PostGIS Spatial Overlay

The Next Generation

Martin Davis, Senior Spatial Engineer
Crunchy Data

November 2020

crunchy data

Metadata

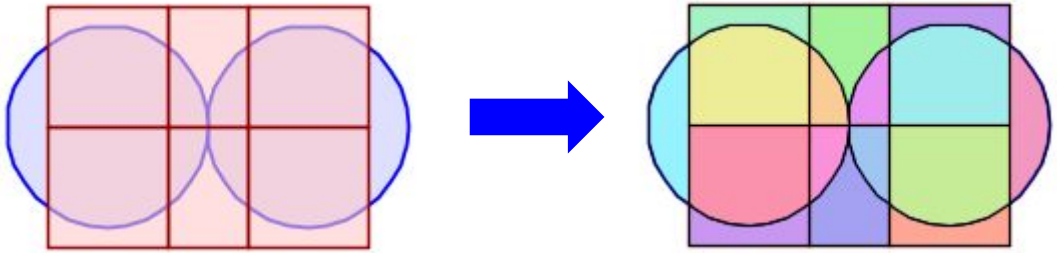
- **Martin Davis**
  - Senior Spatial Engineer at Crunchy Data
  - 25 years of Geospatial software development
  - Lead developer for open source **JTS Topology Suite** Java geometry API
  - Also GEOS, PostGIS, Crunchy Spatial platform

- **Outline**
  - Overview of Overlay
  - PostGIS Overlay
  - Overlay Robustness Issues
  - Overlay - Next Generation

What is Overlay?

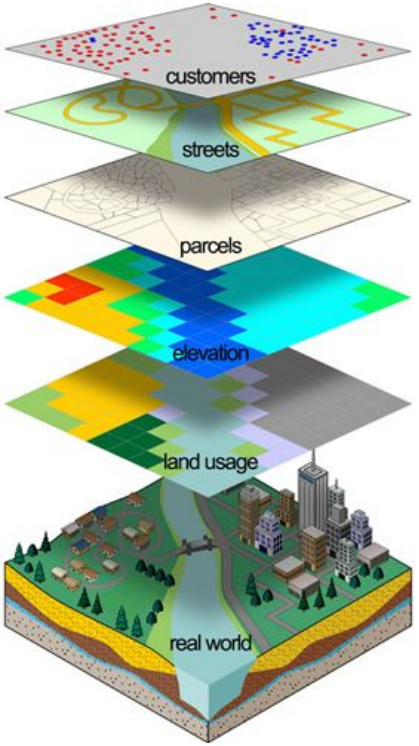
> *"Overlay superimposes multiple datasets (layers) together for the purpose of identifying relationships between them. An overlay creates a composite map by combining the geometry and attributes of the input layers"*
>
> http://wiki.gis.com/wiki/index.php/Overlay

- Defined in **vector** and **raster** domains
  - This talk is about **vector overlay**



Why Overlay?

- Implemented by most full-featured spatial systems
- The **fundamental** tool for **geospatial data analysis**
- Answers questions about “what’s on top of what”:
  - *What land use is on what soil type?*
  - *What roads are within what counties?*
  - *What wells are in what watersheds?*
- Tobler’s First Law of Geography
  - *“Near things are more related than far things”*
  - Coll: Coincident things are almost **certainly** related

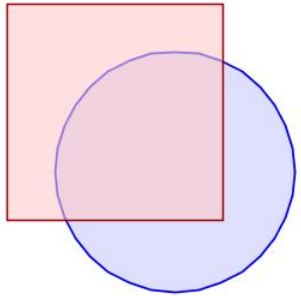


Overlay in PostGIS

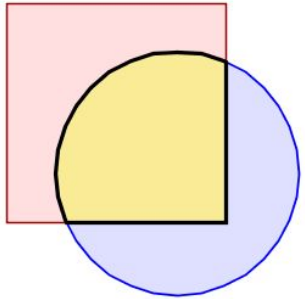
- PostGIS implements the OGC *Simple Features Implementation Specification for SQL* (SFS)
- Spatial analysis functions include:
  - `ST_Intersection, ST_Union, ST_Difference, ST_SymDifference`
- Formally called "*point-set-theoretic binary Boolean operations*"
  - Let's just call them "overlay functions"
- Overlay functions operate on pairs of Geometry values
  - Use SQL for "layer-on-layer" overlay

PostGIS Overlay functions

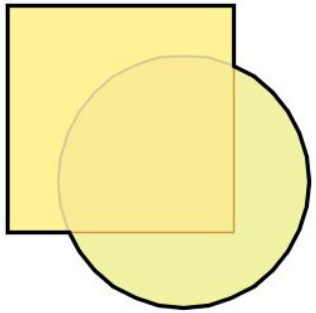
Inputs
A and B



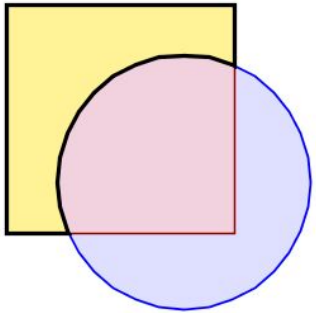
ST_Intersection(A,B)



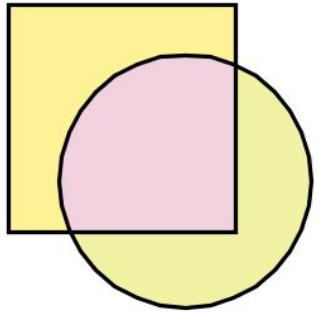
ST_Union(A,B)



ST_Difference(A,B)

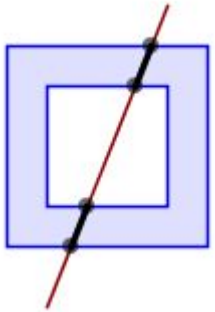


ST_SymDifference(A,B)

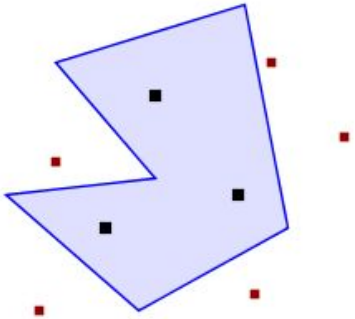


Overlay - all Geometry types

ST_Intersection
(Polygon, Line)

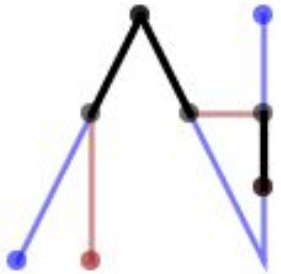


ST_Intersection
(Polygon, Point)



Line / Line

ST_Intersection(A,B)

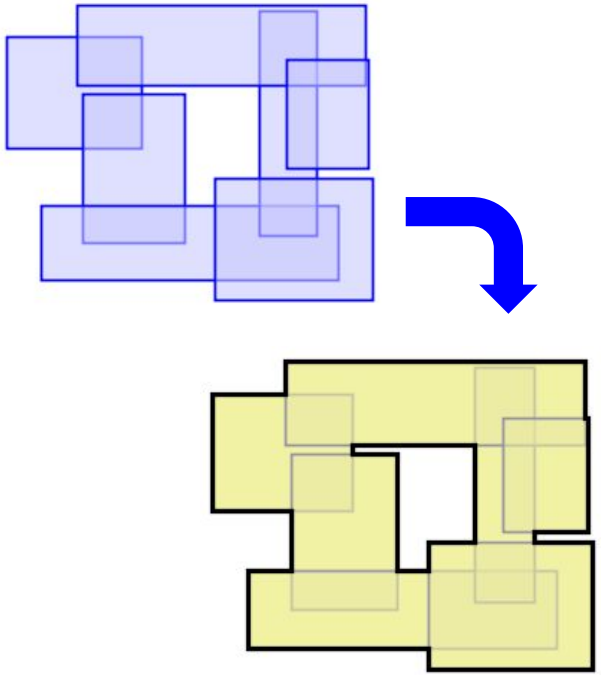


ST_Union(A,B)

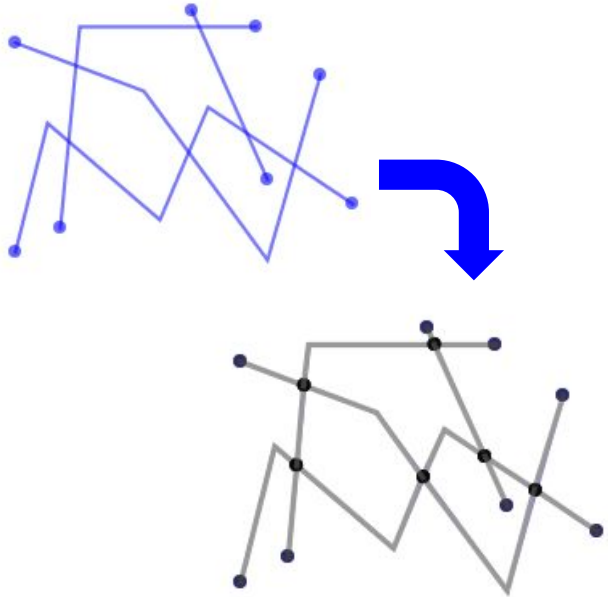
PostGIS Overlay functions (additional)

ST_Union(A)

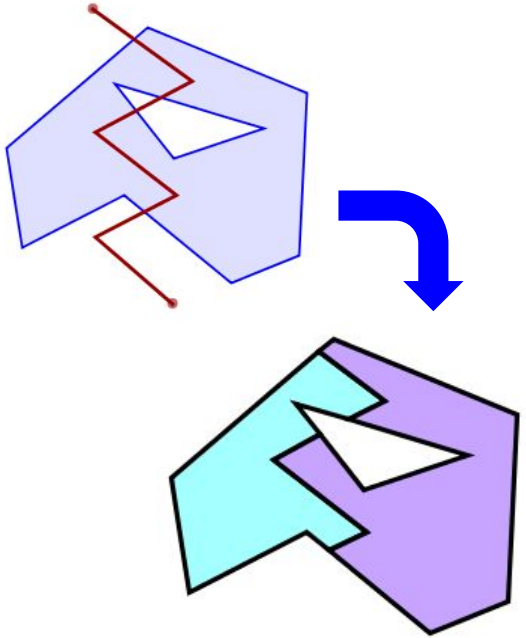
(unary/aggregate union)



ST_Node(lines)



ST_Split(polygon, line)

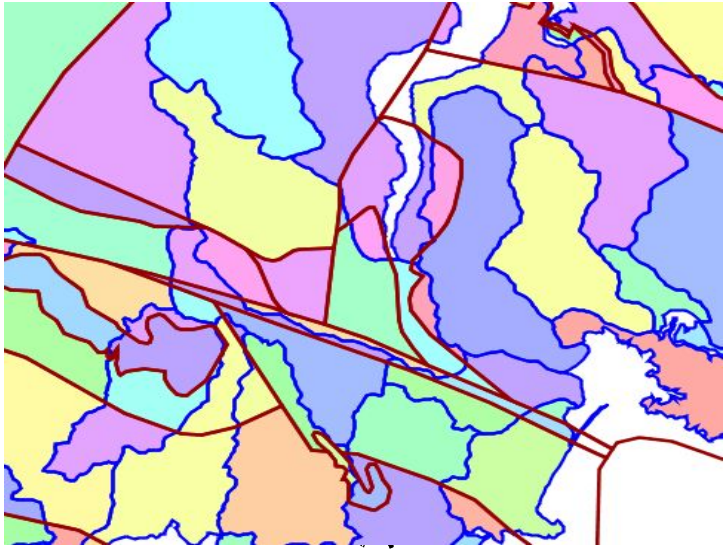


Overlay Analysis - Layer-on-Layer

What geological zones occur in what watersheds?

```
SELECT w.ws_id, g.zone_id,
    Sum( ST_Area( ST_Intersection( g.geom, w.geom ))) AS area,
 FROM geo_zone g
 JOIN watershed w ON ST_Intersects( g.geom, w.geom )
GROUP BY ws_id, zone_id;
```

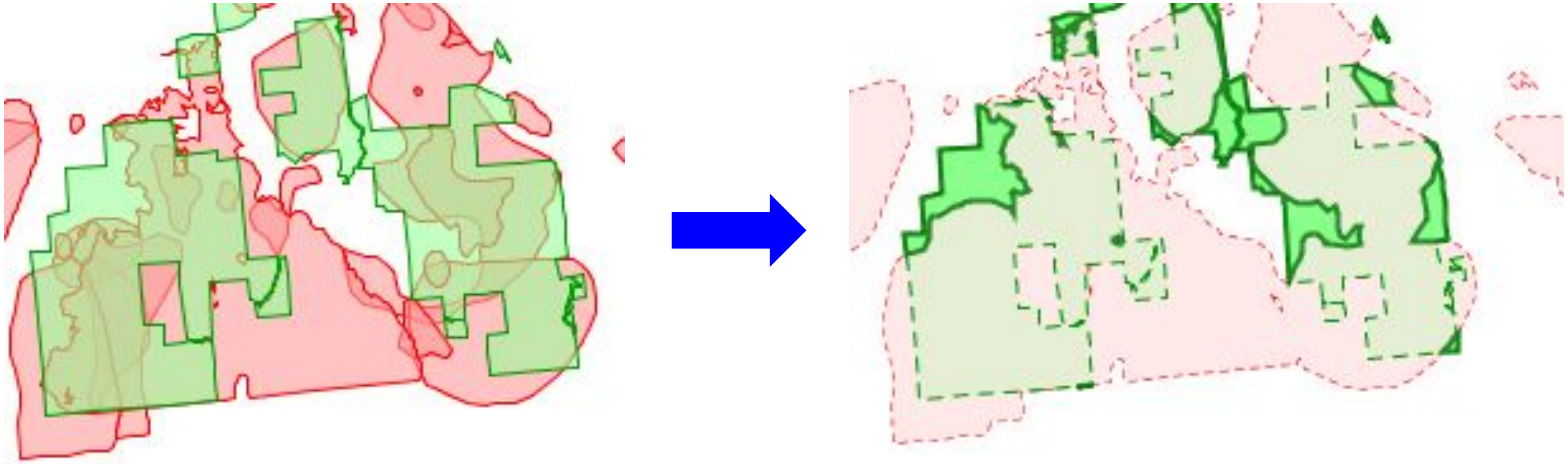
ws_id	zone_id	area
WSD085	ESSF	6660733.8225773545
WSD085	ICH	44534270.11719399
WSD085	IDF	20415532.02766427
WSD087	ICH	13762879.084600706
WSD087	IDF	5620124.330770972
...		



Overlay Query - Union & Difference

- For each Park, how much area is **not** affected by Wildfire?

```
SELECT ST_Difference( park.geom, burnt.geom ) AS geom
  FROM park JOIN LATERAL
    ( SELECT ST_Union(geom) AS geom
        FROM fire WHERE ST_Intersects(park.geom, fire.geom)
    ) ON true AS burnt;
```



PostGIS, GEOS and JTS

- **PostGIS** overlay functions are provided by the **GEOS** geometry library

- **GEOS** (C++) is a port of the **JTS Topology Suite** (Java)

- JTS ⇒ GEOS ⇒ PostGIS *(and Shapely, R-SF, QGIS…)*
- JTS 1.0 was released in 2001
  - The overlay algorithm dates back to the first release

Overlay 1.0 Issues

#4538 new defect

Opened 13 months ago
Last modified 13 months ago

ST_Difference(g1,g2) throws lwgeom_difference: GEOS Error: TopologyException: found non-noded intersection

Get location of PostGIS / GEOS topology exception

Asked 11 months ago Active 8 months ago Viewed 106 times

#4182 new defect

Unexpected TopologyException during ST_Union aggregate function with valid polygons

PostGIS: ST_Union fails with valid polygons?

Asked 3 years, 6 months ago

[postgis-users] Difference throw an error

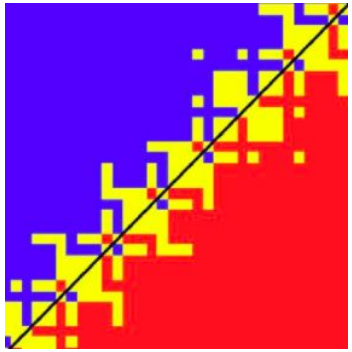
at idu.de
PST 2006

...and many more

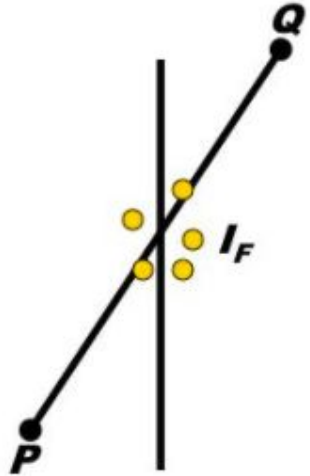
crunchy data

Geometric Non-Robustness

> ***Geometric non-robustness*** *is a problem wherein branching decisions in computational geometry algorithms are based on* ***approximate numerical computations****, leading to various forms of unreliability including* ***ill-formed output*** *and* ***software failure through crashing or infinite loops****.* *- Wikipedia*



- Similar to r**ound-off error** in numerical computing
- Geometric computations which can be non-robust:
  - Compute **orientation** of a point to a line
    - Discrete result: Left, Right, On
    - So, easier to make robust
  - Compute **intersection point** of two lines
    - In general, intersection point cannot be computed OR represented exactly in floating-point arithmetic
    - So, computed intersection point *usually* does not lie exactly on the line segments



crunchy data

Overlay Non-Robustness

- Overlay algorithm phases:
  - **Node the input lines**
  - **Build topology**
  - **Extract result geometry**
- **Noding** requires **point-line orientation** AND **line intersection**
- Original overlay used simple (aka naive) approach
  - Robustness problems !
  - ⇒ Overlay topology may not be computed correctly
  - Detect and throw exception

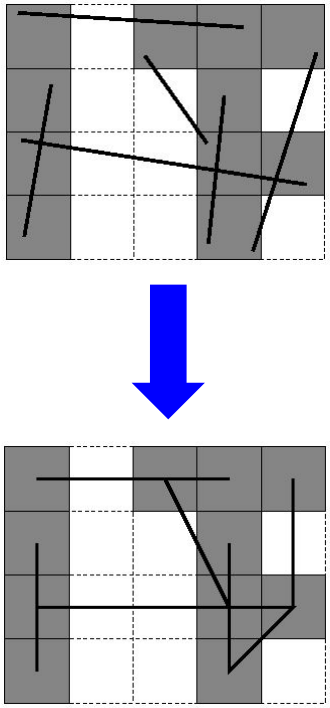
```
ERROR: GEOSUnaryUnion: TopologyException: found non-noded intersection
between LINESTRING (-3.36702e+06 4.94515e+06, -3.36699e+06 4.94512e+06) and
LINESTRING (-3.36699e+06 4.94512e+06, -3.36702e+06 4.94515e+06) at
-3367018.7763304636 4945150.2479274161
```

Overlay Noding Robustness - Solutions?

- Pre-processing heuristics in JTS / GEOS / PostGIS
- **Snapping** geometry vertices and lines
  - Can be slow
  - Does not always work
- **Rounding** vertices to a grid (`ST_SnapToGrid`)
  - ALL vertices rounded - not ideal
  - Does not always work

- **Snap-Rounding** - well-researched noding technique
  - Guaranteed to work
  - ALL vertices rounded - not ideal
  - *Needs full rewrite of overlay*

Snap-Rounding



crunchy data

OverlayNG - Next-Generation Overlay

- Ground-up rewrite of JTS overlay algorithm
- Goals:
  - Robust
  - Performant
  - Simpler and more flexible code
- Initial plan - use **Snap-Rounding noding** for full robustness
- New idea! - **Snapping noding**
  - Avoids rounding coordinates
  - Robust (almost always..)
- Ported to GEOS 3.9
- Used for overlay functions in PostGIS 3.1

OverlayNG - Noding & Robustness

- “Pluggable” Noding
  - **Simple** - fast, but non-robust
  - **Snapping** - much more robust, fairly fast
  - **Snap-rounding** - robust, slower, changes precision
- Combine these strategies in sequence

- **Fully Robust !** (so far…)
  - No more `TopologyException` errors

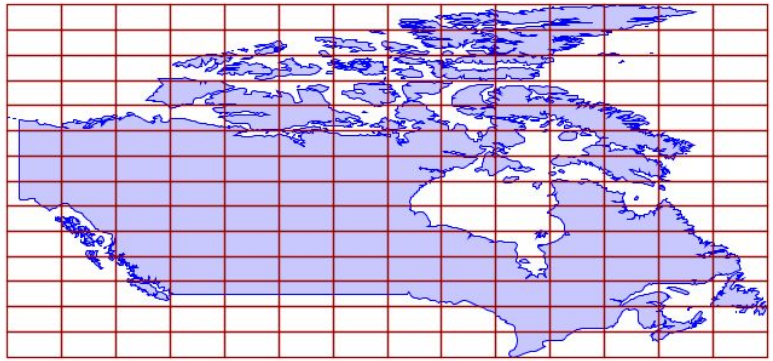
OverlayNG - Performance

- Performance optimizations
  - Pre-clipping for intersection, difference

Example: Tiling a large polygon (68,156 vertices)

```
SELECT sum( ST_Area( ST_Intersection( a.geom, g.geom )))
FROM grid g JOIN admin0 a
ON ST_Intersects( a.geom, g.geom );
```

Grid Size	Old	NG	
42	10.2 s	2.1 s	4.8 x
143	22.7 s	3.6 s	6.3 x
546	55.1 s	10.1 s	5.5 x

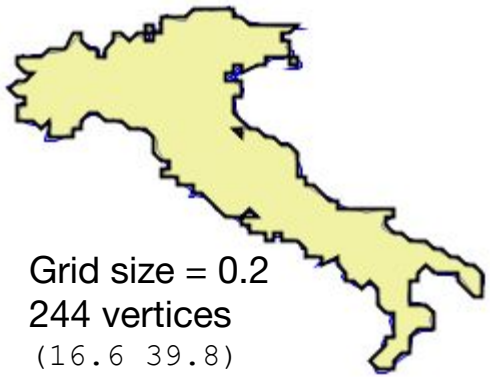


data

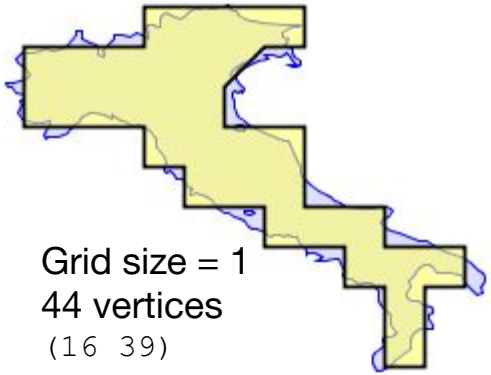
OverlayNG - Precision

- **Snap-Rounding** provides valid result in any fixed precision
- Maintain input precision in overlay result
- Perform precision reduction
  - Avoid or remove irrelevant precision (e.g. 16.23474053845838 ⇒ 16.23474)
  - Reduce data size (vertices, text i.e. WKT, GeoJSON)
  - Vector tile generation

Full precision
544 vertices
(16.63055, 39.843233)



Grid size = 0.2
244 vertices
(16.6 39.8)



Grid size = 1
44 vertices
(16 39)

Future Development

- `ST_ReducePrecision` - using Snap-Rounding
- `ST_Split, ST_Node` – upgrade to use OverlayNG
- `ST_Overlay` - unary / aggregate function for full polygon overlay
- `ST_CoverageUnion` - fast union for polygonal coverages
- `ST_MakeValid` - noding / graph-based
- Streaming Overlay
  - Able to process HUGE datasets

Learn More

PostGIS Manual

https://postgis.net/docs/manual-dev/

PostGIS Workshop

https://postgis.net/workshops/postgis-intro/geometry_returning.html

Blog posts

http://blog.cleverelephant.ca/2019/08/postgis-3-geos.html

http://lin-ear-th-inking.blogspot.com/2020/05/jts-overlay-next-generation.html

http://lin-ear-th-inking.blogspot.com/2020/06/jts-overlayng-noding-strategies.html

http://lin-ear-th-inking.blogspot.com/2020/06/jts-overlayng-tolerant-topology.html

Source Code

https://git.osgeo.org/gitea/geos/geos/src/branch/master/src/operation/overlayng