PostGIS support for Simple Polygonal Coverages

Martin Davis

FOSS4G-NA 2023

October 2023

crunchy data

Martin Davis

martin.davis@crunchydata.com

- **Geospatial Engineer** at crunchy data
- Developer on:
  - **JTS Topology Suite**
  - **GEOS**
  - **PostGIS**
  - `pg_featureserv`

JTS

GEOS

PostGIS

I ❤ Math & Geometry

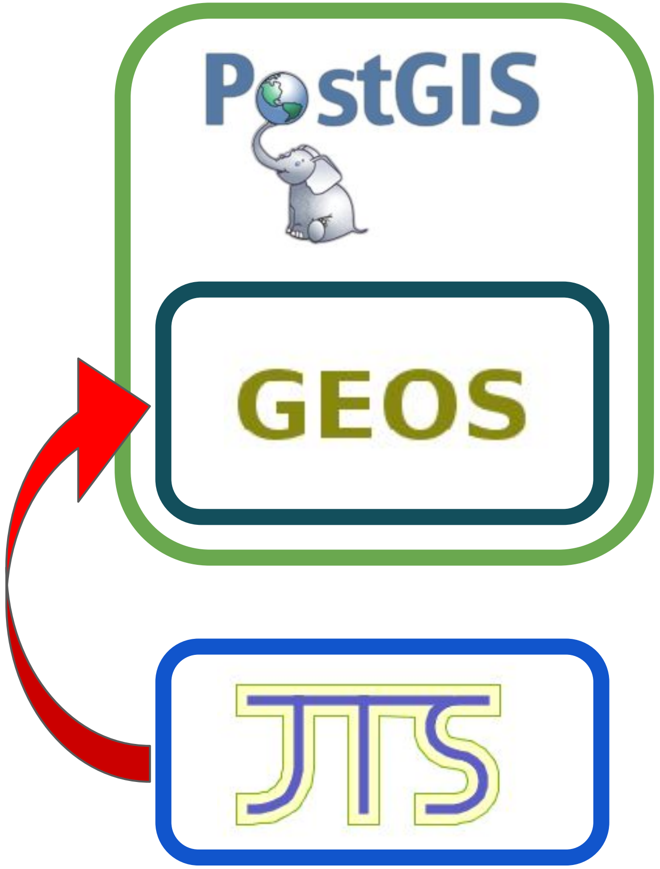
PostGIS

- PostgreSQL extension
- Spatial…
  - Datatypes (geometry, geography)
  - Indexes
  - Functions
- **OGC Simple Features** geometry model
- Also:
  - **Topology**, **Raster**, **pgRouting**
- Supported by most FOSS4G tools
  - **GDAL, QGIS, GeoServer, MapServer**, etc.
- Run **everywhere**: local, datacentre, hosted, cloud

PostgreSQL
PostGIS

JTS & GEOS

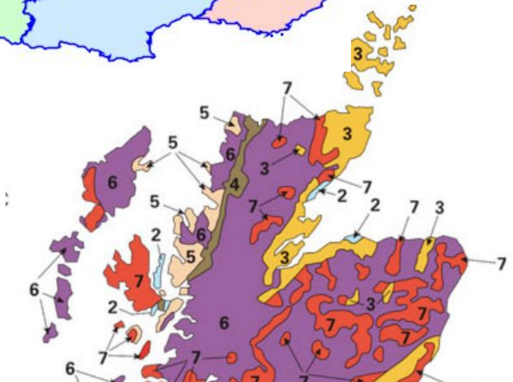
- PostGIS uses **GEOS Geometry Library** for most spatial processing
- Coverage code lives in **GEOS** (C++)
  - GDAL, QGIS, SpatialLite, etc.
  - Shapely, r-SF, Rust, etc.
- Originates in **JTS Topology Suite** (Java)



Polygonal Coverage Model

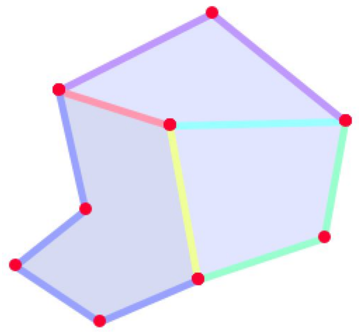
***Coverage** (n., geospatial): a feature that acts as a function to return values from its range for any position in its spatial domain*

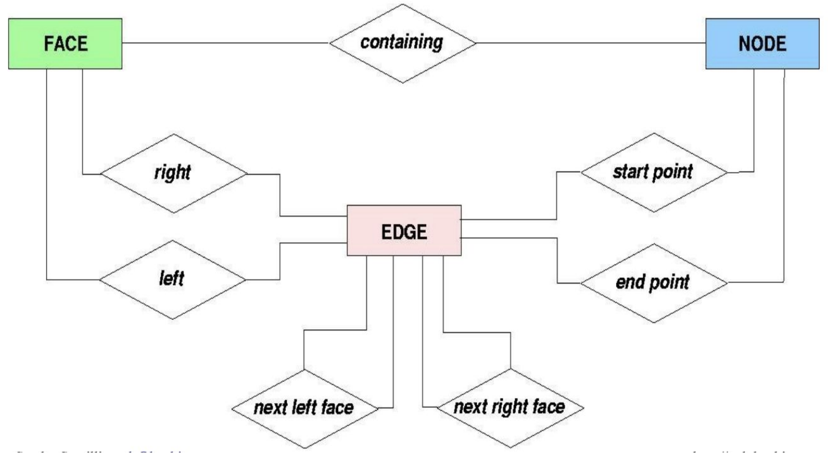
- Spatial model with (usually) **adjacent**, **non-overlapping** polygonal areas
  - Each area has attributes
- Many use cases:
  - *Cadastral parcels*
  - *Political jurisdictions*
  - *Land use*
  - *Geological regions …*



Polygonal Coverage as Topology

- Can represent a polygonal coverage using **PostGIS Topology**
- ISO SQL/MM Topology Model
- A Topology layer is represented by 4 tables:
    - `edge_data, face, node, relation`
    - + 2 metadata tables
- 60+ functions to create and manipulate topology data

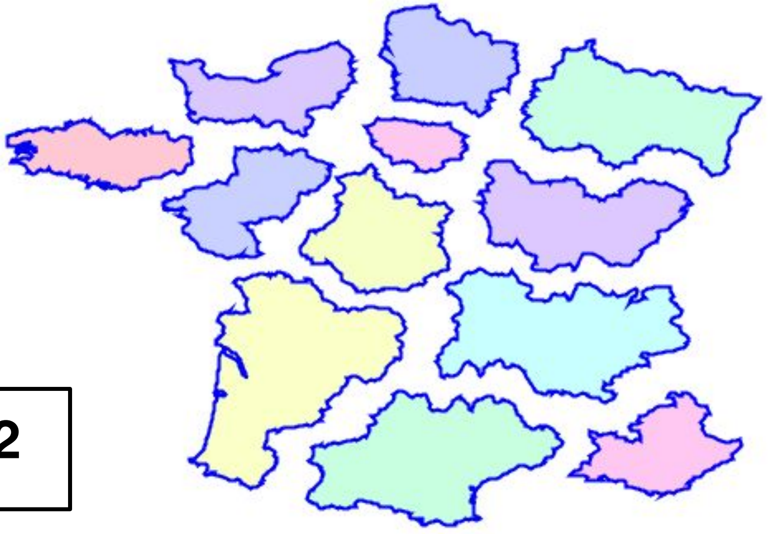




Simple Polygonal Coverage

- Model Polygonal Coverage as **discrete polygons**
  - use **OGC Simple Features** geometry model
- Coverage = single table of Polygons / MultiPolygons, with attributes
- Coverage topology is **implicit**
- Allows holes and disjoint regions
- Works with existing functions and tools

Available in **PostGIS 3.4** with **GEOS 3.12**



Coverage Operations

- Common coverage operations
  - Validation
  - Union
  - Simplification
- Before:
  - Hard (or impossible) to code; complex SQL
  - Poor performance
- Now:
  - Easy-to-use functions
  - High performance

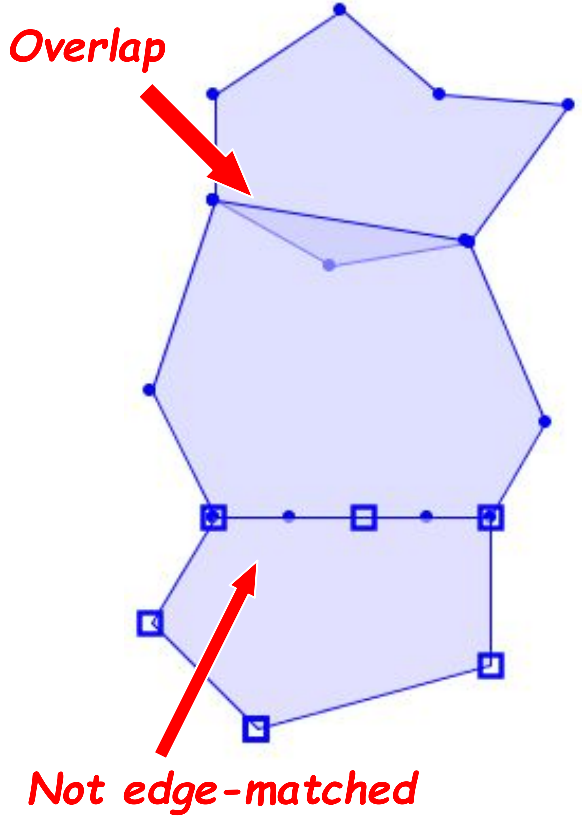
Validation

Coverage Validity

- Coverage Validity is required for:
  - Correct operation of coverage functions
  - Accurate modelling and analysis
- Simple rules:

A set of polygons is a **valid coverage** if:

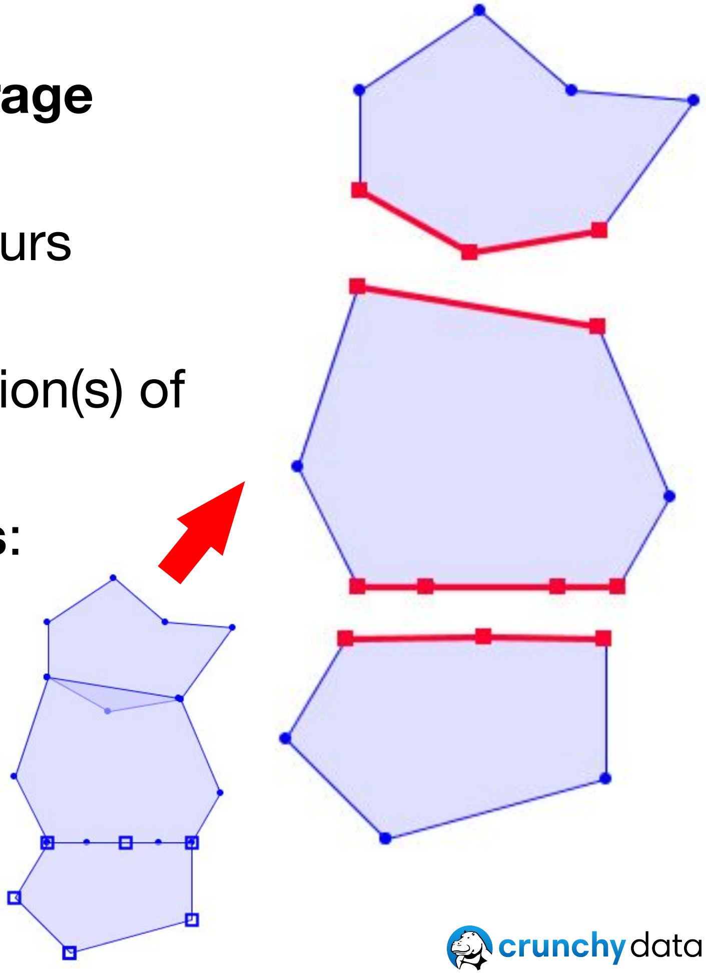
1. Polygons are **valid**
2. Polygons are **non-overlapping** *(interiors do not intersect)*
3. Adjacent polygons are **edge-matched** *(shared lines have identical vertices)*



Coverage Validation

- **Test** if set of ***valid*** polygons is a **valid coverage**
- **Global** operation over coverage dataset
  - Each polygon validated against neighbours

- **Report coverage-invalid** polygons by location(s) of invalidity
- Identify **invalid polygon boundary sections**:
  - Overlapping edges
  - Non-edge-matched adjacent edges



PostGIS - Coverage Validation

```
SELECT id, ...
    ST_CoverageInvalidEdges(geom) OVER () AS invalid_line
    FROM coverage_polys;
```

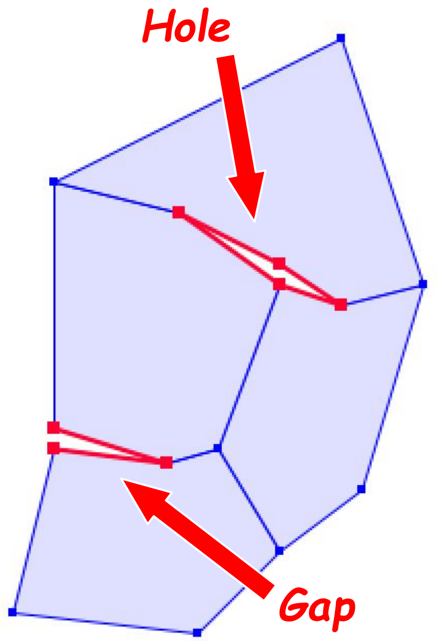
- Window function
  - operates over all or some polygons in table
  - Allows selecting additional polygon attributes (e.g. `id`)
- For each input polygon returns
  - **Invalid**: invalid edges as `(Multi)LineString`
  - **Valid**: `NULL`

```
 id |        invalid_line
----+-------------------------------------
  1 | LINESTRING (40 110, 100 70)
  2 | MULTILINESTRING ((100 130, 140 120, ...
  3 | null
```

crunchydata

Gaps and Holes

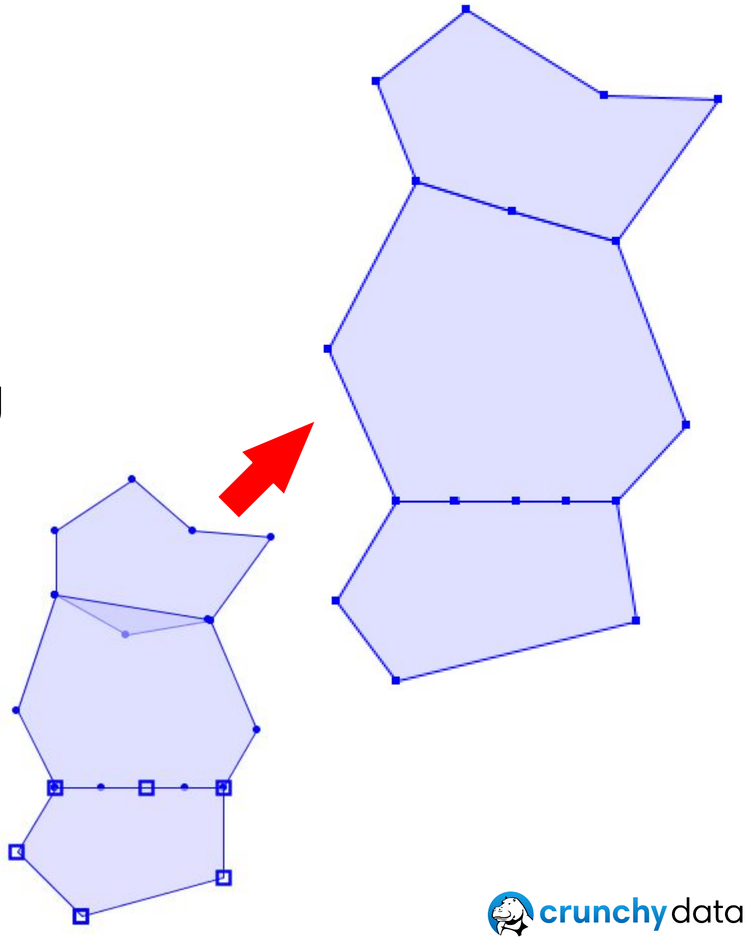
- Coverage validity rules allow **holes** and **gaps**
  - Required to model real-world situations
- Narrow gaps/holes may be errors
- Report using **`ST_CoverageInvalidEdges`** with a distance tolerance
  - *May produce false positives*



```
SELECT ST_CoverageInvalidEdges(geom, tol)
                OVER () AS invalid_line
    FROM coverage_polys;
```

Coverage Cleaning

- Fix invalid polygons
  - **`ST_MakeValid`**
- Fix coverage
  - remove overlaps and gaps
  - add nodes to ensure edge-matching
  - *Coming soon?*
- External tools
  - QGIS **GeometryChecker** tool
  - GRASS GIS **`v.clean`**
  - **`pprepair`**
  - **`MapShaper`**

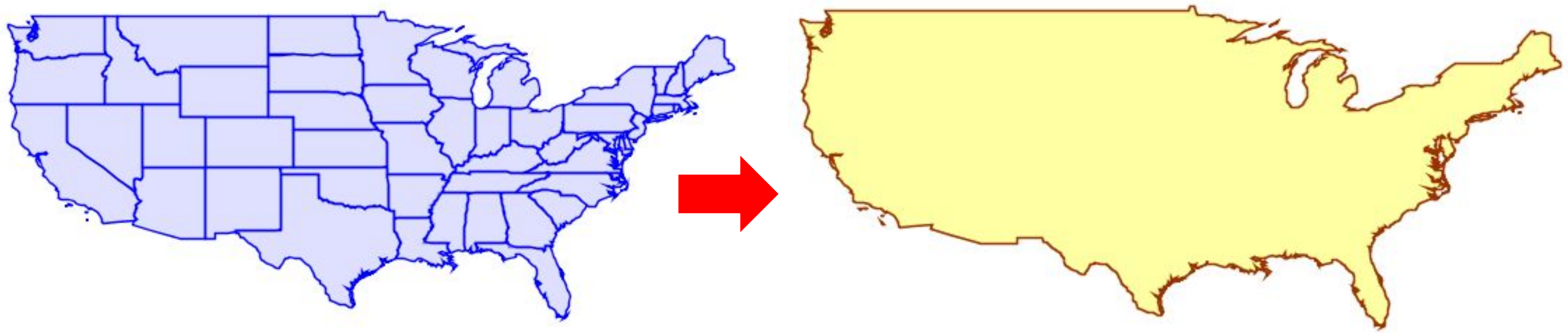


Union

Coverage Union

```
SELECT ST_CoverageUnion(geom) FROM coverage_polys
```

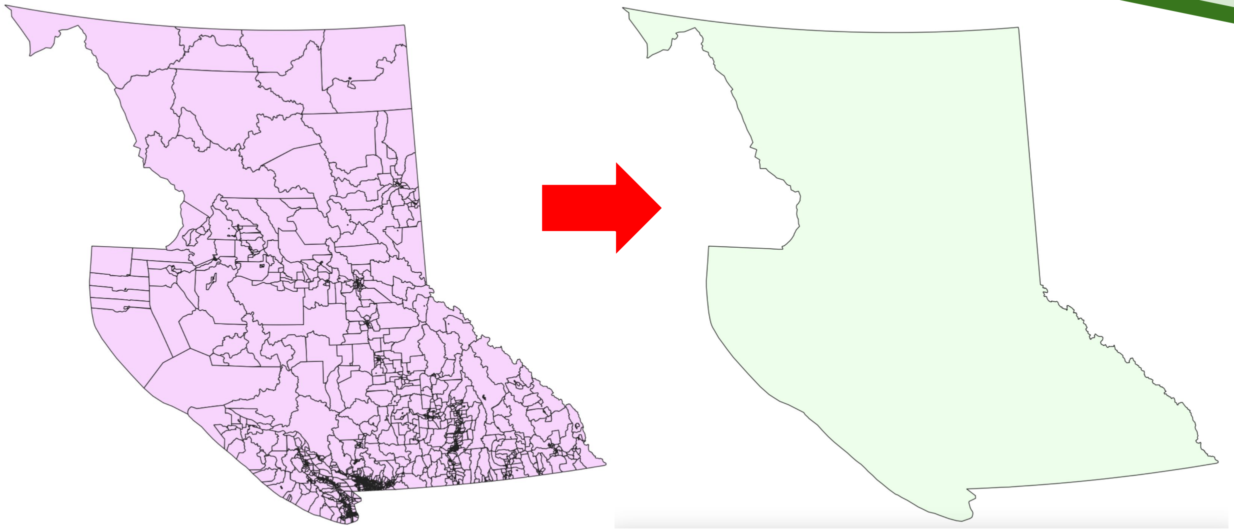
- Computes the union of coverage polygons
- Aggregate function, returns polygonal geometry
- Much faster than `ST_Union`



Coverage Union - Performance

- Dataset: **BC Voting Areas**
- 5,658 polygons with 2,171,572 vertices
- **`ST_Union`**: 5,072 ms
- **`ST_CoverageUnion`**: 411 ms

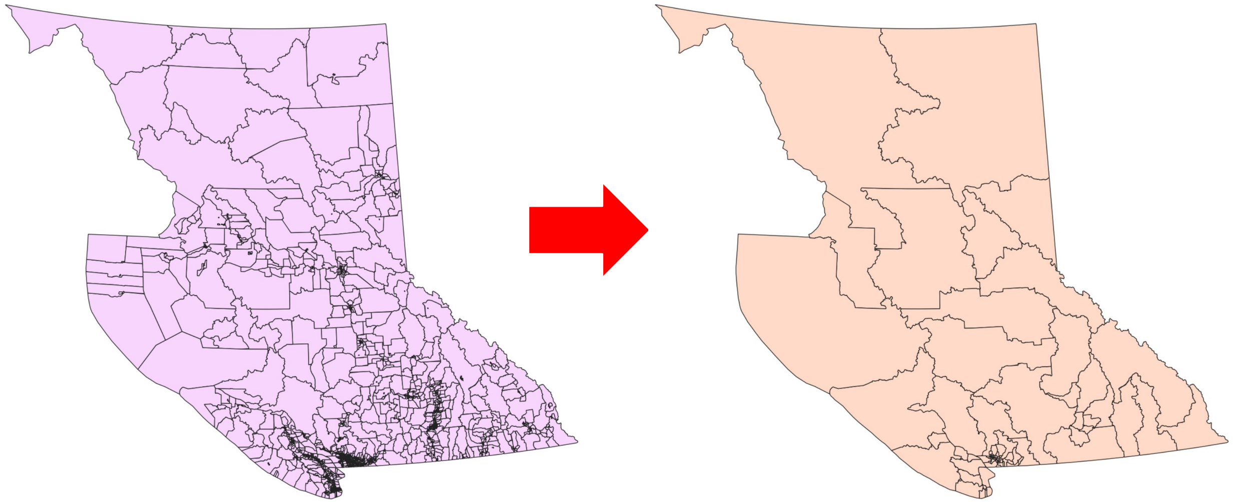
12x Faster



Coverage Union - Rollups

- Union Voting Areas to form Electoral Districts

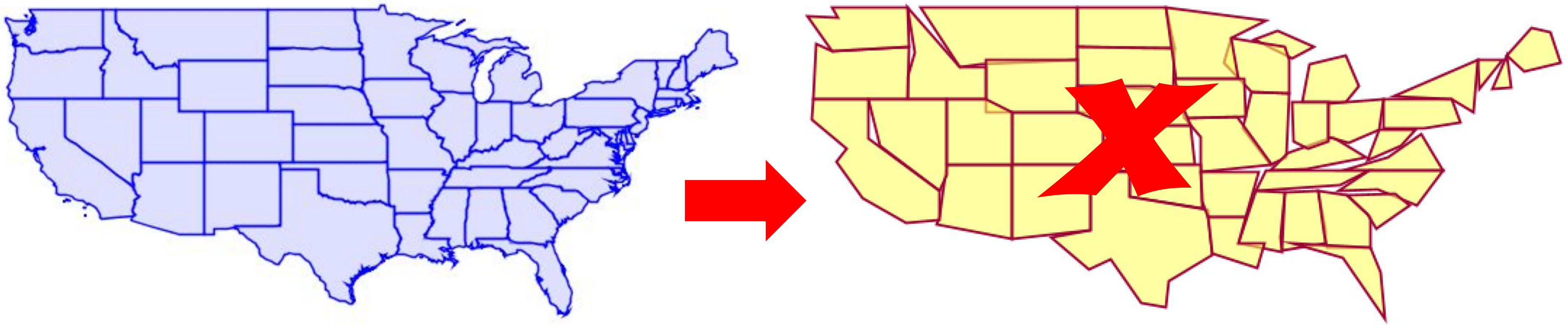
```
SELECT ed_code, ANY_VALUE(ed_name) AS name, SUM(va_tot) AS ed_tot,
        ST_CoverageUnion(geom) AS geom
FROM voting_areas
GROUP BY ed_code;
```



Simplification

Coverage Simplification

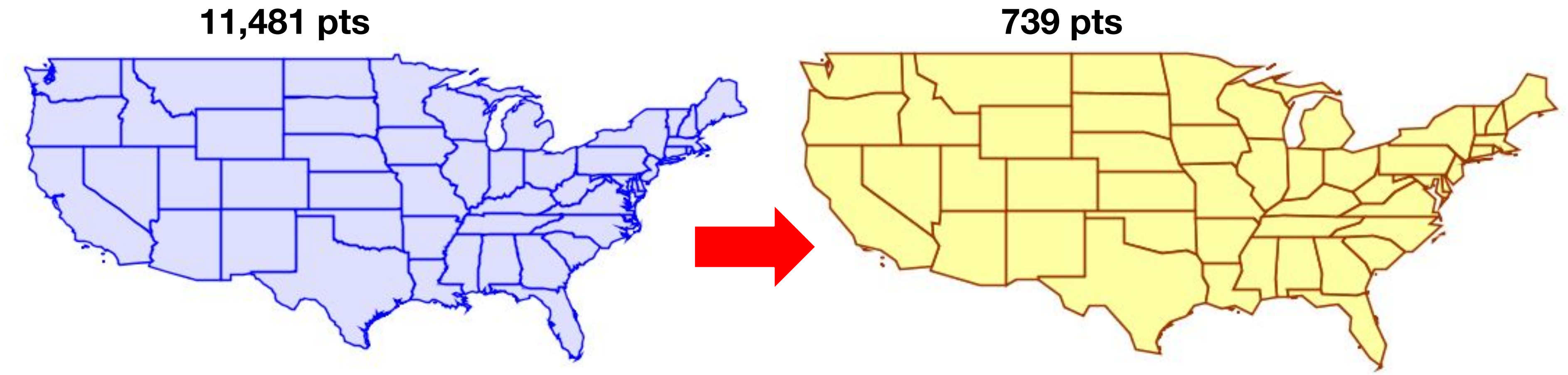
- Reduces the number of vertices in coverage boundaries
- “Killer app” for coverages?
  - e.g. **`MapShaper`**
- Before - no way to do this effectively in PostGIS
  - “Piecewise” doesn’t work!
  - “Dissolve-Simplify-Polygonize” slow, error-prone



Coverage Simplification

```
SELECT ST_CoverageSimplify(geom, tol) OVER ()
    FROM coverage_polys;
```

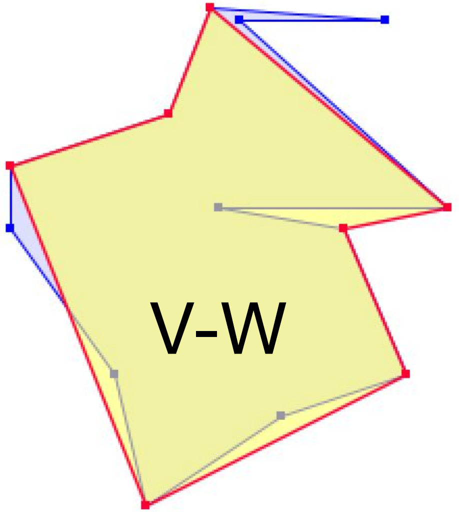
- Simplifies the boundaries of coverage polygons
- Preserves topology; result is a valid coverage with identical structure
- Window function - allows keeping source polygon attributes

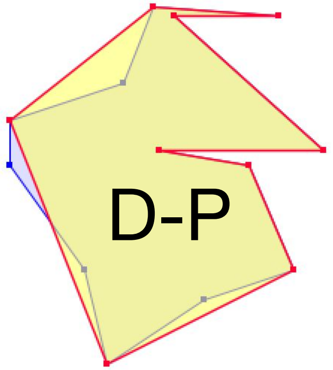


Simplification Algorithm

- Essentially **Visvalingam-Whyatt simplification**
- Tolerance value in distance units
  - *= square root of maximum triangle area to remove*

- Tends to remove **spikes** and **gores**
  - VS Douglas-Peucker, which keeps them
- Better for simplifying areas?

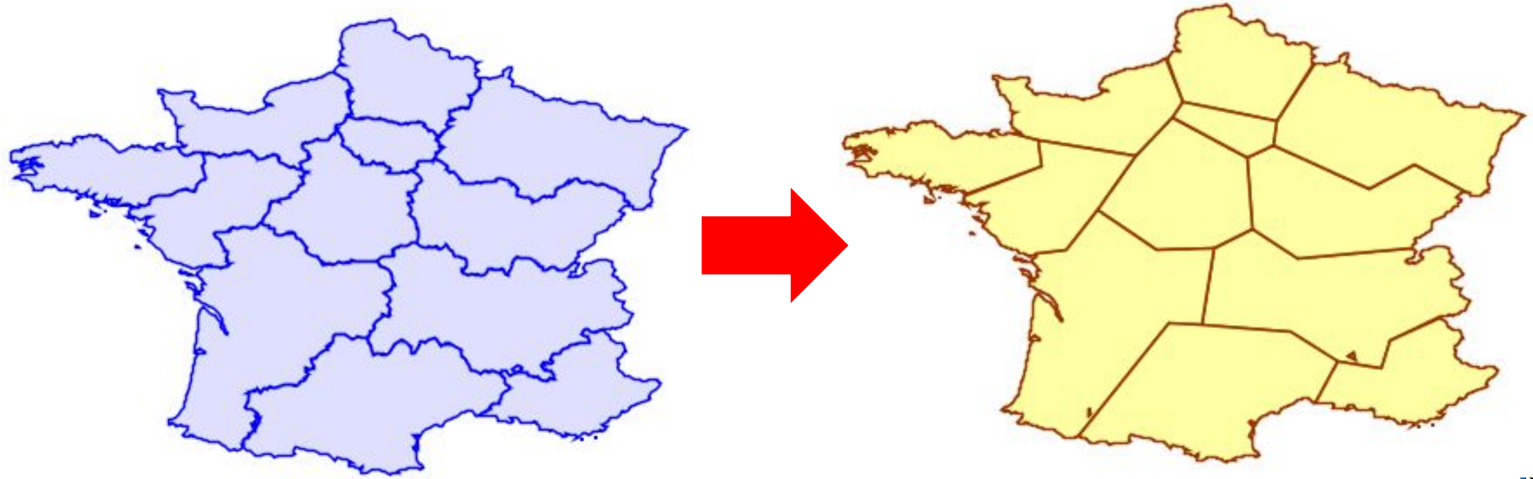




Inner Simplification

```
SELECT ST_CoverageSimplify(geom, tol, FALSE) OVER ()
    FROM coverage_polys;
```

- Simplifies the **inside (shared) boundaries** of a coverage
- Allows simplifying a portion of a coverage
  - Boundary still matches adjacent polygons



Simple Coverage VS Topology

- **Coverage Advantages**
  - Simple geometric model
  - Easy to use with existing data models
  - Works with all spatial functions
  - Performant operations

- **Topology Advantages**
  - Topology maintained ”automatically”
  - Topology hierarchy
  - Edge attributes

crunchy data

Future Work

- More coverage operations
  - Validate single polygon
  - Sliver Merging
  - **Cleaning**
  - Precision Reduction
  - Edge Extraction
  - Export to TopoJSON
  - **Overlay**
- Simple Linear Network ?

Wrap-up

Questions?

Comments?

Ideas?

`martin.davis@crunchydata.com`
`postgis-users@lists.osgeo.org`

crunchy data