Prof. Dr. Florian Heinz
florian.heinz@oth-regensburg.de

Modern Database Concepts

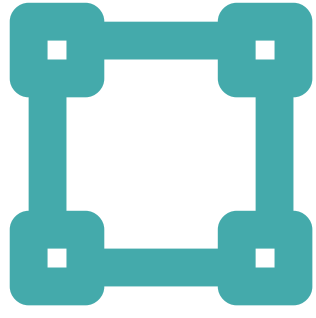
Spatial Data

OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

Spatial Data

Representation of real-world objects

Geometric objects in Euclidean space



Geographic Information System (GIS)

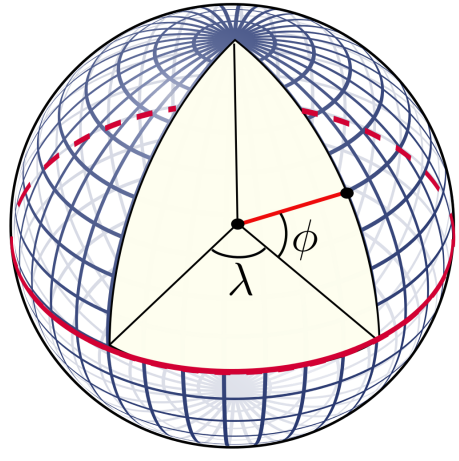
A system for collecting, storing, analyzing and disseminating information about areas of the earth.

Many applications store spatial data: maps, locations of customers / stores / app users / ..., integrated-circuit design, ... In general, this data is based on geometries: points, linestrings, polygons, etc. in an 2D or 3D Euclidean space. Traditional GIS are cartography, government, climatology applications. Business GIS consist of enterprise data with geospatial references (locations).

Coordinate Systems

Geographic

On surface of the earth: Latitude Φ and Longitude λ
Unit: degrees °



Projected

Projected to 2D space; many different projection algorithms

Abstract coordinate systems

Not related to Earth's surface (e.g., location on an image); 2D (x, y) or 3D (x, y, z)

Projection of spheroid coordinates onto another shape (e.g. cylinder) is required to construct 2D maps and simplifies computations. But projection also causes inaccuracies and other problems (e.g. scale distortion at the poles). This is why multiple coordinate reference systems exist.

Spatial Reference Systems

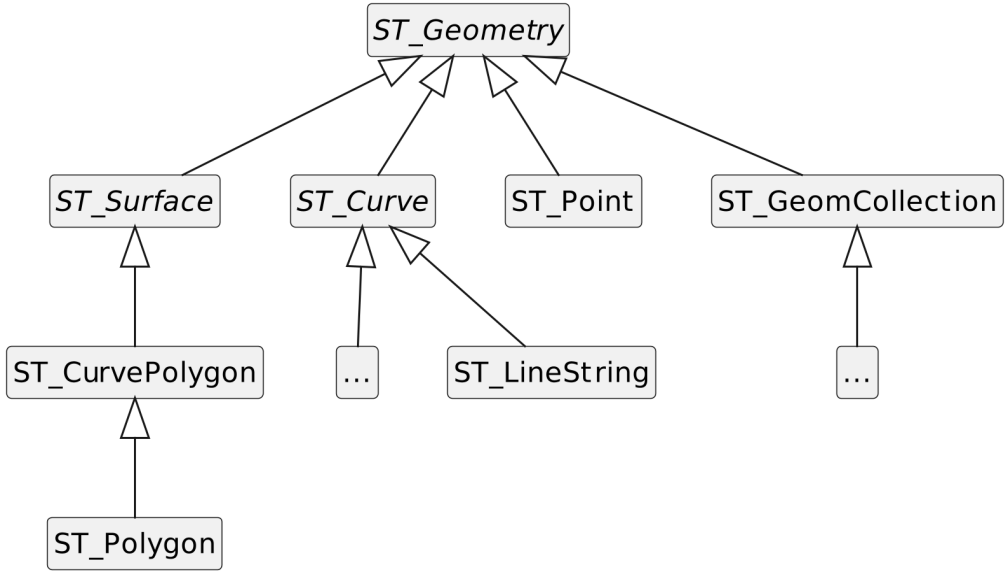
Local, regional or global system to locate geographical entities.

SRID	Spatial Reference System	Description
4326	WGS-84	World-wide; used by GPS, ... (unit: degrees)
31466	DHDN / Gauß-Krüger Zone 2	Germany (west of 7.5° E)
31467	DHDN / Gauß-Krüger Zone 3	Germany (between 7.5° E and 10.5° E)
31468	DHDN / Gauß-Krüger Zone 4	Germany (between 10.5° E and 13.5° E)
31469	DHDN / Gauß-Krüger Zone 5	Germany (between 13.5° E and 16.5° E)
3857	WGS 84 / Pseudo-Mercator	Used by Google Maps, OpenStreetMap, ... (unit: metre)

There are a lot of different spatial reference systems. There are identified by an SRID (Spatial Reference Identifier). Often, coordinates have to be translated from one reference system into another. WGS-84 uses degrees as units. Often latitude and longitude values are provided in degrees. But, distances and radiuses are often in kilometers. So, coordinates have to be converted into a reference system that uses meters, e.g. SRID 3857.

SQL/MM Spatial

GEOMETRY data type



Routines

Creation, manipulation, retrieval, comparison, conversion

- `ST_EQUALS(g1, g2)`
- `ST_INTERSECTS(g1, g2)`
- `ST_INTERSECTION(g1, g2)`
- `ST_BUFFER(g, dist)`
- `ST_AREA(surface)`
- `ST_X(point)`
- `ST_DISTANCE(g1, g2)`
- …

The SQL/MM Spatial 2003 standard defines data types and routines for working with spatial data. The non-italic data types in the type hierarchy are instantiable. Subtypes of ST_GeomCollection are for example ST_MultiPoint, ST_MultiLineString or ST_MultiPolygon. They are a multiset of the underlying data type.

0-dimensional geometries

ST_Point

```
POINT(5 5)
```

ST_MultiPoint

```
MULTIPOINT(1 1, 1 3, 3 3)
```

Format used here: **WKT** (well-known text)

Routines for points:

- `ST_X(p)`, `ST_Y(p)` for coordinate access

A point represents a single location and has an x and y coordinate.
The well-known text format is often used to describe a geometry object via a string. Other formats are WKB (well-known binary), GML (geographic markup language), or GeoJSON. In SQL/MM Spatial, there are constructors for WKT, WKB, and GML for each geometry data type.

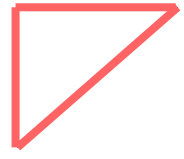
1-dimensional geometries

Represented by a sequence of points (*closed* if start point = end point)

ST_LineString

```
LINESTRING(1 1, 1 3, 3 3)
```

```
LINESTRING(1 1, 1 3, 3 3, 1 1)
```



Routines for curves:

- `ST_StartPoint(c)`, `ST_EndPoint(c)` → ST_Point
- `ST_NPoints(c)` → Integer (number of points)
- `ST_PointN(c, n)` → ST_Point (n-th point)
- `ST_Length(c)` → Float
- `ST_IsClosed(c)` → Boolean

Curves are defined by a sequence of points. For linestrings, linear interpolation between two consecutive points is used. ST_CircularString uses circular interpolation.

2-dimensional geometries

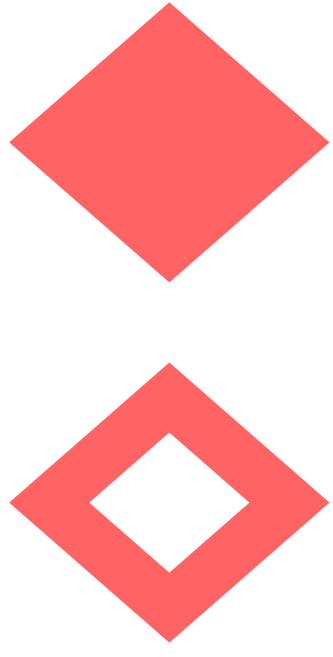
Boundaries are one or more closed curves

ST_Polygon

1 external closed LineString + n holes

```
POLYGON((0 0, 1 1, 2 0, 1 -1, 0 0))
```

```
POLYGON((0 0, 1 1, 2 0, 1 -1, 0 0),
(0.7 0, 1 0.3, 1.3 0, 1 -0.3, 0.7 0))
```



Routines for surfaces:

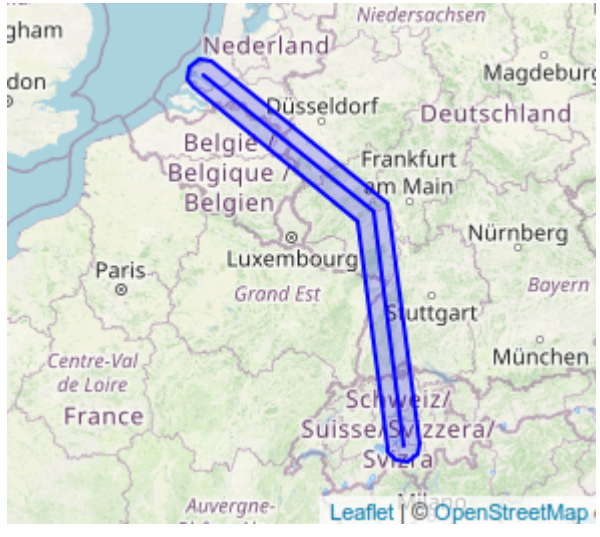
- `ST_Area(s)`,`ST_Perimeter(s)` → Float
- `ST_Centroid(s)` → `ST_Point`

Surfaces are also defined by a sequence of points. If the surface has holes in it, it is a sequence of multiple curves. ST_CurvePolygon is a generalized surface, for an ST_Polygon, the boundaries are linestrings.

Other Routines

Routines for all geometries:

- `ST_Envelope(g)` → ST_Polygon (bounding rectangle)
- `ST_Buffer(g, dist)` → ST_Geometry
  (encircles geometric at a specific distance)
- `ST_Difference(g1, g2)`,
  `ST_Intersection(g1, g2)`,
  `ST_Union(g1, g2)` → ST_Geometry
- `ST_Intersects(g1, g2)`,
  `ST_Crosses(g1, g2)`,
  `ST_Overlaps(g1, g2)`,
  `ST_Touches(g1, g2)`,
  `ST_Contains(g1, g2)`,
  `ST_Within(g1, g2)` → Boolean
- `ST_IsSimple(g)` → Boolean (no self-intersection)



The image on the right shows a 20km buffer around a line string.

PostGIS

Spatial and Geographic Objects for PostgreSQL.

```
CREATE TABLE buildings (
customer_name VARCHAR(50) PRIMARY KEY,
geo GEOMETRY(POINT, 4326));
```

```
INSERT INTO buildings VALUES ('Rita',
ST_GEOMFROMTEXT('POINT(12.096944 49.017222)', 4326));
INSERT INTO buildings VALUES ('Mary',
ST_GEOMFROMTEXT('POINT(7.221258 50.329615)', 4326));
```

```
SELECT ST_Distance(ST_Transform(r.geo, 3857),
                   ST_Transform(w.geo, 3857))/1000     -- in km
FROM buildings r, buildings w
WHERE r.customer_name = 'Rita'
AND w.customer_name = 'Mary';
```

PostGIS is an extension for PostgreSQL that offers a `GEOMETRY` data type and the `ST_%` functions. In this example, we use the spatial reference system with SRID 4326. It uses the unit degrees for longitude and latitude values. To retrieve a distance in meters (or kilometers), we need to transform the coordinates into SRID 3857. The query returns a distance of 587 km between Rita's and Mary's cities Regensburg and Mayen. Without the transformation, the query would return a distance of 5.05 (degrees).

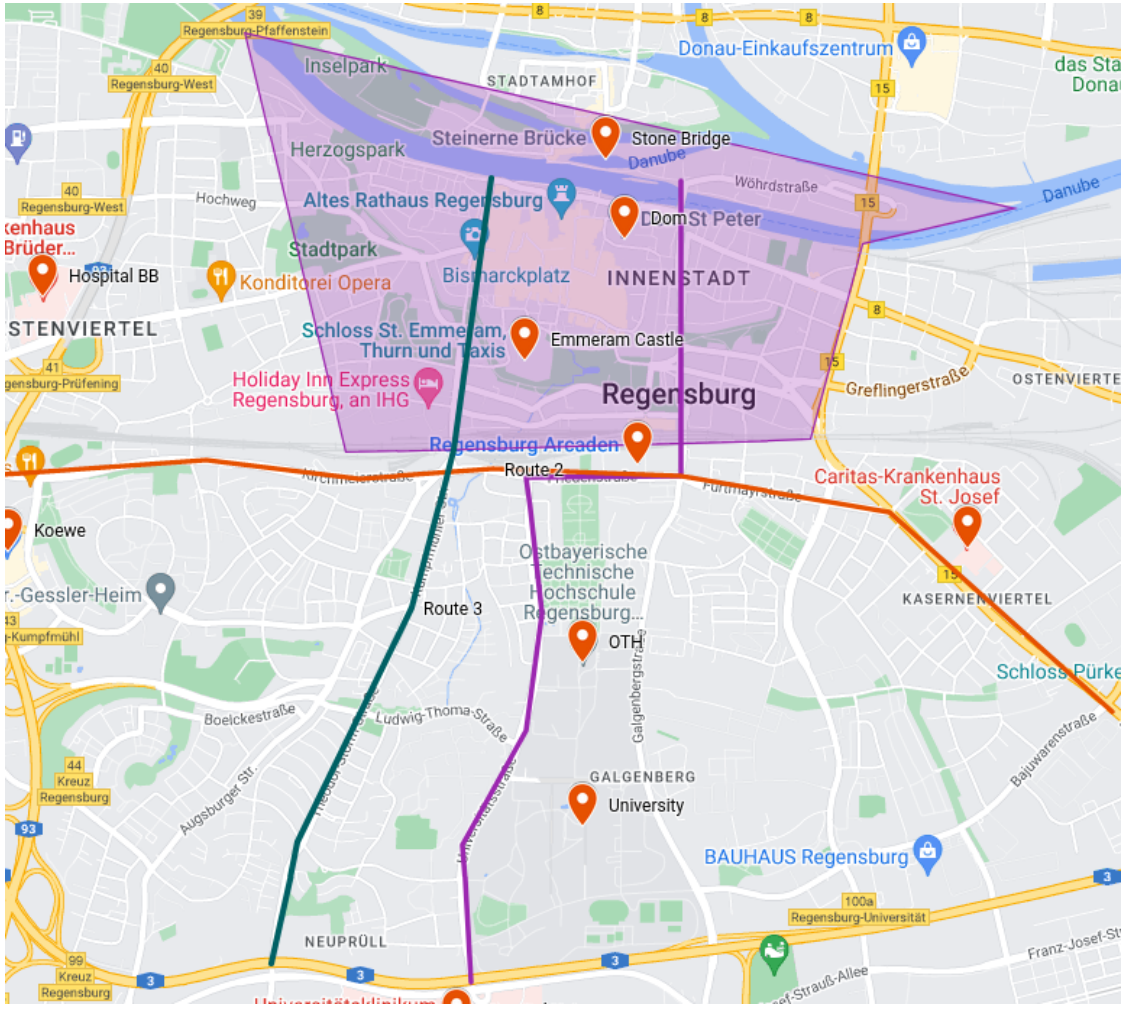
Regensburg example

Objects:

10 POIs

3 routes

1 area



Regensburg example

Schema:

```
CREATE TABLE poi (
       name text PRIMARY KEY,
       location GEOGRAPHY(POINT, 4326)
);

CREATE TABLE areas (
       name text PRIMARY KEY,
       area GEOGRAPHY(POLYGON, 4326)
);

CREATE TABLE routes (
       name text PRIMARY KEY,
       route GEOGRAPHY(LINESTRING, 4326)
);
```

Regensburg example

POIs:

```
INSERT INTO poi VALUES
('Arcaden',       ST_GEOMFROMTEXT('POINT(12.0990973 49.0106664)', 4326)),
('Koewe',         ST_GEOMFROMTEXT('POINT(12.0617609 49.0072884)', 4326)),
('OTH',           ST_GEOMFROMTEXT('POINT(12.0958357 49.002953 )', 4326)),
('Hospital BB',   ST_GEOMFROMTEXT('POINT(12.0638209 49.0171965)', 4326)),
('Hospital Uni',  ST_GEOMFROMTEXT('POINT(12.0883684 48.9885927)', 4326)),
('Hospital SJ',   ST_GEOMFROMTEXT('POINT(12.1186667 49.007401 )', 4326)),
('Dom',           ST_GEOMFROMTEXT('POINT(12.0983248 49.0194481)', 4326)),
('Stone Bridge',  ST_GEOMFROMTEXT('POINT(12.0971924 49.0225646)', 4326)),
('Emmeram Castle',ST_GEOMFROMTEXT('POINT(12.0923859 49.0147405)', 4326)),
('University',    ST_GEOMFROMTEXT('POINT(12.0958191 48.9966107)', 4326));
```

Regensburg example

Routes:

```
INSERT INTO routes VALUES
('Route 1', ST_GEOMFROMTEXT('LINESTRING(12.0893674 48.9905156,
 12.0888524 48.9958658,12.092629 49.0003144,12.0935731 49.0048753,
 12.092629 49.010167,12.1018128 49.010224,12.1018128 49.0217075)',4326)),
('Route 2', ST_GEOMFROMTEXT('LINESTRING(12.0614724 49.0102802,
 12.0736604 49.0108432,12.0827584 49.0101113,12.0906548 49.0105054,
 12.1018128 49.010223,12.1141725 49.008816,12.1273904 49.001046)',4326)),
('Route 3', ST_GEOMFROMTEXT('LINESTRING(12.0775227 48.9912478,
 12.0790677 48.9959784,12.0858483 49.0051005,12.0883374 49.0112373,
 12.090569 49.0218201)', 4326));
```

Areas:

```
INSERT INTO areas VALUES ('Innenstadt', ST_GEOMFROMTEXT('
 POLYGON((12.075892 49.0274926,12.0819001 49.0111128,
          12.1095376 49.0116195,12.1126275 49.0192189,
          12.1218114 49.0206261,12.075892 49.0274926))', 4326));
```

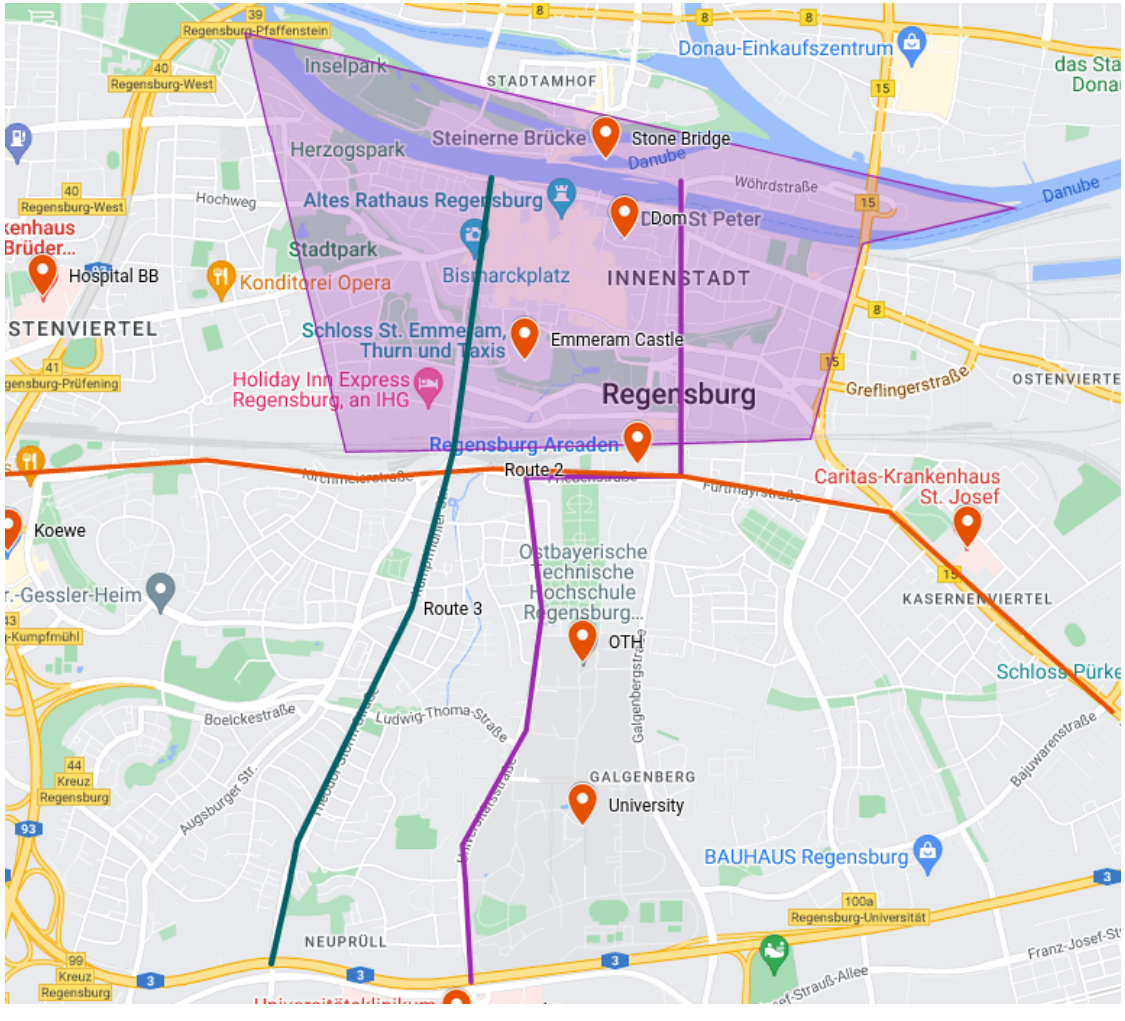
Regensburg example

Objects:

10 POIs

3 routes

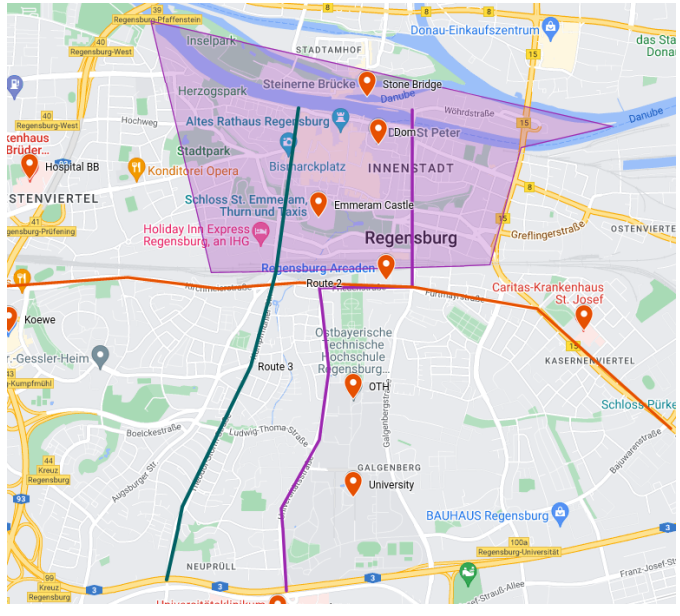
1 area



Regensburg example

Some operations:

- Which POIs are in a 1000m radius from "Dom"?
- Which POIs are in "Innenstadt"?
- Which part of "Route 1" is in "Innenstadt"?
- Where do "Route 2" and "Route 3" intersect?
- Where is the point of closest approach of "Route 2" to Innenstadt
- What is the shortest connection between "Route 1" and "Route 3"?
- Which POIs are in a 300m vincity around Route 1?
- Which POIs have the shortest distance at all?
- What is the minimal enclosing polygon of all POIs?



Insurance Example

```
CREATE TABLE rivers (
 name VARCHAR(30) PRIMARY KEY,
 river_line GEOMETRY(LineString, 4326),
 flood_zones GEOMETRY(Polygon, 4326));
```

```
INSERT INTO rivers (name, river_line) VALUES ('Rhine',
 ST_GEOMFROMTEXT('LINESTRING(8.561667 46.559167,
                             7.895 49.966944,
                             4.179428 51.861992)', 4326));
```

Extend the flood zones for the rivers by 3 kilometers:

```
UPDATE rivers SET flood_zones = ST_Transform(
ST_BUFFER(ST_Transform(river_line, 3857), 3000), 4326);
```

Find all customers that have a building in the flood zones:

```
SELECT customer_name FROM buildings b, rivers r
WHERE ST_Within(b.geo, r.flood_zones)
```

Source: STOLZE, Knut. SQL/MM spatial: The standard to manage spatial data in a relational database system. In: BTW 2003

In reality, the river Rhine has a more complex shape and is not close to the town Mayen ;-)

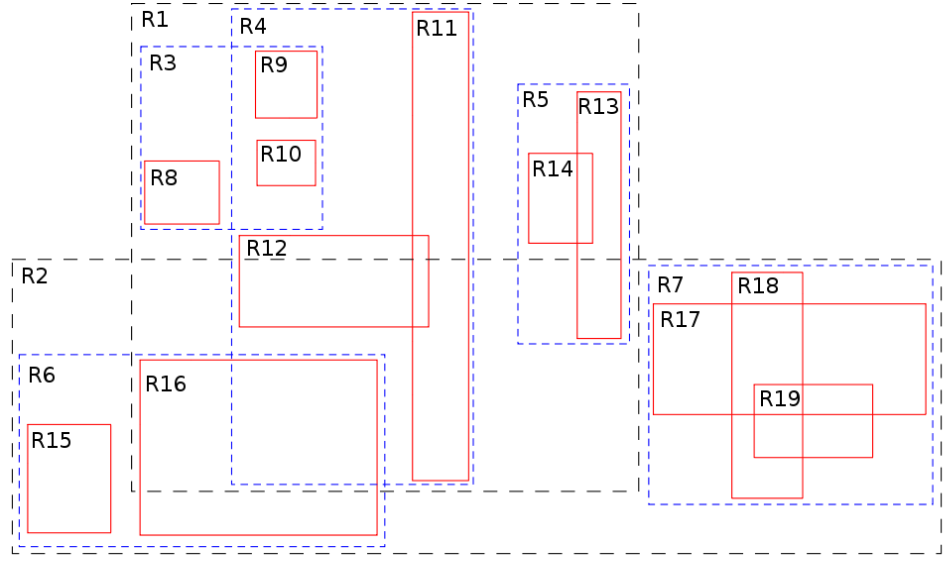
Geo-Indexes

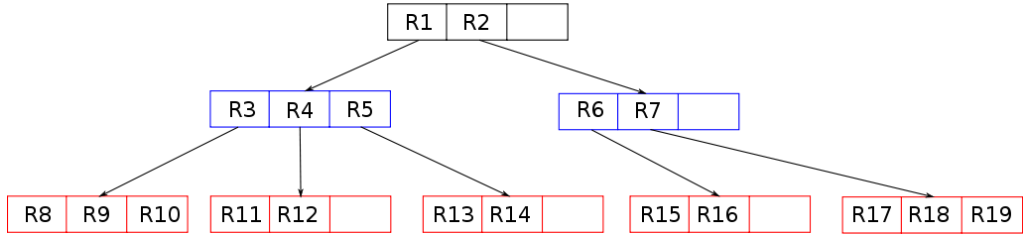
R-Tree

Leaves: minimum bounding rectangles of the geo-objects

Inner nodes: minimum bounding rectangles of the child nodes

Example:
R1 = (10,100), (100,5))
R2 = (0,50), (170,0))
Find objects in ((120, 100), (150, 0)):
⇒ continue search only in R2





R-Tree stands for Rectangle tree and is a multi-dimensional index structure, not only for geo data. For indexing geo data, we compute minimum bounding rectangles ((x1, y1), (x2, y2)) -- the coordinates of its top left and bottom right corner. After that, be build a tree that hierarchically organizes nearby rectangles within a larger rectangle. A search query starts at the root and inspects all children with overlapping areas.

Geo-Indexes in PostgreSQL

```
CREATE INDEX buildings_geo_idx ON buildings USING GIST (geo);
```

Find all buildings within a 10 km radius from Regensburg

```
SELECT * FROM buildings WHERE ST_WITHIN(geo,
ST_Transform(ST_BUFFER(ST_Transform(
ST_GEOMFROMTEXT('POINT(7.22 50.32)', 4326), 3857), 10000), 4326))
```

Geo-index can be used for filters or geo-joins using binary predicates like ST_Within, ST_Overlaps, ST_Intersects, ... The index would not be used for a query that uses `WHERE ST_DISTANCE(...) <= 10000`. The index on this slide can also be used for the query that finds buildings in the flood zones of our rivers.

Summary

- GIS
- Coordinate Systems, SRID
- SQL/MM Spatial: Data Types and Routines
- `ST_Point`, `ST_LineString`, `ST_Polygon`, ...
- WKT
- PostGIS
- Geo-Indexes: R-Trees