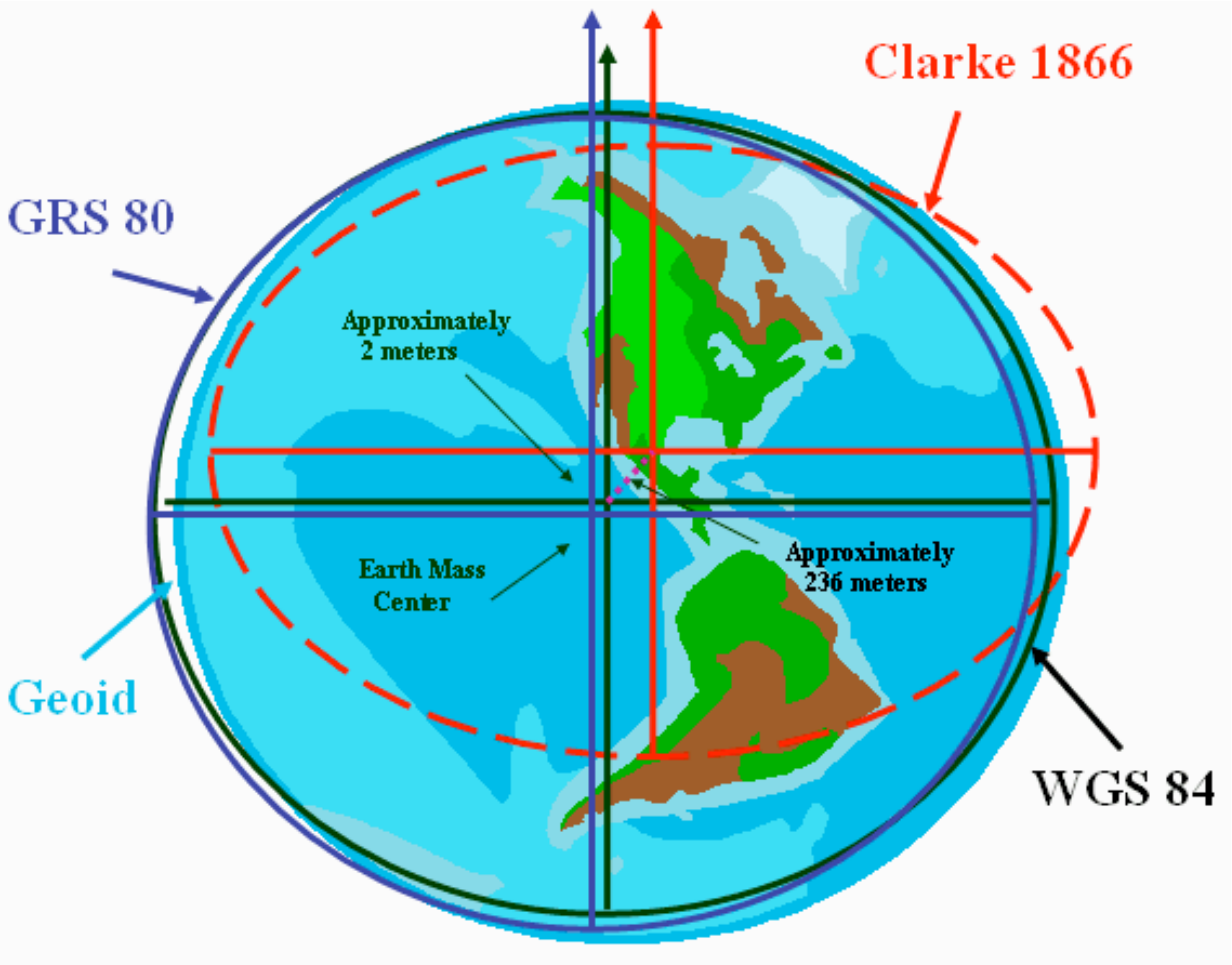
Which of these are proper linestrings?



Spatial data

Coordinate reference systems

- *Spherical* systems (degrees) vs. *planar* systems (meters)
- Catalogued by EPSG (European Petroleum Survey Group)
- Referred to using SRIDs
- Common SRIDs: EPSG codes (format: *EPSG:xxxx*)

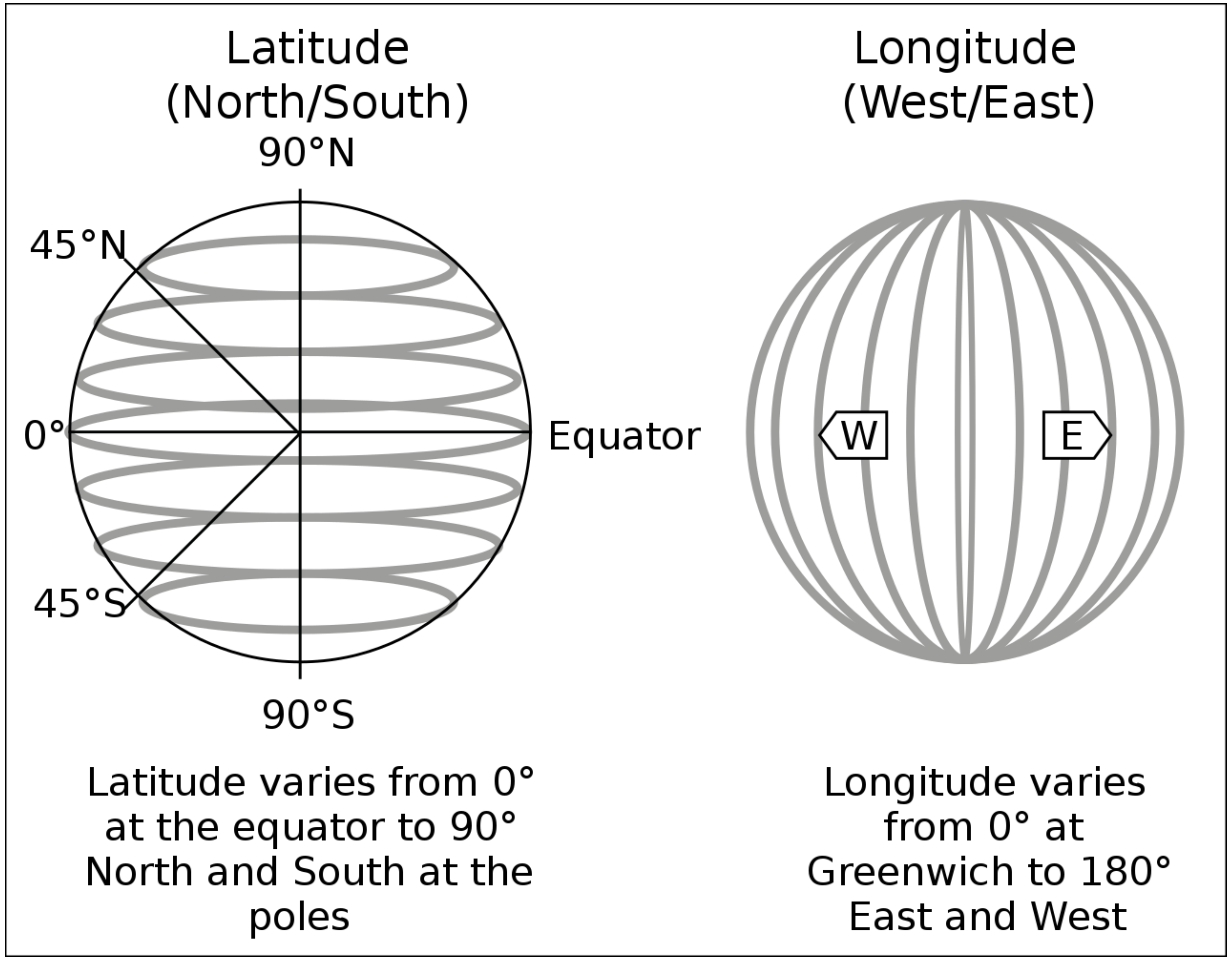


source: National Oceanic and Atmospheric Administration

Spatial data

Spherical CRSs

- Locations defined by providing two coordinates (*latitude* and *longitude*) in *degrees*
- Sometimes: also *elevation* (in *meters*)
- Can express any location on Earth…
- …but pretty useless for measurements: *"What is the distance from Krakow to Katowice in degrees?"*
- Most popular spherical CRS: <u>EPSG:4326 (WGS-84)</u>

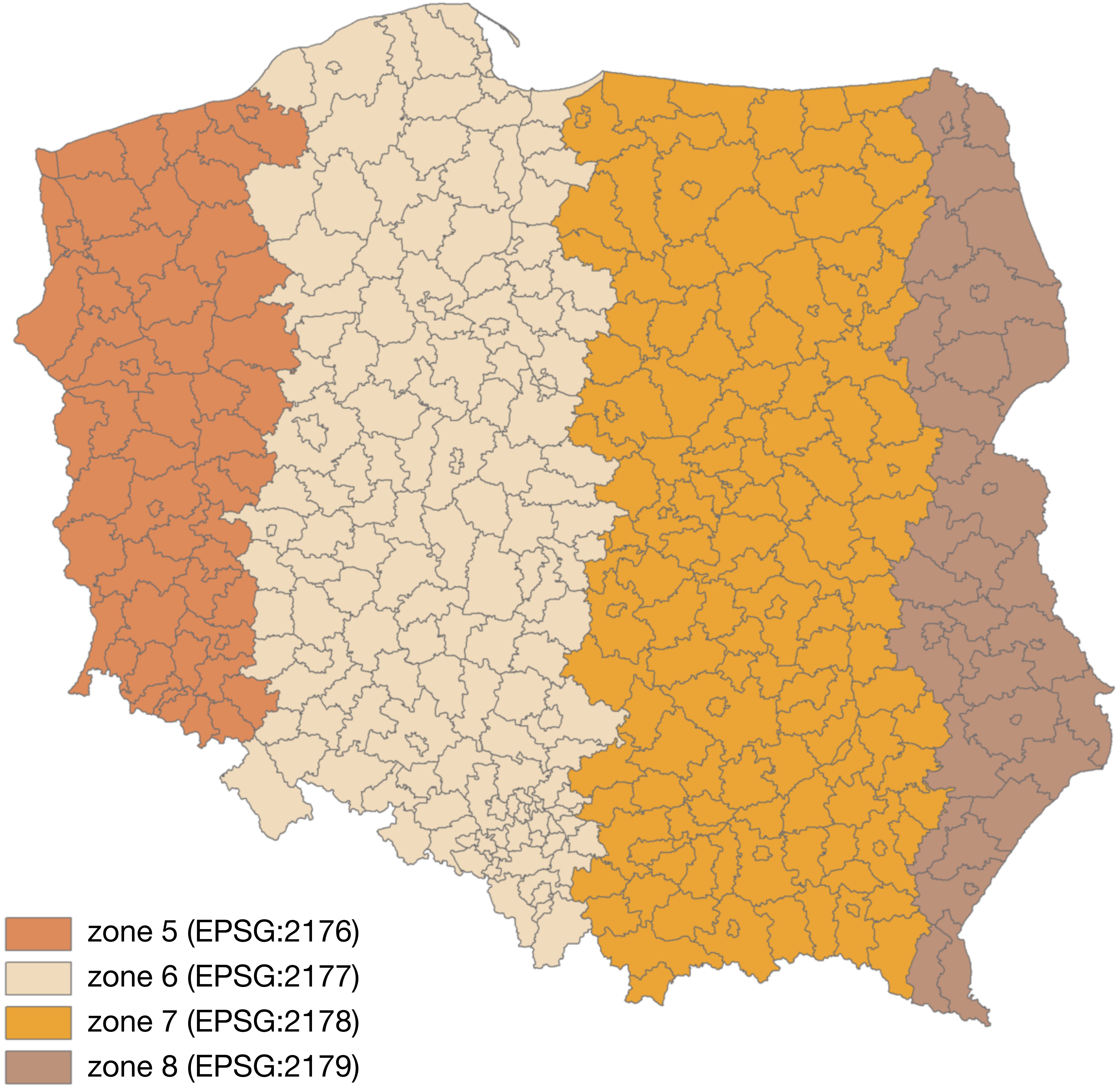

Latitude
(North/South)
90°N
45°N
0°
Equator
45°S
90°S
Latitude varies from 0° at the equator to 90° North and South at the poles
Longitude
(West/East)
W
E
Longitude varies from 0° at Greenwich to 180° East and West

Spatial data

Planar CRSs

- Project *a certain area* onto X/Y (isometric) coordinates, usually in *meters* (sometimes, also Z for elevation)
- Enable easy measurements
- Precision varies

Four zones of the Polish 2000 Coordinate System (PUWG 2000)



Spatial data

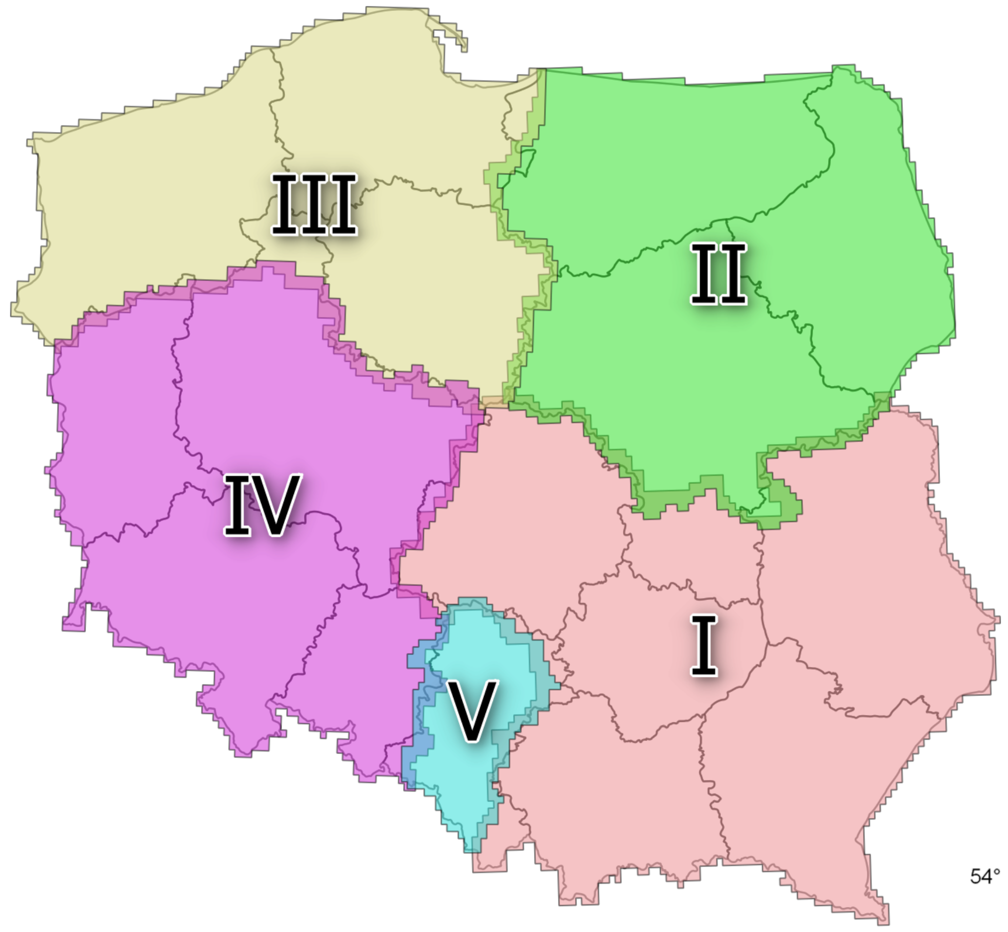
Precision of planar CRSs

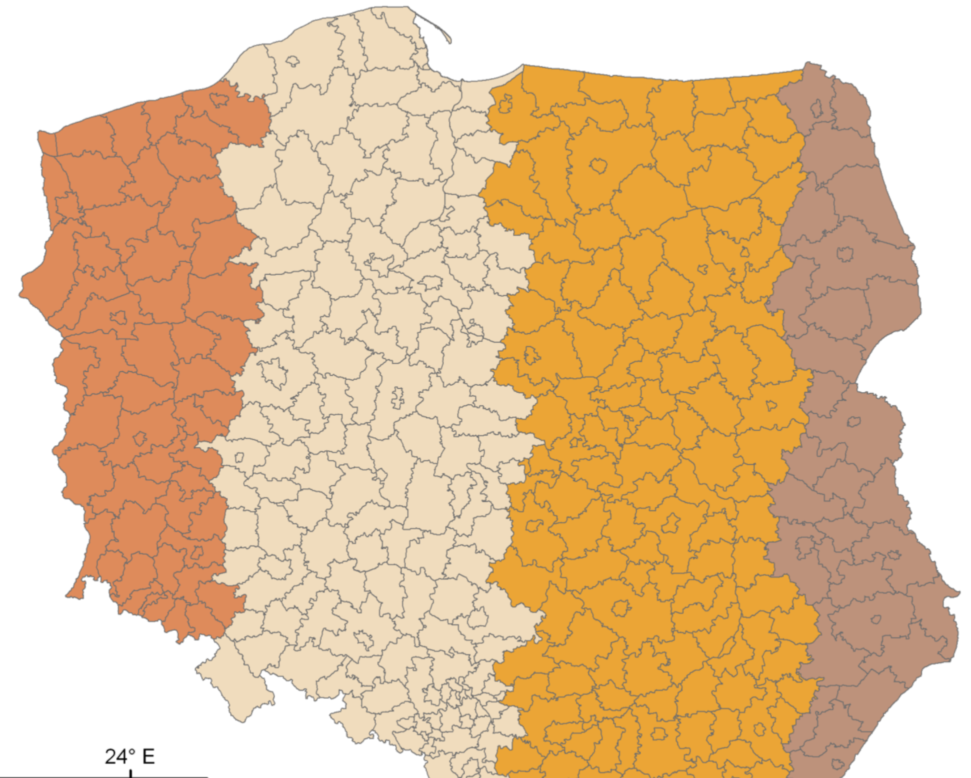
- **Polish CS 1965** (EPSG:3120, EPSG:2172–2175): distortion up to **20 cm/km (0.2 ‰)**

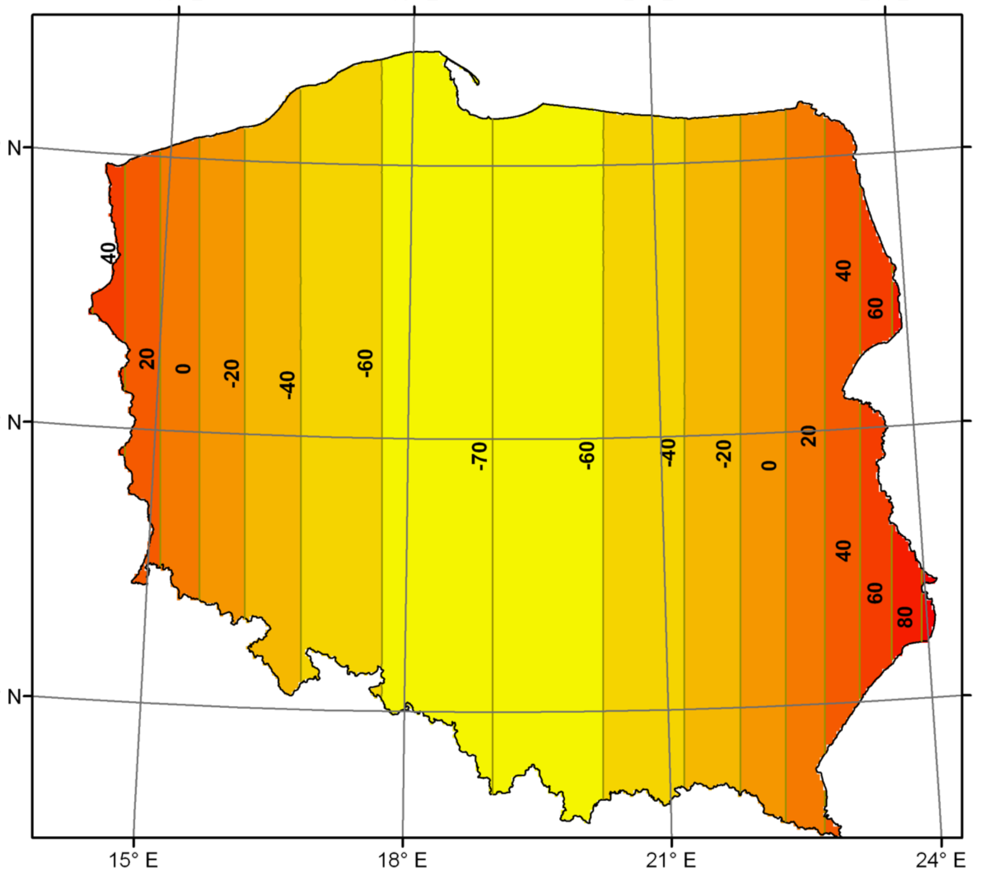
- **Polish CS 1992** (EPSG:2180): distortion up to **90 cm/km (0.9 ‰)**

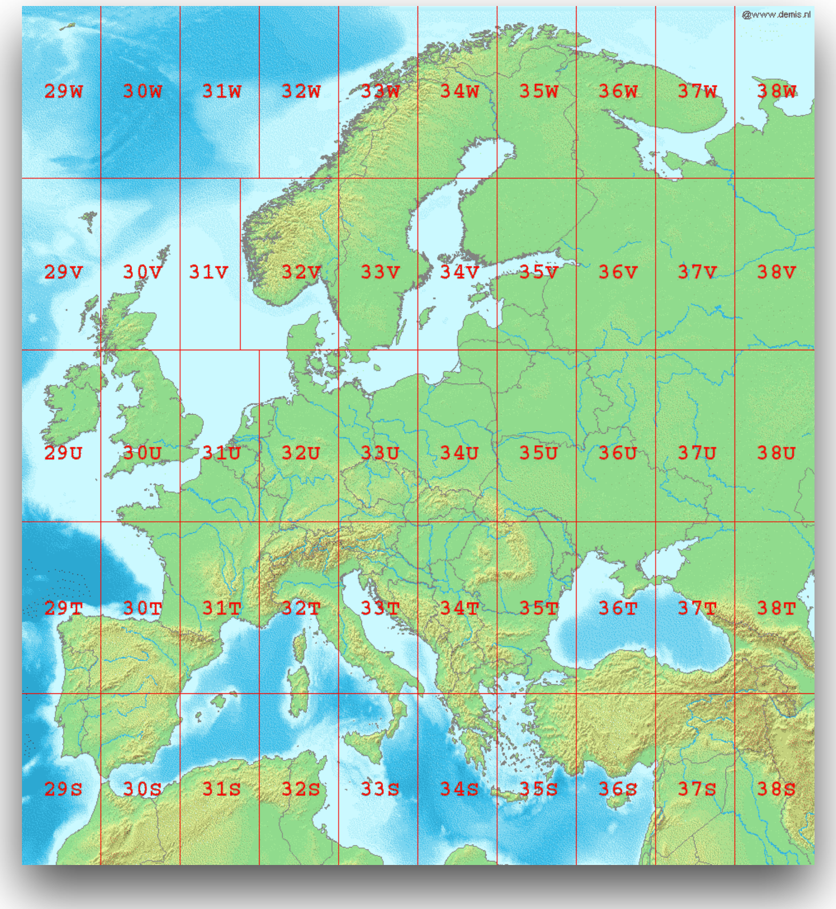
- **Polish CS 2000** (EPSG:2176–2179): up to **7.7 cm/km (0.077 ‰)**

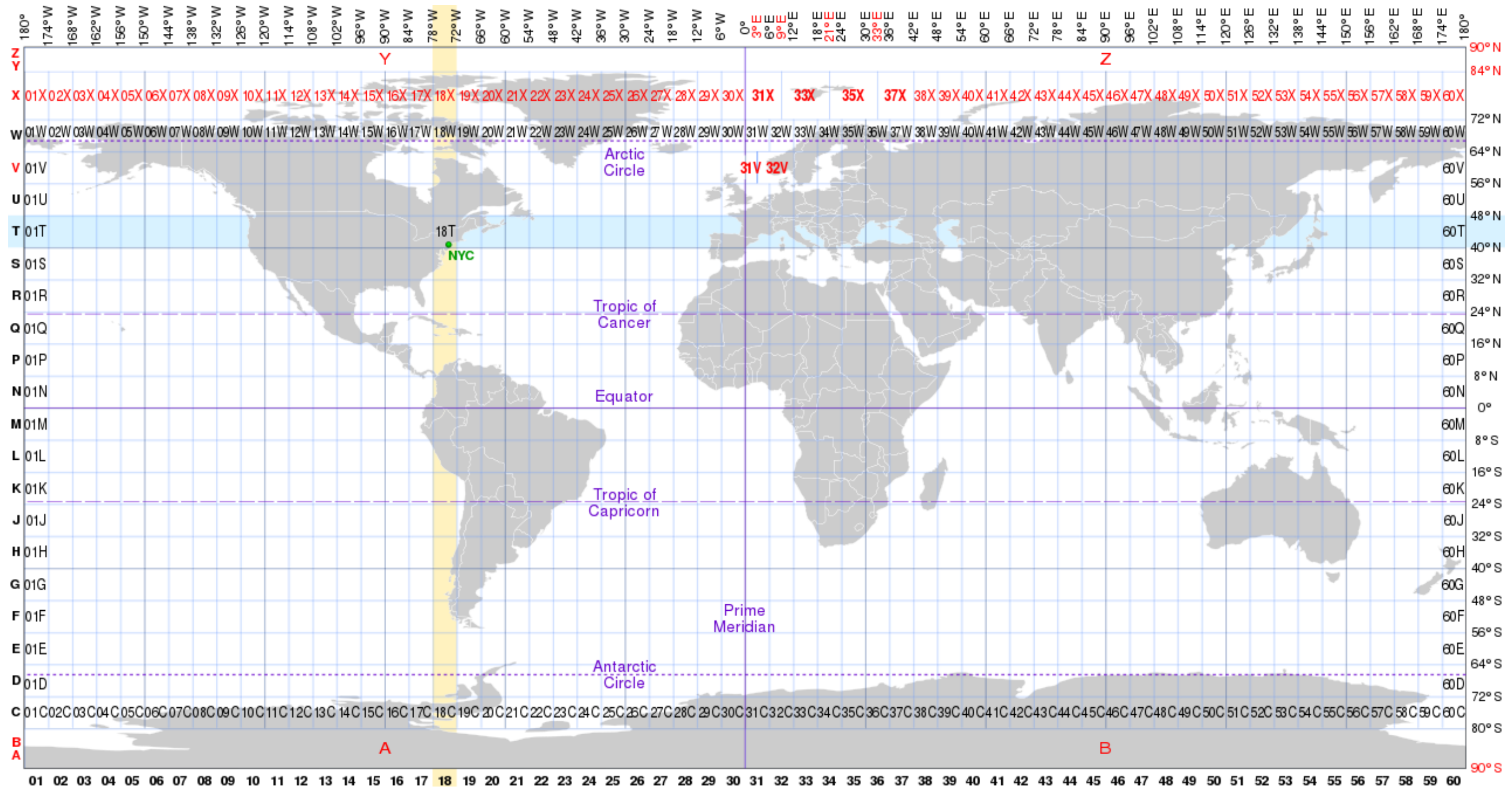
- **Universal Transverse Mercator** (60 zones): below **1 cm/km (1 ‰)**








29W 30W 31W 32W 33W 34W 35W 36W 37W 38W
29V 30V 31V 32V 33V 34V 35V 36V 37V 38V
29U 30U 31U 32U 33U 34U 35U 36U 37U 38U
29T 30T 31T 32T 33T 34T 35T 36T 37T 38T
29S 30S 31S 32S 33S 34S 35S 36S 37S 38S



Spatial data formats

Well-known text (WKT)

- Simple, text representation of vector geometries
- Designed to be human-readable
- Has a binary counterpart: **WKB (Well-known binary)**
- Often used inside SQL queries or when browsing spatial data in non-visual interfaces

```
GEOMETRYCOLLECTION(POINT(4 6),LINESTRING(4 6,
7 10))
POINT ZM (1 1 5 60)
POINT M (1 1 80)
POINT EMPTY
MULTIPOLYGON EMPTY
TRIANGLE((0 0 0,0 1 0,1 1 0,0 0 0))
TIN (((0 0 0, 0 0 1, 0 1 0, 0 0 0)), ((0 0 0,
0 1 0, 1 1 0, 0 0 0)))
POLYHEDRALSURFACE Z ( PATCHES
    ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)),
    ((0 0 0, 0 1 0, 0 1 1, 0 0 1, 0 0 0)),
    ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
    ((1 1 1, 1 0 1, 0 0 1, 0 1 1, 1 1 1)),
    ((1 1 1, 1 0 1, 1 0 0, 1 1 0, 1 1 1)),
    ((1 1 1, 1 1 0, 0 1 0, 0 1 1, 1 1 1))
  )
```

Spatial data formats

EWKT/EWKB

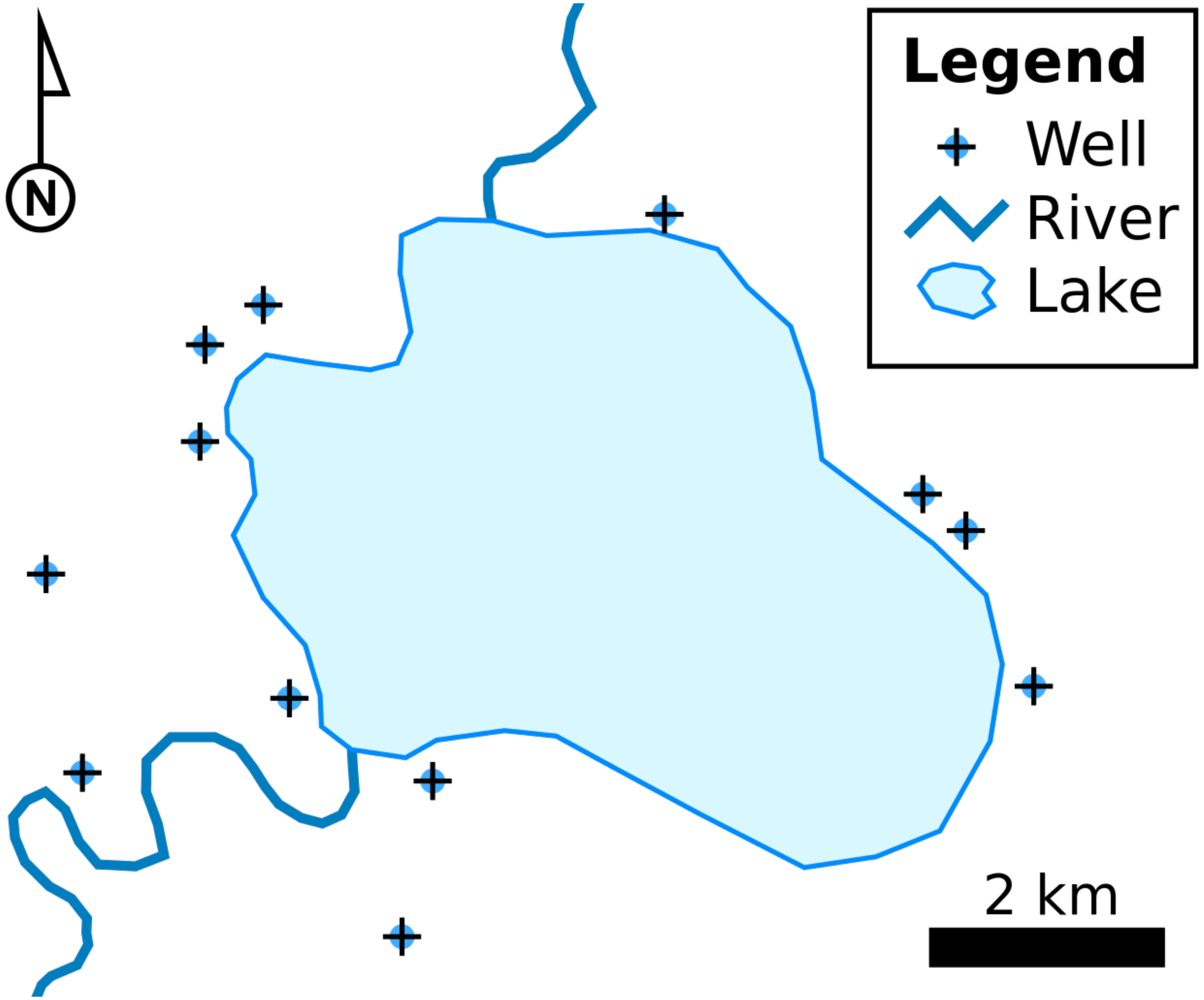
- Variation of WKT/WKB introduced by PostGIS
- Adds SRID, e.g.: SRID=4326;POINT(-44.3 60.1)
- Supports up to four ordinate values (XYZM)

```
GEOMETRYCOLLECTION(POINT(4 6),LINESTRING(4 6,
7 10))
POINT ZM (1 1 5 60)
POINT M (1 1 80)
POINT EMPTY
MULTIPOLYGON EMPTY
TRIANGLE((0 0 0,0 1 0,1 1 0,0 0 0))
TIN (((0 0 0, 0 0 1, 0 1 0, 0 0 0)), ((0 0 0,
0 1 0, 1 1 0, 0 0 0)))
POLYHEDRALSURFACE Z ( PATCHES
    ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)),
    ((0 0 0, 0 1 0, 0 1 1, 0 0 1, 0 0 0)),
    ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
    ((1 1 1, 1 0 1, 0 0 1, 0 1 1, 1 1 1)),
    ((1 1 1, 1 0 1, 1 0 0, 1 1 0, 1 1 1)),
    ((1 1 1, 1 1 0, 0 1 0, 0 1 1, 1 1 1))
  )
```

Spatial data formats

ESRI Shapefile

- Commonly-used vector geometry interchange format, introduced in the 1990s
- Dataset consists of several files
- Attribute names limited to 10 characters
- Mandatory files:
    - `.shp` – geometry itself
    - `.shx` – shape index
    - `.dbf` – attributes (dBase IV format)
- Additional files: projection (.prj), indexes, code page specification, etc.



Spatial Data Formats

GeoPackage

- “New standard” format, defined in 2014
- Supports vector and raster data
- Everything in a single file (`.gpkg`)
- Extended SQLite 3 database file underneath

Spatial data formats

GeoJSON

- JSON-based standard format for geographic data interchange
- Defined by RFC 7946
- Supports commonly-used primitives and collections
- Each feature can have arbitrary JSON properties

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

Spatial data formats

Geography Markup Language

- Abbreviated as GML
- Primary use: vector features
- Also supported:
    - coverages (bitmap layers)
    - sensor data streams

```
<gml:Polygon>
  <gml:outerBoundaryIs>
    <gml:LinearRing>
      <gml:coordinates>
        0,0 100,0 100,100 0,100 0,0
      </gml:coordinates>
    </gml:LinearRing>
  </gml:outerBoundaryIs>
</gml:Polygon>
<gml:Point>
  <gml:coordinates>
    100,200
  </gml:coordinates>
</gml:Point>
<gml:LineString>
  <gml:coordinates>
    100,200 150,300
  </gml:coordinates>
</gml:LineString>
```

Spatial data formats

OpenStreetMap XML

- Not really a GIS format, but a format for OpenStreetMap data

- More on OpenStreetMap will follow

- XML-based text format (`.osm`) that represents all OpenStreetMap (but not OGC) primitives

- Files can be huge!

- Has a binary counterpart: PBF (Protocolbuffer Binary Format)

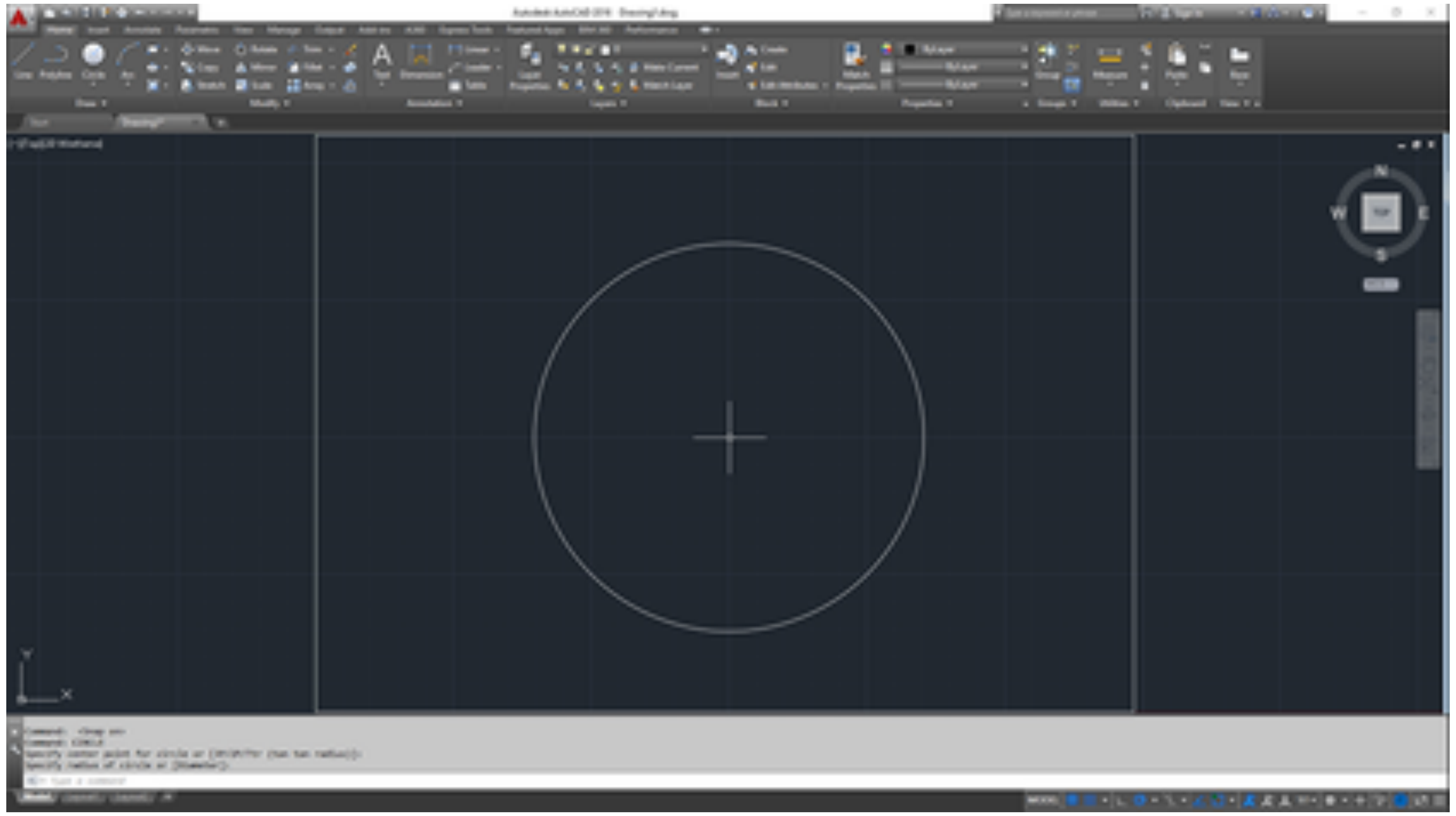
```
<node id="273470204" lat="50.0679716"
lon="19.9312968" version="1"
changeset="302227" user="Rafal Olearski"
uid="12349" visible="true"
timestamp="2008-06-26T19:21:27Z"/>

<way id="25117187" visible="true"
timestamp="2008-06-26T19:21:30Z" version="1"
changeset="302227" user="Rafal Olearski"
uid="12349">
  <nd ref="273470204"/>
  <nd ref="273470207"/>
  <nd ref="262201253"/>
  <tag k="name" v="Kremerowska"/>
  <tag k="highway" v="residential"/>
  <tag k="oneway" v="yes"/>
</way>
```

Spatial data formats

AutoCAD DWG/DXF

- Also not really a GIS format
- CAD drawings can have geographic (planar) coordinates
- Data organised in layers
- DXF is the ‘open’ format; DWG can be converted e.g. using the ODA File Converter tool



```
Command: <Snap on>
Command: CIRCLE
Specify center point for circle or [3P/2P/Ttr (tan tan radius)]:
Specify radius of circle or [Diameter]:
```

Spatial data formats

Converting between formats

- GDAL has ogrinfo & ogr2ogr shell tools
- Python: Fiona
- Shapefiles: shp2pgsql (bundled w/PostGIS)
- OSM data: osmosis, osm2pgsql

Spatial RDBMSs

Spatial database features

- **Data types.** A spatial database must support data types which allow for storing map elements.
- **Spatial operations.** We need functions which construct, process and analyse spatial objects (area, distance, etc.).
- **Exchange of spatial data.** The system must be able to exchange spatial data with other systems.
- **Spatial data indexing.** As already mentioned, traditional indexes are not optimised for spatial queries.

Spatial RDBMSs

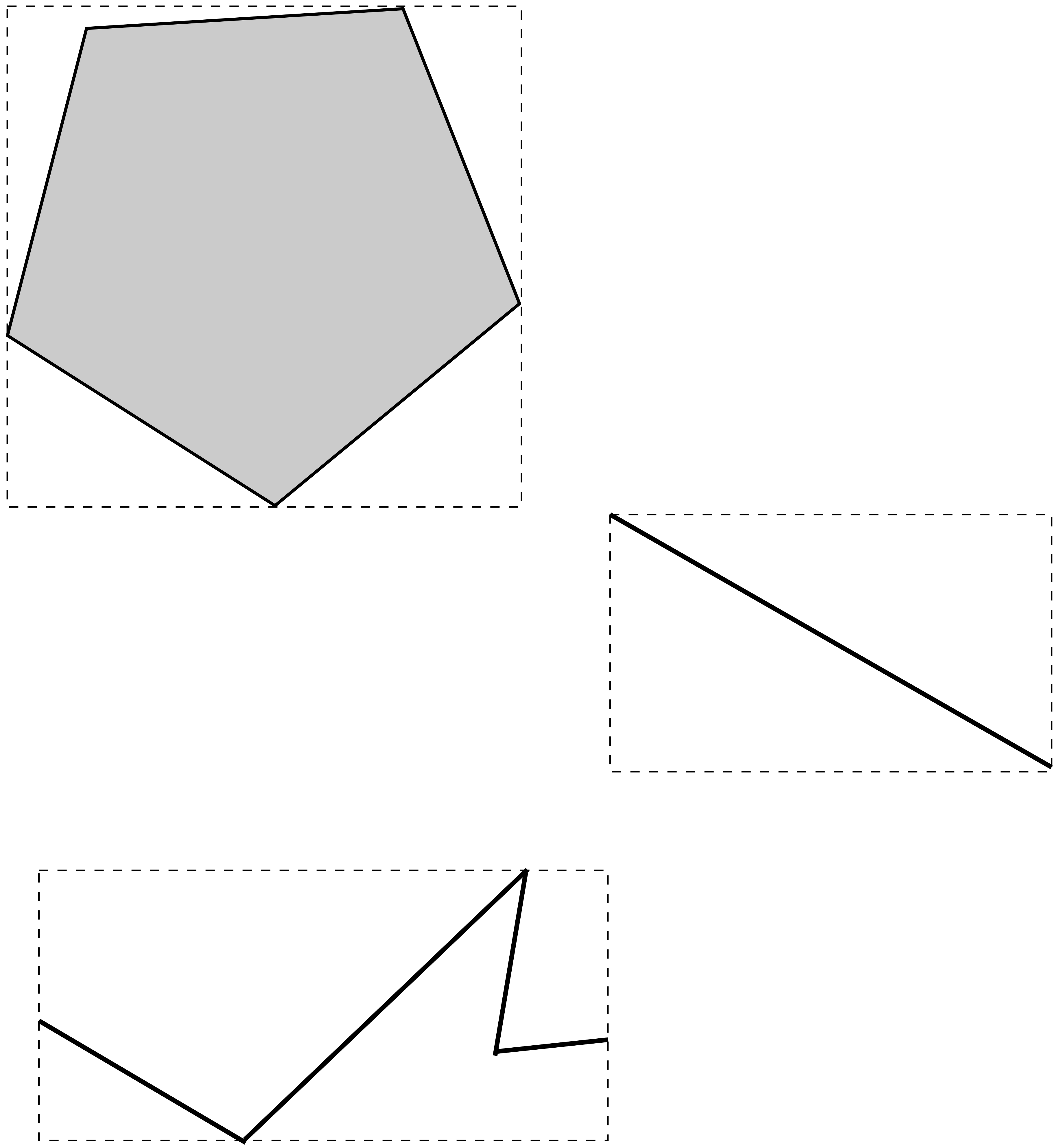
Most popular implementations

- MySQL supports spatial data since version 5.0 (not entirely OpenGIS compliant).
- PostGIS – a PostgreSQL extension, provides OpenGIS-compliant data types and analytic functions.
- Oracle Spatial – an OpenGIS-compliant Oracle extension.
- IBM DB2 Spatial Extender, Geodetic Extender.
- Spatialite – spatially-enabled SQLite

Spatial RDBMSs

Data indexing

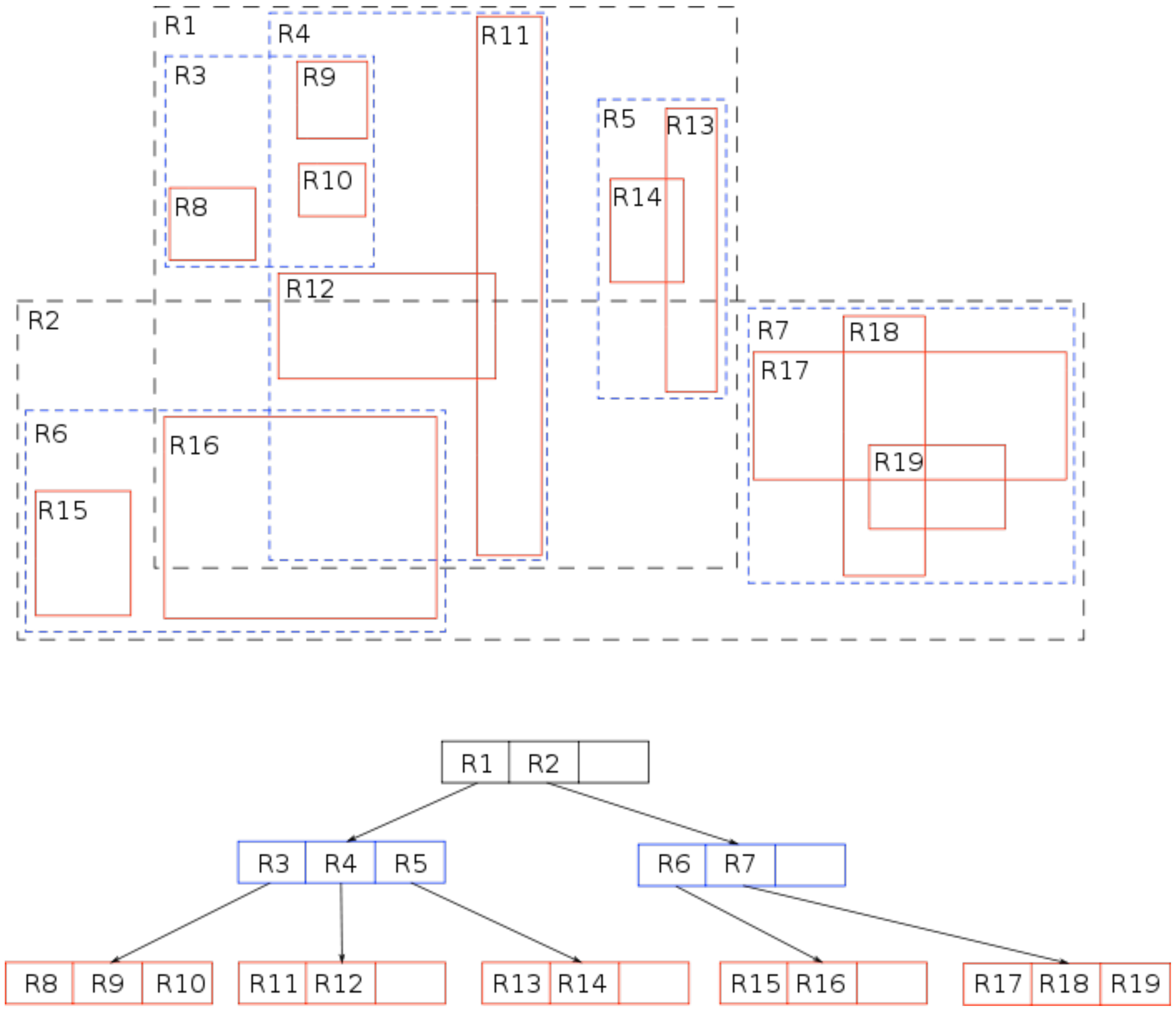
- Classical indexes are usually based on structures like B-trees, only applicable for one-dimensional data.
- Spatial data is usually 2D or 3D.
- We may use two independent 1D indexes, but that limits query flexibility.
- Instead, spatial indexes are based on the concept of the MBR (Minimum Bounding Rectangle)



Spatial RDBMSs

Indexing using R-trees

- Similar to B-trees, but used to index multidimensional data.
- Divide space into nested, overlapping regions.
- Close regions are placed in one tree node.



PostGIS

General characteristics

- PostgreSQL add-on.
- GPL license.
- Available for all platforms supported by PostgreSQL
- Supports (E)WKT/(E)WKB, GeoJSON
- GIST-based indexing

PostGIS

Database structure

- Three system tables spatial_ref_sys, geometry_columns and geography_columns store data regarding spatial reference system and spatial columns

- Before version 2.x, spatial columns had to be added using the AddGeometryColumn function:

  ```
  SELECT AddGeometryColumn('parks','park_geom',
      128, 'GEOMETRY', 2 );
  ```

- Now, everything is done in the CREATE statement.

PostGIS

Geometry vs. geography

- The GEOMETRY type may use any of the available reference systems; distances are calculated as straight-line distances on a plane in a given system.

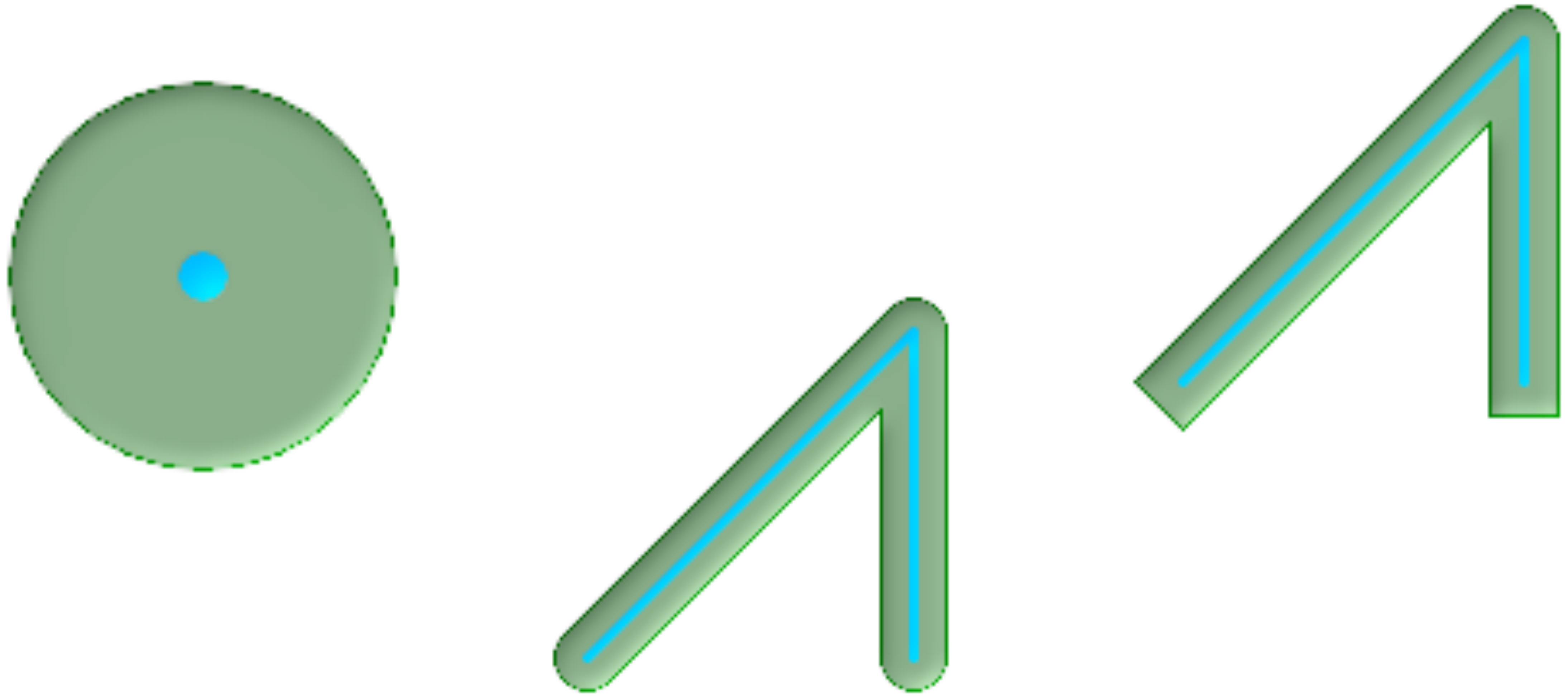
- The GEOGRAPHY type always stores longitude/latitude in degrees (WGS-84), but distances are calculated using the shape of the spheroid.

- More in PostGIS docs.

PostGIS

Practical usage

- Link: Reference manual
- *Buffer* function – used more commonly than you might think
- Some more complex examples: link



GeoPandas

General characteristics

- GeoPandas extends Pandas by adding geometric types and operations
- Uses:
  - shapely for operations
  - fiona for I/O
  - descartes and matplotlib for plotting
- Supports spatial joins, reprojecting CRSs… and all that Pandas can do
- Pandas: Comparison with SQL

QGIS

Open-source GIS software

- Formerly known as Quantum GIS
- I/O in many supported formats
- Visualisation and editing
- Data filtering, labelling and rule-based rendering
- Supports Linux (repos/binaries for most distributions), Windows (directly or via OSGeo4W) and macOS (use HomeBrew Caskroom!)

OSGeo

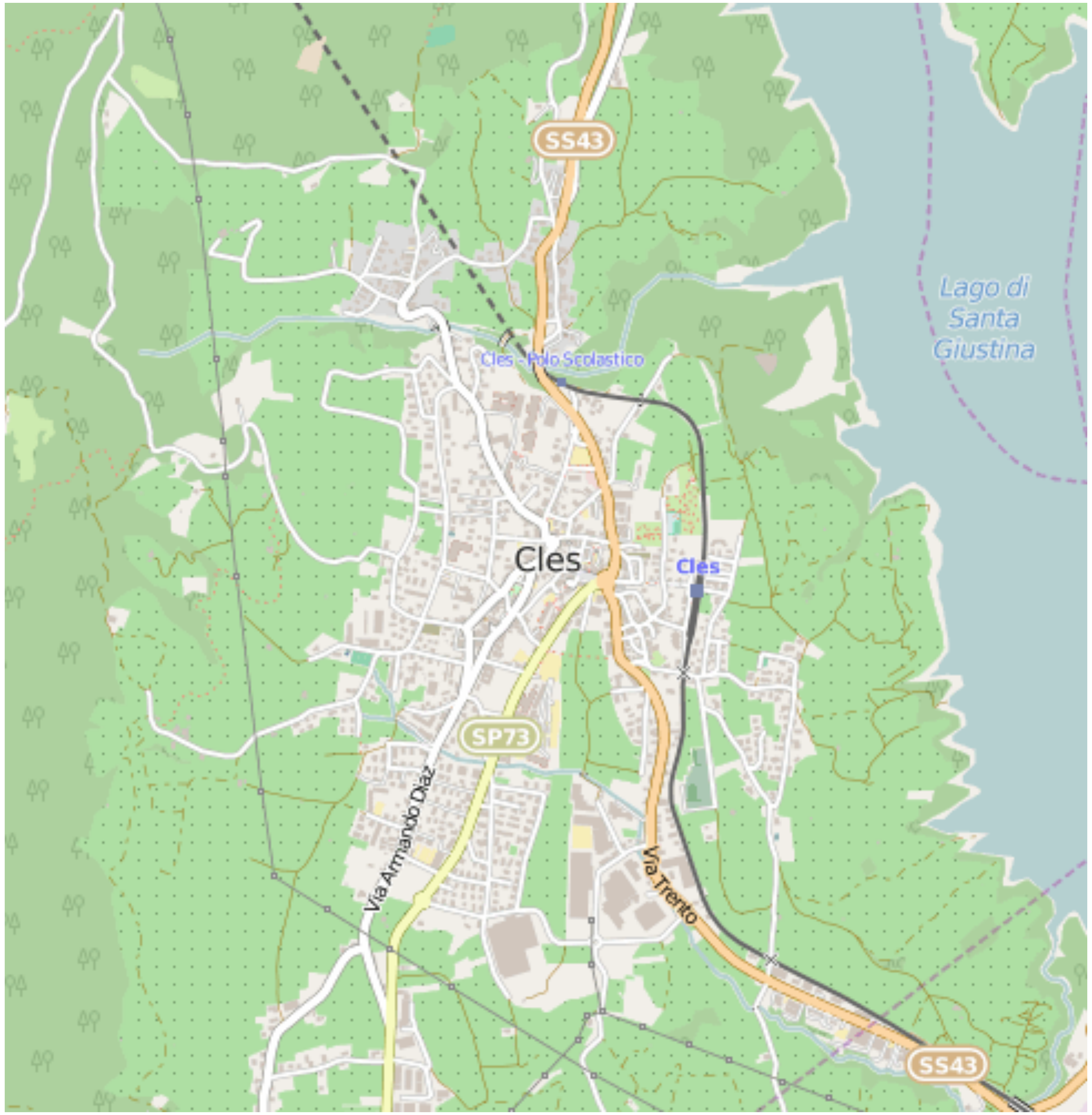
Open Source Geospatial Foundation

- Provides tools, teaching materials and promotes the use of open-source GIS tools
- OSGeo4W, OSGeo4mac, OSGeoLive are easy-to-install sets of GIS tools (including e.g. QGIS)

Mapnik

Open-source map renderer

- Used as “industry standard” for rendering maps
- Many supported input formats
- Cross-platform



OpenStreetMap

A collaborative map of the world

- Edited by the community, just like Wikipedia
- Central repository (uses MySQL), not really interesting for us
- Web interface allows for map browsing, edition and (small) downloads
- Extra tools: converters, editors (more advanced than the web-based one)

OpenStreetMap

The Web interface

- http://www.openstreetmap.org
- Uses Leaflet.js
- Provides basic editing tools
- Can export some OSM XML

OpenStreetMap

Data model

- Basic elements:
  - nodes,
  - ways,
  - relations.
- Parameters for each element (e.g. road class, building type) are assigned as key/value pairs.
- Link: map features and their attributes.

OpenStreetMap

Tracking changes

- OSM keeps all history of changes
- Changesets describe changes introduced in a given editing session by a given user
- Operations: *add*, *modify*, *delete*
- Each changeset has a bounding box

OpenStreetMap

Getting the data

- Download entire globe: *planet.osm*
- Currently approximately 95 GB (XML/bz2), 54 GB (PBF)
- Better solution: download only the region you need, e.g. from GeoFabrik...
- …and crop using the Osmosis utility

OpenStreetMap

Tools

- Editors: Potlatch, JOSM
- Data processing: Osmium, Osmosis
- Data conversion: osm2pgsql, osm2pgrouting, but also Osmosis
- Python support: OSMnx