Spatial Databases via PostGIS
DD2471
KTH

Michael Minock

Tuesday, May 8, 2012

Organization of lecture

- What is a spatial database?
- Query types, indexes and evaluation strategies

- The PostGIS project
- Examples, examples, examples...
- Some fun ideas, starting points, etc...

What is a spatial database?

For our purposes, it is a *relational database* that models entities in space.

- Extends attribute types beyond traditional types (e.g. `varchar`, `integer`,...) system to geometric primitives (e.g. `point`, `polygon`,...)
- Supports *ad hoc* querying (e.g. some extension to SQL)
- Inherits all the other things that come along with databases (e.g. transactions, persistence, recovery, etc.)
- Supports and uses spatial indexes
- By these requirements, many stand alone GIS visualization tools (e.g. Arcinfo) are not spatial databases

Example spatial database application areas

- Cartographic, map-based data is often 2-dimensional, used within so called *geographical information systems* (GIS) applications.
- *Computer assisted design* (CAD) and *virtual world* data is often three dimensional and consists of wire frames with associated surface properties, etc.
- Scientific data is often 3 spatial and 1 time dimensional and might have non-Cartesian (e.g. spherical, relativistic) coordinate systems.
- Image understanding systems often represent image content in a layering of forms: raw pixels, spectral coefficients, edges, segmented regions, camera and objects in three space.
- ...

Some terminology

Are entities (objects, regions, etc.) represented *discretely* or *continuously*?

- Discrete representations are composed of pixels (or voxels) within the mesh of a two (or three) dimensional grid. Also known as *raster* based.
- Entities represented continuously are usually composed of two (or three) dimensional geometric primitives. Also known as *vector* based.
- A *layered* model describes entities in the same space with both vector and raster representations. Mappings are accomplished via *interpolation* and *discretization*.

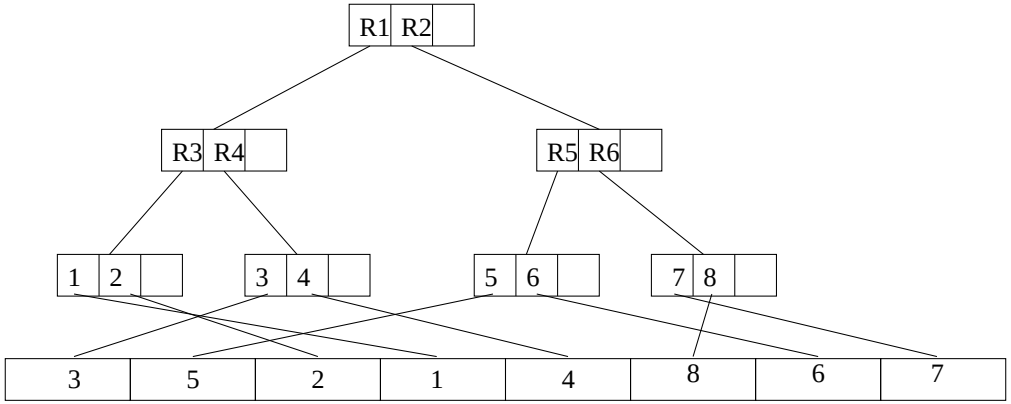
Classic spatial query types

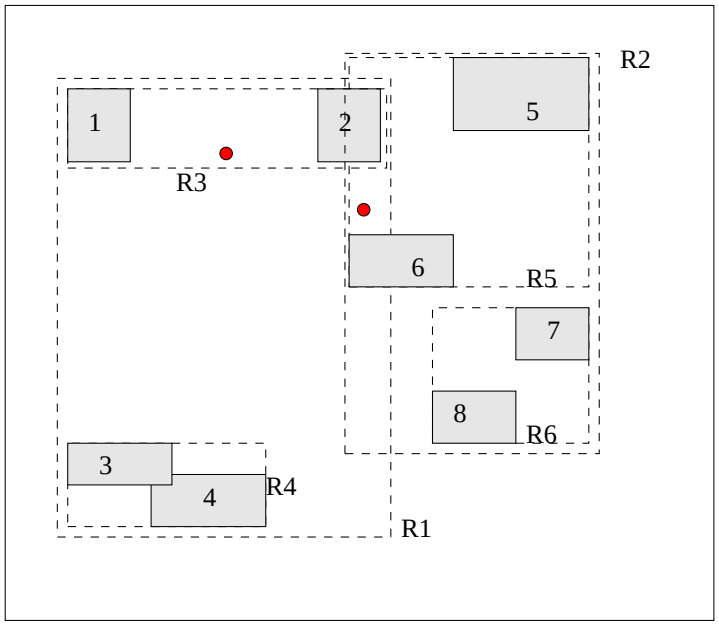
- **Where am I queries**:
  Describe the objects 'associated' with a point in space. (e.g. *What building am I in?*)
- **Range Queries**:
  Find objects of a given type within a specific geographical area or distance from a particular location. (e.g. *all bars within 200 meters of KTH's metro station*)
- **Spatial Joins**:
  Joins objects of two types based on a spatial condition. (e.g. *all bars within 100 meters of a metro station*
- *k*-**Nearest Neighbor Queries**:
  Finds the *k* objects of a particular type that are closest to a given point. (e.g. *what is nearest metro station?*)

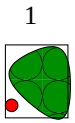
Spatial indexes

- Most spatial indexes use the notion of *spatial occupancy* within 'easy to calculate' bounding boxes
- Solving for an answer set involves the two step process of first *filter* and then *refine*
- The trick is to make sure the index gets used is a smart way by the query optimizer
- Usual method is to use an R-Tree Index (Guttman, 1984)

An example R-Tree







Quick points about R-Trees

- 'Where am I?' queries are trivial
- Searches may follow multiple branches
- On inserts, add to child that will minimize rectangle area growth
- On over-fill split so as to minimize total area and recur to next higher level as new index record added to internal node
- On underfill, typically do a series of re-inserts
- Use *packed* R-Trees for static data

Spatial indexes for distance-based range query

- Filter on bounding box
- Refine with actual distance computation
  - Works for Euclidean distance
  - Requires ‘distance’ can be translated to a containing bounding box.

Spatial indexes for spatial joins

"the churches less than 100 meters from a bar."

- Iterate through the churches
- For each church calculate the bars that are within 100 meters

Spatial indexes for k-nearest neighbors

- A series of expanding range queries (δ governed by parameter that bounds the total number of range queries)
- (Alternative) walk the R-Tree upward increasing the bounding rectangle until k-nearest neighbors are located.
- (in both) beware to apply the refine step!!!

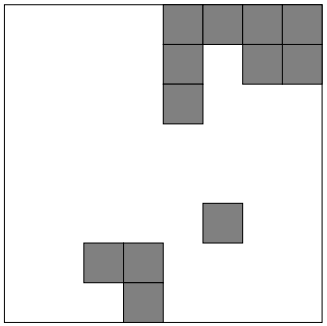
Some more advanced types of queries

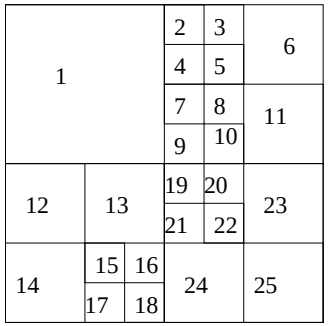
- **Route planning**:
  - In general not first order expressible.
  - Dijkstra's algorithm vs. A*
- **Network distance**:
- **Viewability**:
  - Isovists
  - Viewsheds based on 2.5 height map raster data
  - An active research area! (see `spacebook-project.eu`)

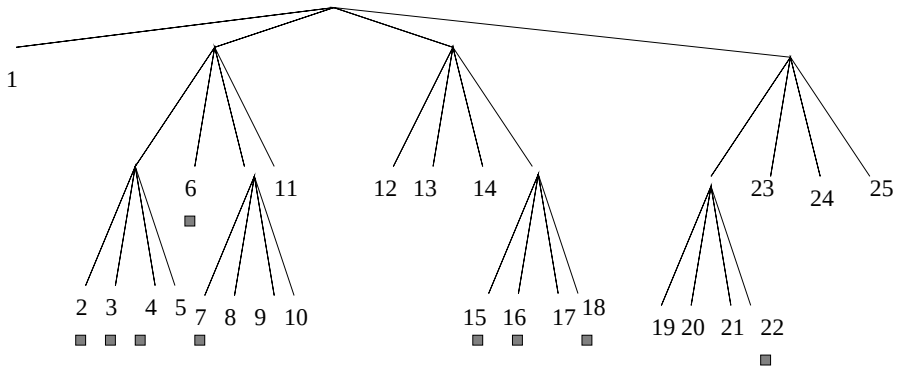
Rasters

- Commonly with regular grid-based mesh
- In multiple layers representing (e.g. height categories, rainfall categories, land cover type, etc)
- Often represented in quad trees (Klinger, 1971)
  - compression
  - fast set operations
  - can be seen as a type of 'compressed bit map index'.

Quad trees

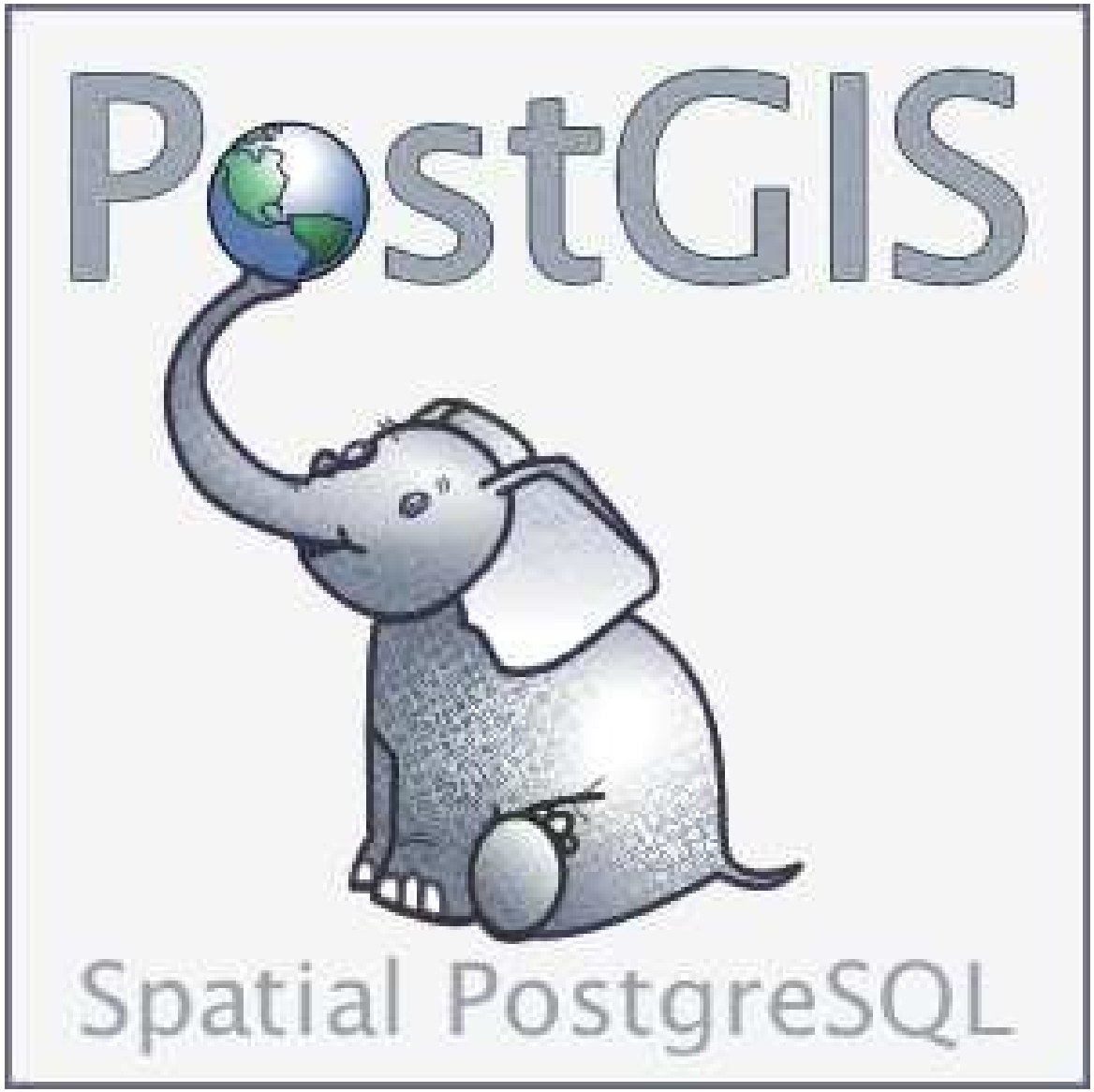






(0 ((1110) 1 (1000) 0) (000 (1101)) ((0001)000)

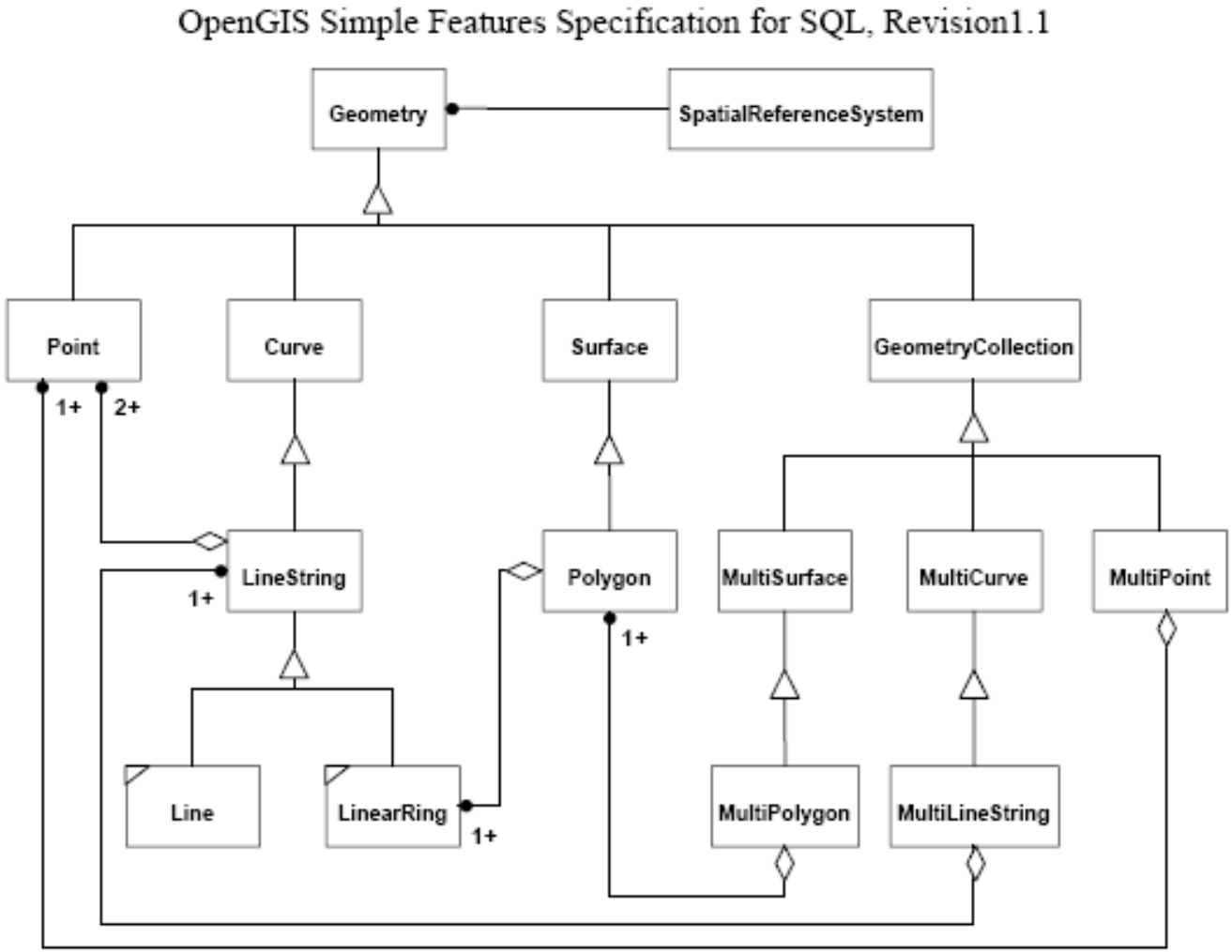
PostGIS



OpenGIS

- Supported by major commercial vendors (Oracle, IBM, etc.)
- Simple types specification
- Basic operators

OpenGIS simple types



OpenGIS operators

OGC spatial operators defined on the class geometry

Classes	*Operators*	*Operator Functions*
Basic Operators	Spatial Reference	Returns the reference system of the geometry
	Envelope	Returns the minimum bounding rectangle of the geometry
	Export	Converts the geometry into a different representation
	IsEmpty	Tests if the geometry is the empty set or not
	IsSimple	Returns TRUE if the geometry is simple
	Boundary	Returns the boundary of the geometry
Topological Operators	Equal	Tests if the geometries are spatially equal
	Disjoint	Tests if the geometries are disjoint
	Intersect	Tests if the geometries intersect
	Touch	Tests if the geometries touch each other
	Cross	Tests if the geometries cross each other
	Within	Tests if a geometry is within another geometry
	Contain	Tests if a given geometry contains another geometry
	Overlap	Tests if a given geometry overlaps another given geometry
	Relate	Returns TRUE if the spatial relationship specified by the 9-Intersection matrix holds
Spatial Analysis Operators	Distance	Returns the shortest distance between any two points of two given geometries
	Buffer	Returns a geometry that represents all points whose distance from the given geometry is less than or equal to a specified distance
	ConvexHull	Returns the convex hull of a given geometry
	Intersection	Returns the intersection of two geometries
	Union	Returns the union of two geometries
	Difference	Returns the difference of two geometries
	SymDifference	Returns the symmetric difference (i.e. the logical XOR) of two geometries

PostGIS project

- PostGIS "spatially enables" PostgreSQL
- Follows the OpenGIS "Simple Features Specification for SQL" - certified as compliant with the "Types and Functions" profile.
- Developed by Refractions Research starting in 2001
- Version 2.0 just released!

PostGIS OpenGIS types

```
POINT(0 0)

LINESTRING(0 0,1 1,1 2)

POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1, 2 1, 2 2, 1 2,1 1))

MULTIPOINT(0 0,1 2)

MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4))

MULTIPOLYGON(((0 0,4 0,4 4,0 4,0 0),
(1 1,2 1,2 2,1 2,1 1)),
((-1 -1,-1 -2,-2 -2,-2 -1,-1 -1)))

GEOMETRYCOLLECTION(POINT(2 3),LINESTRING(2 3,3 4))
```

PostGIS extended types (part 1)

```
POINT(0 0 0) -- XYZ

SRID=32632;POINT(0 0) -- XY with SRID

POINTM(0 0 0) -- XYM

POINT(0 0 0 0) -- XYZM

SRID=4326;MULTIPOINTM(0 0 0,1 2 1) -- XYM with SRID

MULTILINESTRING((0 0 0,1 1 0,1 2 1),(2 3 1,3 2 1,5 4 1))

POLYGON((0 0 0,4 0 0,4 4 0,0 4 0,0 0 0),(1 1 0,2 1 0,2 2 0,1 2 0,1 1 0)

MULTIPOLYGON(((0 0 0,4 0 0,4 4 0,0 4 0,0 0 0),
  (1 1 0,2 1 0,2 2 0,1 2 0,1 1 0)),
  ((-1 -1 0,-1 -2 0,-2 -2 0,-2 -1 0,-1 -1 0)))
```

PostGIS extended types (part 2)

```
GEOMETRYCOLLECTIONM( POINTM(2 3 9), LINESTRINGM(2 3 4, 3 4 5) )

MULTICURVE( (0 0, 5 5), CIRCULARSTRING(4 0, 4 4, 8 4) )

POLYHEDRALSURFACE(
 ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)),
 ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)),
 ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
 ((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
 ((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)),
 ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)) )

TRIANGLE ((0 0, 0 9, 9 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)) )
```

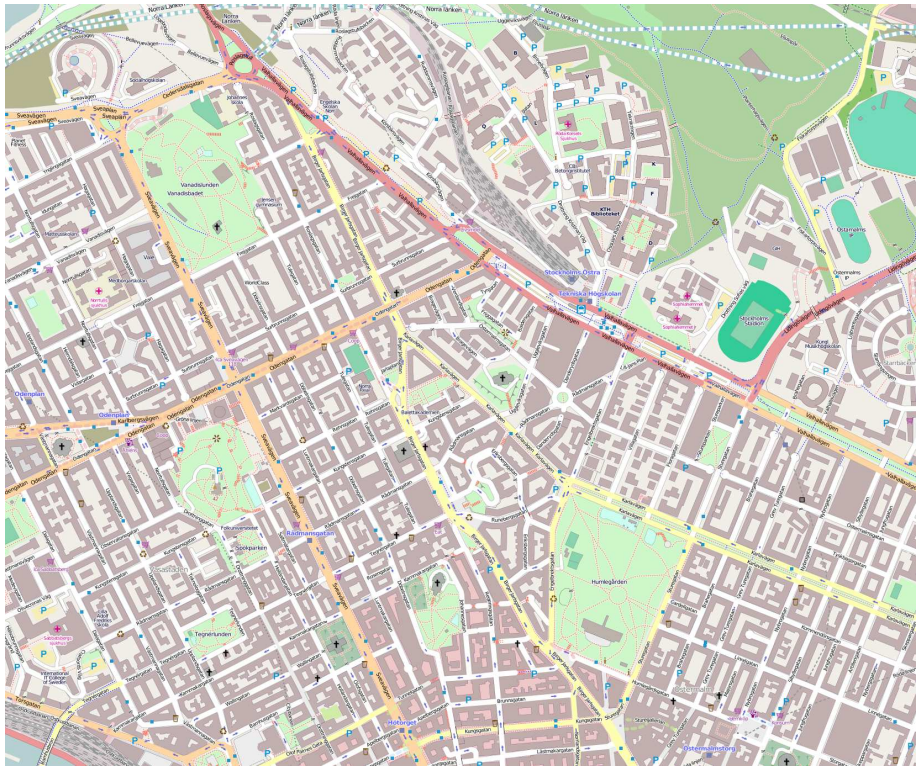
PostGIS creating a database

After installing PostGIS:

```
createdb kth

createlang plpgsql kth

psql -d kth -f /usr/share/postgresql/8.4/contrib/postgis.sql
psql -d kth -f /usr/share/postgresql/8.4/contrib/postgis_comments.sql
psql -d kth -f /usr/share/postgresql/8.4/contrib/spatial_ref_sys.sql

psql -d kth -f kth.sql
```

Our example build: kth.sql

- Exported from OpenStreetMaps
- 3875 objects (including 542 buildings, 38 bars, 24 hotels, 7 churches, etc.)



(Some) table definitions

```
create table Entity(
  id integer primary key
);

create table IsA(
  id integer references Entity(id),
  type text
);

create table HasPoint(
  id integer references Entity(id) primary key,
  geom geometry
);

create table HasPolyline(
  id integer references Entity(id) primary key,
  geom geometry
);

create table HasPolygon(
  id integer references Entity(id) primary key,
  geom geometry
);
```

PostGIS inserting tuples

```
insert into Entity values (1);
insert into isA values (1,'building');
insert into HasPolygon values(1,
  ST_Transform(
    ST_Polygon(
      ST_GeomFromText(
        'LINESTRING(
          18.0642389 59.3493935,
          18.0640184 59.349472,
          18.0644599 59.3498025,
          18.0646781 59.3497238,
          18.0642389 59.3493935)'),
        4326),
    3006)
);
```

4326 – geographic coordinate systems.

3006 – SWEREF 99

Let's do some ad hoc querying!

PostGIS Creating Indexes

```
create index isA_types_index on isA(type);
create index points_geom_index on HasPoint using gist (geom);
create index polylines_geom_index on HasPolyline using gist (geom);
create index polygons_geom_index on HasPolygon using gist (geom);
```

Example ‘where am I?’ query

```
explain analyse
select id
from hasPolygon
where ST_Contains(geom,
    ST_Transform(
      ST_GeomFromText(
        'POINT(18.0643 59.3496)',
        4326),
    3006));

 id
----
  1
```

Example query

```
Index Scan using polygons_geom_index on haspolygon
  (cost=0.00..8.27 rows=1 width=4)
  (actual time=0.251..0.253 rows=1 loops=1)
   Index Cond: (geom && '0101000..C5941'::geometry)
   Filter: _st_contains(geom, '0101000...C5941'::geometry)
 Total runtime: 0.197 ms
```

Example query

```
drop index polygons_geom_index;

Seq Scan on haspolygon
  (cost=0.00..24.13 rows=1 width=4)
  (actual time=0.038..0.940 rows=1 loops=1)
   Filter: ((geom && '0101000..C5941'::geometry) AND
   _st_contains(geom, '010100..C5941'::geometry))
 Total runtime: 0.966 ms
```

Spatial Join

```
explain analyse
select distinct name
from isnamed as X natural join isa as y natural join haspoint as v,
  haspoint as z natural join isa as w
where y.type ='church' and w.type='hotel' and
  st_distance(z.geom,v.geom)< 100;
```

Spatial Join

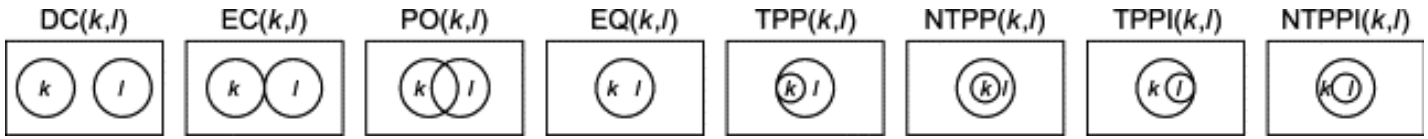
```
explain analyse
select distinct name
from isnamed as X natural join isa as y natural join haspoint as v,
  haspoint as z natural join isa as w
where y.type =’church’ and w.type=’hotel’ and
  st_dwithin(z.geom,v.geom, 100.0);
```

PostGIS extras

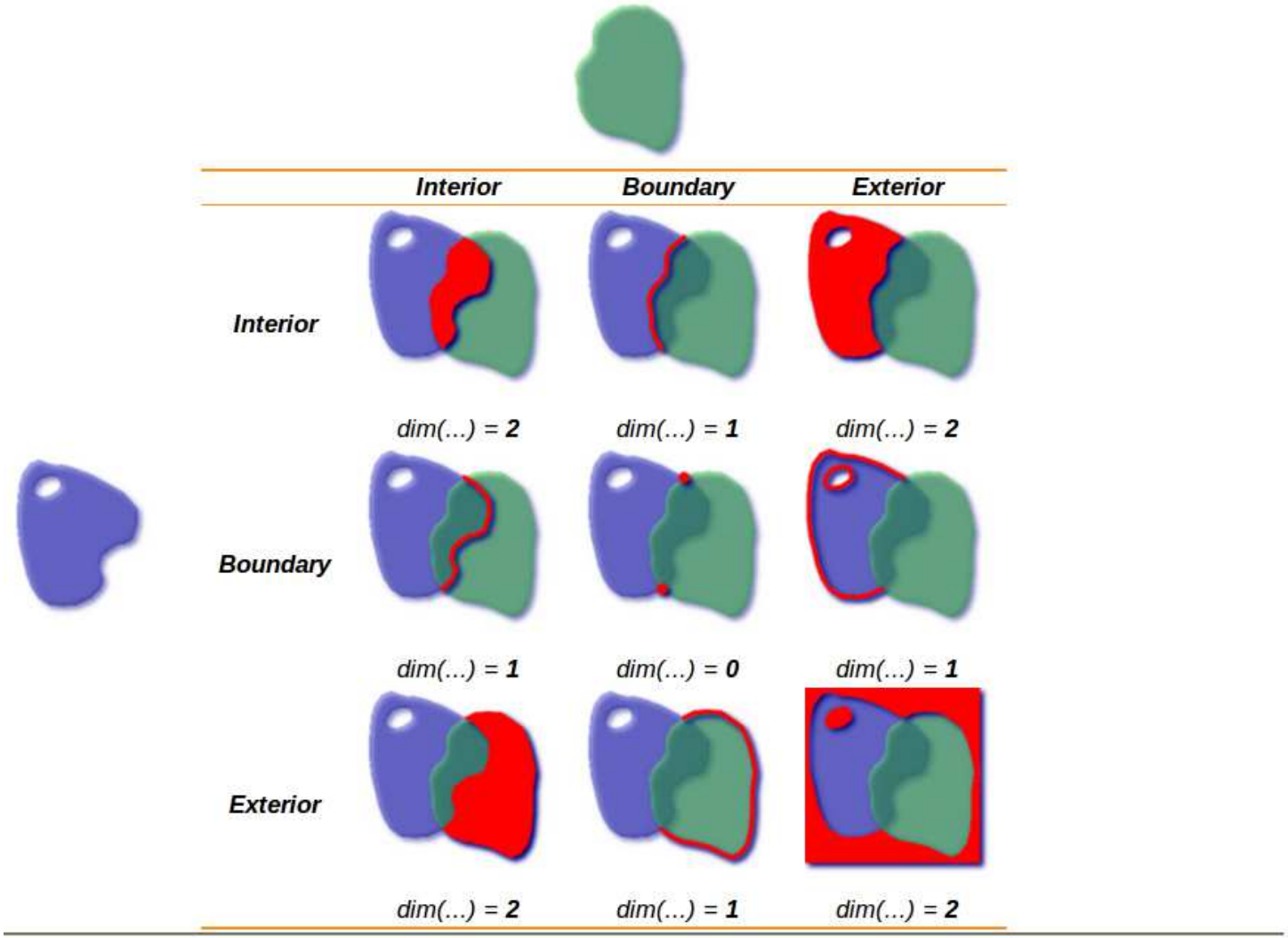
- `pg_routing`
- rasters!
- Map Server

Region Connection Calculus

Analogous to *Allen's Interval Algebra*



Dimensionally Extended 9 Intersection Model (DE-9IM)



Conclusions

- Spatial indexes work (kind of)
- Active area!
- Lots of research and development on-going