From HTML to PostGIS
Introduction do GIS
Spatial data formats and databases

Michał Okulewicz

Wydział Matematyki i Nauk Informacyjnych
Politechnika Warszawska

Spatial data and databases

Definitions I

Most definitions according to PTIP/PASI lexicon or INSPIRE DIRECTIVE 2007/2/EC:

[PL] Obiekt przestrzenny

Abstrakcja obiektu geograficznego jako zjawiska świata rzeczywistego, stanowi figurę geometryczną utworzoną przez wyodrębniony zbiór punktów w rozpatrywanej przestrzeni dwuwymiarowej lub trójwymiarowej i opisany danymi przestrzennymi.

[PL] INSPIRE - Obiekt przestrzenny

Abstrakcyjna reprezentacja zjawiska świata rzeczywistego związaną z określonym miejscem lub obszarem geograficznym.

Definitions II

[EN] INSPIRE - Spatial object

An abstract representation of a real-world phenomenon related to a specific location or geographical area.

[PL] Dane przestrzenne / Spatial data

Dane dotyczące obiektów przestrzennych, w tym zjawisk i procesów, znajdujących się lub zachodzących w przyjętym układzie współrzędnych.

[PL] INSPIRE Dane przestrzenne

Dane odnoszące się bezpośrednio lub pośrednio do określonego położenia lub obszaru geograficznego.

Definitions III

[EN] INSPIRE Spatial data

Any data with a direct or indirect reference to a specific location or geographical area.

Geometric simple types

- Point - 0-dimensional object
- Arc - 1-dimensional object
- Surface - 2-dimensional object
- Solid - 3-dimensional object

Definitions IV

Spatial data types

- Vector - made of geometric simple types
- Raster - values defined on an ordered set of cells (pixels, voxels) covering a given area

Definitions V

Are the vector data truly scalable?

That whole GIS... I

Geoinformatics

The science and technology dealing with the structure and character of spatial information, its capture, its classification and qualification, its storage, processing, portrayal and dissemination, including the infrastructure necessary to secure optimal use of this information.

Raju, Fundamentals of Geographic Information Systems

That whole GIS... II

Geomatics

Discipline concerned with the collection, distribution, storage, analysis, processing, presentation of geographic data or geographic information.

ISO/Technical Committee 211 Geographic information/Geomatics

That whole GIS... III

Geographical Information System(s)

System of:

- acquisition,
- storing,
- verifying,
- integrating,
- analyzing,
- transfer,
- sharing

for spatial data.

In broad sense it covers:

- methods,
- hardware and software,
- spatial database,
- companies,
- financial assets,
- stakeholders.

Why do we need GIS?

- To model and simulate environment and make data driven analyses and decisions
- To easily compare same information coming from various sources (i.e. cadastre data with an aerial map)
- To efficiently process spatial data (structures and algorithms)
- To share spatial data in an organized way (data types and services standards)

Why should we learn about and study GIS?

- They are common and well established by various legal entities - possibilities for stable business opportunities.
- Efficiency is an important issue - and who better to design new algorithms?
- They are one of the applicable areas for Artificial Intelligence, *Deep Learning* in particular

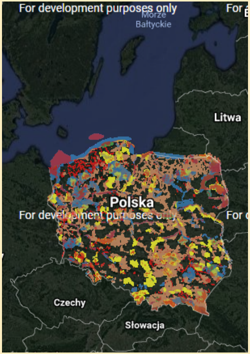
Applications

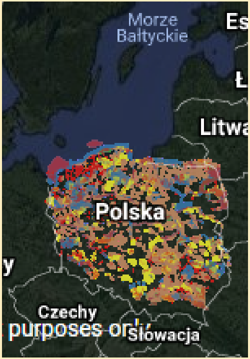
- LOKKOM: Data visualization
- LOKKOM: Data reports
- Google: Creating your own maps
- MiNI Plan [with all the consequences of relaying on GMaps]
- jakdojade.pl
- Middle-earth - with events
- Middle-earth - with artistic approach
- Westeros - with POVs travels

GIS Standards

- Coordinate Reference Systems and Coordinate Transformations - EPSG codes maintained by International Association of Oil & Gas Producers
- File formats and services - Open Geospatial Consortium
- Spatial SQL - ISO/IEC 13249 norm

Are all the maps the same?





EPSG codes

- Official registry
- Useful registry
- Useful codes:
  - EPSG:4326 – WGS84 - the most typical, used in GPS
  - EPSG:2180 – ETRS89 / Poland CS92 - large maps of Poland
  - EPSG:2176–EPSG:2179 – Poland CS200 - detailed maps of Poland
  - EPSG:3857 – WGS84 / Pseudo-Mercator / GoogleMaps

Standards for file formats (and services)

- Standard in the GIS business are set by the Open Geospatial Consortium (OGC)
- OGC is an association of GIS experts and companies and regulates the *de facto* technical standards in the GIS community
  - Even Google's own KML has been passed to standardization within OGC,
- OGC are not ISO norms, in order to remain accessible for free
- OGC's standards are selected as the source for technical guidelines for INSPIRE EC Directive

KML: Keyhole Markup Language

Documentation:

- Google
- OGC - KML 2.2.

Goals:

- spatial data,
- presentation (style),
- views (camera setting),

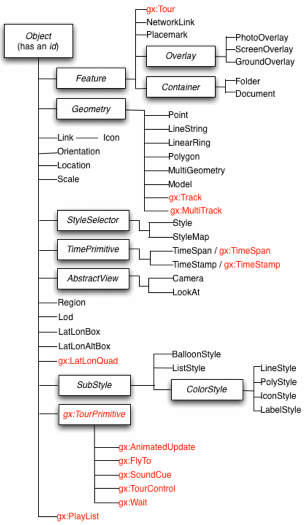
Not only for vector data...

KML - basic example

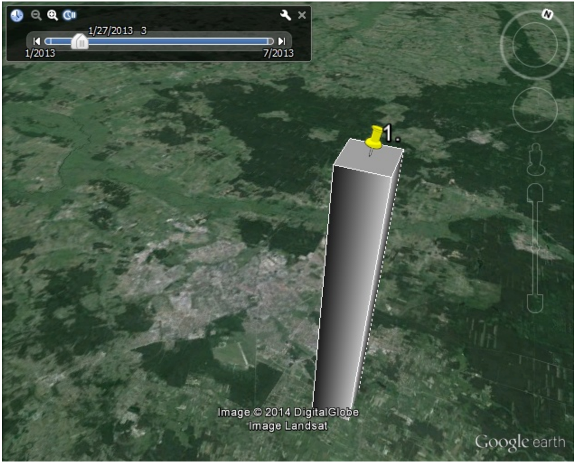
Labeled point

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2"
     xmlns:gx="http://www.google.com/kml/ext/2.2">
<Document>
  <name>Marker</name>
  <Placemark>
    <name>Marker</name>
      <Point>
        <coordinates>
        17.39125903192781,53.19437478927049,0
        </coordinates>
      </Point>
  </Placemark>
</Document>
</kml>
```

KML - abilities



KML - spatio–temporal presentations



http://www.mini.pw.edu.pl/˜okulewiczm/downloads/augis/files/3D.kml

GML: Geographic Markup Language

- Documentation
- GML is a data format allowing for a transfer and serialization of a more detailed data structures then KML
- GML concentrates on data, instead of its presentation (like KML).
- GML directly extends table like approach of RDBMS databases

GML - basic example

Point location with attributes

```
<wfs:FeatureCollection xmlns:wfs="http://www.opengis.net/wfs"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:ows="http://www.opengis.net/ows"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:qgs="http://www.qgis.org/gml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wfs
     http://schemas.opengis.net/wfs/1.0.0/wfs.xsd
     http://www.qgis.org/gml>
     <gml:boundedBy .../>
     <gml:featureMember .../>
</wfs:FeatureCollection>
```

GML - basic example

/wfs:FeatureCollection/gml:boundedBy

```
<gml:Box srsName="EPSG:4326">
 <gml:coordinates cs="," ts=" ">
   18.01020400725687765,52.23902294775036381
   20.99333372859214819,53.13053438171262144
 </gml:coordinates>
</gml:Box>
```

GML - basic example

/wfs:FeatureCollection/gml:featureMember

```
<qgs:test fid="test.0">
 <gml:boundedBy>
  <gml:Box srsName="EPSG:4326">
   <gml:coordinates cs="," ts=" ">18.01,53.13 18.01,53.13
   </gml:coordinates>
  </gml:Box>
 </gml:boundedBy>
 <qgs:geometry>
  <gml:Point srsName="EPSG:4326">
   <gml:coordinates cs="," ts=" ">18.01,53.1</gml:coordinates>
  </gml:Point>
 </qgs:geometry>
 <qgs:id>2</qgs:id>
 <qgs:Adres>RównieżNieMa 3 m 8</qgs:Adres>
 <qgs:Ocena>3</qgs:Ocena>
 <qgs:Nazwa>Cafe2</qgs:Nazwa>
 <qgs:DataOtwarcia>2012-01-01</qgs:DataOtwarcia>
```

Other popular data types

- GeoJSON - basically like GML - only a JSON instead of XML
- Shapefile - a format maintained by ESRI - one of the main commercial organizations in the GIS world
- GeoTIFF - geotagged TIFF images

Storing spatial data in databases I

Standards

- Simple Feature Access (Simple Features) - OGC standard and ISO 19125:2004 norm
- SQL Multimedia and Application Packages (SQL/MM) - ISO/IEC 13249-3:2011 SQL/MM Spatial standard

Contain

- Definitions of storing and presenting data as text (WKT) and binary data (WKB)
- Definitions of possible spatial operations

Storing spatial data in databases II

Example

- (E)WKB - `0101000020E61000009A9999999999F13F00000000000029C0`
  Byte order (little/big endian), Type (01 = point), Coordinate Transformation System, First coordinate, Second coordinate.
- (E)WKT - `SRID=4326;POINT(1.1 -12.5)`

RDBMS systems implementing SQL/MM Spatial

- MS SQL Server (from 2008 version)
- Oracle Spatial (addition to Enterprise)
- **PostGIS (extension of PostgreSQL)**
- and others (including MySQL or SQLite)

Object types

- Points - `POINT(-118.4079 33.9434)`
- Lines - `LINESTRING(-118.4079 33.9434, 2.5559 49.0083)`
- Figures - `POLYGON((0 0, 10 0, 10 10, 0 10, 0 0),(1 1, 1 2, 2 2, 2 1, 1 1))`
- Solids - `POLYHEDRALSURFACE Z ( ((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)) ))`

Data types

- Geometry - spatial objects considered in a Cartesian coordinate system
- Geography - spatial objects considered to be on a sphere defined by WGS 84 coordinate system

Spatial operations

Set theory operations

- Intersection
- Union
- Inclusion
- Buffer

Geometrical operations

- Distance
- Area

Others

- Conversion from/to KML
- Konwersja from/to GMLa

Example of a spatial SQL query I

Source 1 Source 2

Create data

```
 CREATE TABLE augis.dane
(
  point geometry(Point,4326),
  line geometry(LineString,4326),
  geopoint geography(Point)
)
```

Example of a spatial SQL query II

Adding points

```
 INSERT INTO augis.bus_stops
  (name, point)
VALUES (
  'Politechnika_01',
  ST_GeometryFromText('POINT(21.0113 52.2200)',4326)
)
```

Example of a spatial SQL query III

Spatial search

```
 SELECT augis.roads.name
FROM augis.roads
WHERE ST_Intersects(
   (SELECT point
     FROM augis.bus_stops
     WHERE augis.bus_stops.name = 'Politechnika_01'),
   line);
```

Example of a spatial SQL query IV

Spatial measurements

```
 SELECT
   augis.roads.name as street,
   ST_Length(line::geography) as length,
   augis.bus_stops.name as bus_stop,
   ST_Distance
     (point,
     line) as cartesian,
   ST_Distance
     (point::geography,
     line::geography) as meters
FROM augis.roads, augis.bus_stops;
```

street character varying	length double precision	bus_stop character varying	cartesian double precision	meters double precision
Niepodległości	308.796863269	Politechnika 01	0.00648183012	443.154371035
Nowowiejska	443.154371034	Politechnika 01	0	0

Selected spatial operations

- ST_Union(geometry),ST_Union(geometry A, geometry B), ST_Union(geometry[ ])
- ST_Difference(geometry A, geometry B)
- ST_SymDifference(geometry A, geometry B)
- ST_Intersection(geometry A, geometry B)
- ST_Buffer(geometry A, float distance)
- ST_ConvexHull(geometry A)
- ST_Simplify(geometry A, float tolerance)

Selected spatial data operations *geography*

- `ST_AsSVG(geography)`
- `ST_AsGML(geography)`
- `ST_AsKML(geography)`
- `ST_AsGeoJson(geography)`
- `ST_Distance(geography, geography)`
- `ST_Area(geography)`
- `ST_Length(geography)`
- `ST_Transform(geography, srid)`

Selected spatial queries

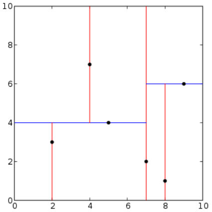
- `ST_Intersects(geometry A, geometry B)`
- `ST_Touches(geometry A, geometry B)`
- `ST_Crosses(geometry A, geometry B)`
- `ST_Disjoint(geometry A, geometry B)`
- `ST_Contains(geometry A, geometry B)`
- `ST_Within(geometry A, geometry B)`
- `ST_DWithin(geometry A, geometry B, float radius)`

Spatial indexes

- Efficient data retrieval is a really important feature of a database
- Spatial data call for another method of indexing
- How can we improve spatial operations without spatial indexing?
  - You can compute four bounding values on X and Y axes and build a classic index upon them.
  - This should improve finding intersections, but not necessarily nearest neighbors.
- In practice special types of data structures are involved: k-d trees, **R-trees**, X-trees, . . .

k-d tree

- Easy to implement
- Tree levels are created by repetitive iterating over subsequent dimensions
- It might be imbalanced



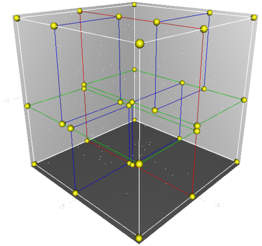
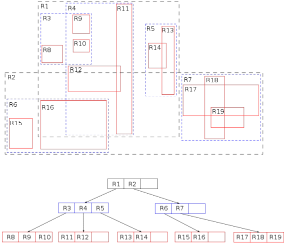


Image source: en.wikipedia.org

R-tree

- Always balanced
- Pages are enhanced with the minimizing hyper-rectangles volume in mind
- In higher dimensions hyper-rectangles overlap might tamper its performance



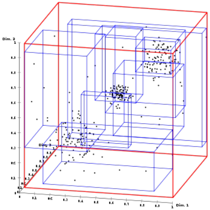


Image source: en.wikipedia.org

B-trees / B+trees

- Always balanced
- Ordered elements on the pages
- Adjusted for data storage on hard-drive
- Values stored in leaf nodes (in B+tree)

B-tree operations (M = 2) I

Image source: de.wikipedia.org

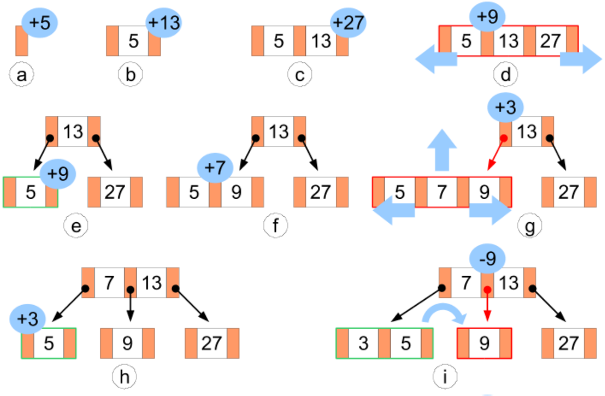
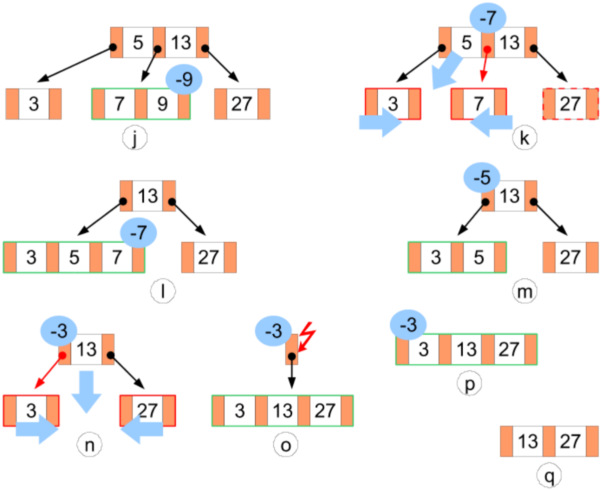


Image source: de.wikipedia.org

B-tree operations (M = 2) II



R-trees features

- Adjusted for data storage on hard-drive
- Elements on the pages are not ordered
- Pages store bounding hyper-rectangles
- Spatial objects are in leaf nodes

R-tree operations I

The following operations are supported by R-trees

- Search for intersecting objects
- Search for nearest neighbors
- Adding/removing elements
- **Tree restructuring**

Search for intersecting objects

At every level rejecting all hyper-rectangles which do not overlap with bounding hyper-rectangle of a search object.
Problematic data - long objects not aligned with coordinate system.

Note to self: draw it on board.

R-tree operations II

Inserting new element

On every level (which is not a leaf) we find a hyper-rectangle which will be the least extended by expanding its boundary to encapsulate this additional object

Rebuilding trees

This might be a costly operation due to not ordered element placement on a single page. Though, it is performed only when the tree needs expanding or contracting. strony).

Podział strony I

All possible division of hyper-rectangles

Checking all possible divisions of hyper-rectangles into two subsets. Selecting the one in which hyper-rectangles volume is the lowest.
Exponential complexity against the page size.

Selecting the best pair

Iterate through all pairs of hyper-rectangles. Selecting such a pair which has the highest difference between overlapping hyper-rectangle and the sum of hyper-rectangles. Proceed by selecting those with smallest raise in the overlapping hyper-rectangle volume.
Square complexity (against page size).

Podział strony II

Simplified pair selection

Selecting a pair of hyper-rectangles which are furthest away from one another in each dimension. Selecting such a pair as the one for making a division (and proceed as earlier).
Linear complexity (against page size).

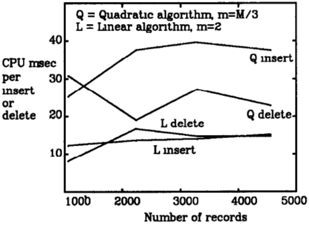


Figure 4 7
CPU cost of inserts and deletes
vs amount of data

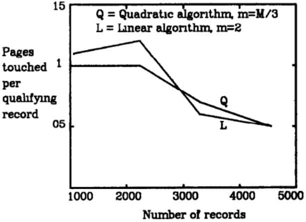
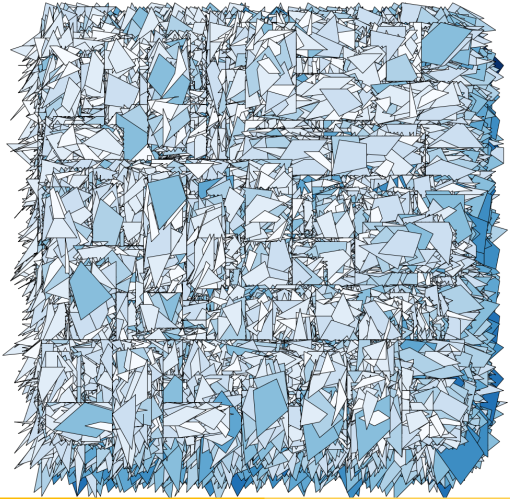


Figure 4 8
Search performance vs amount of data·
Pages touched

Utilizing spatial index in PostgreSQL

Example: 100 000 random figures.



Utilizing spatial index in PostgreSQL I

Creating table

```
CREATE TABLE data.wielokaty
(
  id serial primary key,
  poly geometry(Polygon)
);
```

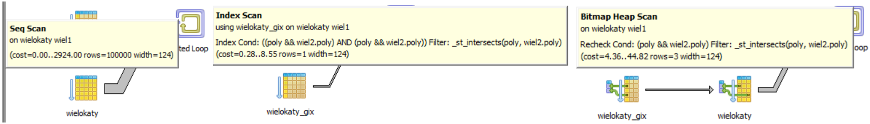
Creating index

```
CREATE INDEX wielokaty_gix
  ON data.wielokaty USING GIST (poly);
VACUUM ANALYZE data.wielokaty;
```

Utilizing spatial index in PostgreSQL II

Query for utilizing spatial index

```
SELECT wiel1.id
  FROM data.wielokaty wiel1, data.wielokaty wiel2
  WHERE
    wiel1.poly && wiel2.poly AND
    ST_INTERSECTS(wiel1.poly, wiel2.poly) AND
    wiel2.id = 300403
```



More info