New, Improved and Future PostGIS Functions for Hulls, Triangulations & Coverages

Martin Davis

PostGIS Day 2022

November 2022

crunchy data

Martin Davis

martin.davis@crunchydata.com

- **Geospatial Engineer** at 

- Developer on:
  - **JTS Topology Suite**
  - **GEOS**
  - **PostGIS**
  - `pg_featureserv`

I ❤ Math & Geometry!

Outline

New and **Improved**

- Concave Hulls
- Polygon Hulls
- Polygon Triangulation

Requires Version
PostGIS 3.3
with **GEOS 3.11**

Future

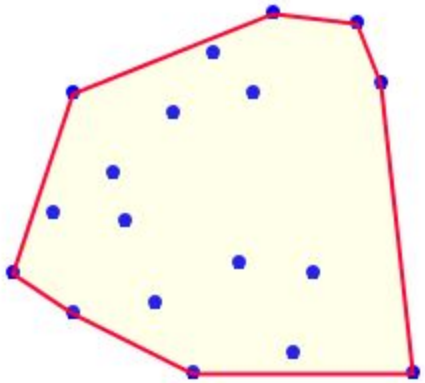
- Polygonal Coverage Functions

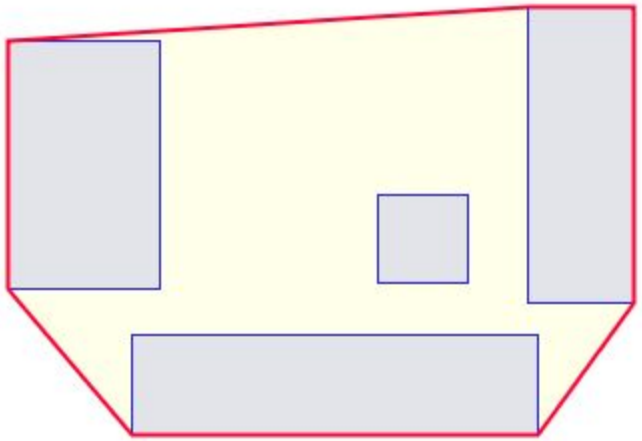
Hulls

Convex Hull

- The *(unique)* convex polygon containing input vertices
- As per the Simple Features specification
- Works for all geometry types

```
SELECT ST_ConvexHull( geom );
```





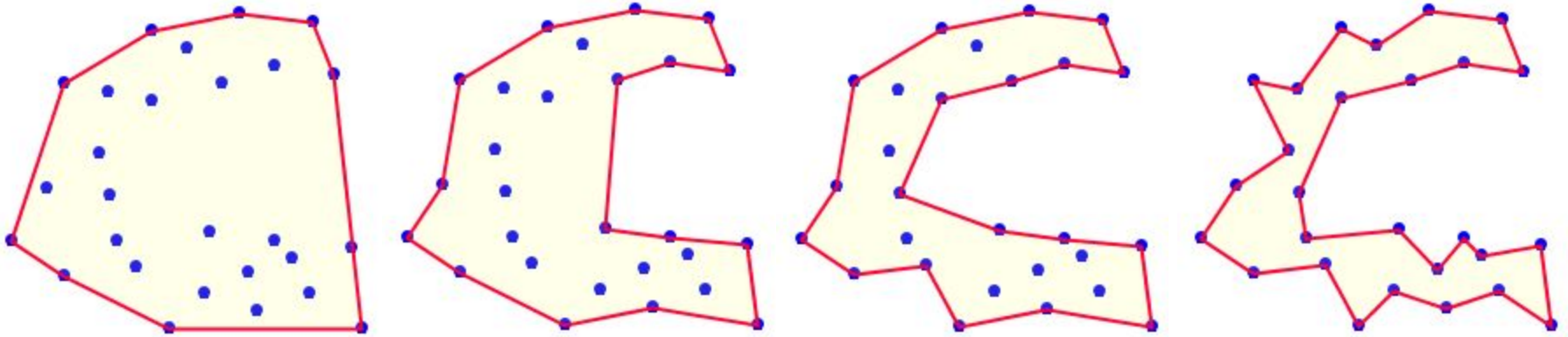
Concave Hull - Points

Improved

- A *(possibly)* concave polygon containing input vertices
- Many possible hulls, determined by param `pctconvex`

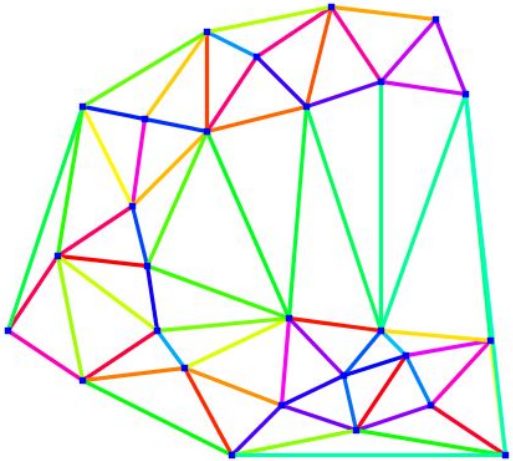
```
SELECT ST_ConcaveHull( geom, pctconvex );
```

pctconvex= 1.0 0.6 0.4 0.0

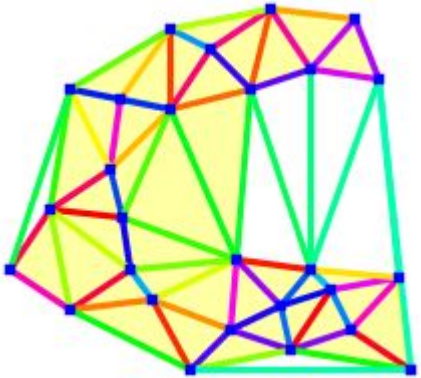


Concave Hull - Points: How it works

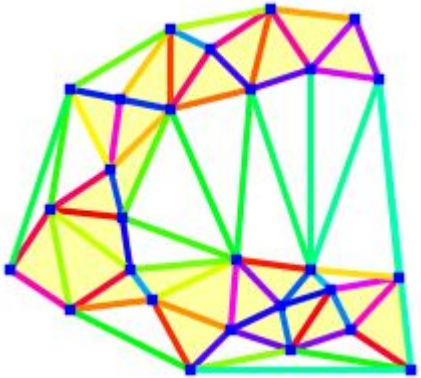
- Build Delaunay Triangulation on points
- Sort triangles by longest edge length
- Remove triangles, until tolerance is reached



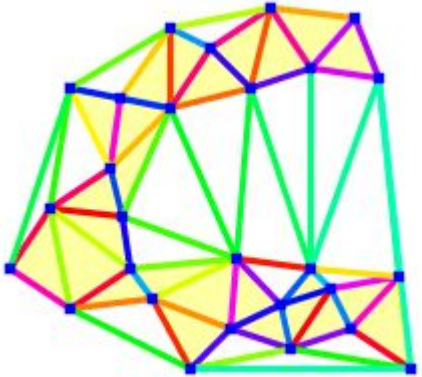
Pctconvex = 0.6



0.4



0.0



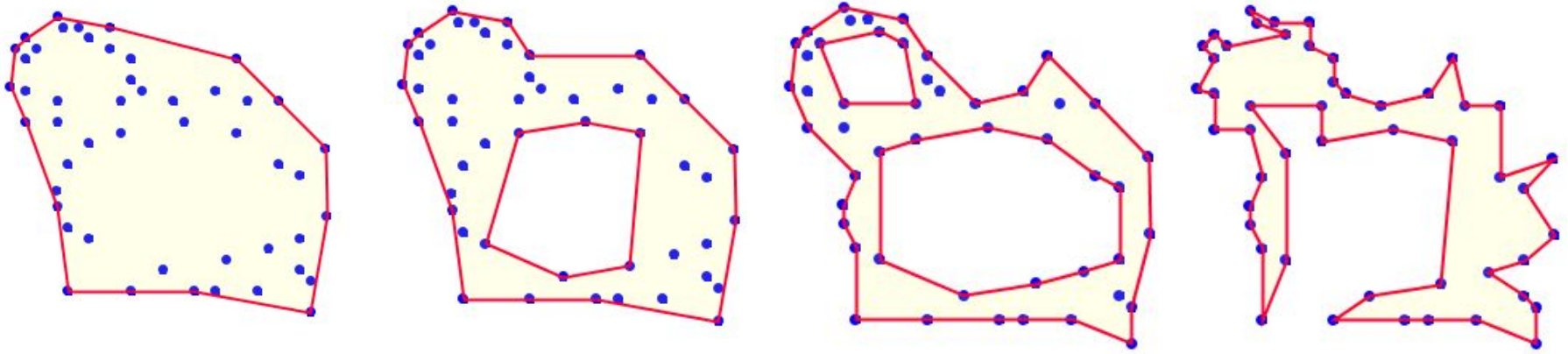
Concave Hull - Points, allowing holes

Improved

- Concave hull can contain holes
  - via optional parameter `allow_holes = true`

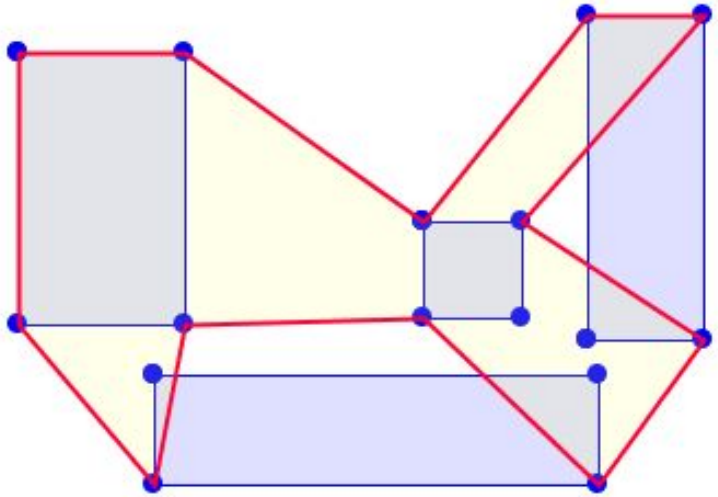
```
SELECT ST_ConcaveHull( geom, pctconvex, true );
```

pctconvex = 0.6 = 0.5 = 0.25 = 0.0



Concave Hull - Polygons?

- Standard Concave Hull algorithms only process points
- **Problem**! Does **not** respect polygon boundaries

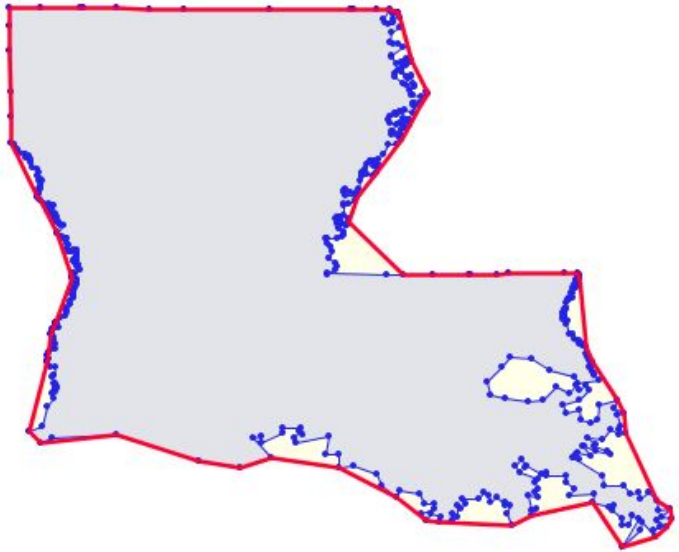


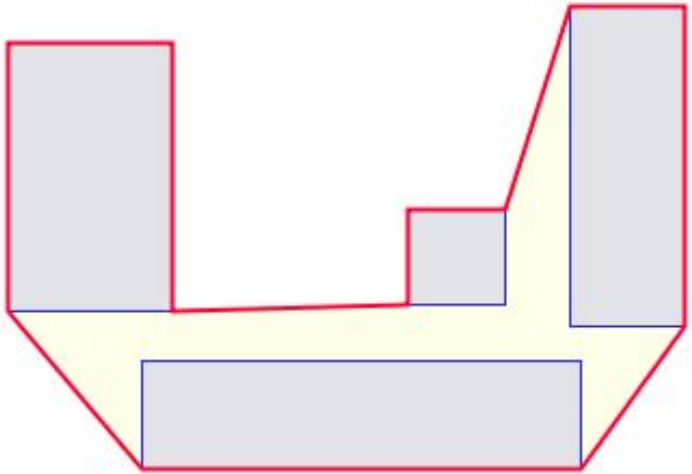
Concave Hull - Polygons

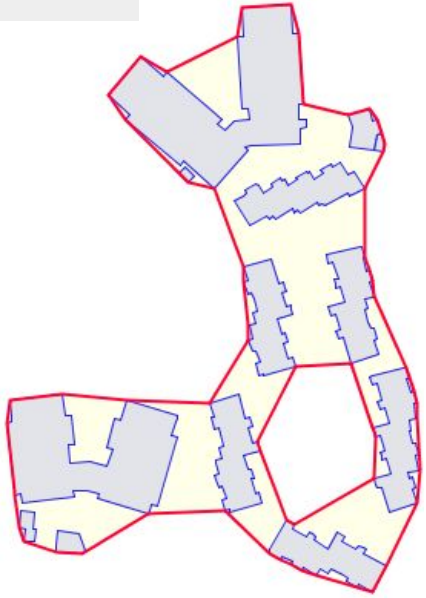
New

- PostGIS has new algorithm to compute Concave Hull for polygon(s)
  - constrained by polygon boundaries

```
SELECT ST_ConcaveHull( geom, pctconvex, allow_holes );
```

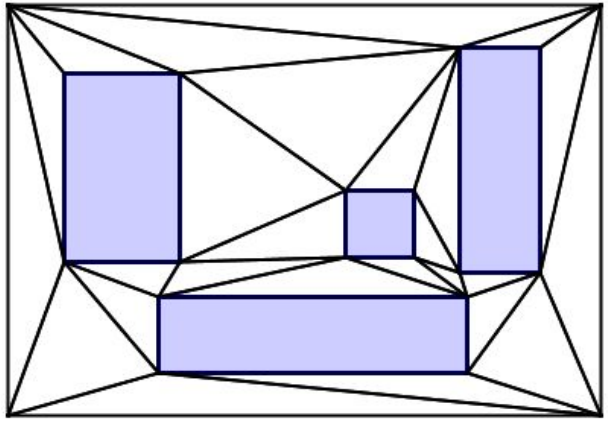


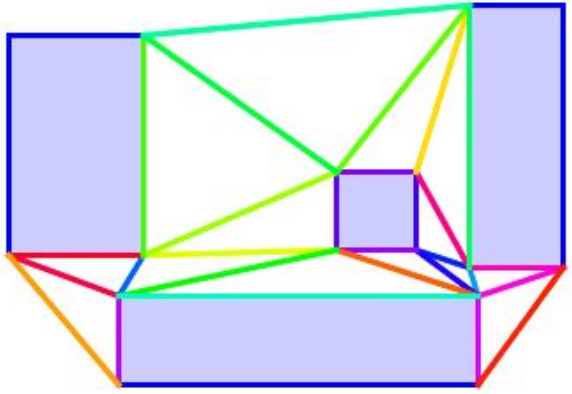


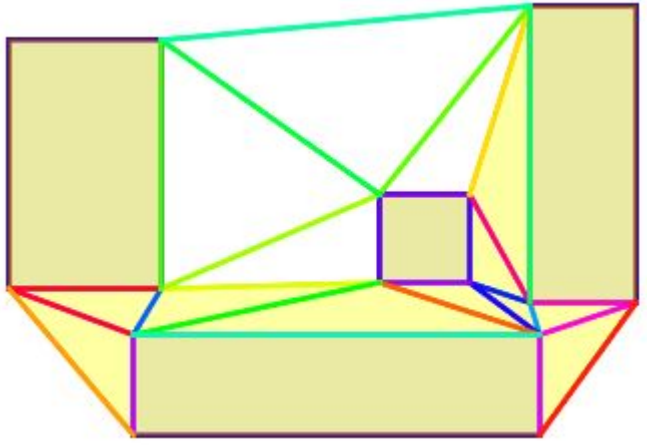


Concave Hull - Polygons: How it works

- Triangulate space around polygons
- Sort triangles by longest edge length
- Remove triangles until tolerance is reached







Polygon Hull Simplification

New

- Computes **Outer** and **Inner Hulls** of polygonal geometry
- Preserves polygonal topology, including holes and MultiPolygons
- Parameter: `vertex_fraction` = fraction of vertices kept

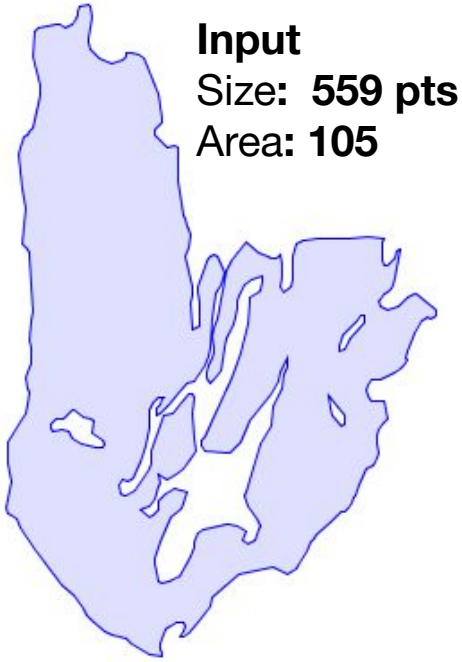
```
SELECT ST_SimplifyPolygonHull( geom, vertex_fraction, is_outer );
```

Outer Polygon Hull

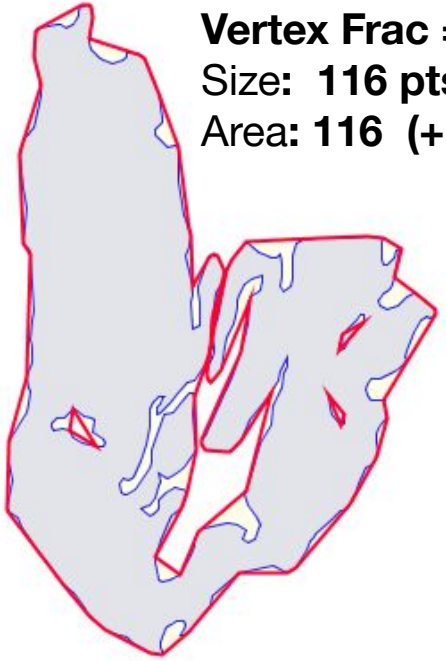
New

- **Fully contains** input geometry
- Preserves topological structure

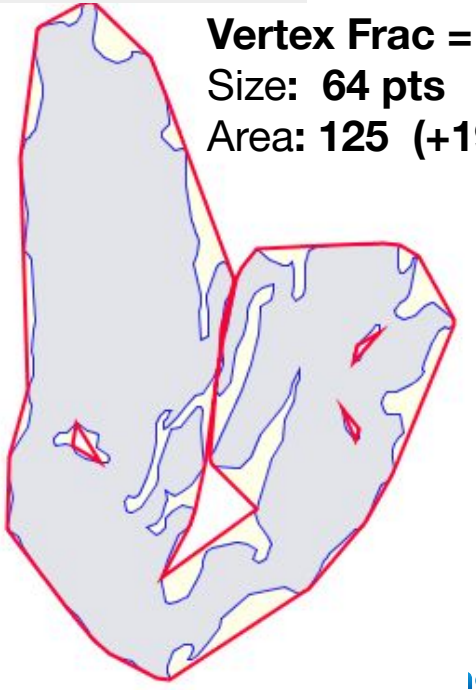
```
SELECT ST_SimplifyPolygonHull( geom, vertex_fraction, true );
```



Input
Size: **559 pts**
Area: **105**



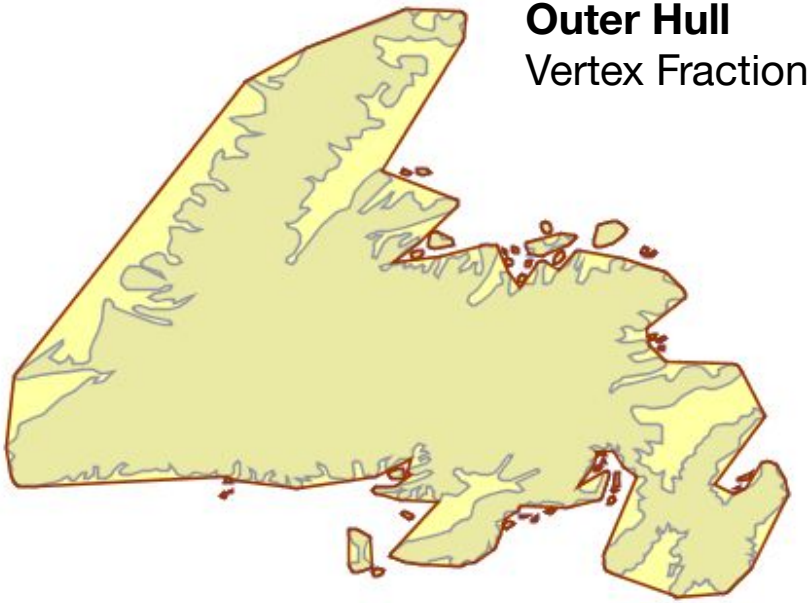
Vertex Frac = 0.2
Size: **116 pts**
Area: **116 (+10%)**



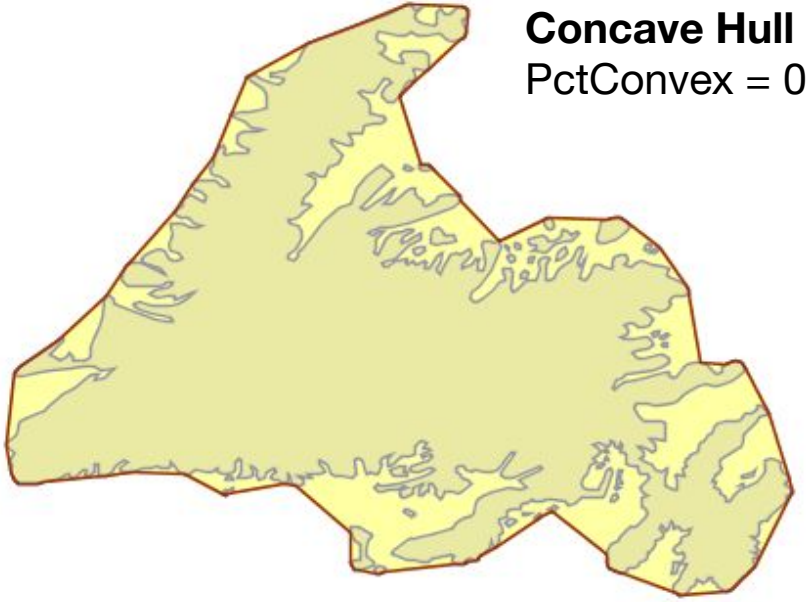
Vertex Frac = 0.1
Size: **64 pts**
Area: **125 (+19%)**

Polygon Outer Hull VS Concave Hull

- Preserves Holes/MultiPolygon VS Single Polygon
- Parameter: Vertex Fraction VS Percent Convex



Outer Hull
Vertex Fraction = 0



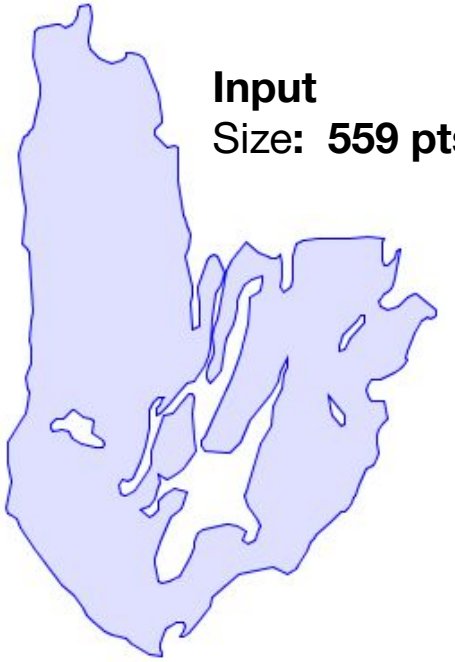
Concave Hull
PctConvex = 0.2

Inner Polygon Hull

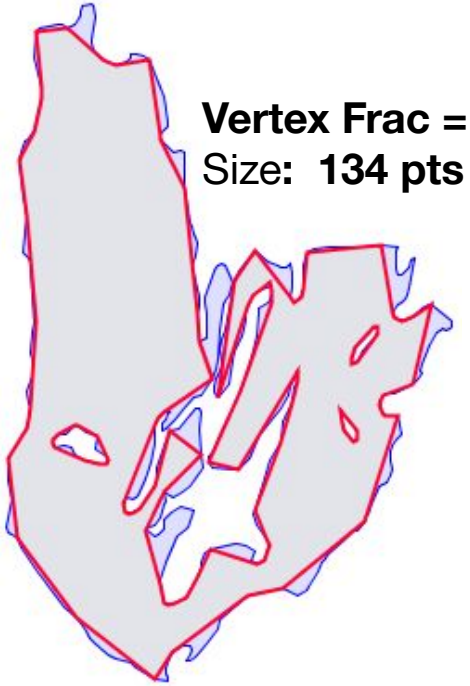
New

- **Fully within** input geometry
- Preserves topological structure

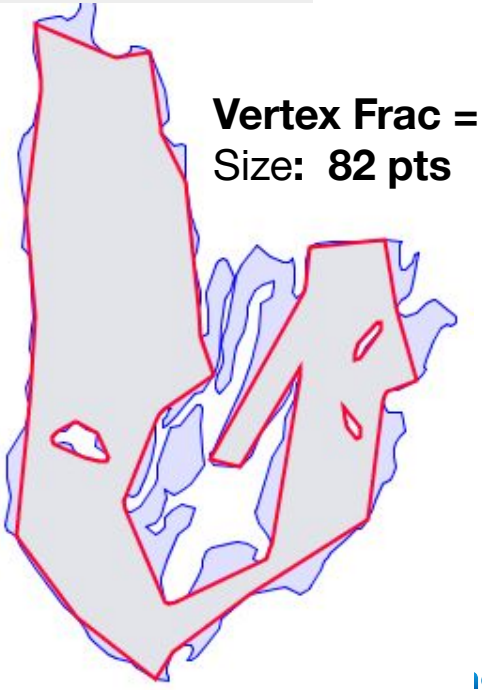
```
SELECT ST_SimplifyPolygonHull( geom, vertex_fraction, false );
```



Input
Size: **559 pts**



Vertex Frac = 0.2
Size: **134 pts**



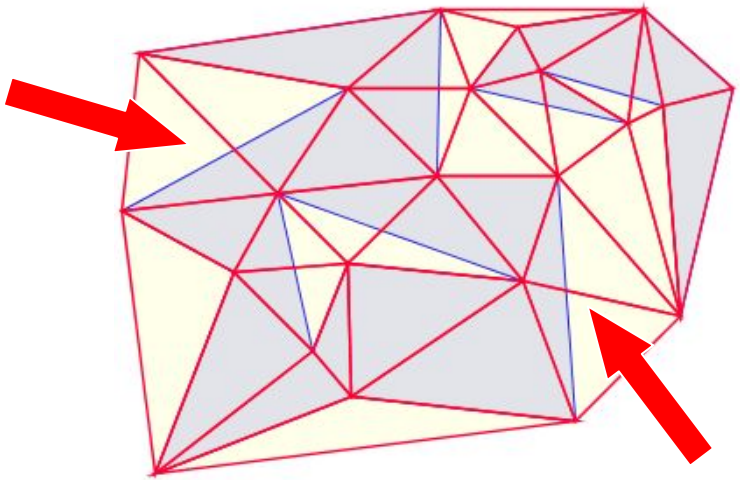
Vertex Frac = 0.1
Size: **82 pts**

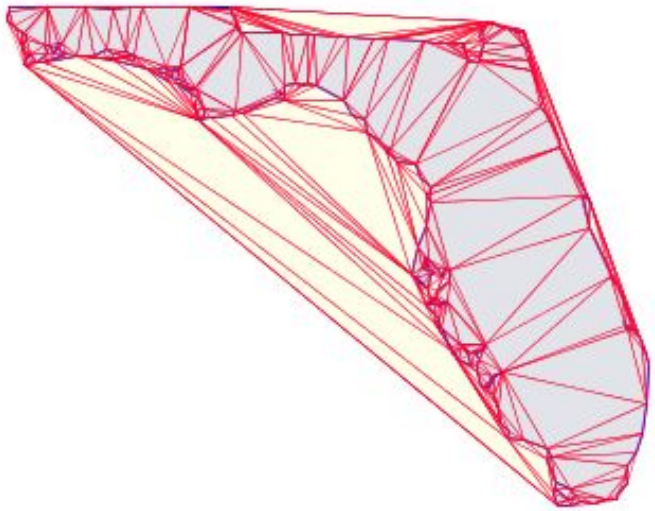
Triangulations

Delaunay Triangulation

- Computes the **Delaunay Triangulation** of points
- Processes vertices **only**
  - *does not respect polygon linework*
  - *does not handle holes or MultiPolygons*

```
SELECT ST_DelaunayTriangles( geom );
```



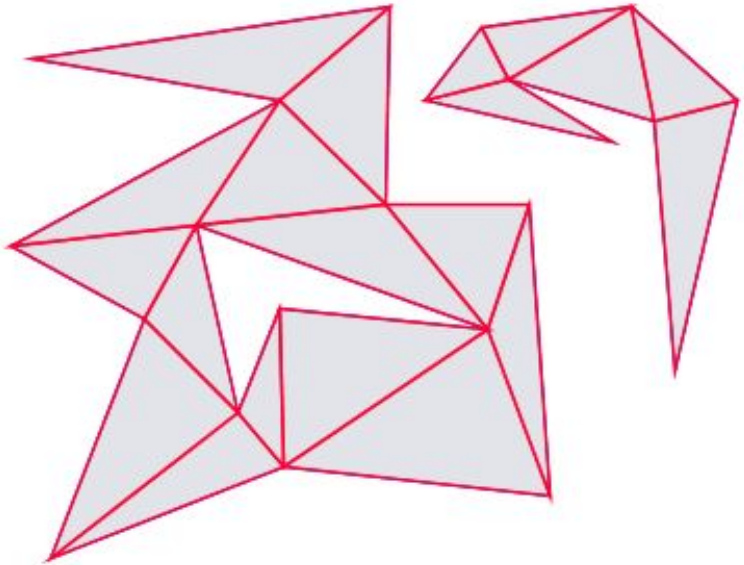


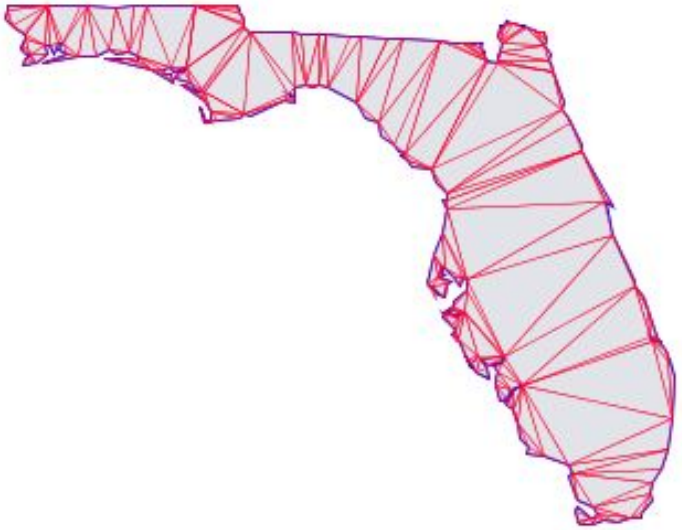
Polygon Triangulation

New

- Computes the **Constrained Delaunay Triangulation** of polygons
  - *respects polygon linework*
  - *handles holes and MultiPolygons*

```
SELECT ST_TriangulatePolygon( geom );
```





crunchy data

FUTURE

Polygonal Coverages

Polygonal Coverages

- A set of non-overlapping polygons
- Many use cases
  - *Cadastral parcels*
  - *Political jurisdictions*
  - *Land use*
  - *Geological regions*
  - *Etc, etc*
- Represent as a topological model?
  - **PostGIS Topology**
- Another option…

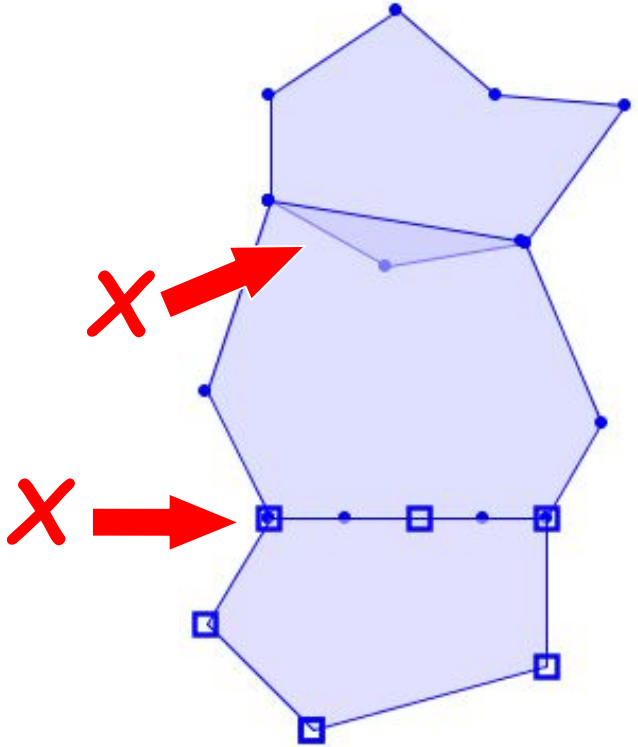
Simple Polygonal Coverage

- Represent Polygonal Coverage as discrete polygons
  - Standard PostGIS table
  - one Polygon/MultiPolygon per row, with attributes *(= Simple Feature)*
  - Supports holes, disjoint regions
  - Topology is implicit - maintained externally
- Advantages
  - *Simple*
  - *Performant*
  - *Matches existing datasets*
  - *Allows all PostGIS geometry functions*
  - *Good fit for relational model*
- But… needs more support!

Polygonal Coverage - Validity

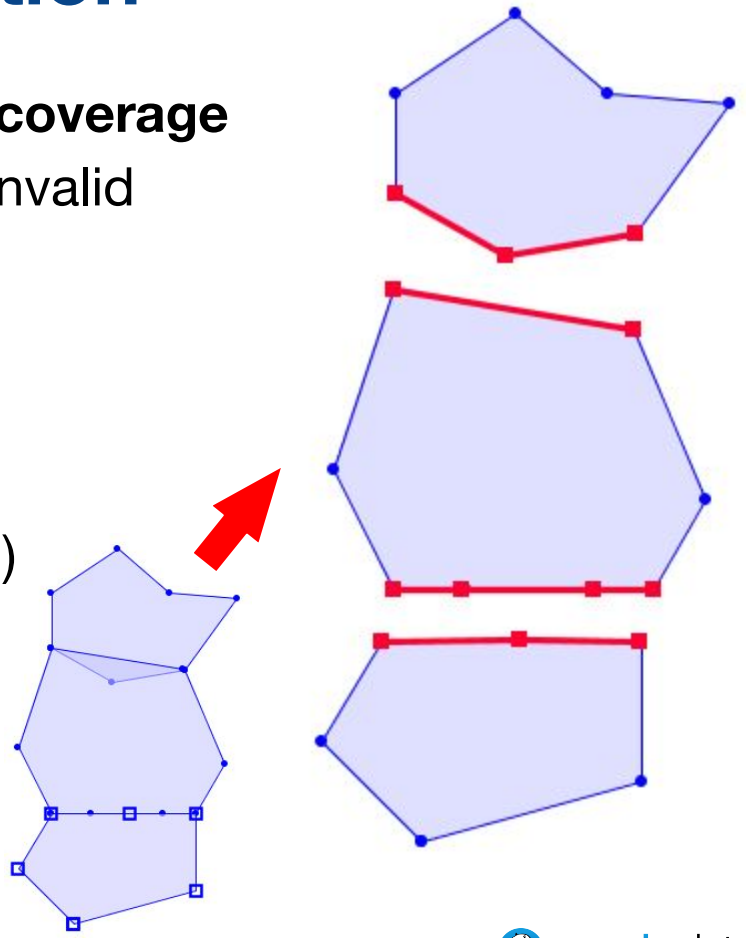
- Coverage Validity required for:
  - Correct operation of coverage functions
  - Accurate modelling and analysis

- A set of polygons is a valid coverage if:
  - Polygons are **valid**
  - Polygons are **non-overlapping**
    - *interiors do not intersect*
  - Adjacent polygons are **edge-matched**
    - *shared lines have identical vertices*



Polygonal Coverage - Validation

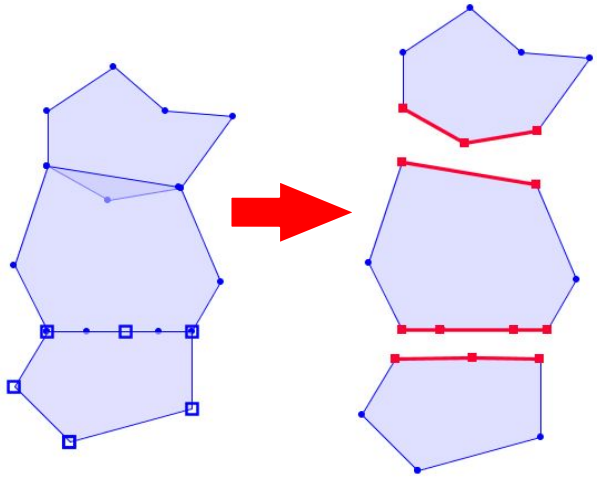
- Tests if a set of **valid** polygons is a **valid coverage**
- For **coverage-invalid** polygons, reports invalid sections of polygon boundary:
  - Overlapping edges
  - Non edge-matched adjacent edges
- For each polygon returns
  - Invalid: invalid edges (MultiLineString)
  - Valid: `NULL`



Coverage Validation - Window Function

- Window function
  - process a set of rows, return a result for **each** row
- **`ST_CoverageValidate(geom)`**
  - Process entire coverage, or subset
  - Returns invalid edges, or `NULL` if valid
  - Keep some/all attributes for each coverage feature
  - Filter to include only invalid results
  - Empty result set => coverage is valid

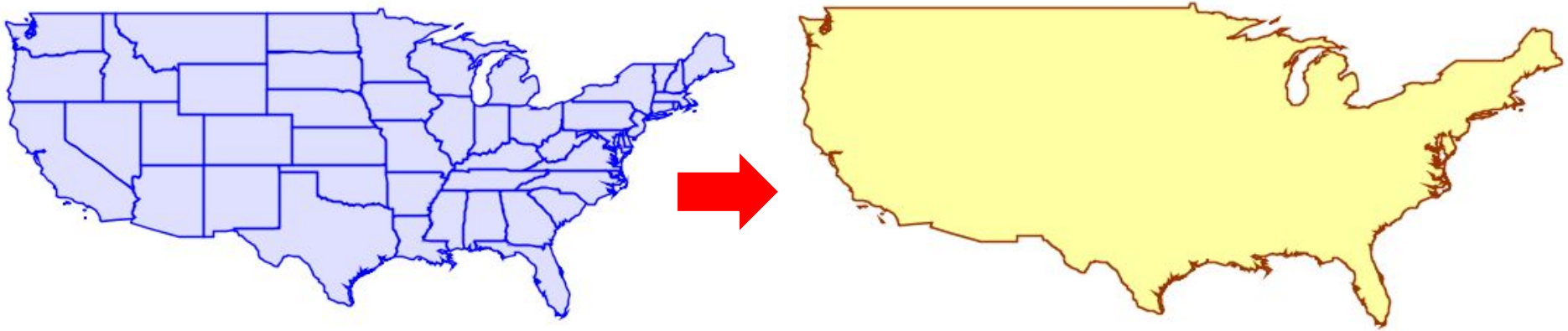
```
SELECT * FROM (
    SELECT id …,
        ST_CoverageValidate(geom) OVER()
            AS invalid_edges
    FROM poly_cov) AS t
WHERE invalid_edges IS NOT NULL;
```



Polygonal Coverage - Union

- Computes the union of a set of coverage polygons
- Aggregate function, returns polygonal geometry
- Very fast (can be 100x faster than `ST_Union`)

```
SELECT ST_CoverageUnion( geom ) FROM poly_cov;
```



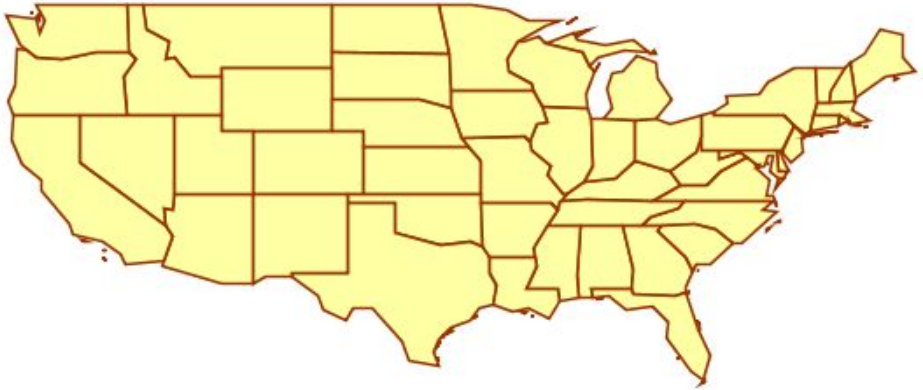
Polygonal Coverage - Simplification

- Simplifies the boundaries of a set of coverage polygons
- Preserves topology; result is a valid coverage with identical structure
- Window function, returning each simplified geometry

```
SELECT id, attr, …, ST_CoverageSimplify(geom, tolerance) OVER() AS geom
    FROM poly_cov;
```

Size: **11,481 pts**

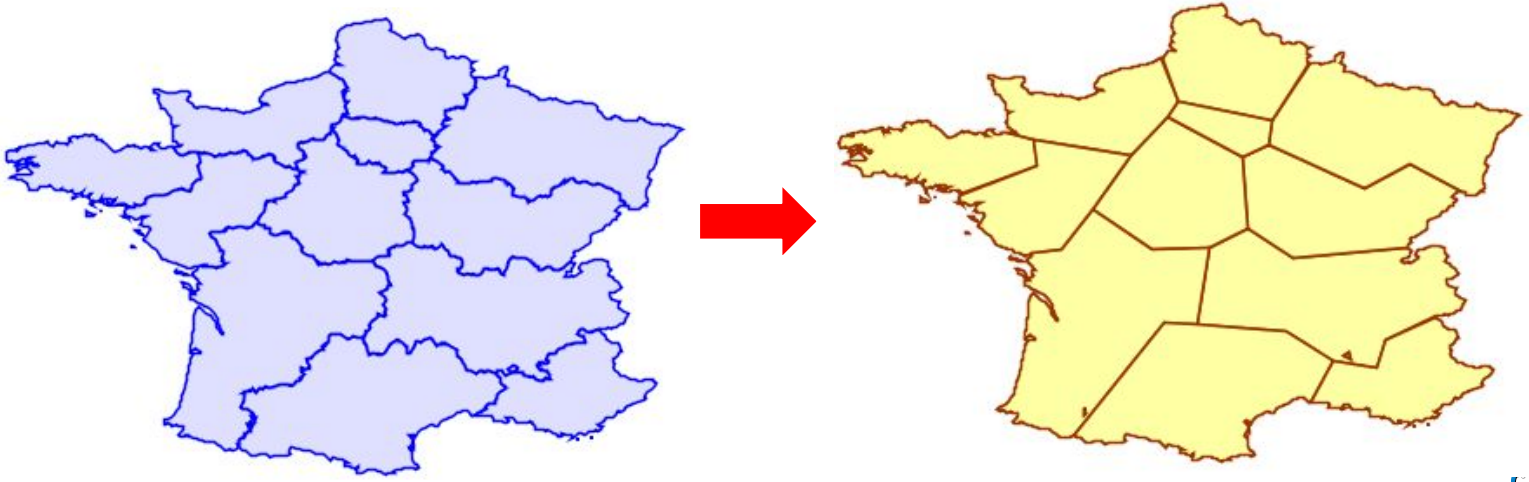
Size: **739 pts**



Polygonal Coverage - Inner Simplification

- Simplifies the **inside boundaries** of a set of coverage polygons
- Preserves topology; result is a valid coverage with identical structure
- Window function, returning each simplified geometry

```
SELECT id, attr, …, ST_CoverageSimplifyInner(geom, tolerance) OVER() AS geom
    FROM poly_cov;
```



crunchy data

Future Work

- Polygonal Coverage functions
  - Find Gaps
  - Clean
  - Precision Reduce
  - Overlay ?