https://bit.ly/2JYfqCA

Everything About PostGIS

paul.ramsey@crunchydata.ca

Motivation

I want a computer map!

By golly, my friend, you need to buy a...

GIS!

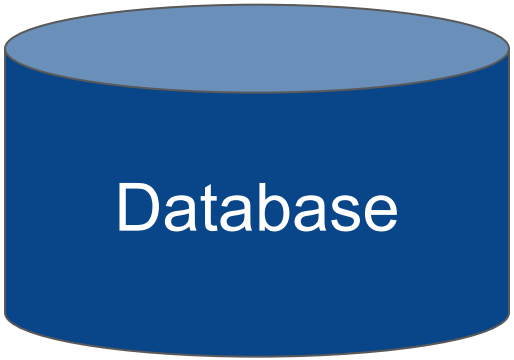
Wait! Stop!

- Do you have PostgreSQL?
- Do you have PostGIS?

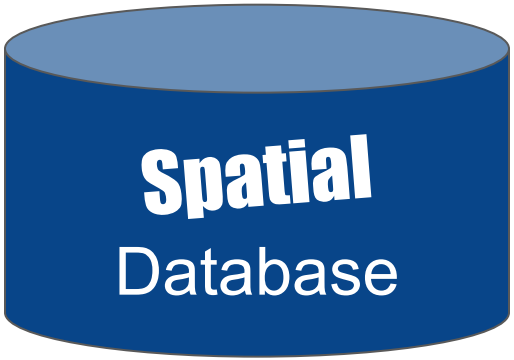
You already have a GIS!

What? (is PostGIS?)


“spatial”
database



- **Types**
  - string, float, date
- **Indexes**
  - b-tree, hash
- **Functions**
  - strlen(text)
  - pow(float, float)
  - now()



- **Spatial Types**
  - geometry, geography
- **Spatial Indexes**
  - t-tree, quad-tree, kd-tree
- **Spatial Functions**
  - ST_Length(geometry)
  - ST_Buffer(geometry, R)
  - ST_X(geometry)

Reasoning by Analogy

PostGIS  is to PostgreSQL 

as

ORACLE SPATIAL is to ORACLE

Spatial Database Marketplace

ORACLE SPATIAL

Microsoft® SQL Server®

And **everybody** else:

- SpatiaLite
- BigQuery
- Teradata
- Netezza
- OmniSci
- ....

Spatial Database Marketplace

Open Geospatial Consortium
“Simple Features for SQL
Version 1.2”

International Standards
Organization
SQL/MM Part 3:
Spatial/Temporal

Spatial Database Marketplace

ST_Function()

ISO SQL/MM Part 3: Spatial
13249-3

except ORACLE

Google BigQuery

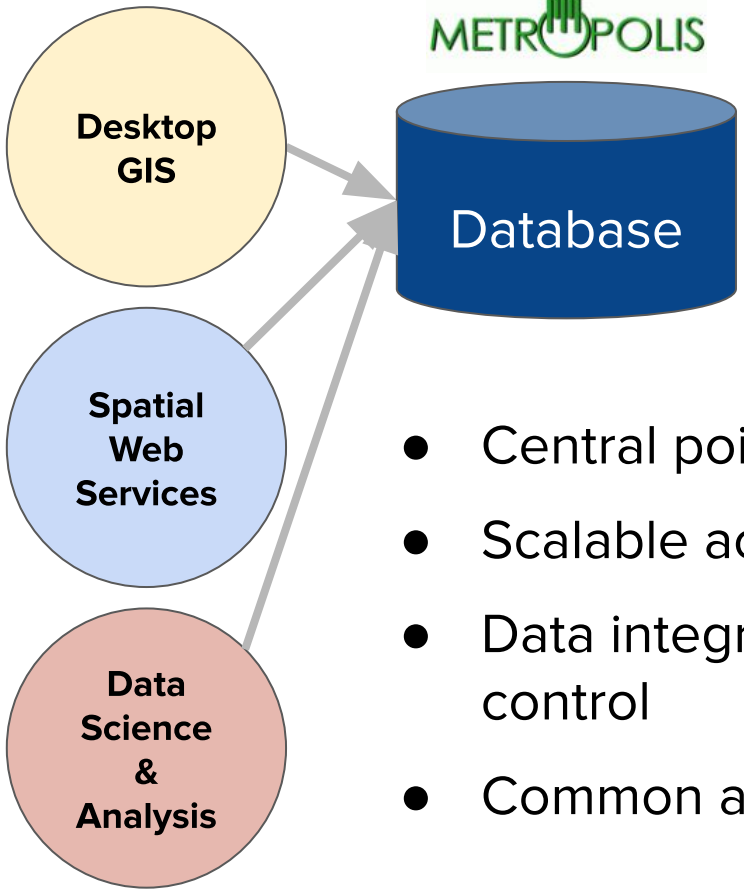
We now support the following geographic types: Point, Linestring, Polygon, Multi-polygon, and Collections. Functions will look and feel very familiar to users of PostGIS as they follow the SQL/MM Spatial standard.

Why? (PostGIS?)

Spatial Enterprise Architecture

Why does City of Metropolis manage their spatial data as a collection of files?

Spatial Enterprise Architecture



That's better!

- Central point of truth
- Scalable access
- Data integrity and transactional control
- Common access pattern (SQL)

Spatial Enterprise Database

- Integration with third party applications
  - Desktop
  - Middleware
  - Scripting
- Baseline Functionality
  - Spatial types to support business
  - Spatial indexes to support business
- Enterprisey-ness
  - Scalability, security, reliability, support

PostGIS Desktop Integration

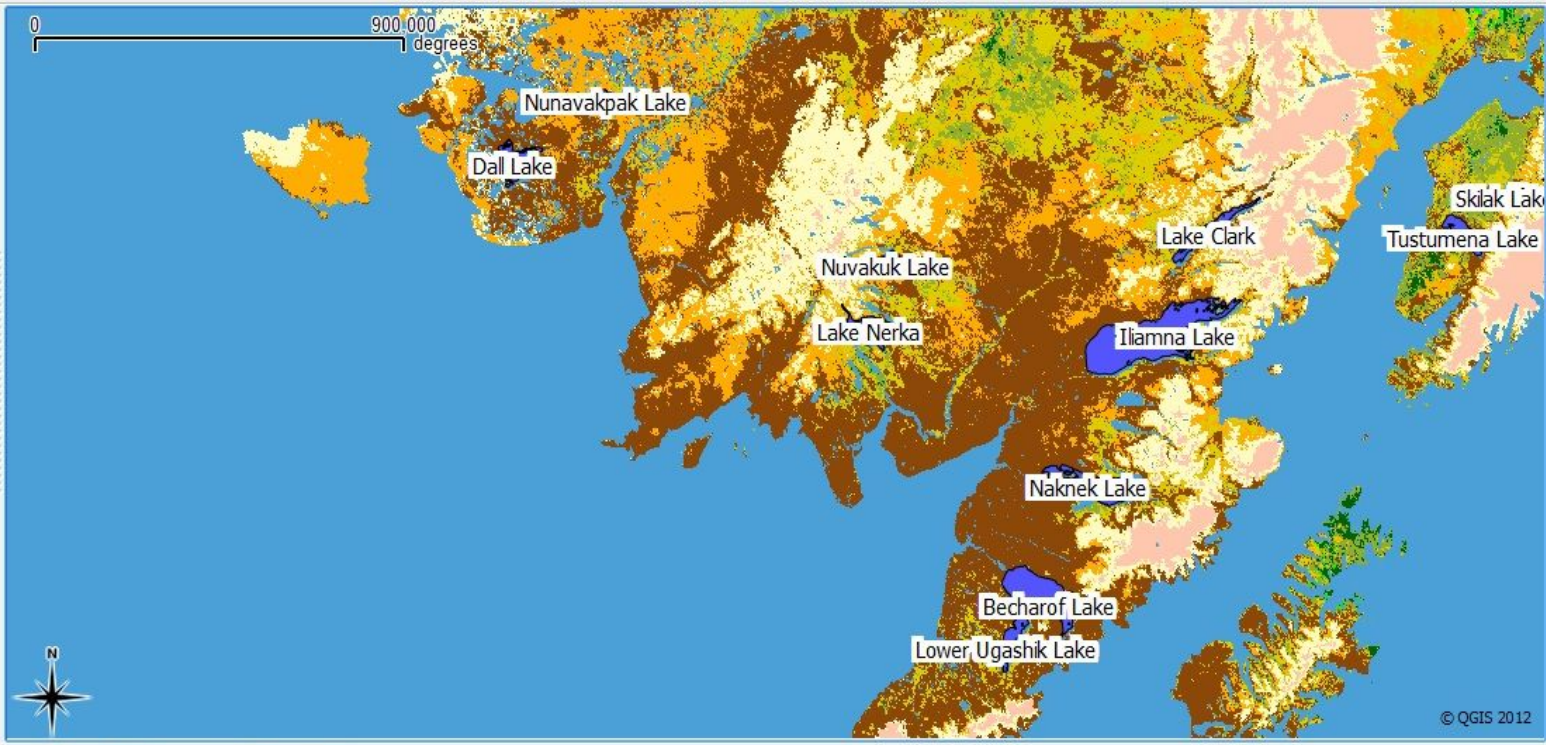
Quantum GIS 1.7.3-Wroclaw

File Edit View Layer Settings Plugins Raster Database Vector Help

Layers

lakes

landcover



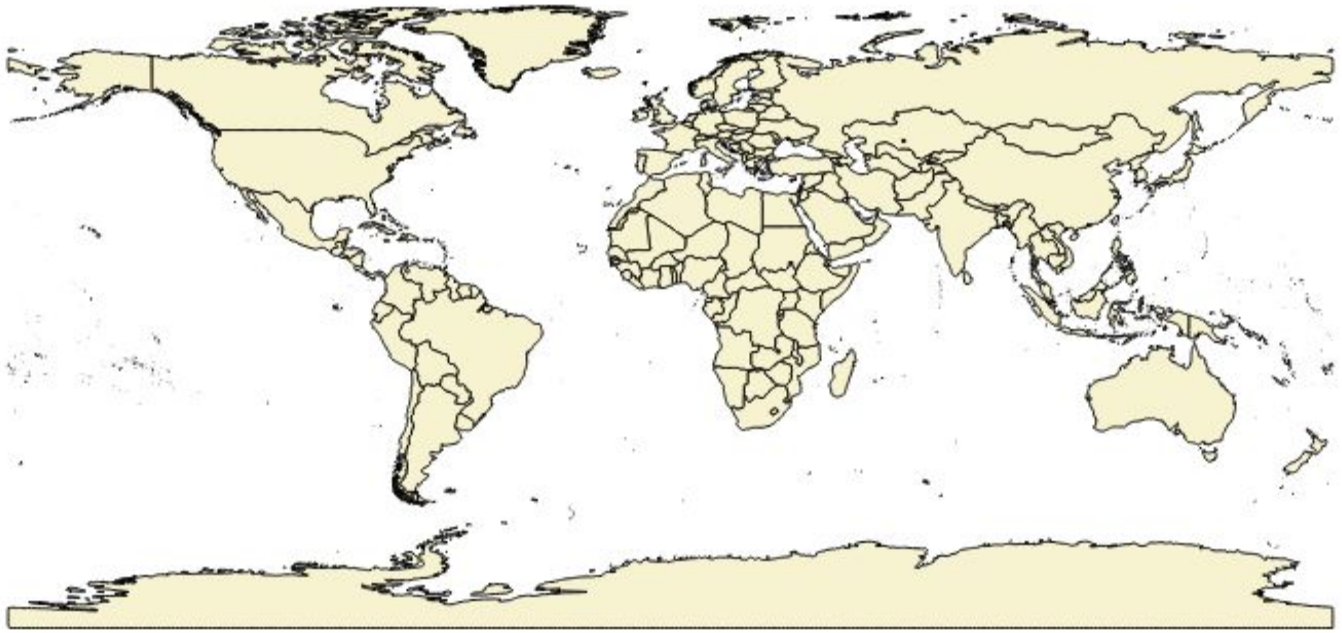
Coordinate: -2273060,3010907 Scale :1034230518564 Render EPSG:4326

world - QGIS a09cb6f364

Project Edit View Layer Settings Plugins Vector Raster Database Web Processing Help

Layers Panel

ne_10m_admin_0_cou...



Type to locate (Ctrl+K)

-27.6,-99.8

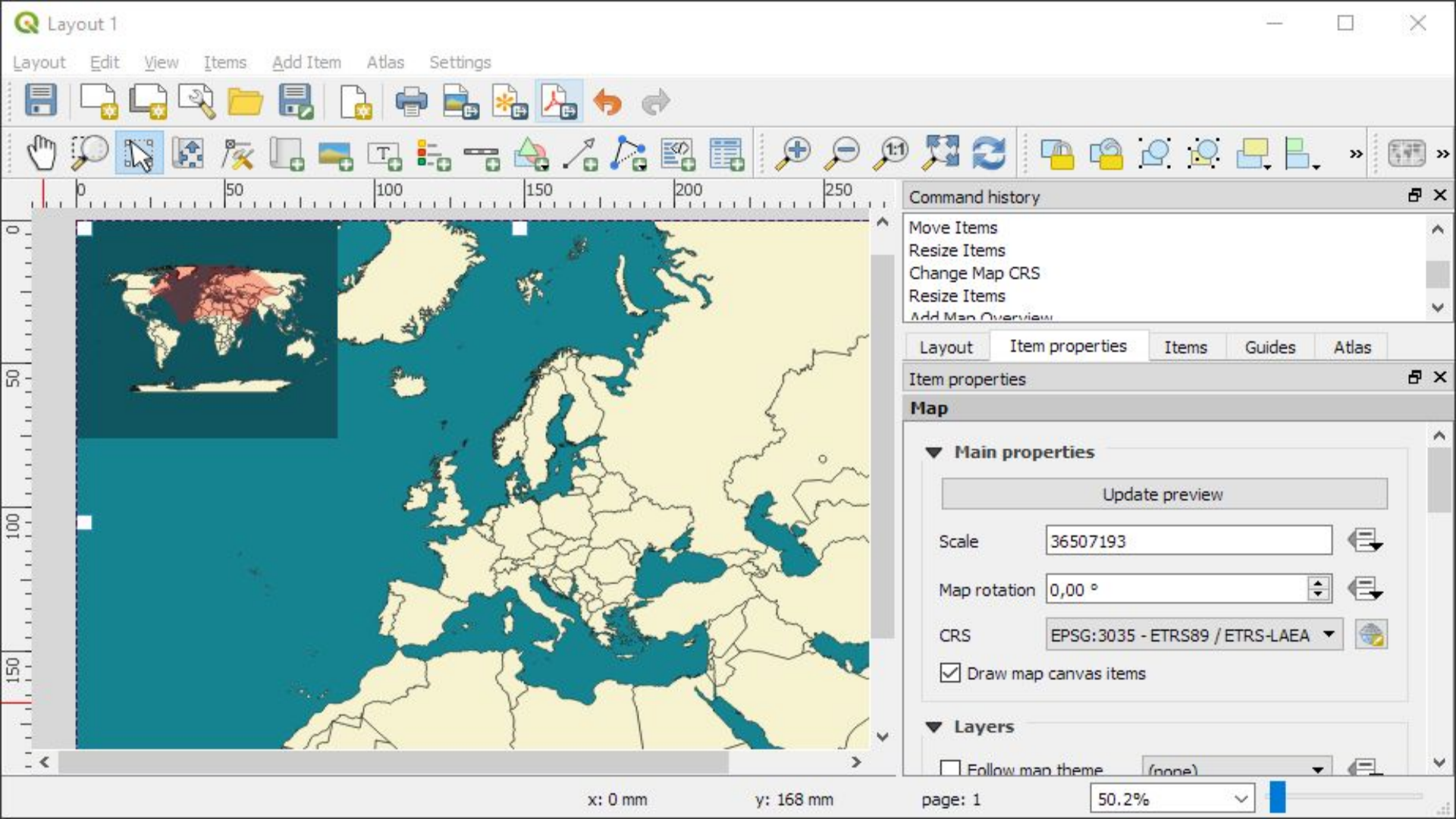
Scale 225 119 774

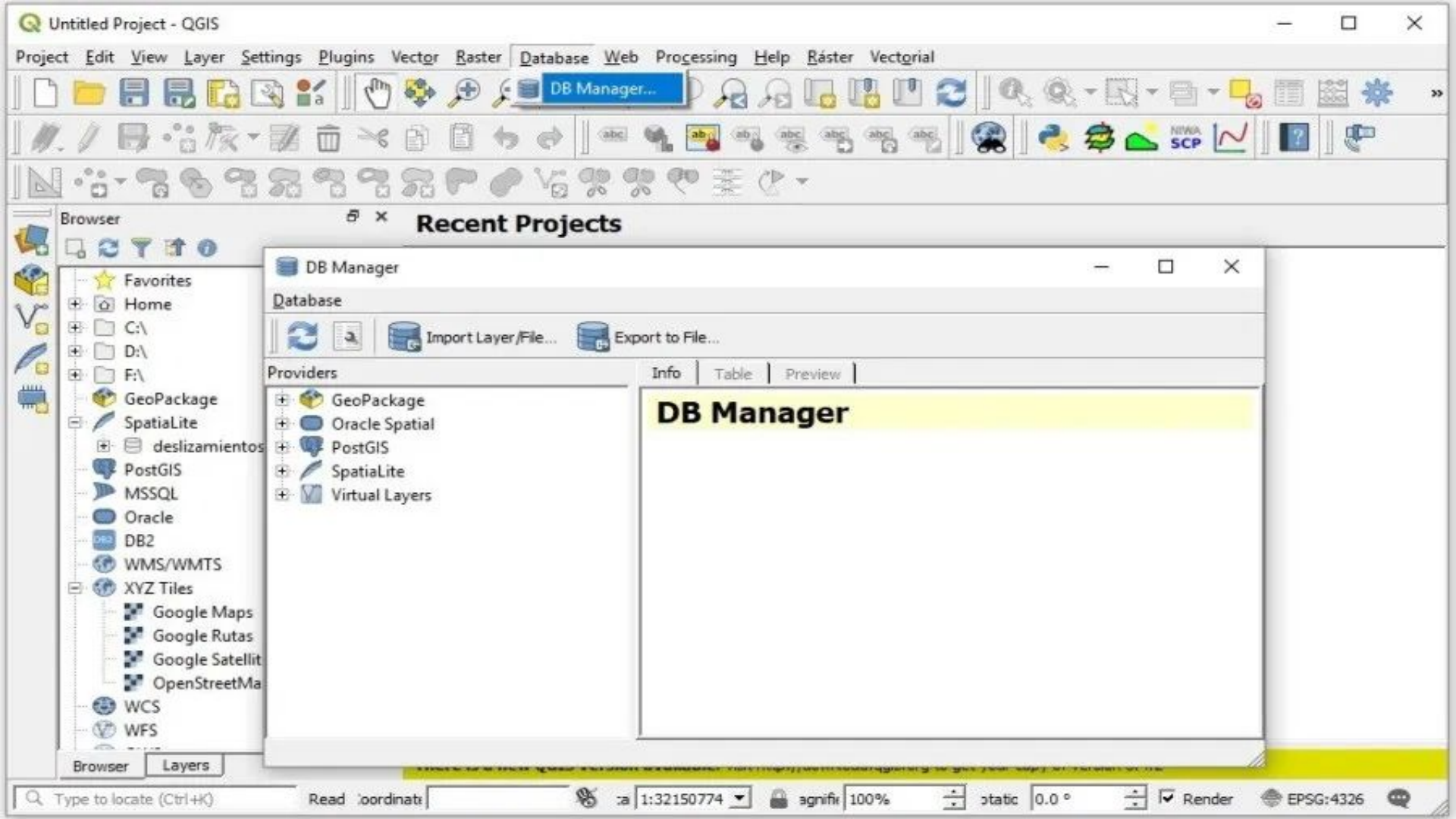
100%

0,0 °

Render

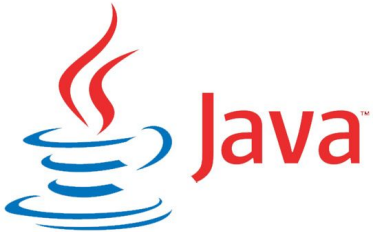
EPSG:4326

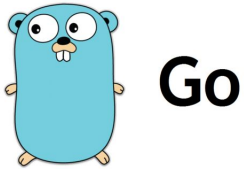

Layout 1
Layout
Edit
View
Items
Add Item
Atlas
Settings
Command history
Move Items
Resize Items
Change Map CRS
Resize Items
Layout
Item properties
Items
Guides
Atlas
Item properties
Map
Main properties
Update preview
Scale
36507193
Map rotation
0,00 °
CRS
EPSG:3035 - ETRS89 / ETRS-LAEA
Draw map canvas items
Layers
Follow map theme
(none)
x: 0 mm
y: 168 mm
page: 1
50.2%


Untitled Project - QGIS
Project Edit View Layer Settings Plugins Vector Raster Database Web Processing Help Ráster Vectorial
DB Manager...
Browser
Recent Projects
Favorites
Home
C:\
D:\
F:\
GeoPackage
SpatiaLite
deslizamientos
PostGIS
MSSQL
Oracle
DB2
WMS/WMTS
XYZ Tiles
Google Maps
Google Rutas
Google Satellit
OpenStreetMa
WCS
WFS
Browser
Layers
DB Manager
Database
Import Layer/File...
Export to File...
Providers
GeoPackage
Oracle Spatial
PostGIS
SpatiaLite
Virtual Layers
Info
Table
Preview
DB Manager
Type to locate (Ctrl+K)
1:32150774
100%
0.0 °
Render
EPSG:4326

PostGIS Middleware Integration

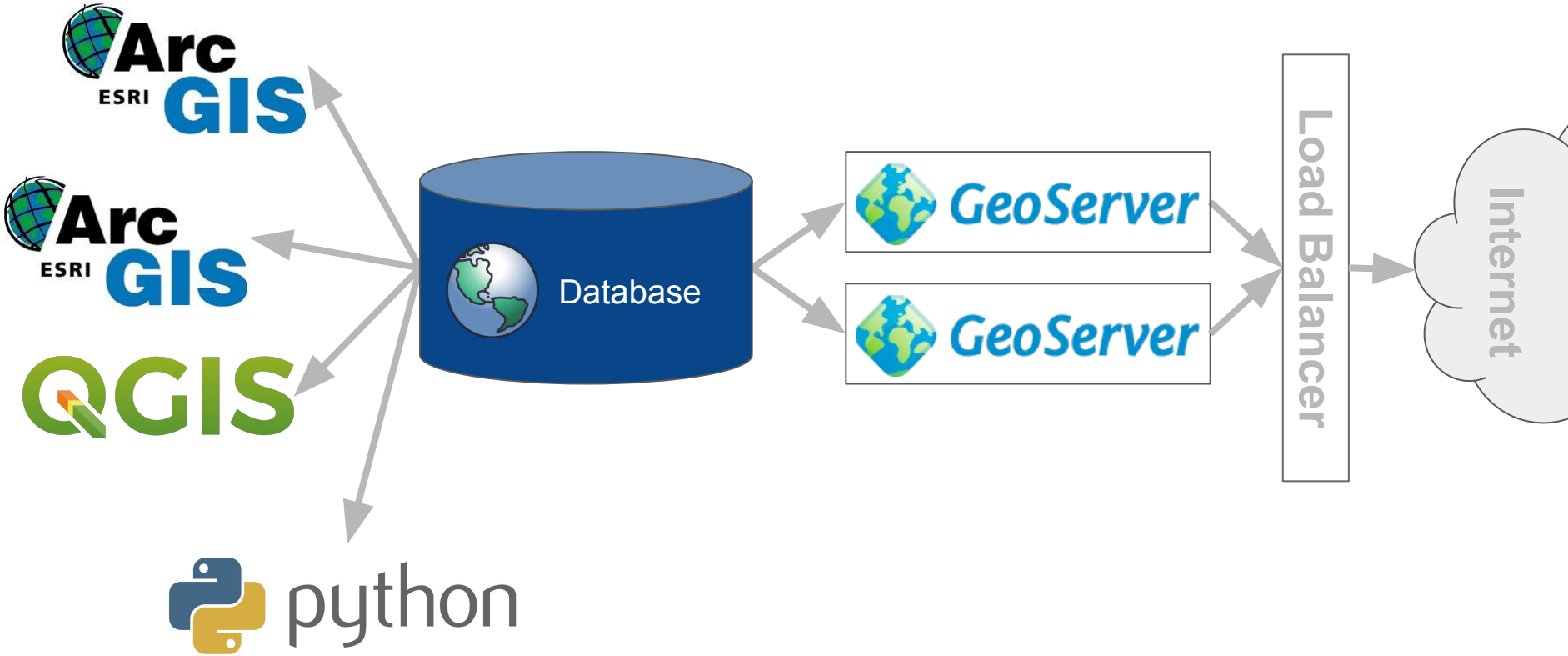
PostGIS Language Integration





Ruby, C, C++, DCOM, JDBC, ODBC, node.js, Scala, Groovy, Mono, Haskell, Rust, ...

Example PostGIS Hybrid Architecture

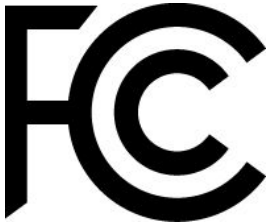


Who? (PostGIS?)

PostGIS Enterprise Reference Users (Gov't)

Natural Resources Canada | Ressources naturelles Canada

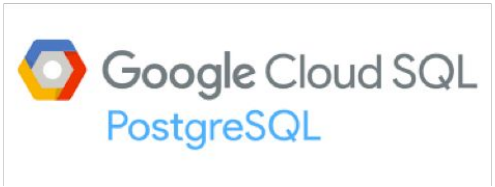
Canada



PostGIS Enterprise Reference Users (Corp)

PostGIS Cloud Users

Google Cloud Platform



Amazon RDS

PostGIS Cloud Options

Your own managed cluster?

kubernetes

OPENSHIFT

Deployed on?

Google Cloud Platform

Azure

amazon web services™

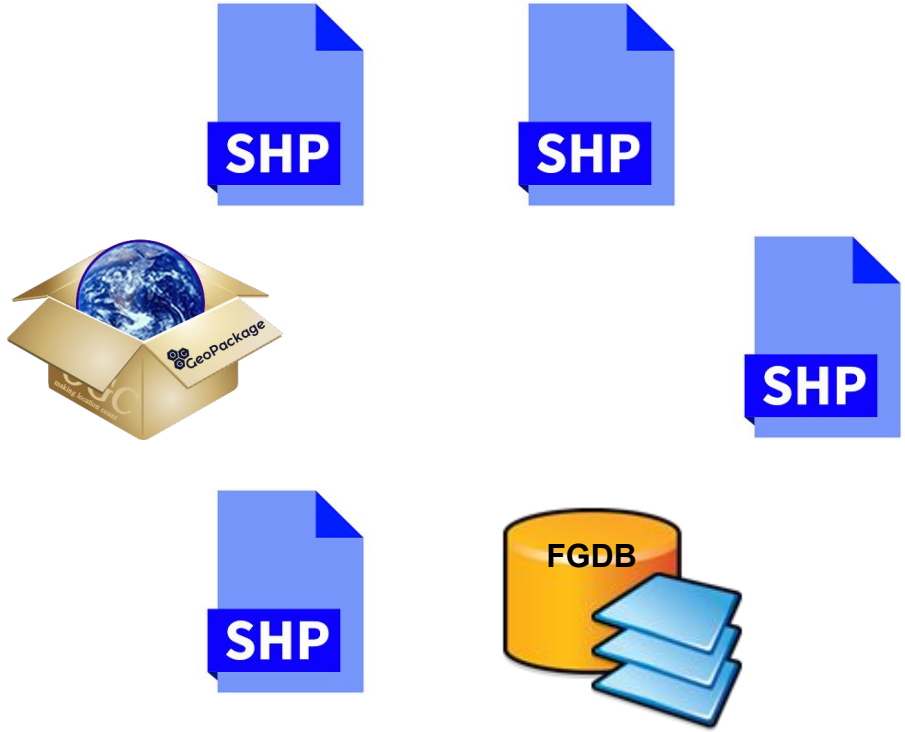
crunchy data

- Database-as-a-service
- Crunchy PostgreSQL “operator” for k8s
- Deploy on any cloud


History

Refractions RESEARCH

“Managing changing data in shape files is a pain in the _____!”



Refractions RESEARCH & BRITISH COLUMBIA

BC Watershed Atlas

"we need a spatial type!

BC Digital Road Atlas

distance()
indexes (v0.1)

BC Corporate Watersheds

"lightweight"
geometry (v1.0)

GIS without the GIS

GIS without the GIS

- **Move workload** from low-performance scripts to high-performance SQL
- Pipe analysis directly from database to the web, **without spatial middleware**
- Give developers **direct access** to powerful GIS analytics engine

How? (PostGIS?)

```
CREATE EXTENSION postgis;
```

it's a feature frenzy!!! (ack!)

freedom!

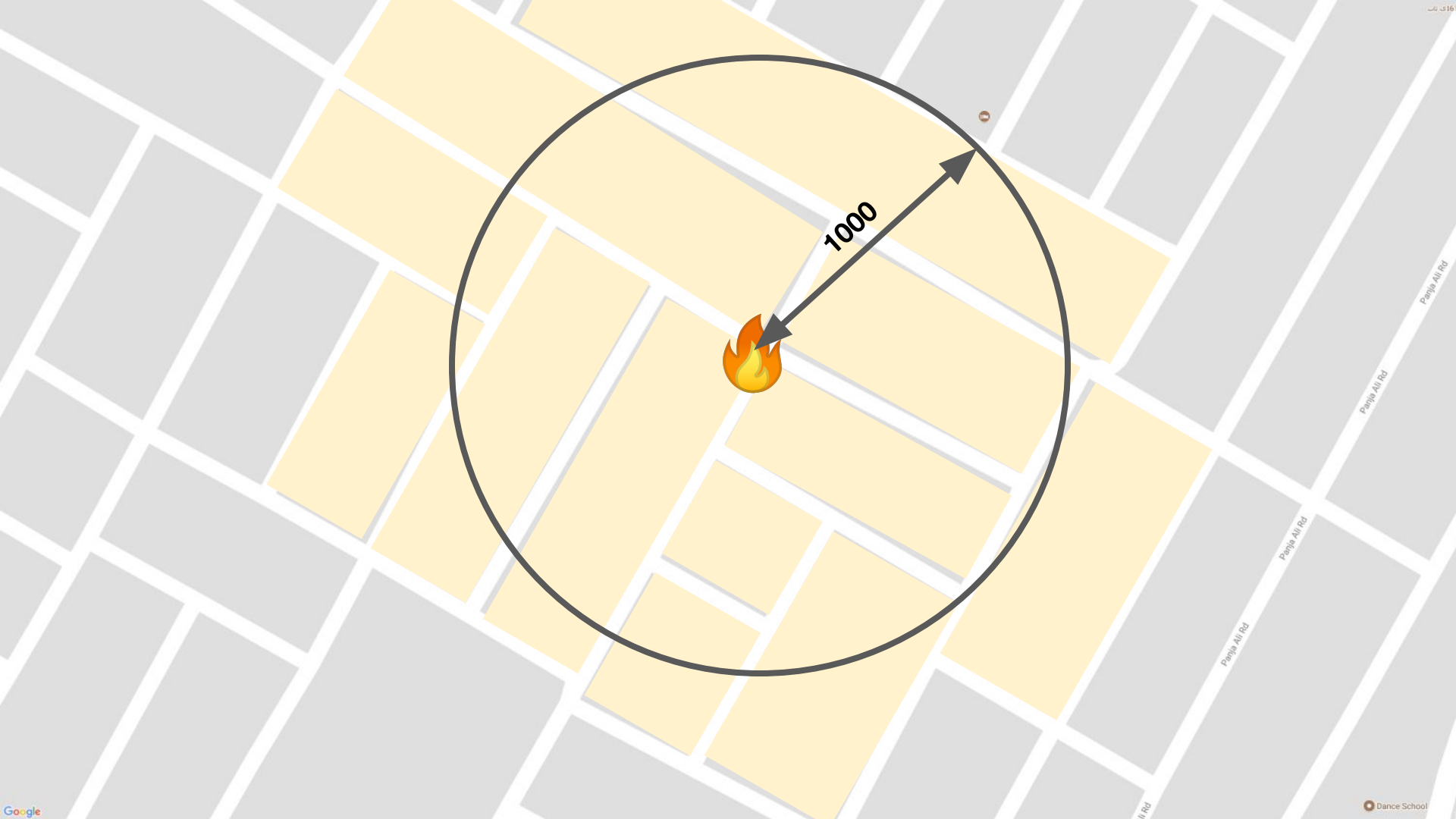
NoSQL

is the ultimate hotness


YeSQL!

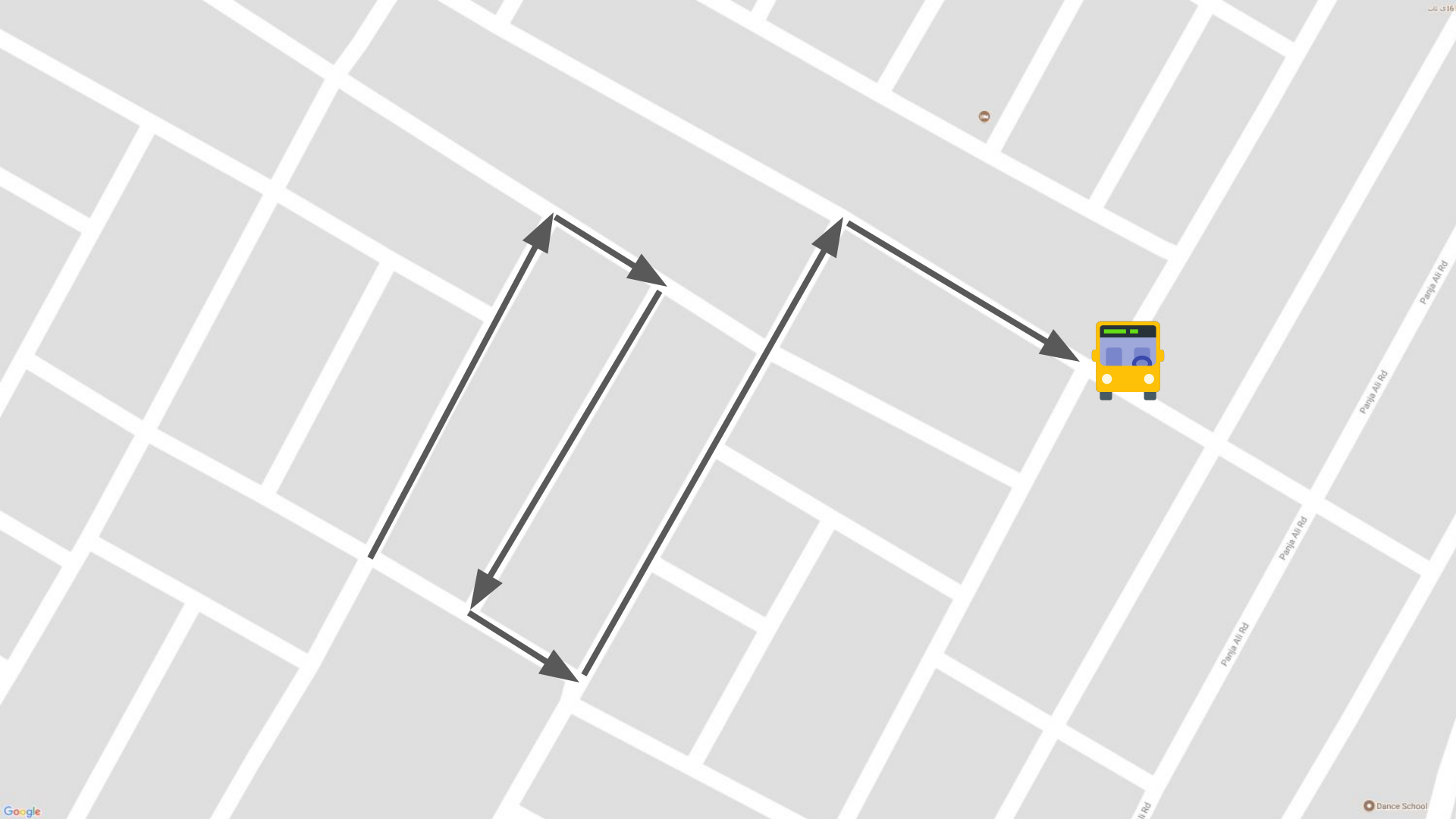
GIS without the GIS

“What parcels are within 1km of the fire?”


1000
Panja Ali Rd
Panja Ali Rd
Panja Ali Rd
Google
Dance School

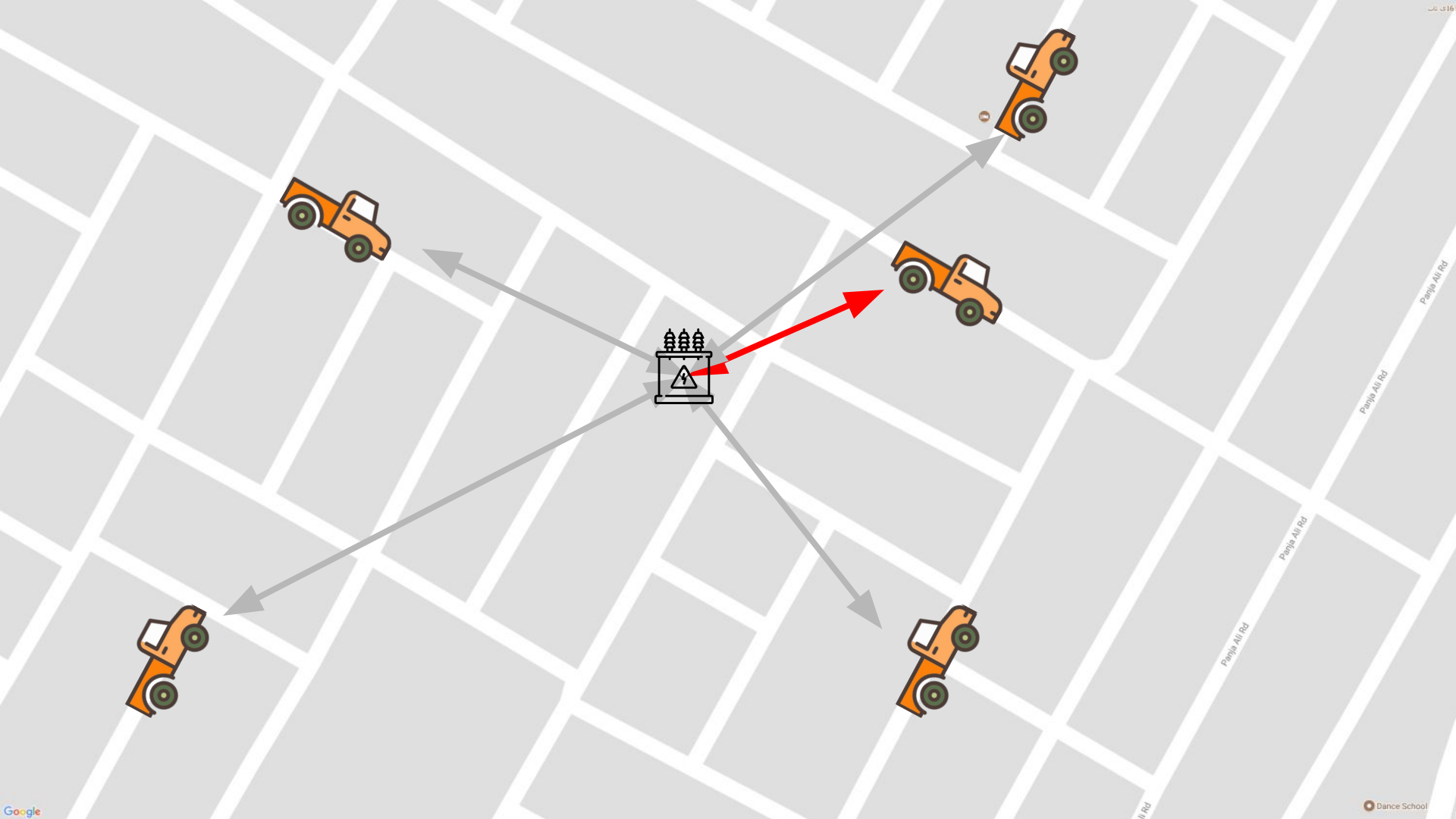
```
SELECT owner_phone
FROM parcels
WHERE ST_DWithin(
          geom,
          'POINT()',
          1000
      );
```

“How far did the bus travel last week?”


Panja Ali Rd
Panja Ali Rd
Panja Ali Rd
Panja Ali Rd
Google
Dance School

```
SELECT Sum(ST_Length(geom))
  FROM vehicle_paths
 WHERE id = 12
   AND date > Now() - '7d';
```

“Find me the nearest truck to the transformer.”

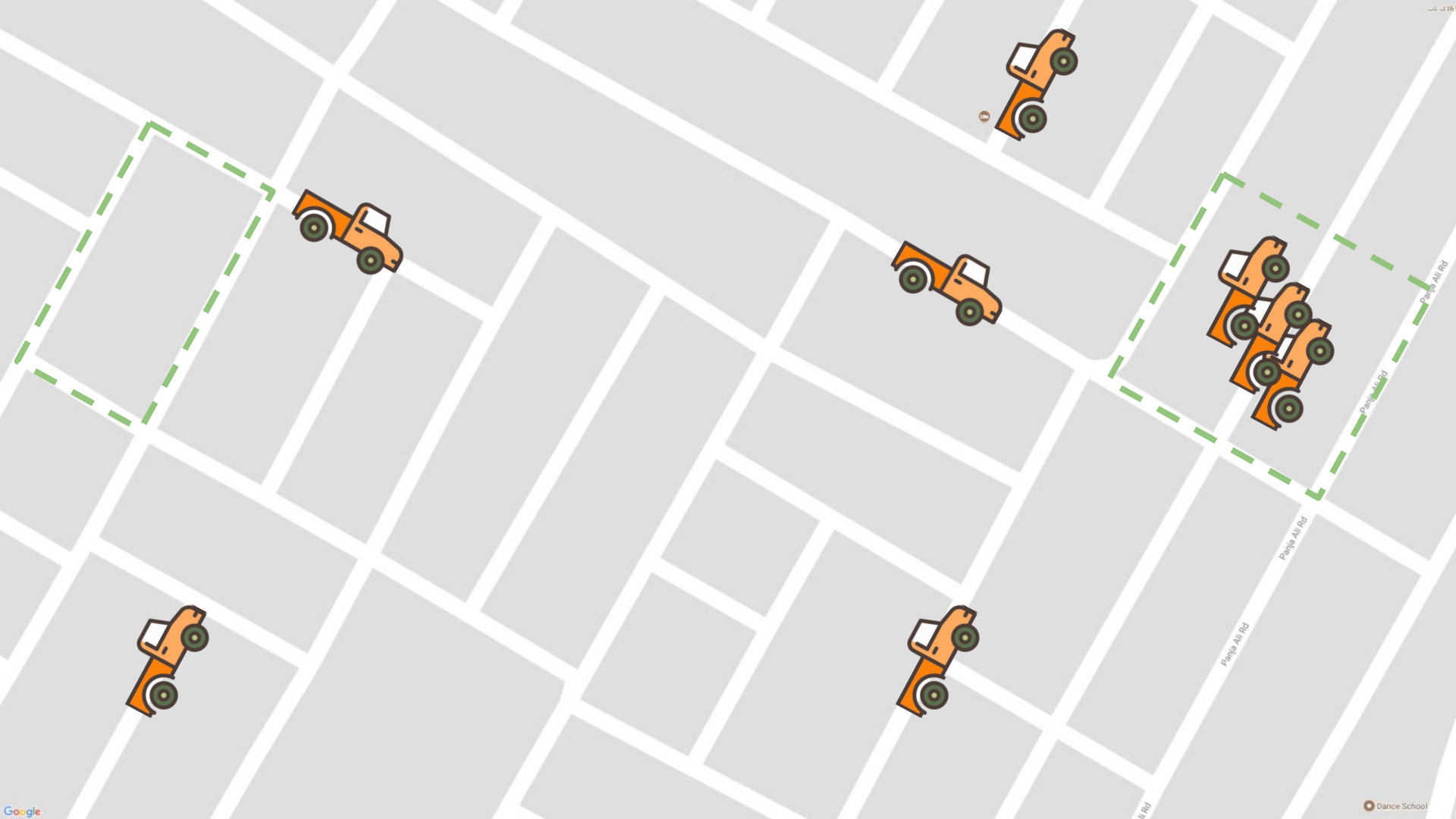

Panja Ali Rd
Panja Ali Rd
Panja Ali Rd
Google
Dance School

```
SELECT t.*
FROM trucks t
ORDER BY t.geom <-> (
  SELECT geom
  FROM transformers
  WHERE trans_id = 12
  )
LIMIT 1;
```

find the transformer

find the truck

“What trucks are in the service depots?”


Panja Ali Rd
Panja Ali Rd
Panja Ali Rd
Google
Dance School

```
SELECT yards.id, trucks.*
FROM trucks t
JOIN yards y
ON ST_Contains(y.geom,
               t.geom)
WHERE y.code = 'service';
```

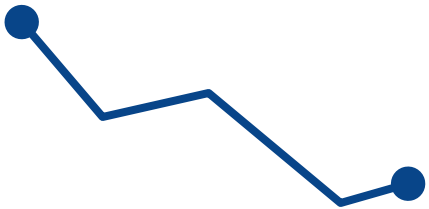

Spatial
Database
Basics
crunchydata.com

PostGIS Core - SFSQL

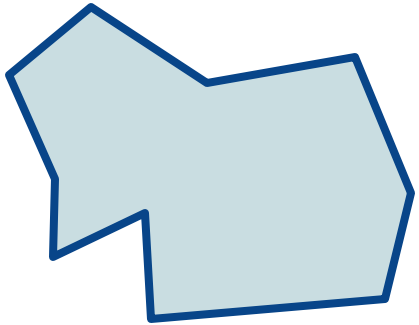
```
CREATE TABLE my_table (
  id INTEGER SERIAL PRIMARY KEY,
  name VARCHAR(28),
  geom GEOMETRY(Point, 26910)
)

CREATE TABLE my_table (
  geog GEOGRAPHY(Point, 4326)
)
```

PostGIS Core - SFSQL

POINT

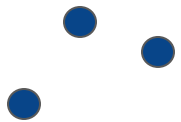


LINESTRING

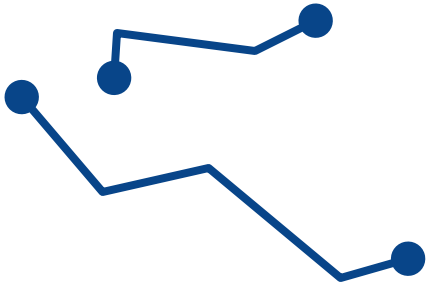


POLYGON

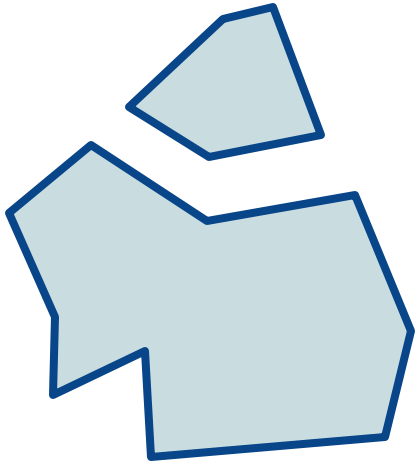
PostGIS Core - SFSQL



MULTI POINT

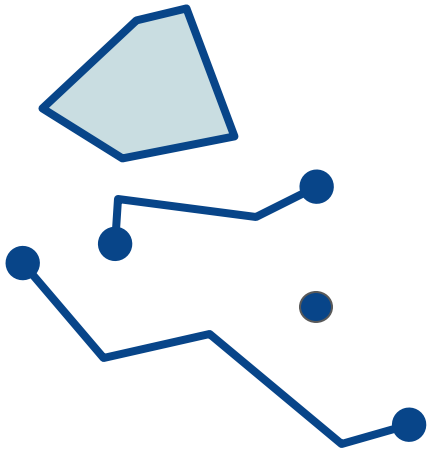


MULTI LINESTRING



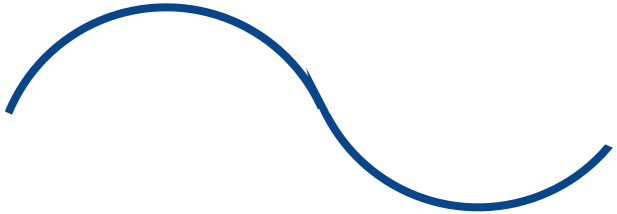
MULTI POLYGON

PostGIS Core - SFSQL

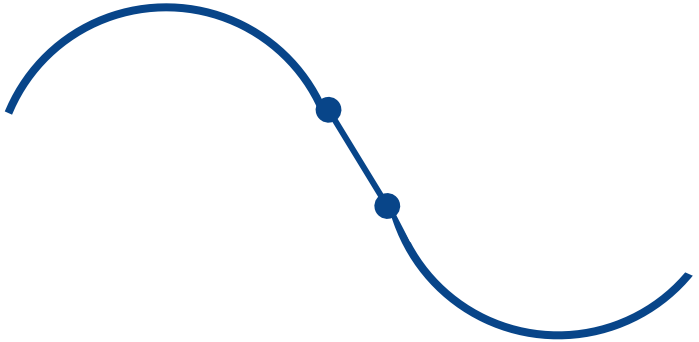


GEOMETRY COLLECTION

PostGIS Core - ISO



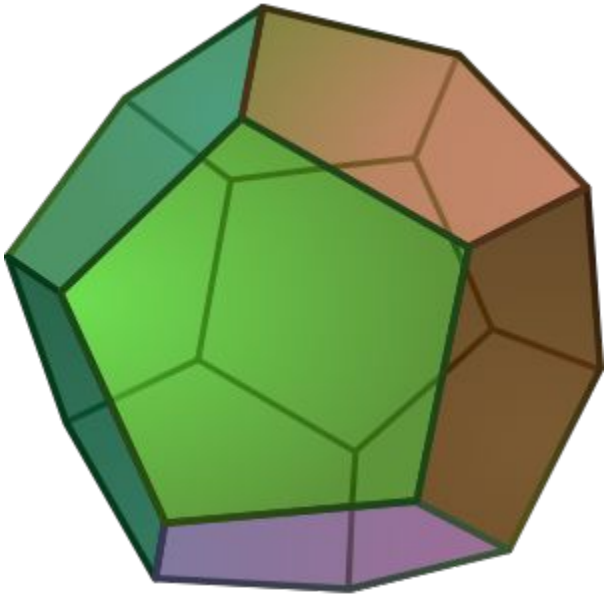
CIRCULAR STRING



COMPOUND CURVE

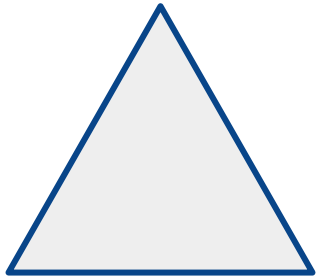
- CURVEPOLYGON
- MULTICURVE
- MULTISURFACE

PostGIS Core - ISO Extended

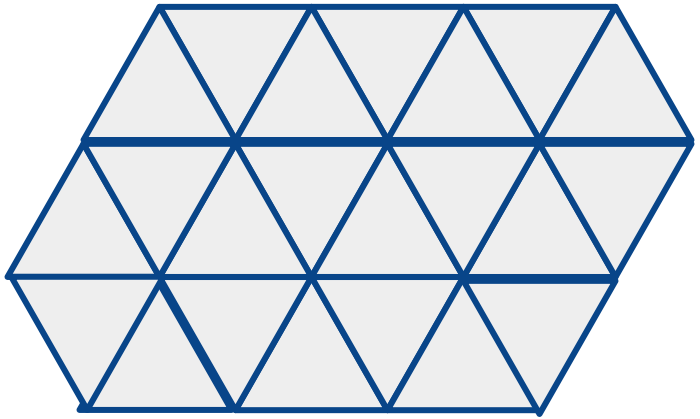


POLYHEDRAL SURFACE

PostGIS Core - ISO Extended



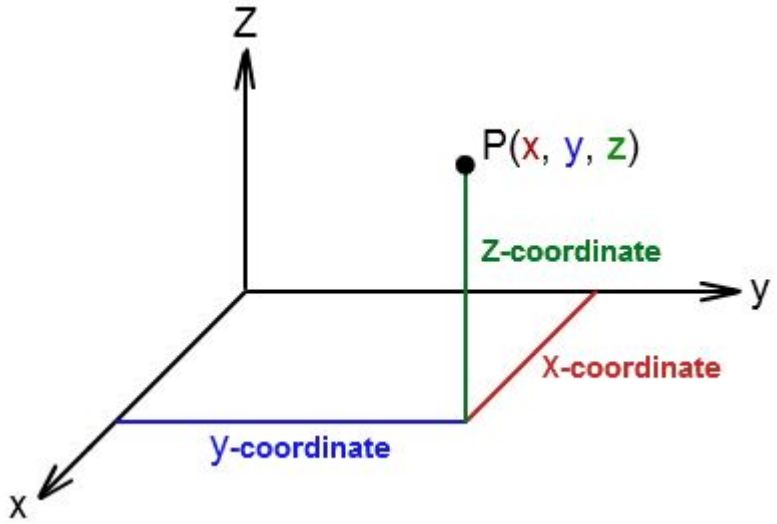
TRIANGLE



TIN

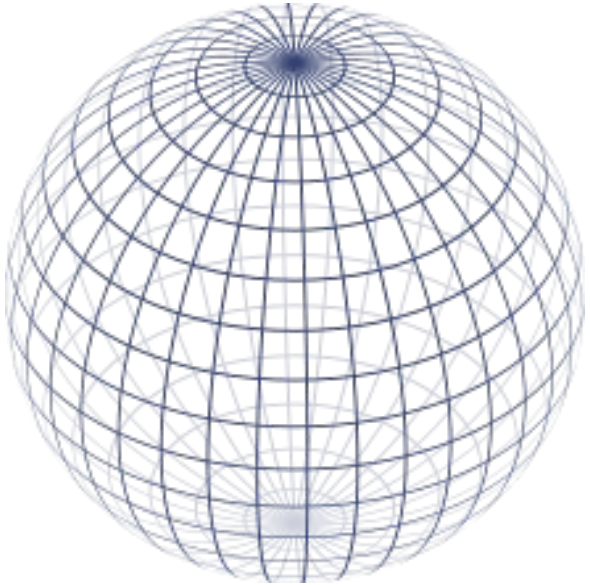
PostGIS Core - OSS

GEOMETRY



R_3

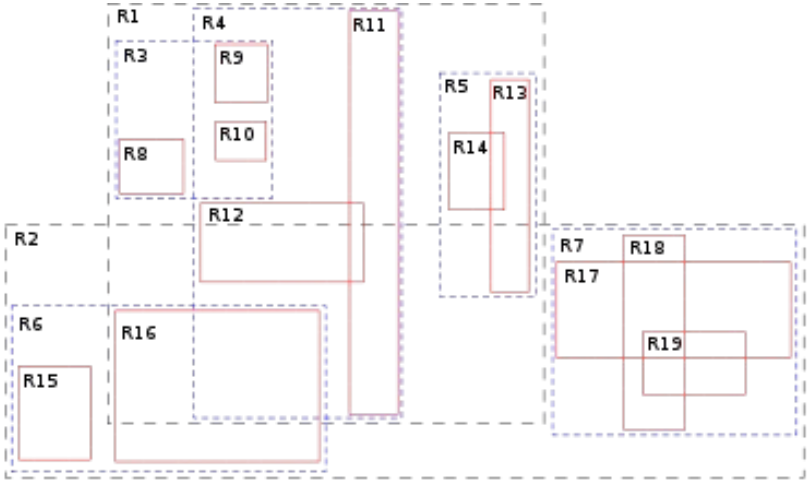
GEOGRAPHY

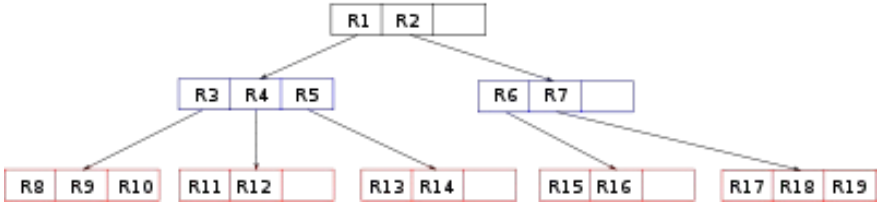


S_2

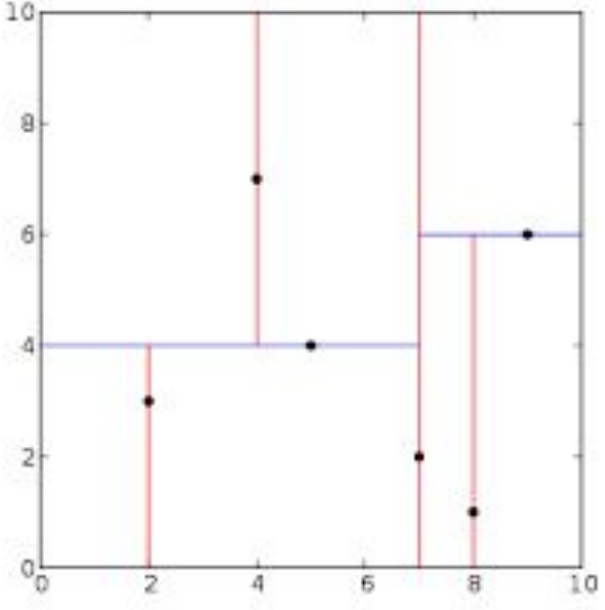
PostGIS Core - Indexing

R-TREE

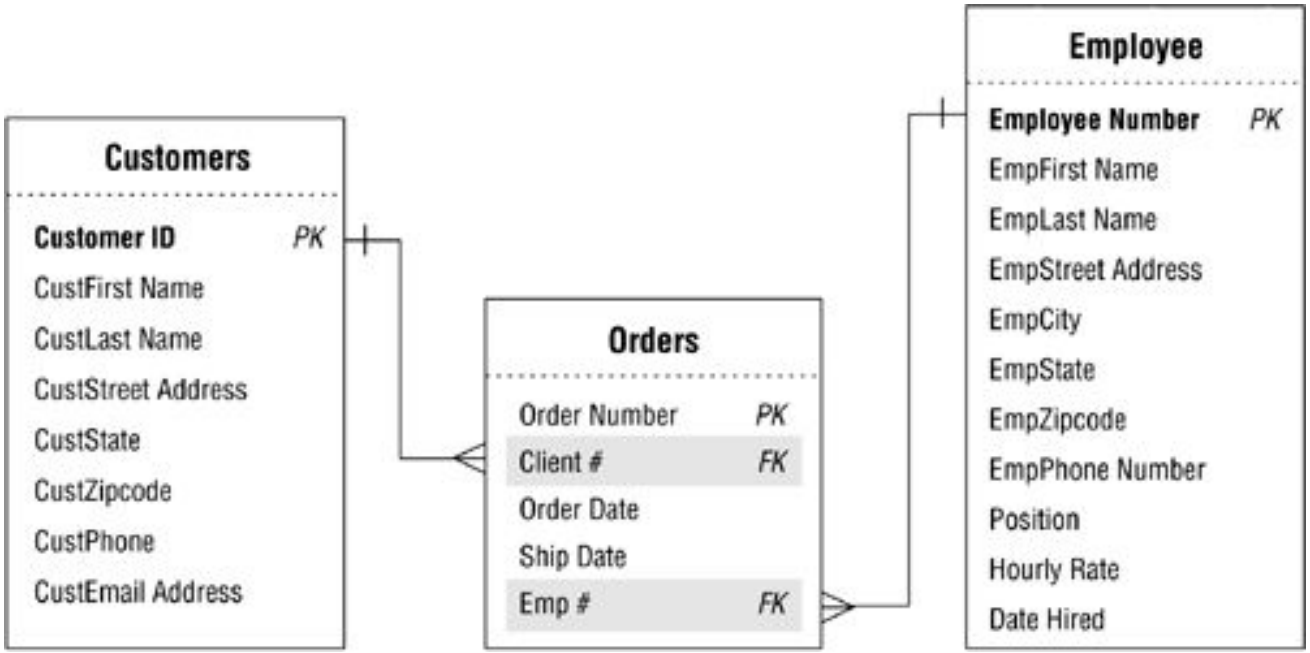


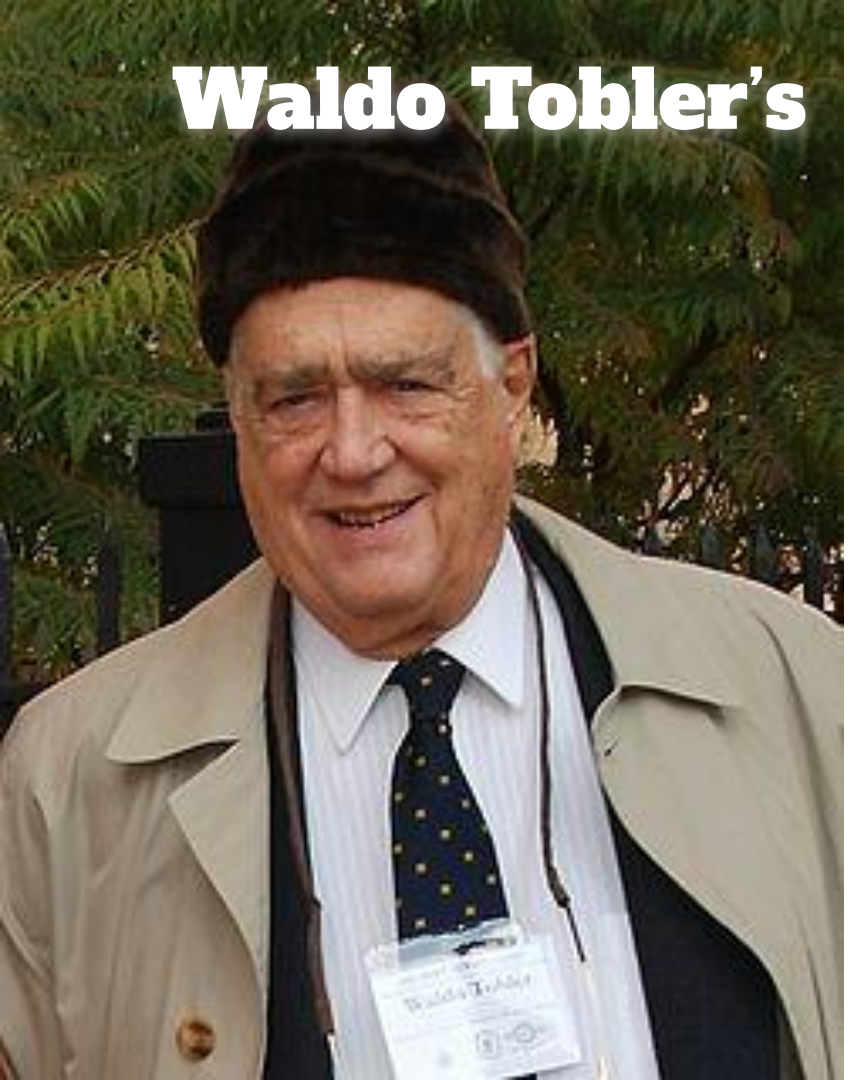


KD-TREE



GIS without the GIS


Customers
Customer ID PK
CustFirst Name
CustLast Name
CustStreet Address
CustState
CustZipcode
CustPhone
CustEmail Address
Orders
Order Number PK
Client # FK
Order Date
Ship Date
Emp # FK
Employee
Employee Number PK
EmpFirst Name
EmpLast Name
EmpStreet Address
EmpCity
EmpState
EmpZipcode
EmpPhone Number
Position
Hourly Rate
Date Hired



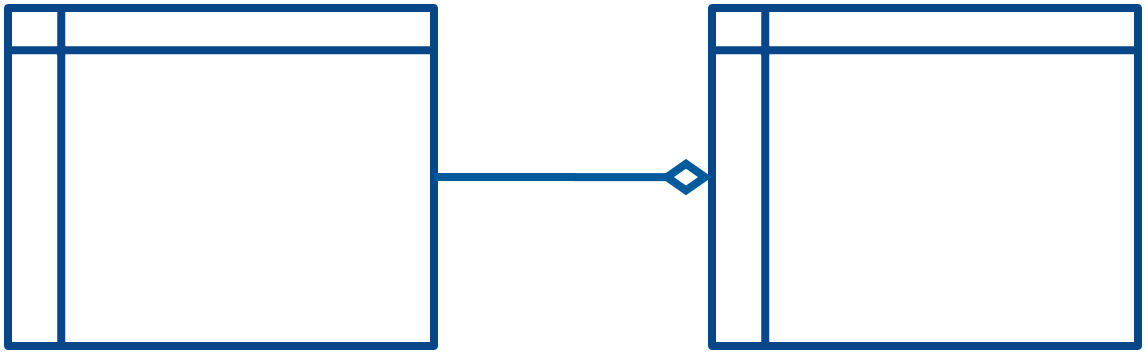
First Law of Geography

"Everything is related to everything else, but **near** things are **more related** than **distant** things."

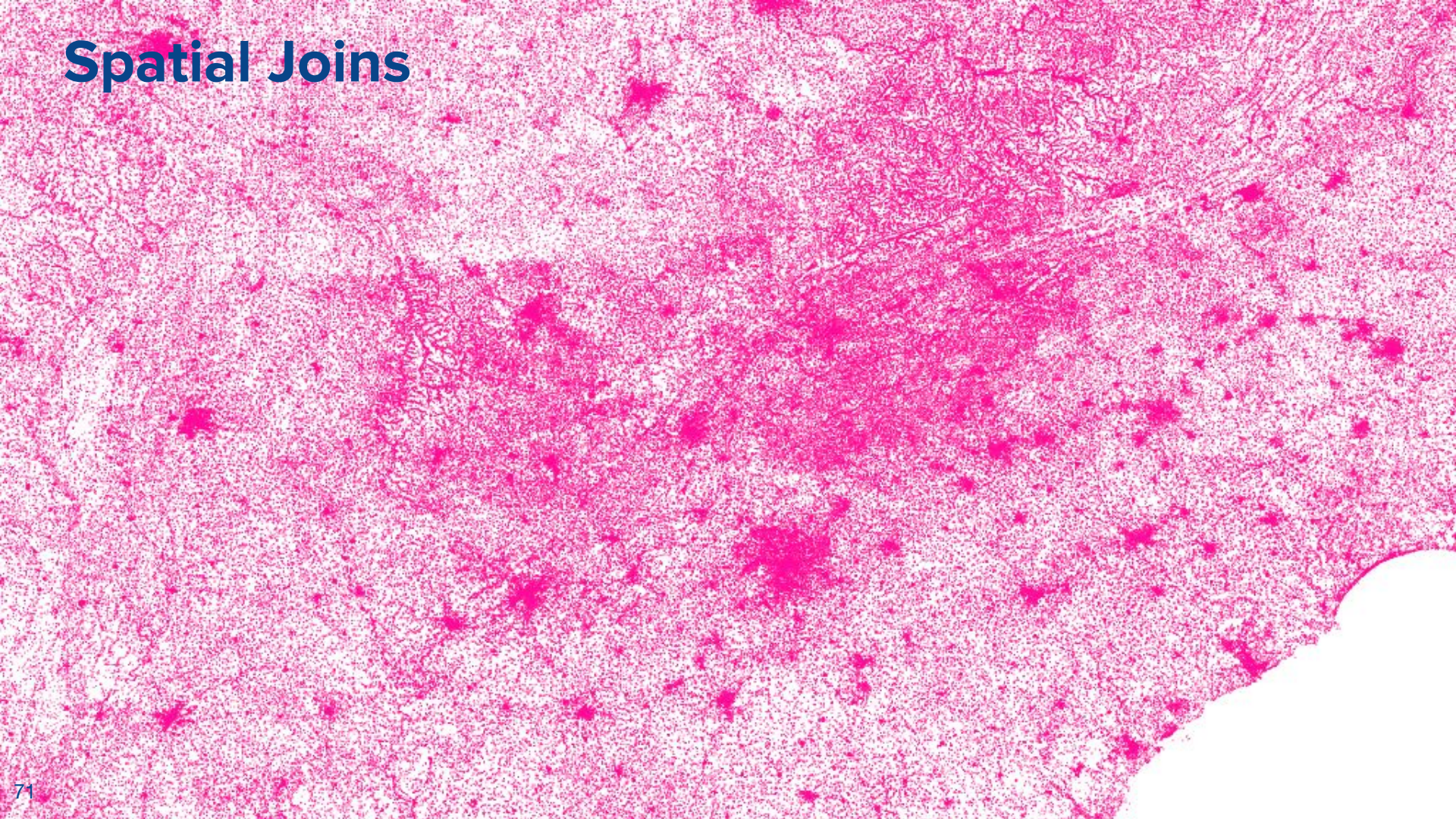
Spatial is the ~~foreign primary~~ universal key

Spatial Joins

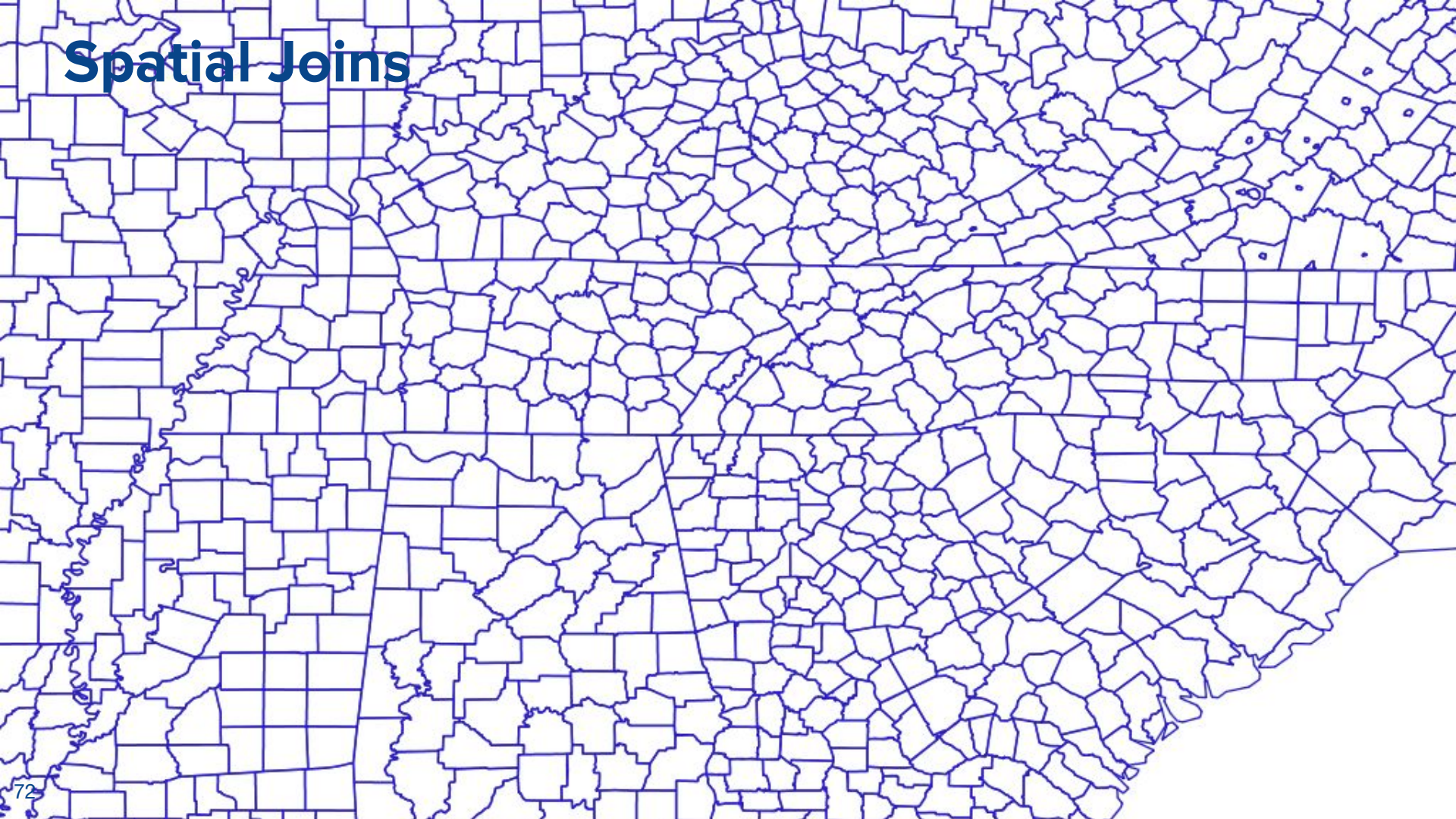
```
ST_Intersects(geom, geom)
ST_Contains(geom, geom)
ST_DWithin(geom, geom, radius)
```



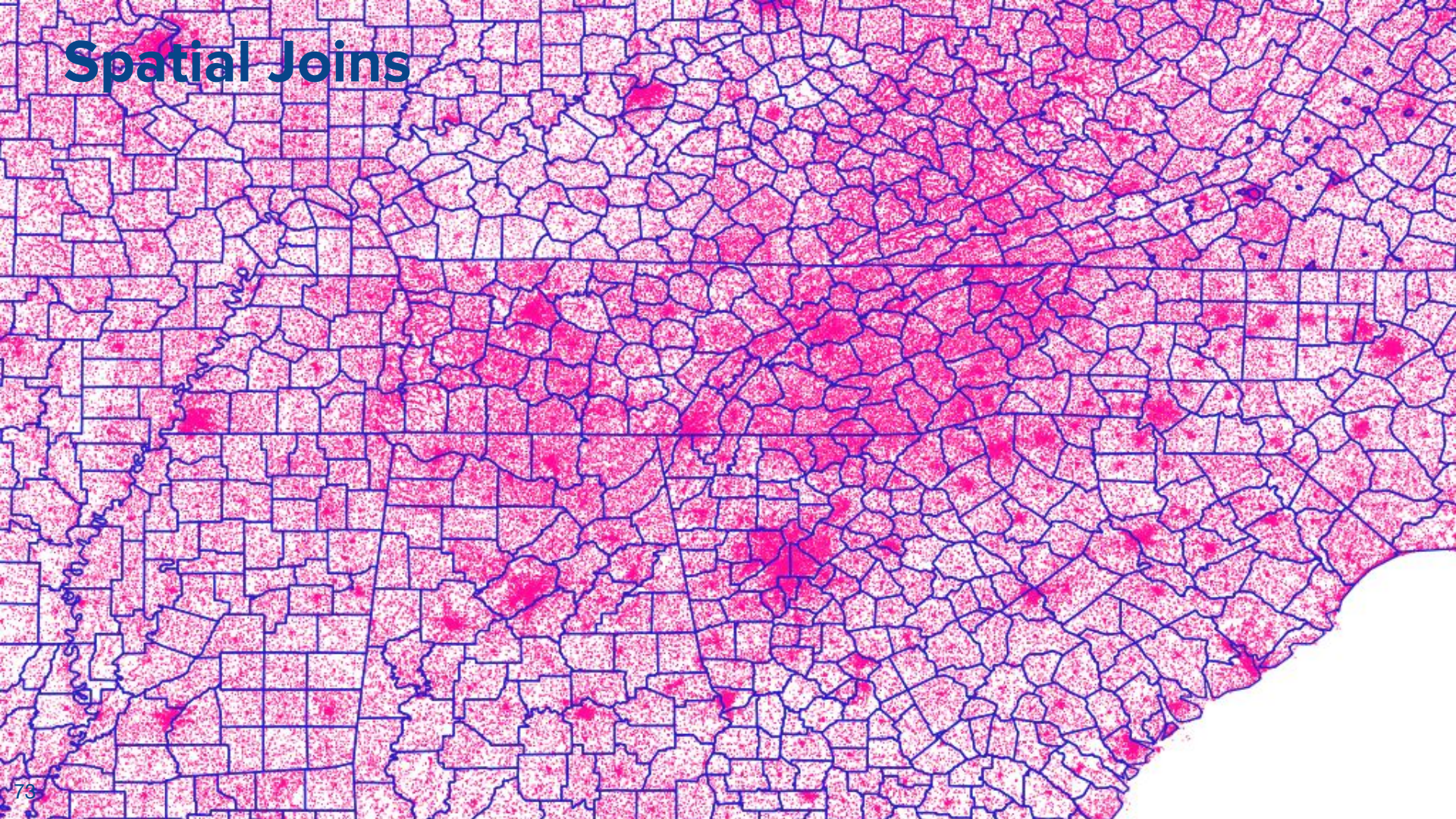
Spatial Joins



Spatial Joins



Spatial Joins



Spatial Joins

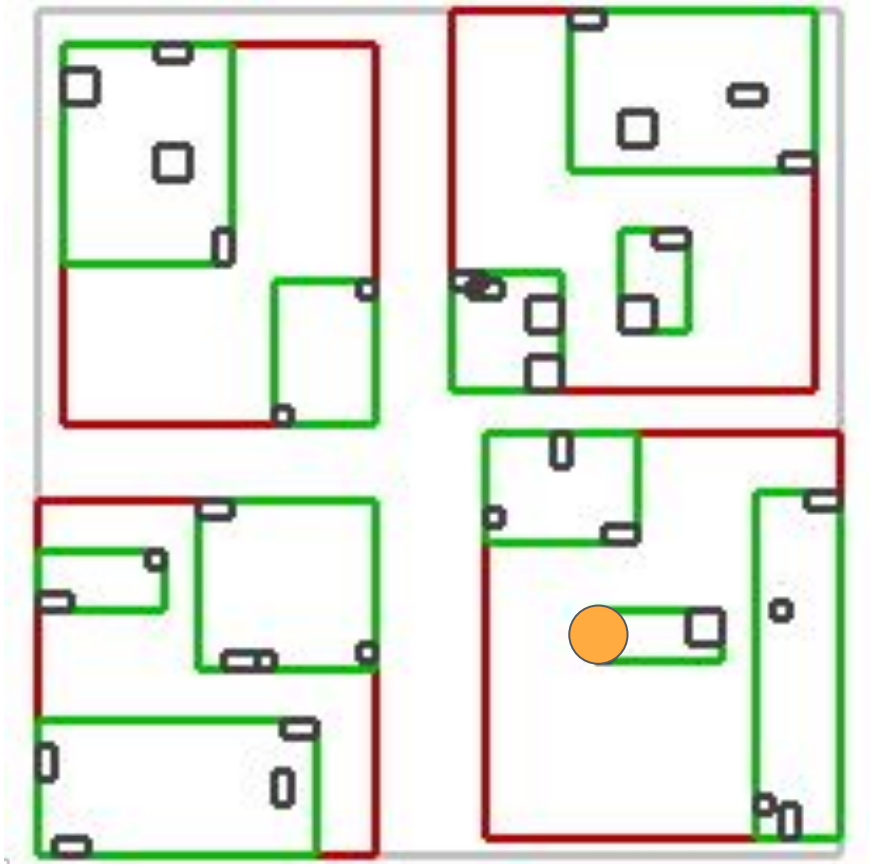
```
SELECT
   census.*, customers.*
FROM census
JOIN customers
ON ST_Contains(
     census.geom,
     customers.geom );
```


find the
nearest

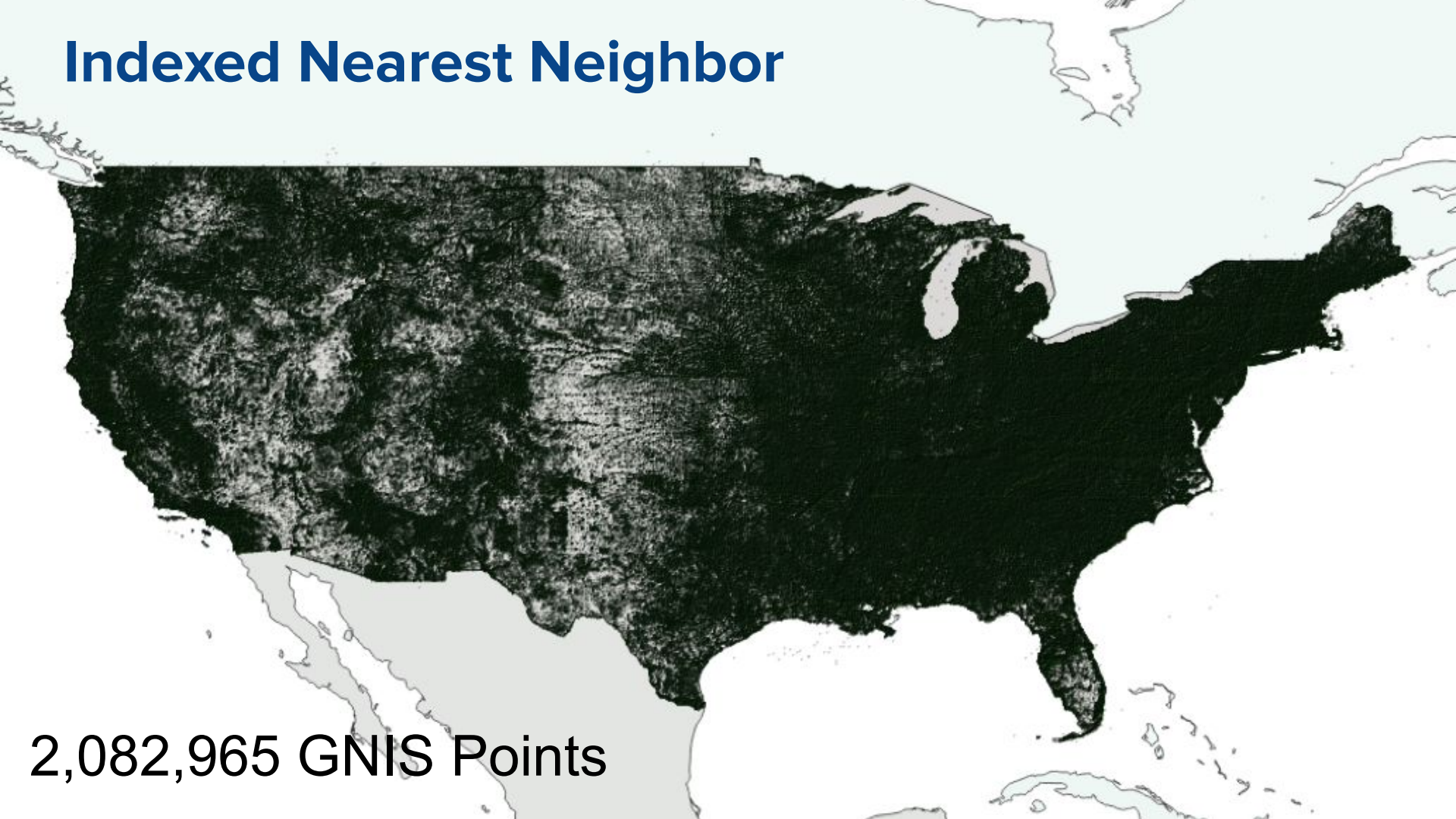
Indexed Nearest Neighbor

```
SELECT *
FROM customers
ORDER BY
   geom <->
   ST_MakePoint(-124, 42)
LIMIT 1
```

Indexed Nearest Neighbor



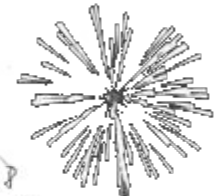
Indexed Nearest Neighbor



2,082,965 GNIS Points

Indexed Nearest Neighbor

```
SELECT id, name, state, kind
FROM geonames
ORDER BY
   geom <->
   (SELECT geom FROM geonames
    WHERE id = 4781416)
LIMIT 10
```

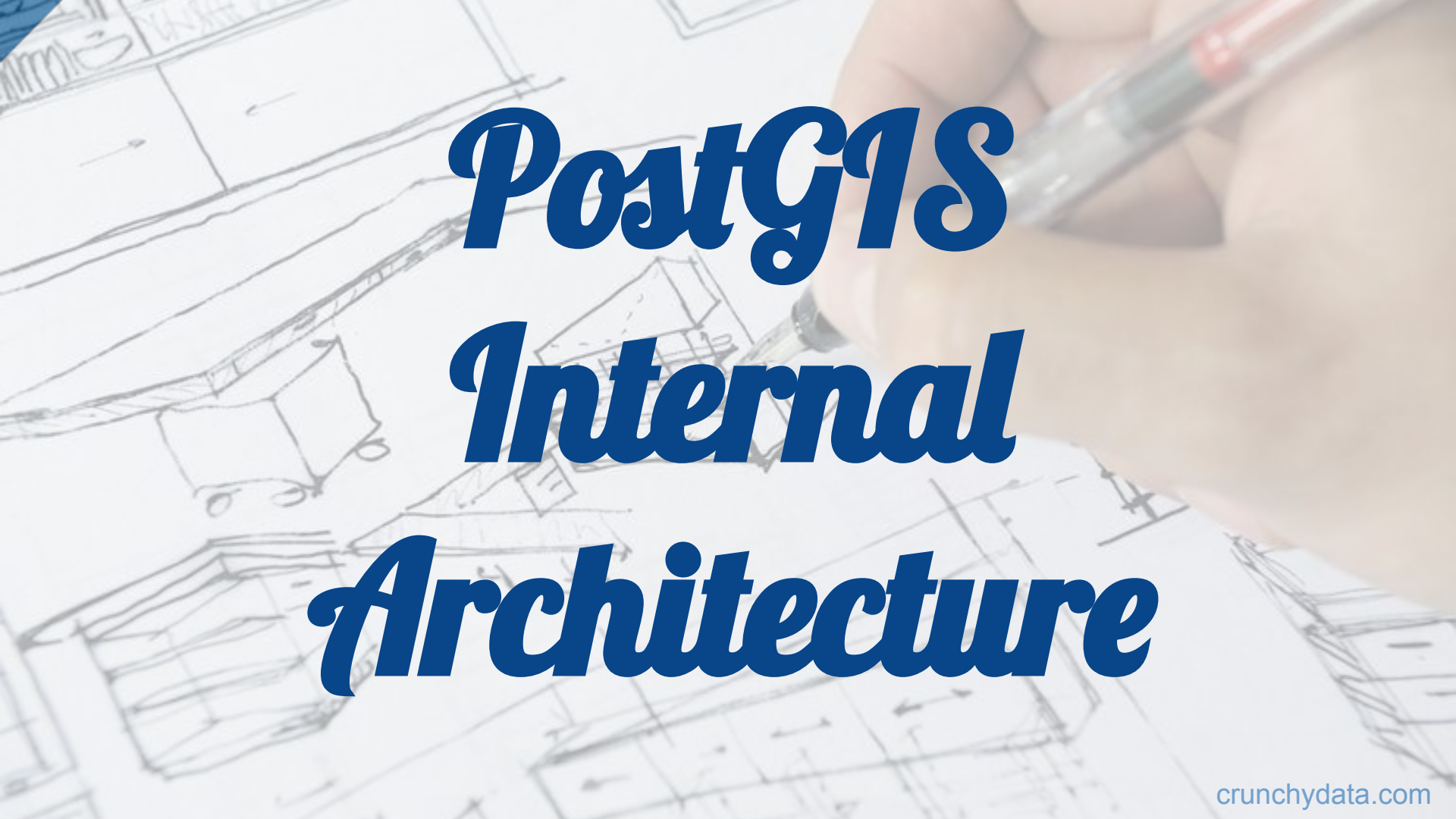


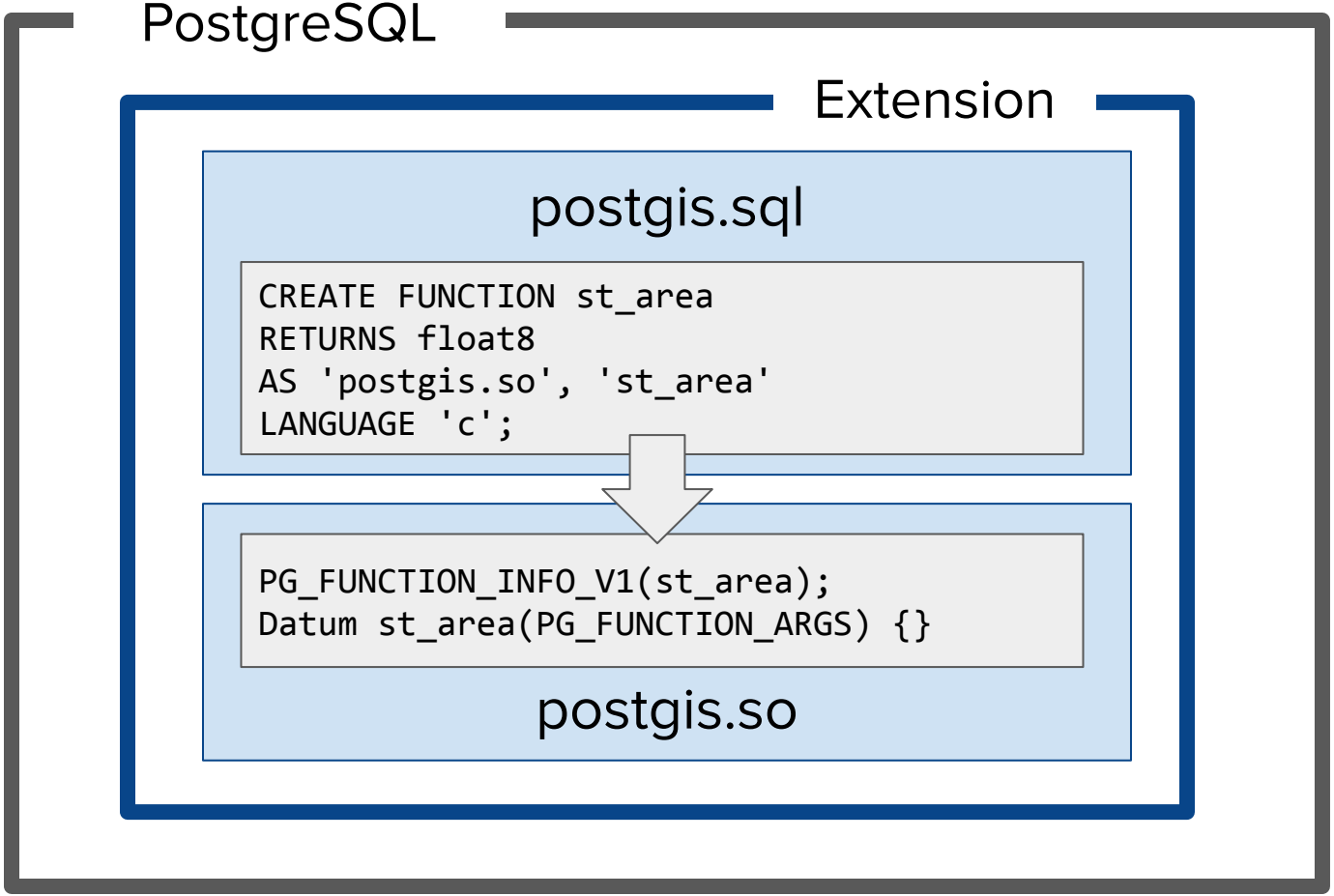
2,082,965 GNIS Points

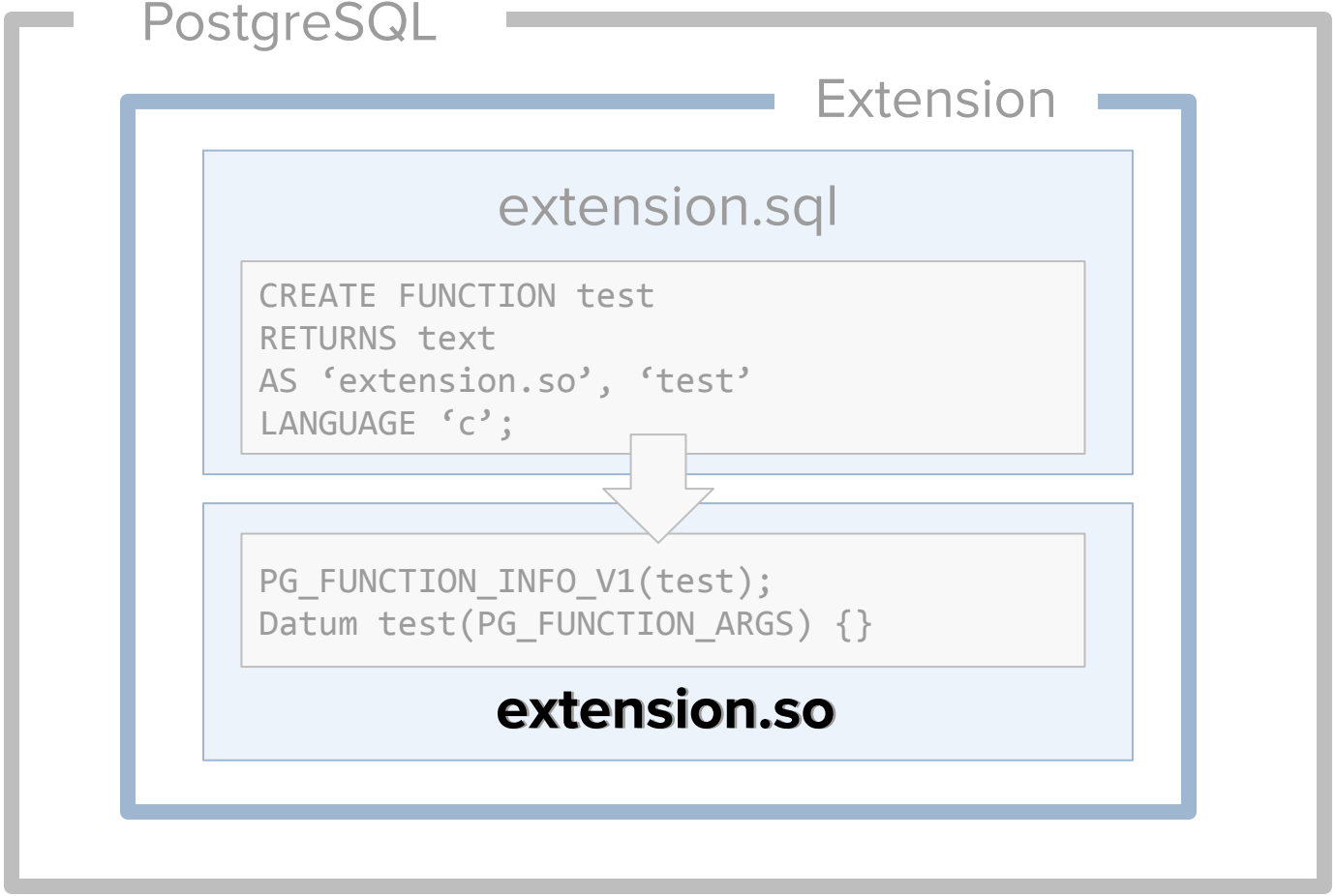
10ms

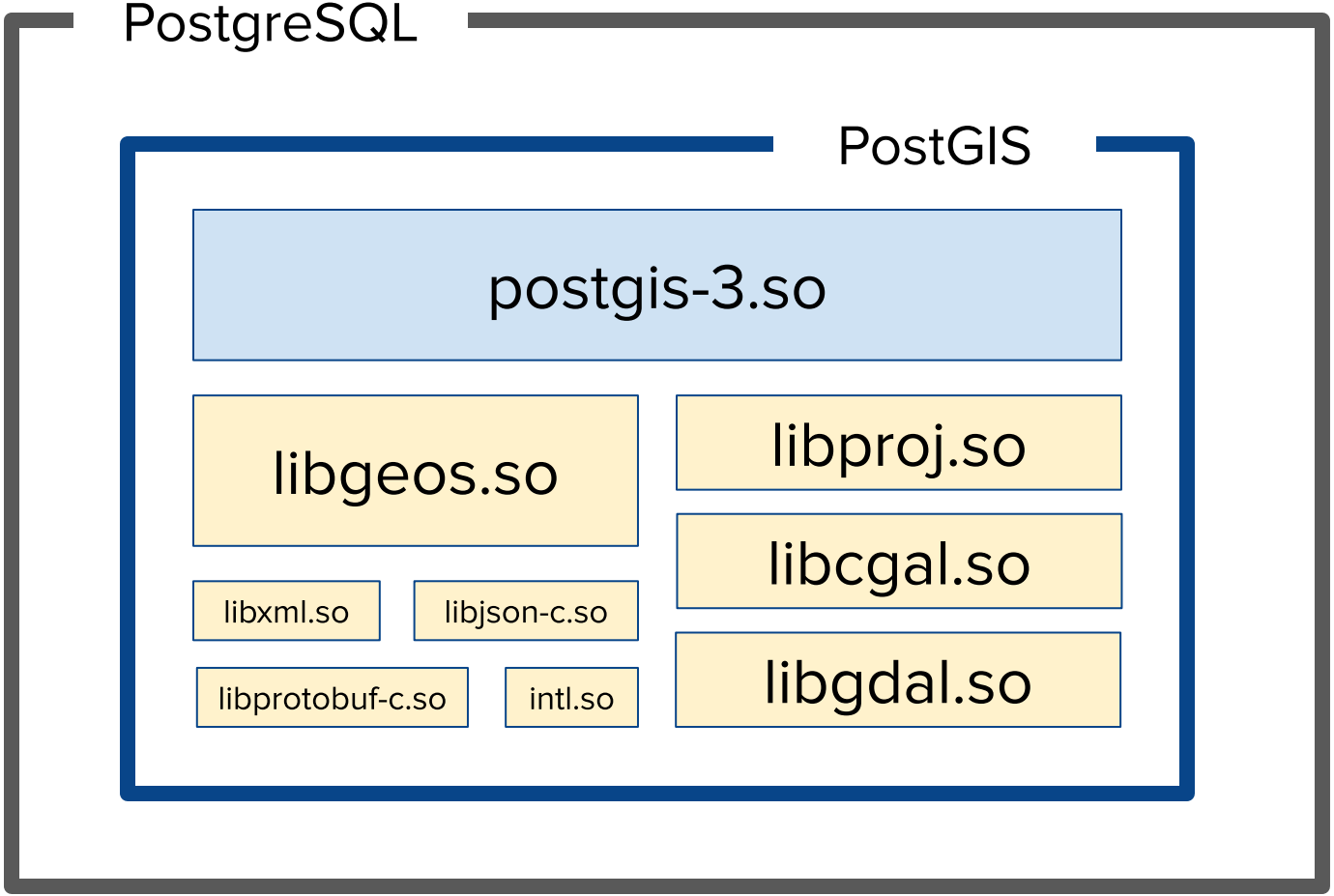
Spatial Joins

```
SELECT c.*, s.store_id
  ST_Distance(c.geom, s.geom) AS dist
FROM
  customers AS c
CROSS JOIN LATERAL
  (SELECT store_id, geom
   FROM stores
   ORDER BY
     customers.geom <-> stores.geom
   LIMIT 1) AS s
```


PostGIS
Internal
Architecture


PostgreSQL
Extension
postgis.sql
CREATE FUNCTION st_area
RETURNS float8
AS 'postgis.so', 'st_area'
LANGUAGE 'c';
PG_FUNCTION_INFO_V1(st_area);
Datum st_area(PG_FUNCTION_ARGS) {}
postgis.so


PostgreSQL
Extension
extension.sql
CREATE FUNCTION test
RETURNS text
AS ‘extension.so’, ‘test’
LANGUAGE ‘c’;
PG_FUNCTION_INFO_V1(test);
Datum test(PG_FUNCTION_ARGS) {}
extension.so


PostgreSQL
PostGIS
postgis-3.so
libgeos.so
libproj.so
libcgal.so
libxml.so
libjson-c.so
libgdal.so
libprotobuf-c.so
intl.so

Longitude Latitude

(θ, Φ) Forward / Inverse (x, y)

(θ_1, Φ_1) Pipeline (θ_2, Φ_2)

Coordinate Systems

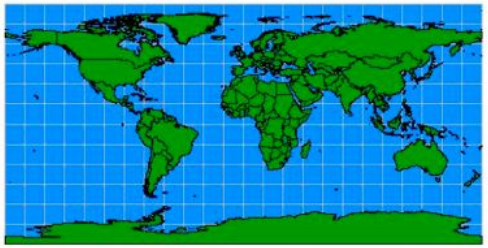
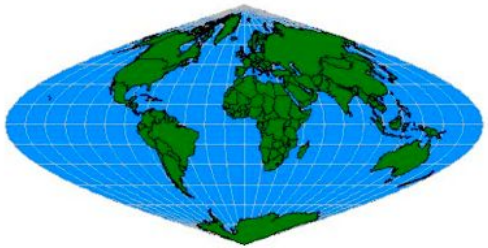
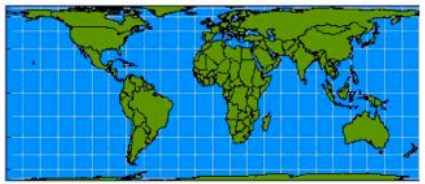


Plate Carree Projection



Sinusoidal Projection

```
ST_Transform(geometry, srid)
```



Behrmann Projection

Albers Equal Area Conic Projection

Proj 6 (proj.org)

- Time dependent datums
- Vertical transformations
- Built in EPSG database
- WKT2, WKT1, URN format support
- Direct datum transformations
    - (no trip through WGS84)

Coordinate Systems

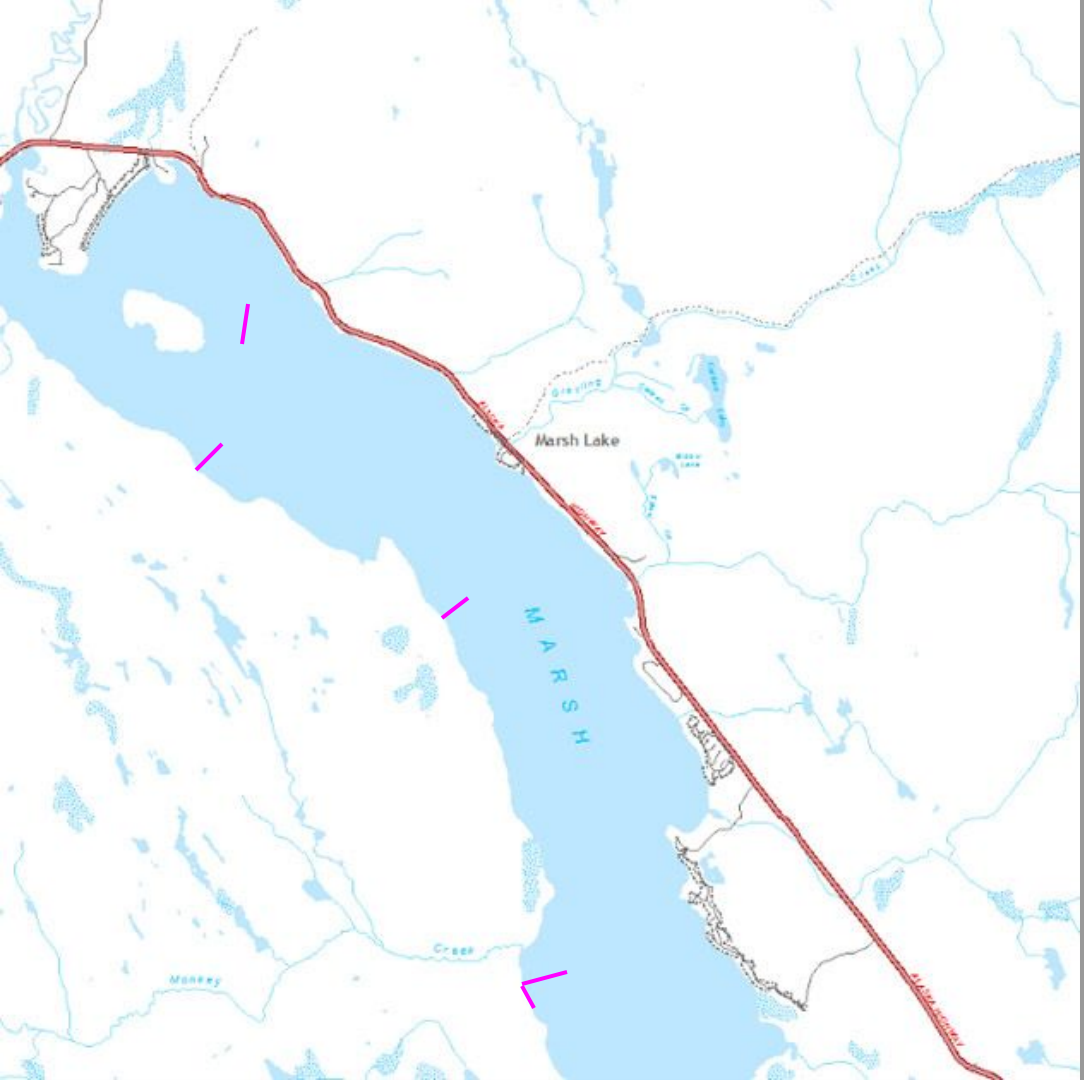
Albers Equal Area, **Azimuthal Equidistant**, Airy, Aitoff, Modified Stererographics of Alaska, Apian Globular I, August Epicycloidal, Bacon Globular, Bipolar conic of western hemisphere, Boggs Eumorphic, Bonne (Werner lat_1=90), Cal Coop Ocean Fish Invest Lines/Stations, Cassini (Cassini-Soldner), Central Cylindrical, Central Conic, Equal Area Cylindrical, Chamberlin Trimetric, Collignon, Craster Parabolic (Putnins P4), Denoyer Semi-Elliptical, Eckert I, Eckert II, Eckert III, Eckert IV, Eckert V, Eckert VI, **Equidistant Cylindrical** (Plate Carrée), Equidistant Conic, Equal Earth, Euler, Extended Transverse Mercator, Fahey, Foucaut, Foucaut Sinusoidal, Gall (Gall Stereographic), Geostationary Satellite View, Ginsburg VIII (TsNIIGAiK), General Sinusoidal Series, Gnomonic, Goode Homolosine, Mod. Stererographics of 48 U.S., Mod. Stererographics of 50 U.S., Hammer & Eckert-Greifendorff, Hatano Asymmetrical Equal Area, HEALPix, rHEALPix, Interrupted Goode Homolosine, International Map of the World Polyconic, Icosahedral Snyder Equal Area, Kavraisky V, Kavraisky VII, Krovak, Laborde, **Lambert Azimuthal Equal Area**, Lagrange, Larrivee, Laskowski, **Lambert Conformal Conic,** Lambert Conformal Conic Alternative, **Lambert Equal Area Conic**, Lee Oblated Stereographic, Loximuthal, Space oblique for LANDSAT, McBryde-Thomas Flat-Polar Sine (No. 1), McBryde-Thomas Flat-Pole Sine (No. 2), McBride-Thomas Flat-Polar Parabolic, McBryde-Thomas Flat-Polar Quartic, McBryde-Thomas Flat-Polar Sinusoidal, **Mercator**, Miller Oblated Stereographic, Miller Cylindrical, Mollweide, Murdoch I, Murdoch II, Murdoch III, Natural Earth, Nell, Nell-Hammer, Nicolosi Globular, Near-sided perspective, New Zealand Map Grid, General Oblique Transformation, Oblique Cylindrical Equal Area, Oblated Equal Area, **Oblique Mercator**, Ortelius Oval, Orthographic, Perspective Conic, Polyconic (American), Putnins P1, Putnins P2, Putnins P3, Putnins P3', Putnins P4', Putnins P5, Putnins P5', Putnins P6, Putnins P6', Quartic Authalic, Quadrilateralized Spherical Cube, Robinson, Roussilhe Stereographic, Rectangular Polyconic, Sinusoidal (Sanson-Flamsteed), Swiss Oblique Mercator, Stereographic, Oblique Stereographic Alternative, Gauss-Schreiber Transverse Mercator (aka Gauss-Laborde Reunion), Transverse Central Cylindrical, Transverse Cylindrical Equal Area, Tissot, **Transverse Mercator**, Two Point Equidistant, Tilted perspective, Universal Polar Stereographic, Urmaev V, Urmaev Flat-Polar Sinusoidal, **Universal Transverse Mercator** (UTM), van der Grinten (I), van der Grinten II, van der Grinten III, van der Grinten IV, Vitkovsky I, Wagner I (Kavraisky VI), Wagner II, Wagner III, Wagner IV, Wagner V, Wagner VI, Wagner VII, Web Mercator / Pseudo Mercator, Werenskiold I, Winkel I, Winkel II, Winkel Tripel

Geometry Engine Open Source

Exactly characterize relationships between any pair of geometries.

aka “dimensionally extended nine-intersection model” (DE9IM)

	Interior	Boundary	Exterior
Interior	 *dim(...) = 2*	 *dim(...) = 1*	 *dim(...) = 2*
Boundary	 *dim(...) = 1*	 *dim(...) = 0*	 *dim(...) = 1*
Exterior	 *dim(...) = 2*	 *dim(...) = 1*	 *dim(...) = 2*



Spatial QA/QC

- A valid dock has one end on the edge of a lake
- The interior of a dock should be entirely on the interior of the lake

ST_Relate(dock, lake)

‘1FF00F212’

Geometry Engine Open Source

Exactly characterize relationships between any pair of geometries.

aka “dimensionally extended nine-intersection model” (DE9IM)

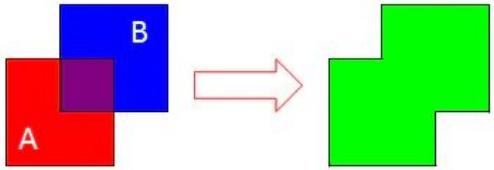
- **ST_Intersects**(A, B)
  - = NOT ST_Disjoint(A, B)
- **ST_Contains**(A, B)
  - = ST_Within(B, A)
- ST_Touches(A, B)
- ST_Overlaps(A, B)
- ST_Covers(A, B)
- ST_Crosses(A, B)

Geometry Engine Open Source

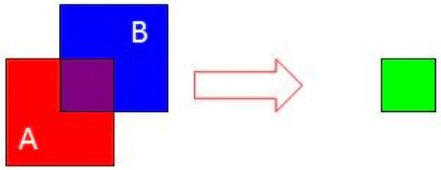
Generate new geometries for set-wise operations.

- Union
- Difference
- SymDifference
- Intersection

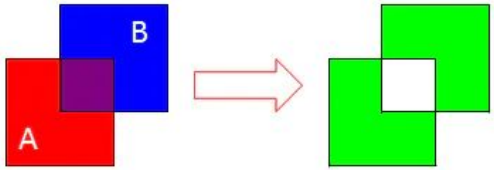
ST_Union(A, B)



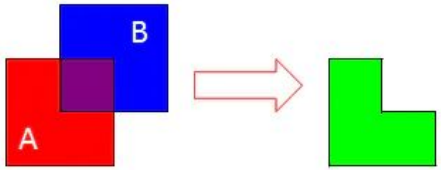
ST_Intersection(A, B)



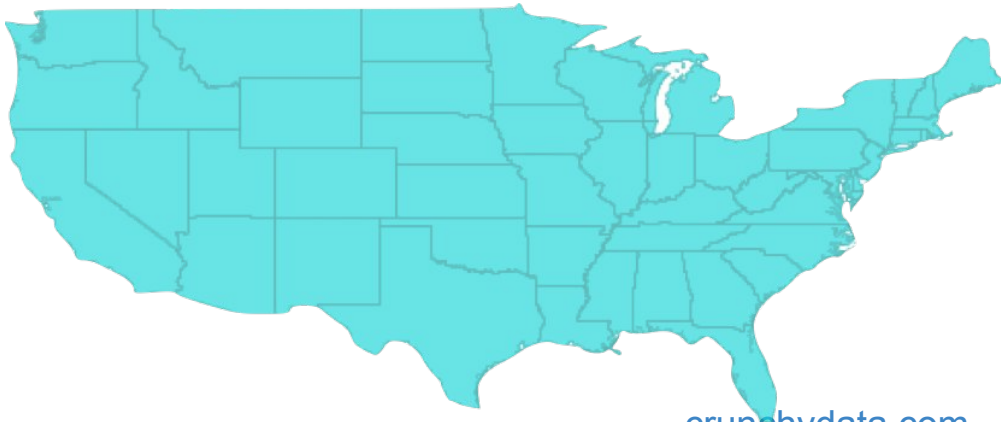
ST_SymDifference(A, B)



ST_Difference(A, B)



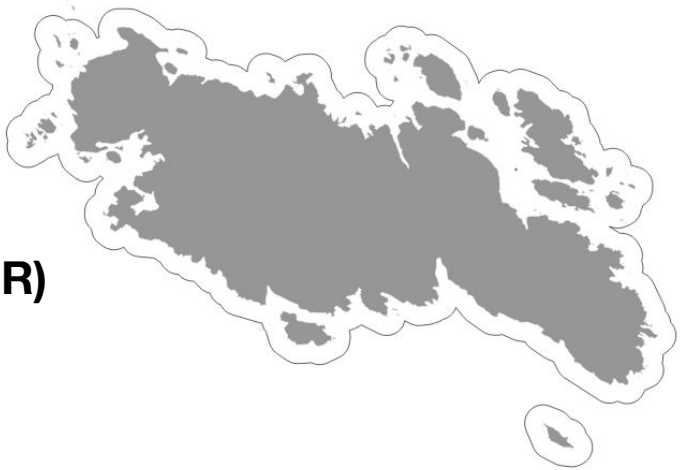
ST_Union(geometry[])



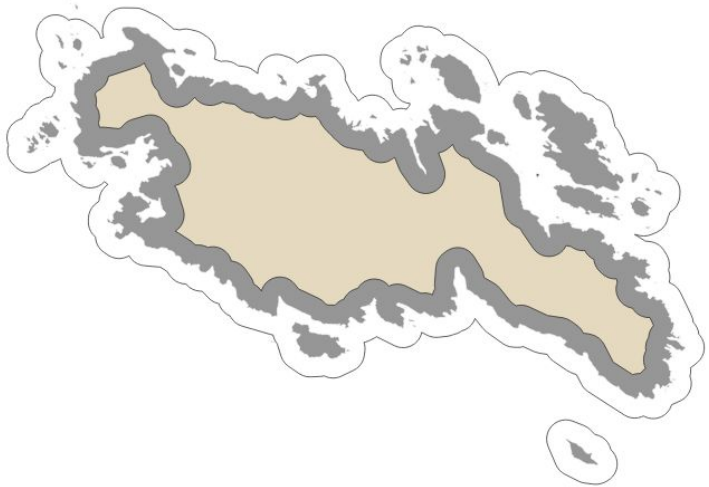
Geometry Engine Open Source

Generate new geometries for fun!

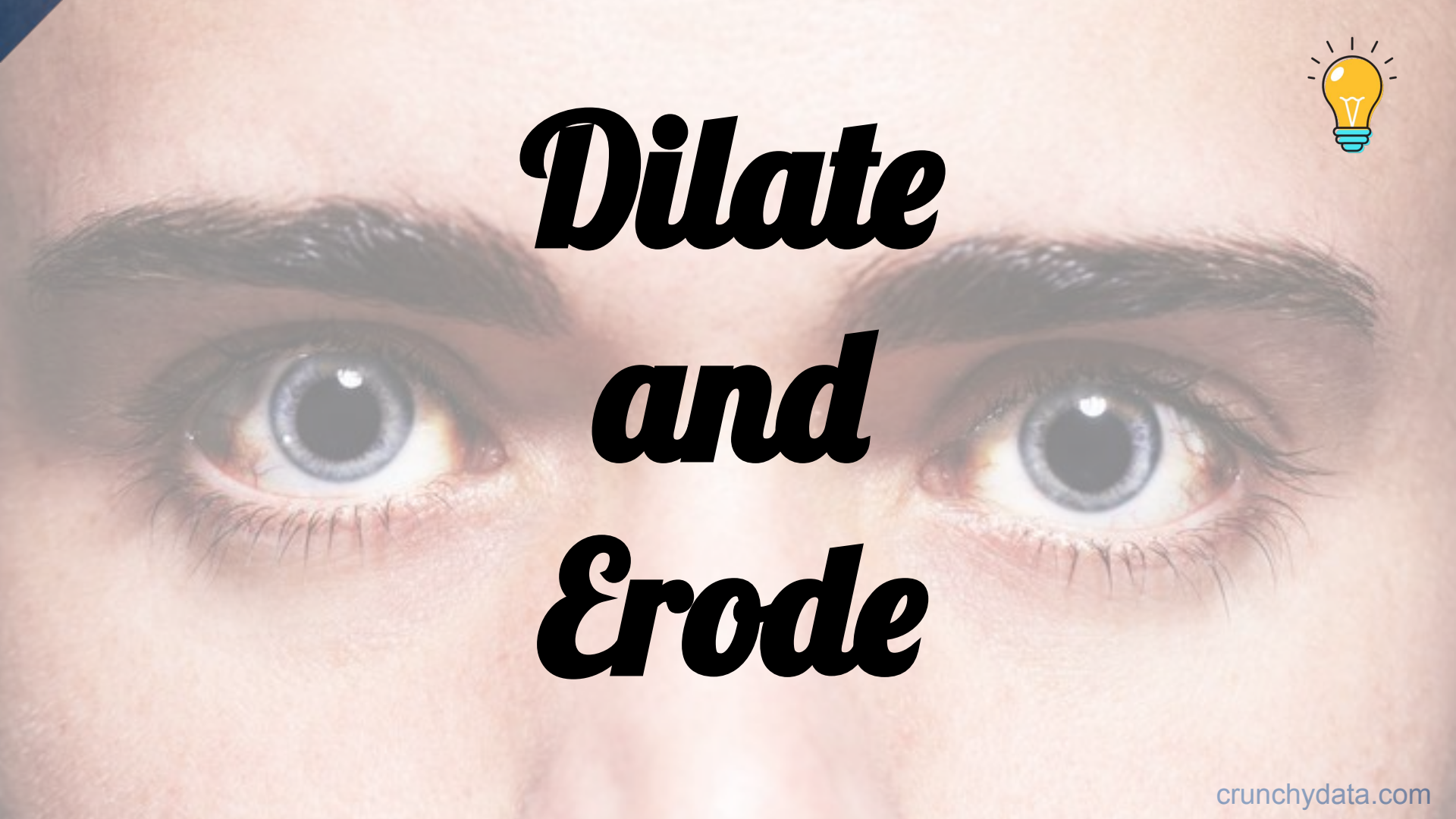
- **Buffer**
- Delauney / Voronoi

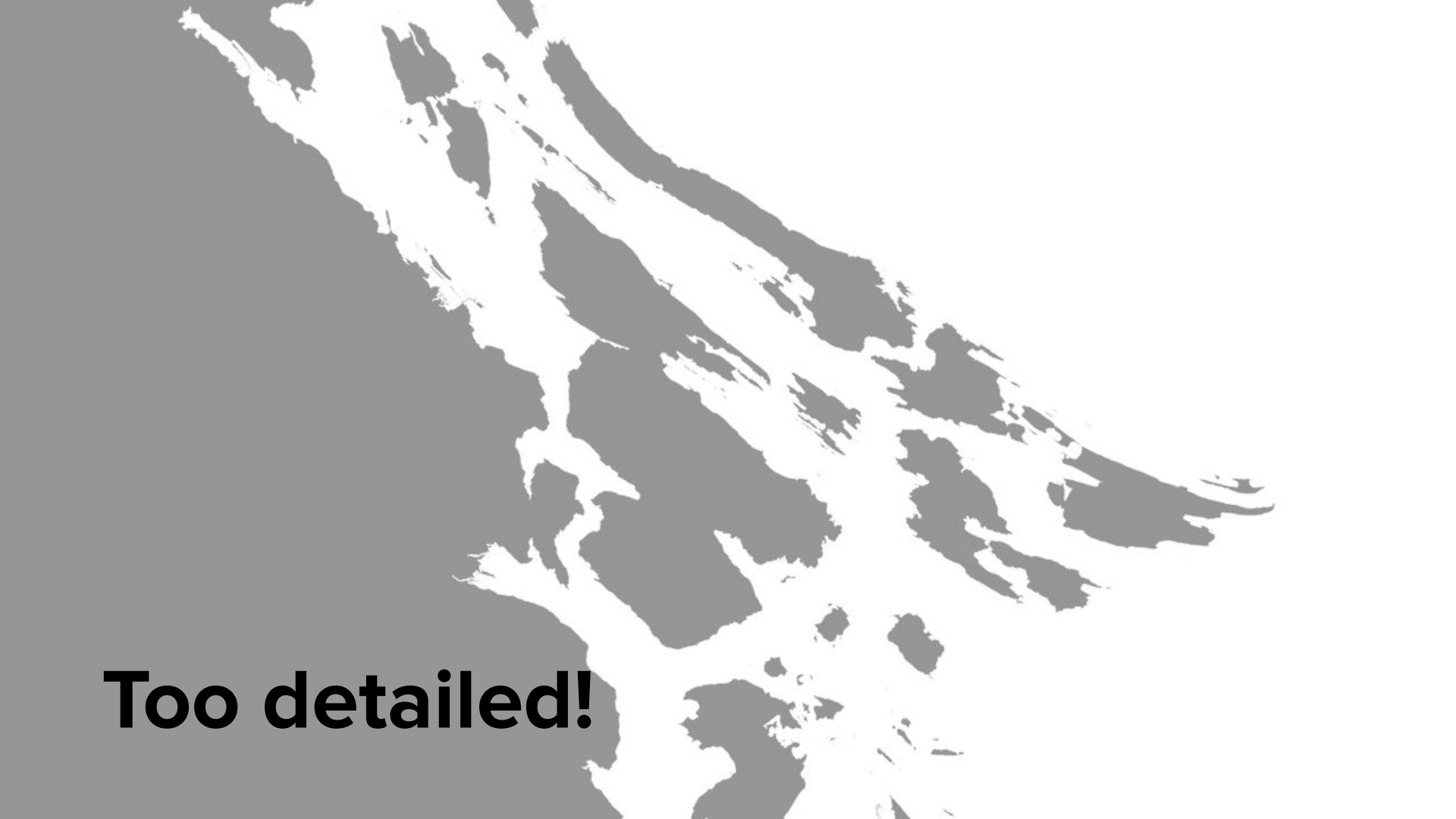


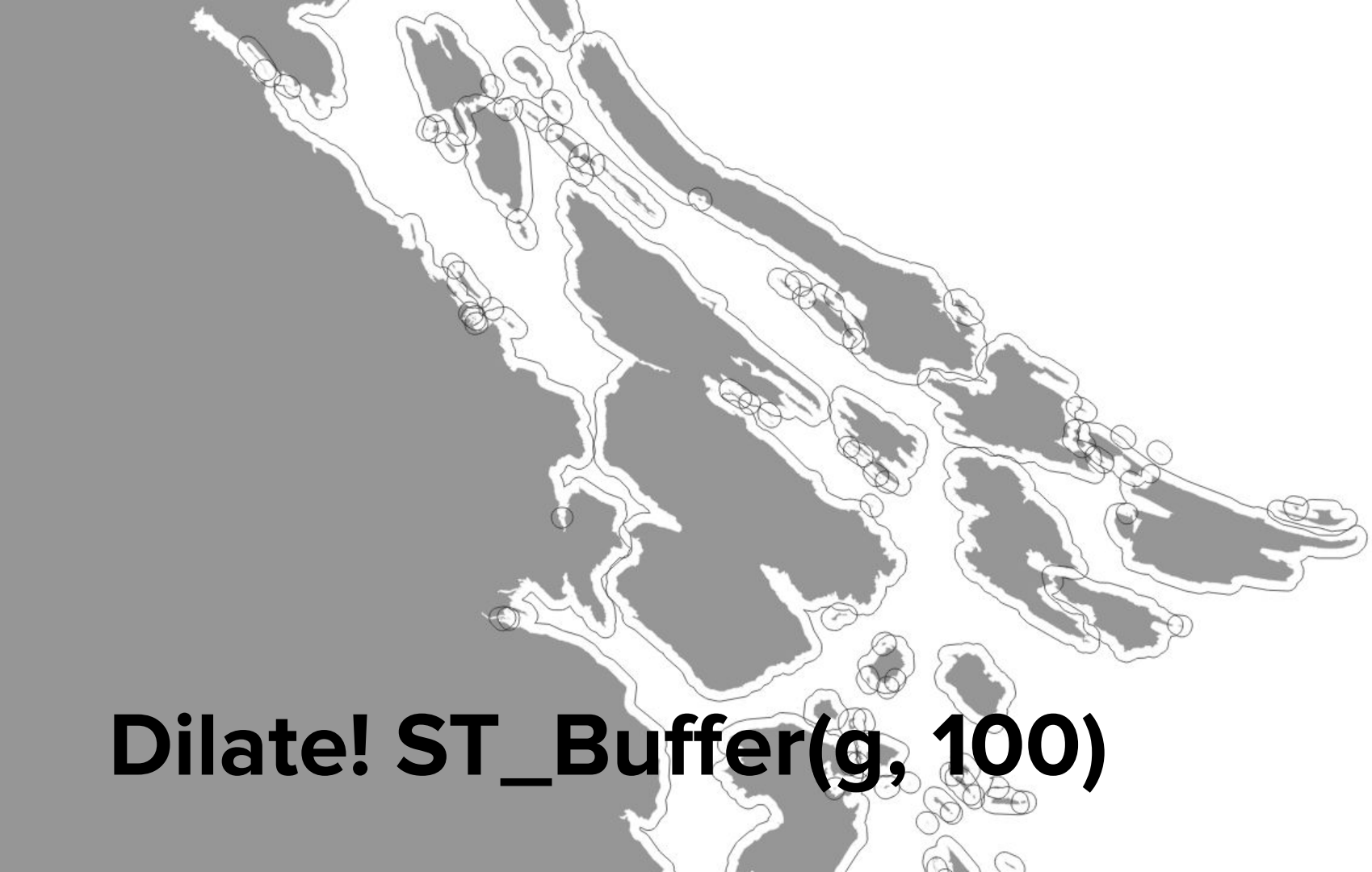
ST_Buffer(A, +R)



ST_Buffer(A, -R)


Dilate
and
Erode


Too detailed!



Dilate! ST_Buffer(g, 100)

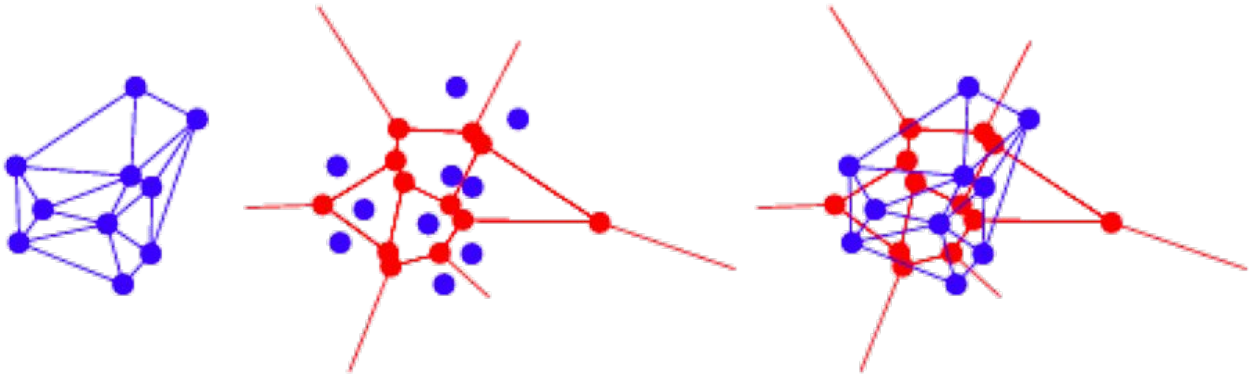
Erode! ST_Buffer(g, -100)

Geometry Engine Open Source

Generate new geometries for fun!

- **Buffer**
- Delauney / Voronoi

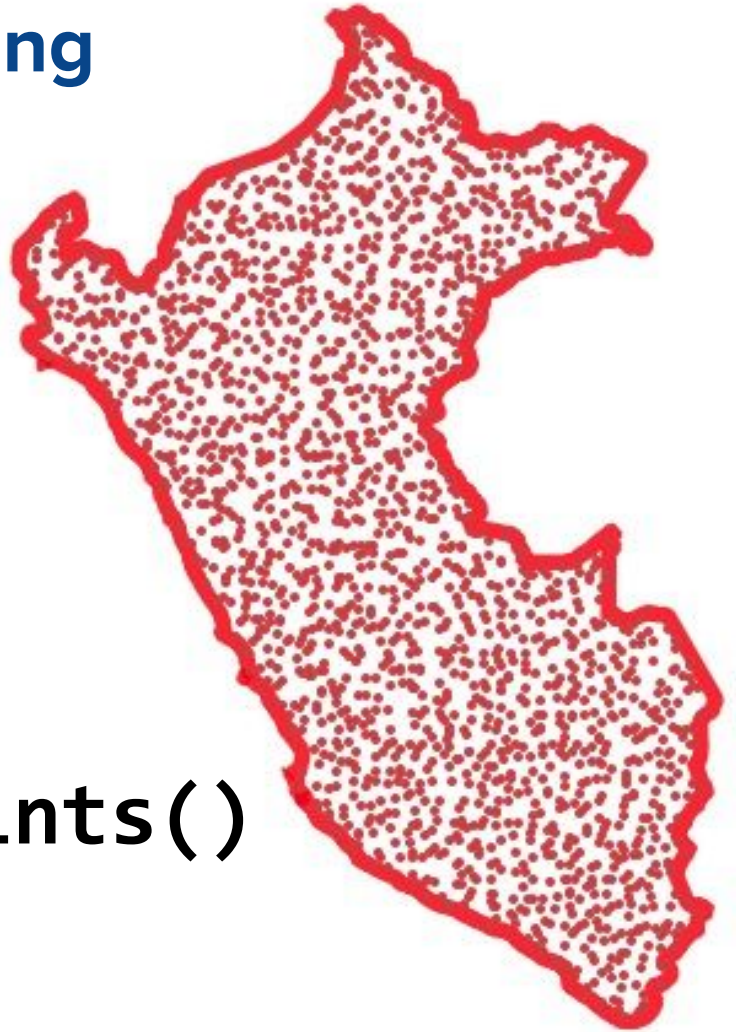
```
ST_VoronoiPolygons()
ST_DelaunayTriangles()
```



Delaunay triangulation *Voronoi diagram* *Delaunay and Voronoi*

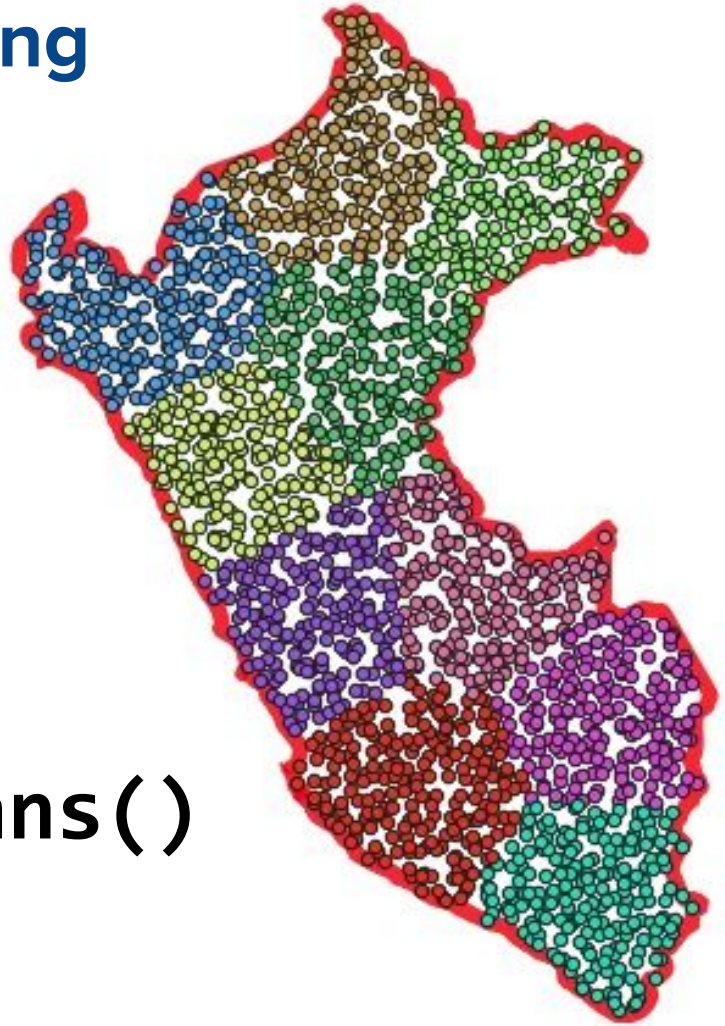
“Divide a polygon into an arbitrary number of similar sub-polygons.”

Geometry Processing



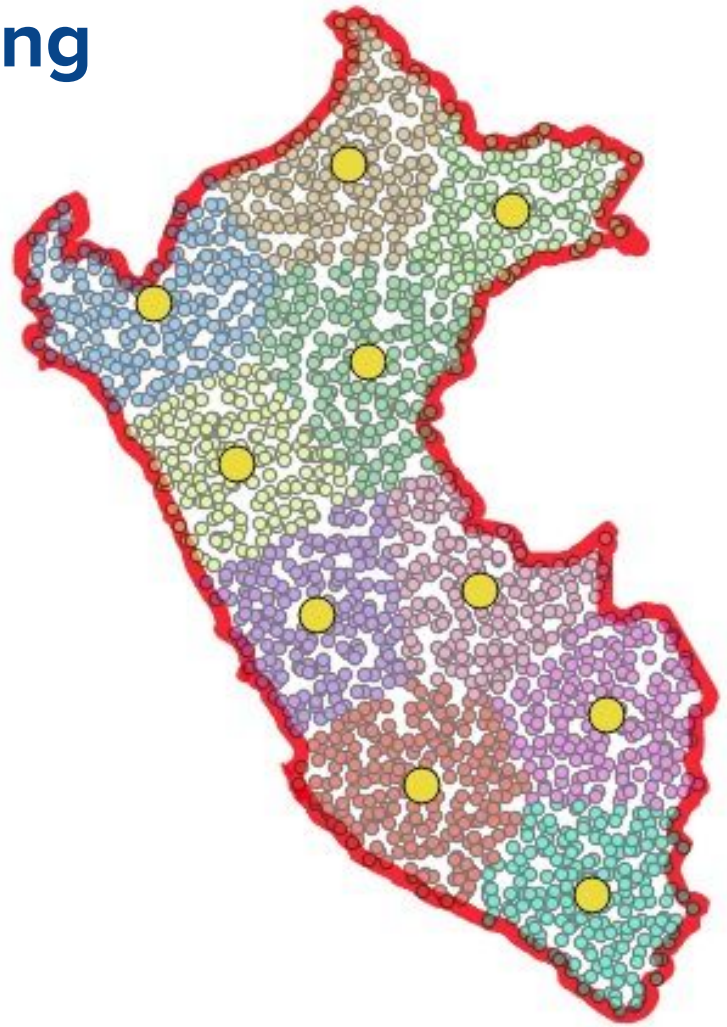
```
ST_GeneratePoints()
```

Geometry Processing



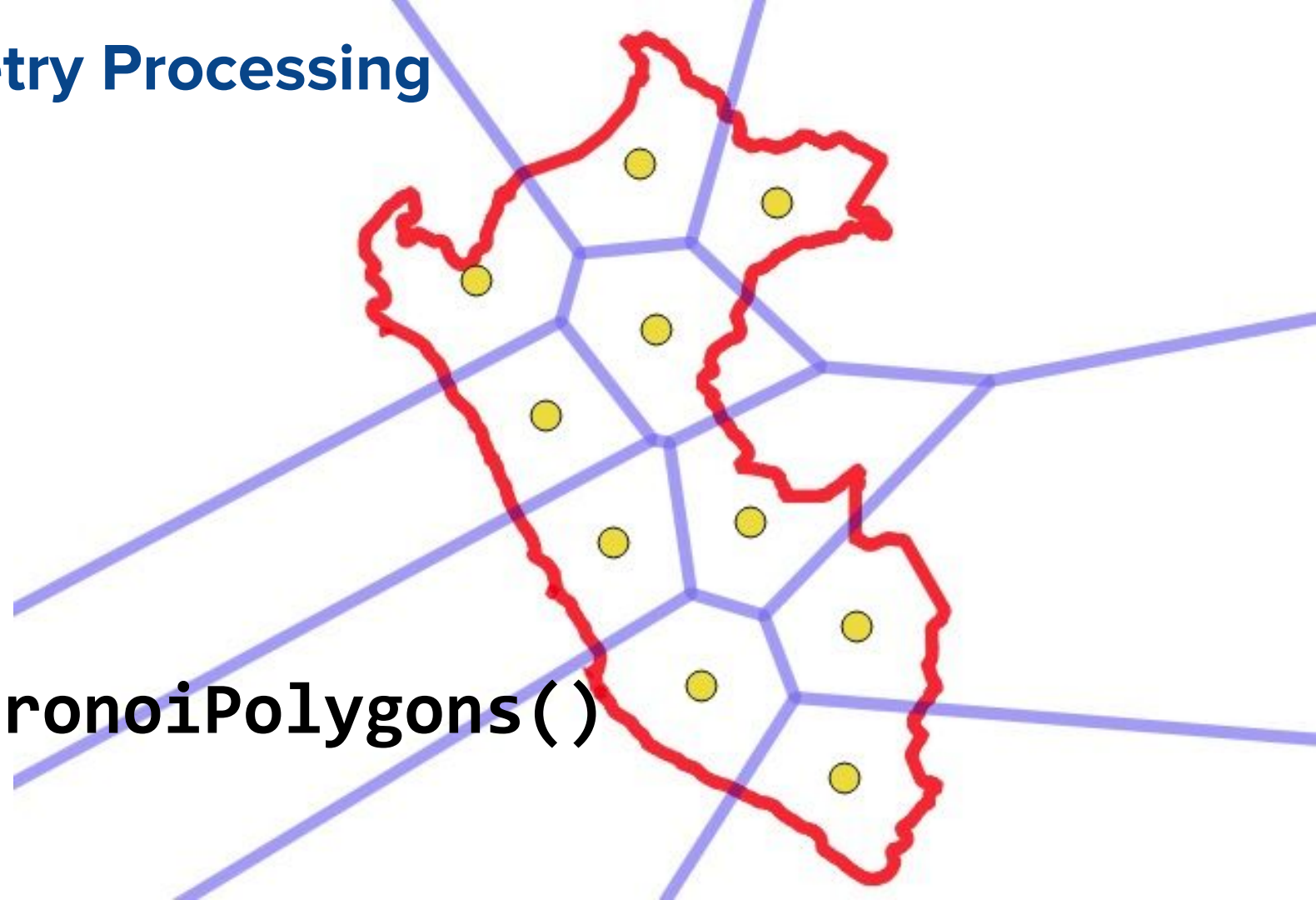
```
ST_ClusterKMeans()
```

Geometry Processing



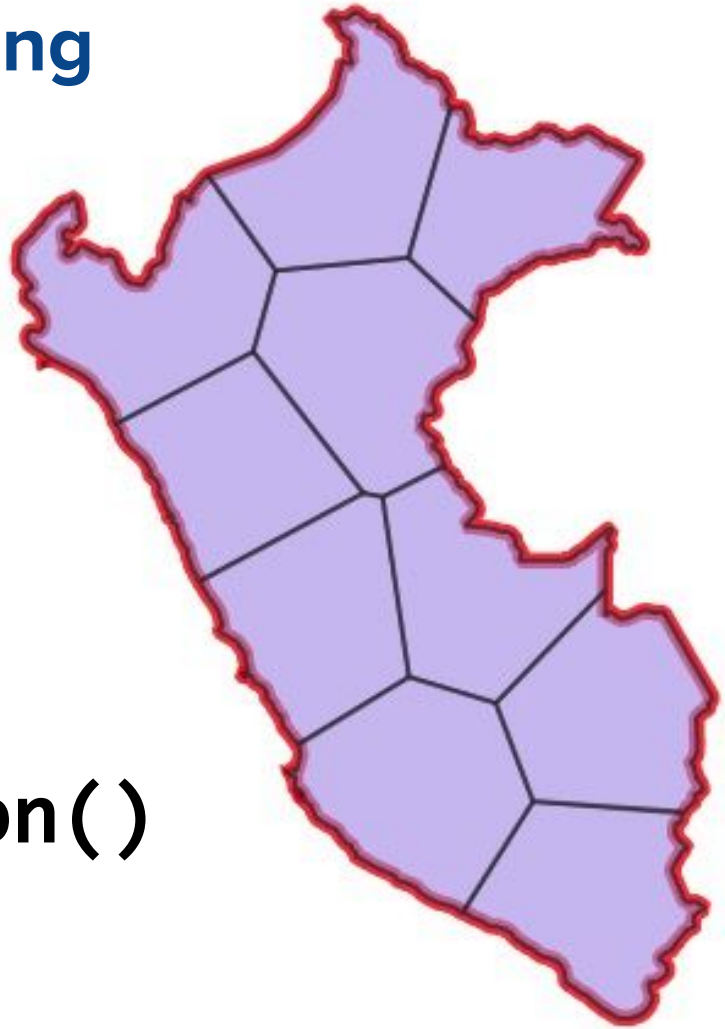
```
ST_Centroid()
```

Geometry Processing



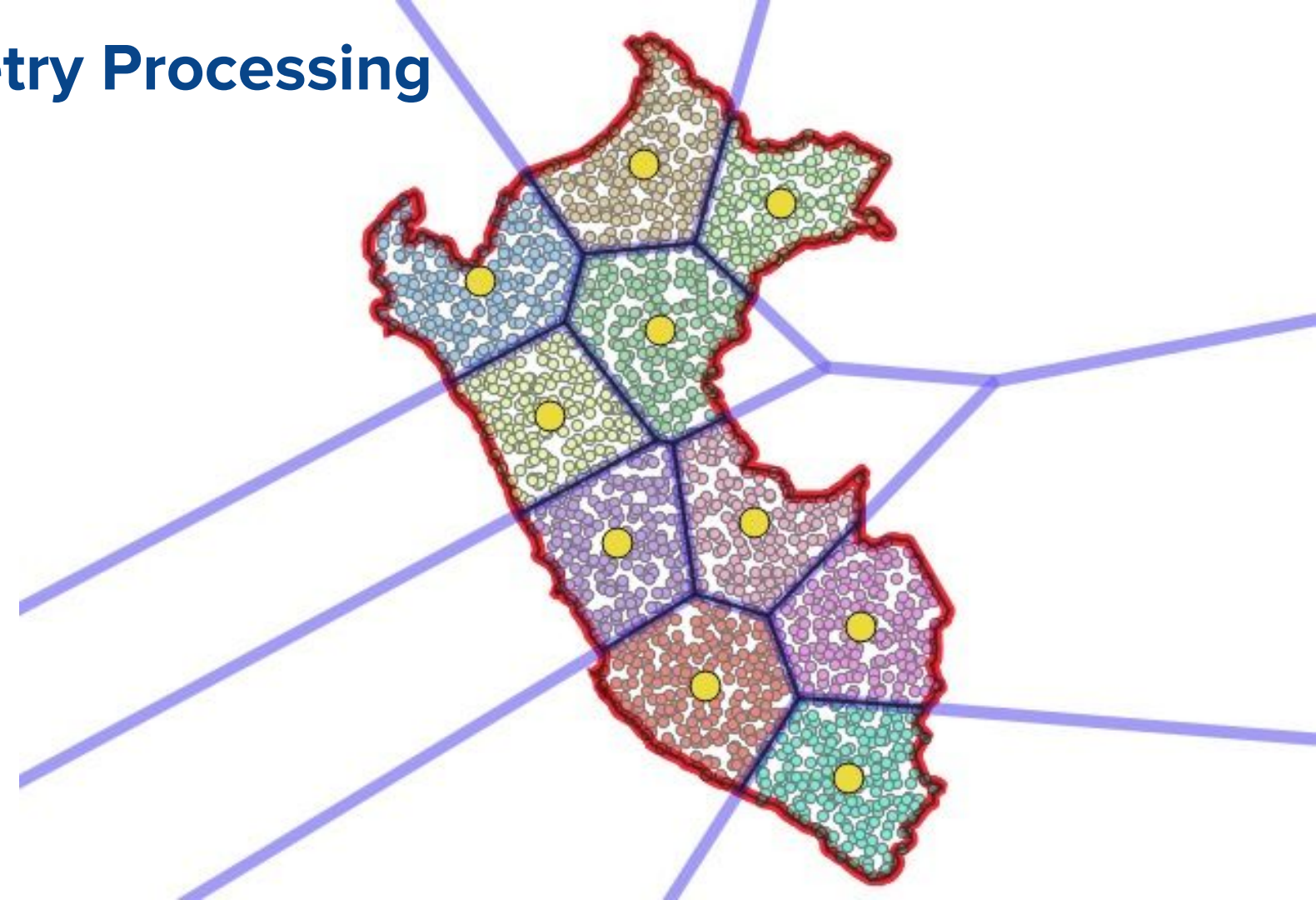
`ST_VoronoiPolygons()`

Geometry Processing

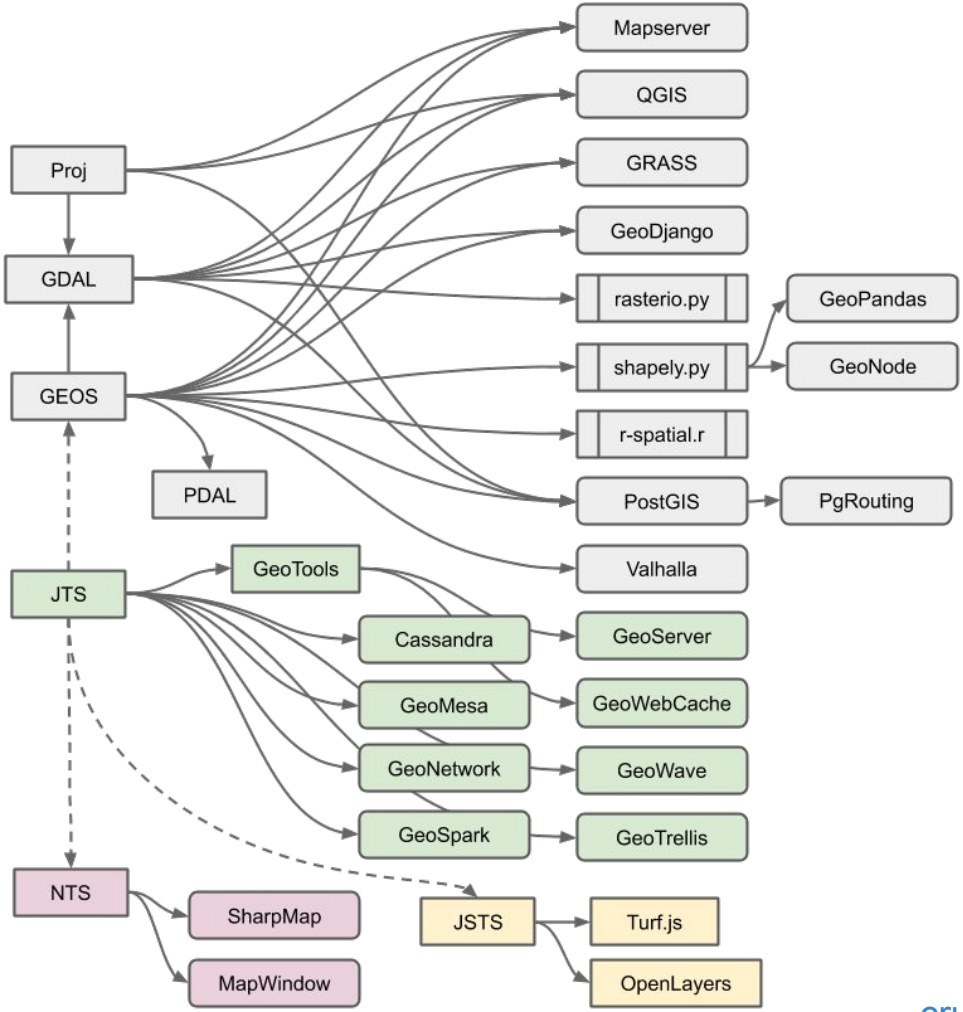


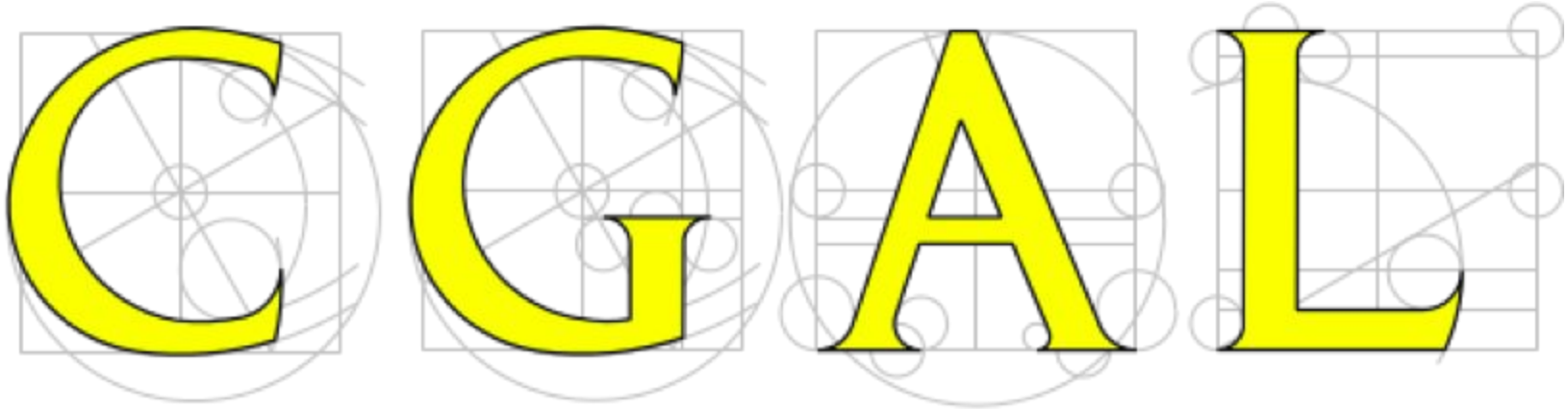
`ST_Intersection()`

Geometry Processing

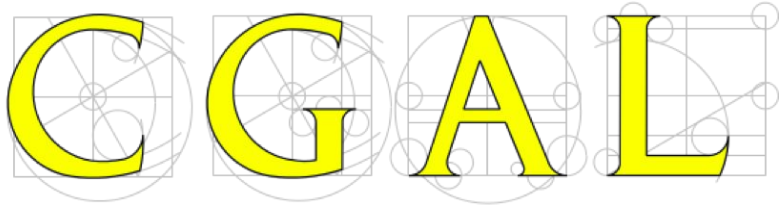


Geometry Engine Open Source





- **Another Library!**
- **3D Support**
- **New Algorithms**

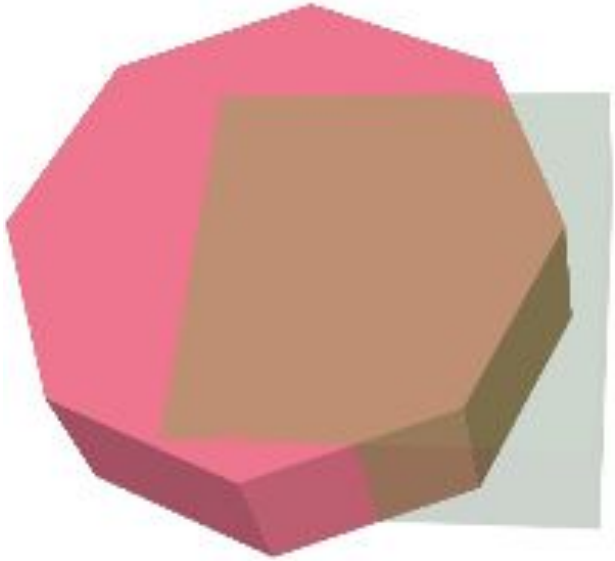



```
CREATE EXTENSION
postgis_sfcgal
```

- ST_3DIntersection
- ST_Tesselate
- ST_3DArea
- ST_3DUnion
- ST_Extrude
- ST_ApproximateMedialAxis
- ST_ForceLHR
- ST_Orientation
- ST_Minkowski
- ST_Volume
- ST_StraightSkeleton

ST_ApproximateMedialAxis

CGAL



ST_Extrude(geometry, x, y, z)

CGAL

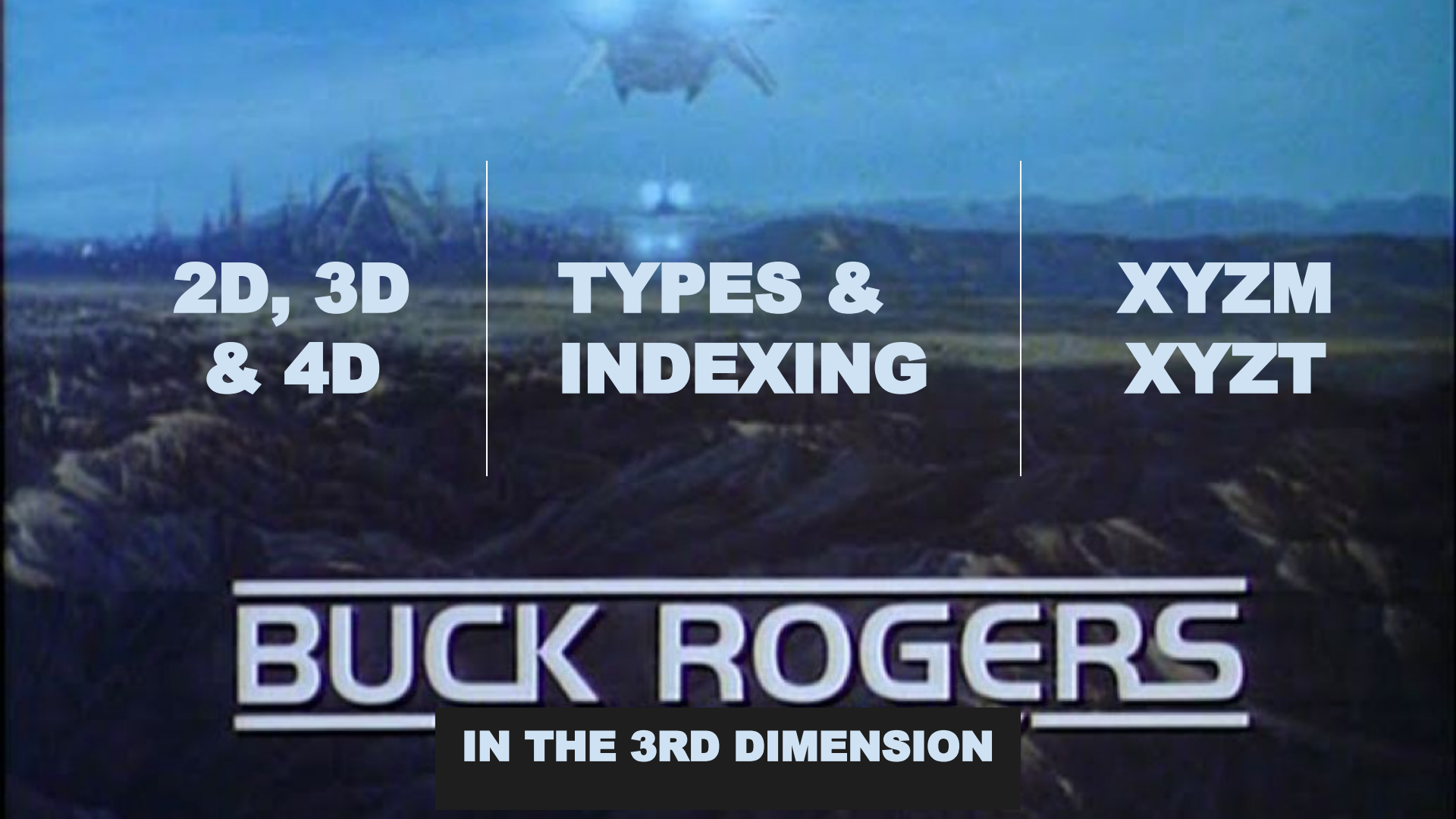
ST_3DIntersection(geometry, geometry)

CGAL

Multi-format Raster Processing

ST_Resample
ST_Rescale
ST_Rotate
ST_Transform
ST_AsJPEG
ST_AsPNG
ST_DumpAsPolygons


THE ZOO


2D, 3D
& 4D
TYPES &
INDEXING
XYZM
XYZT
BUCK ROGERS
IN THE 3RD DIMENSION

Higher dimensionality

```
POINT Z (1 2 3)
LINESTRING Z (1 3 4, 1 4 5)
POLYGON Z ((0 0 2, 1 1 2,
            1 0 2, 0 0 2))
```

```
ST_X('POINT(1 2 3)') = 3
```

Higher dimensionality input/output

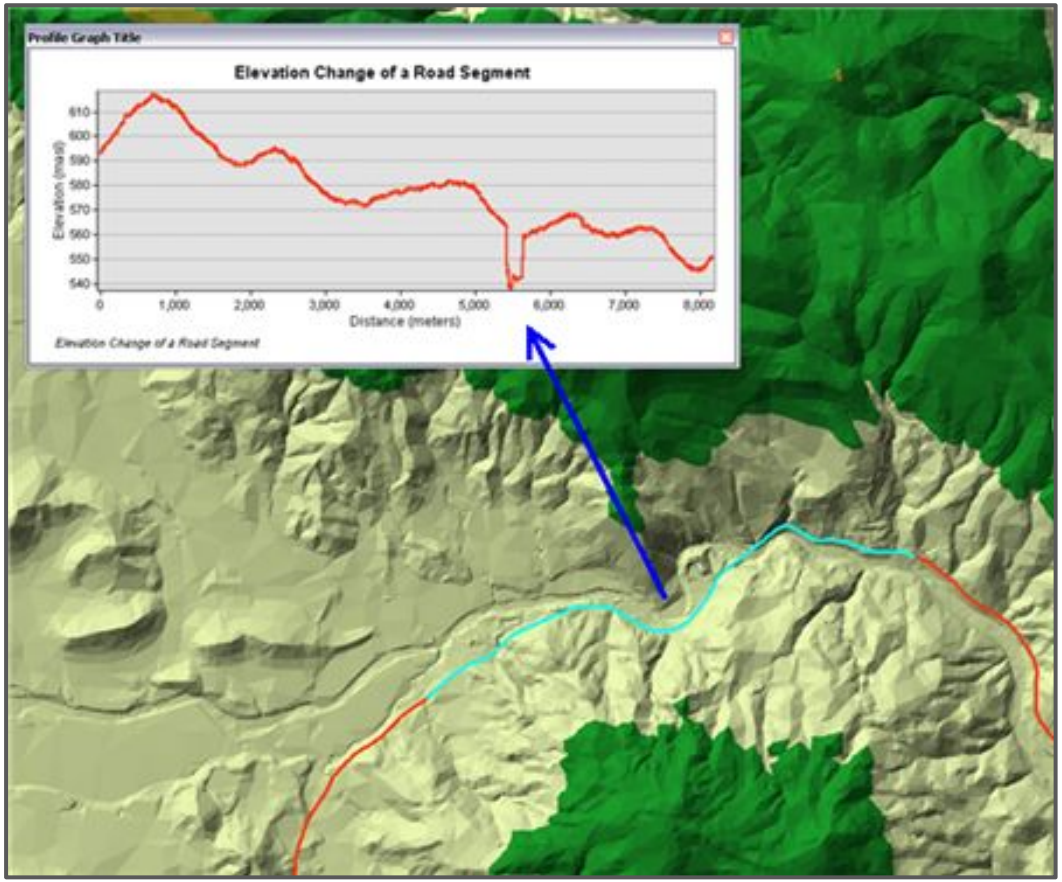
```
ST_AsText()
ST_AsBinary()
ST_AsGML()
ST_AsGeoJSON()
```

```
{"type":"Point",
 "coordinates":[1,1,1]}
```

3D Calculations

```
ST_3dDistance(geom, geom)
ST_3dLength(geom)
ST_3dClosestPoint(geom, geom)
ST_3dPerimeter(geom)
ST_3dIntersects(geom, geom)
ST_3dDWithin(geom, geom, tolerance)
```

3D Calculations




```
ST_Length
(geom, geom)
```

vs

```
ST_3dLength
(geom, geom)
```

Multi-dimensional Indexing

```
CREATE INDEX my_nd_x
ON my_table
USING GIST (
   geom gist_geometry_ops_nd
);
```

Multi-dimensional nearest neighbor!

```
SELECT *
FROM starships
ORDER BY
   geom <->
   'POINT Z (534 492 -166)'
LIMIT 1
```

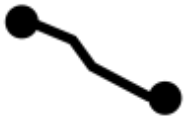
Extra-dimensionality (X, Y, Z, M)

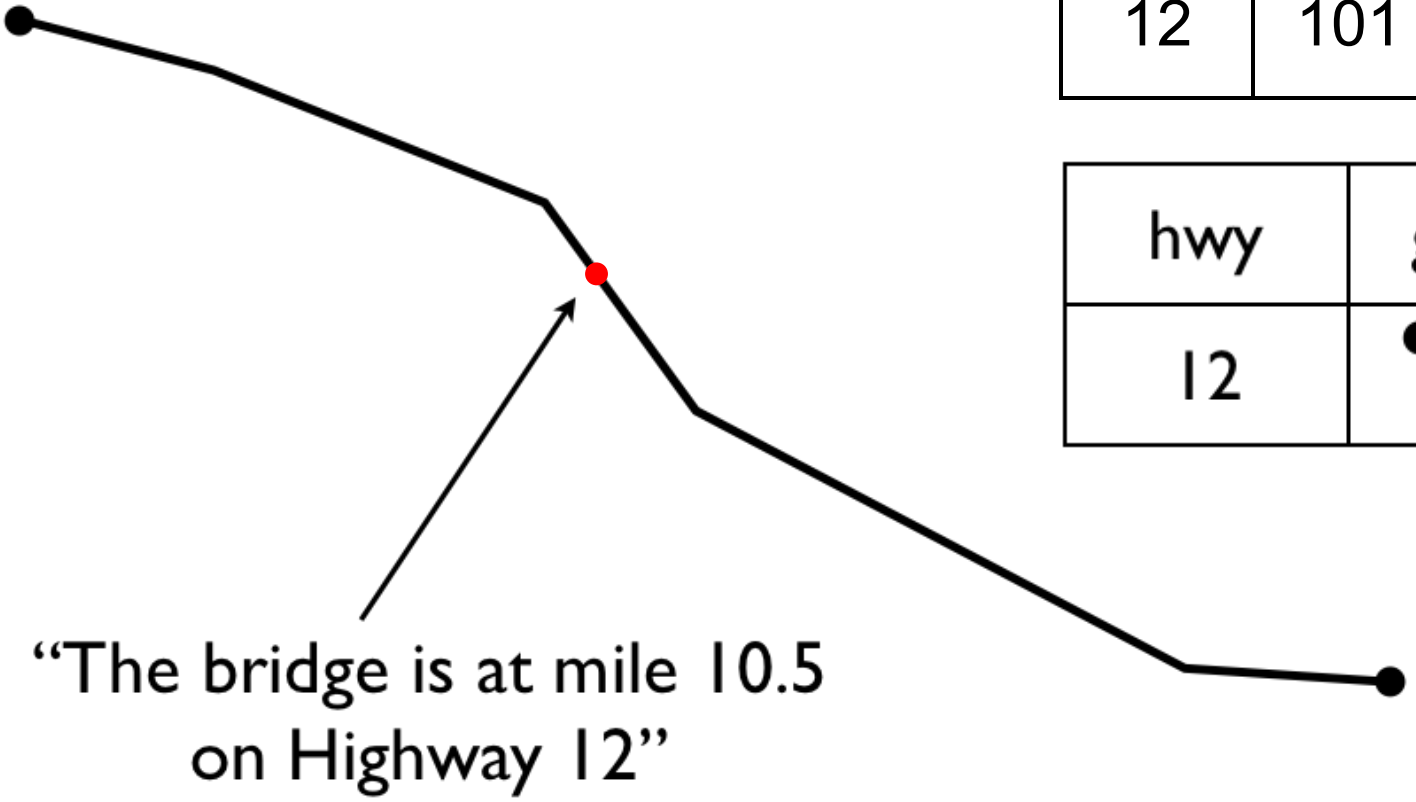
```
POINT ZM (1 2 3 4)
LINESTRING M (1 2 3, 1 4 3)
LINESTRING Z (1 3 4, 1 4 5)
```

```
ST_M('POINT(1 2 3 4)') = 4
```

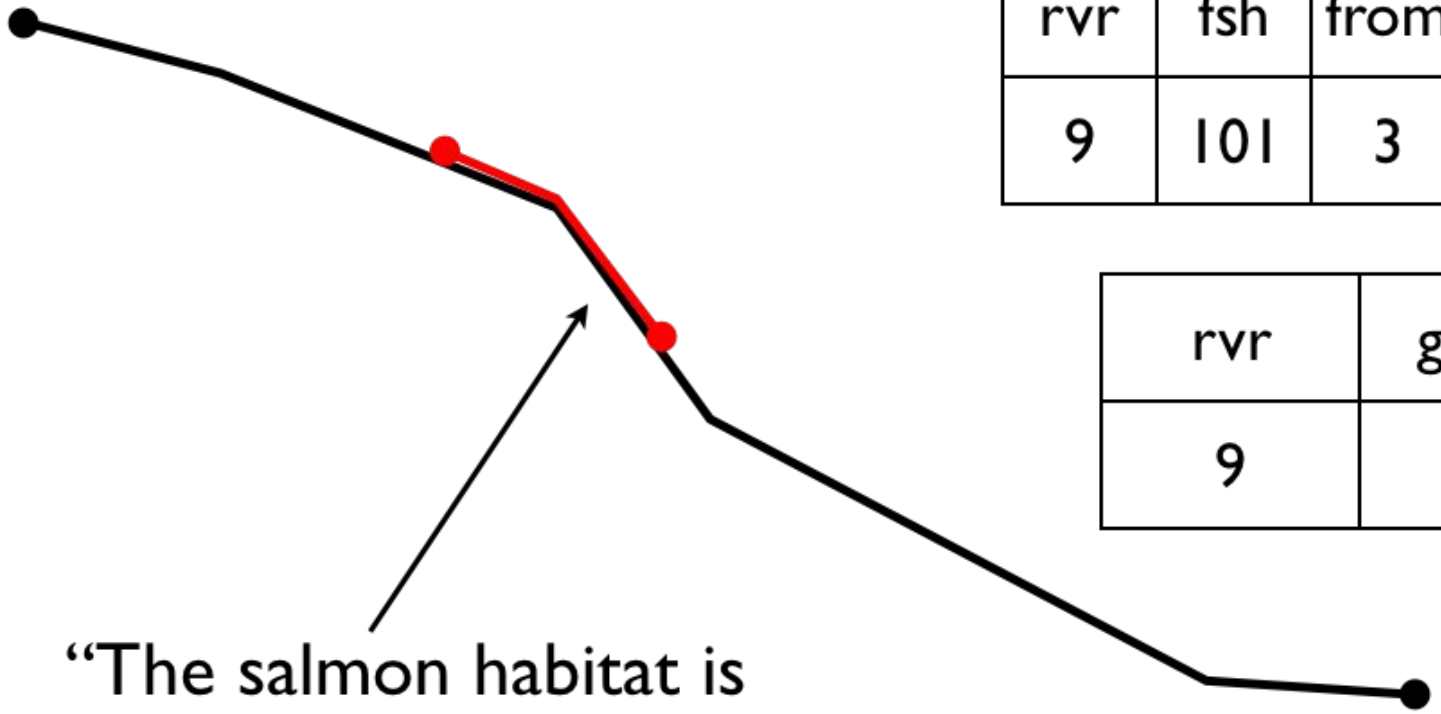
Linear Referencing

hwy	brdg	loc
12	101	10.5

hwy	geom
12	



Linear Referencing



rvr	fsh	from	to
9	101	3	5

rvr	geom
9	

"The salmon habitat is from 3km to 5k above the confluence"

Linear Referencing

LINESTRING M (x_1 y_1 m_1, x_2 y_2 m_2)

ST_LineInterpolatePoint(geom, float)

ST_LineLocatePoint(geom, geom)

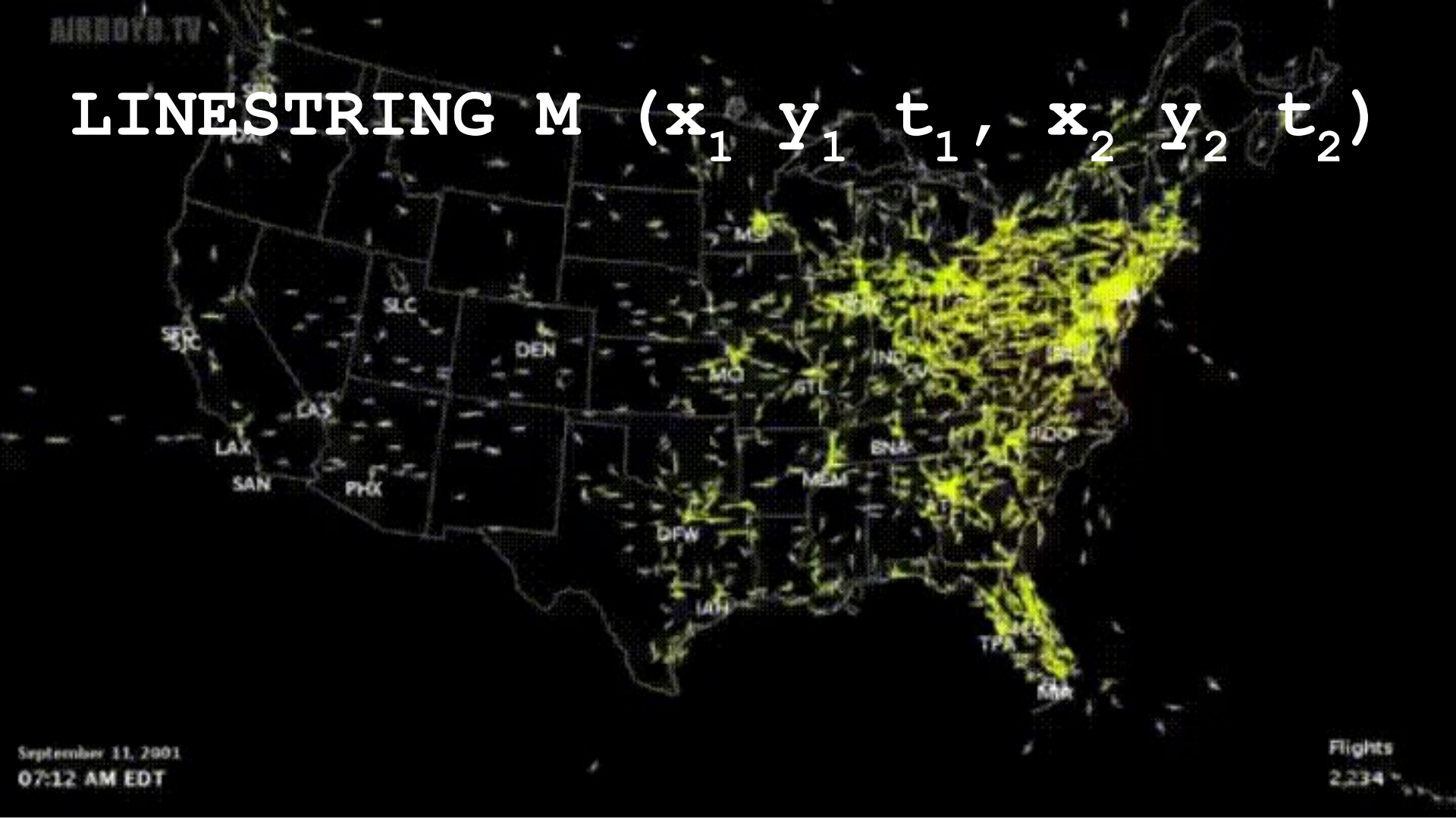
ST_LocateAlong(geom, float, float)

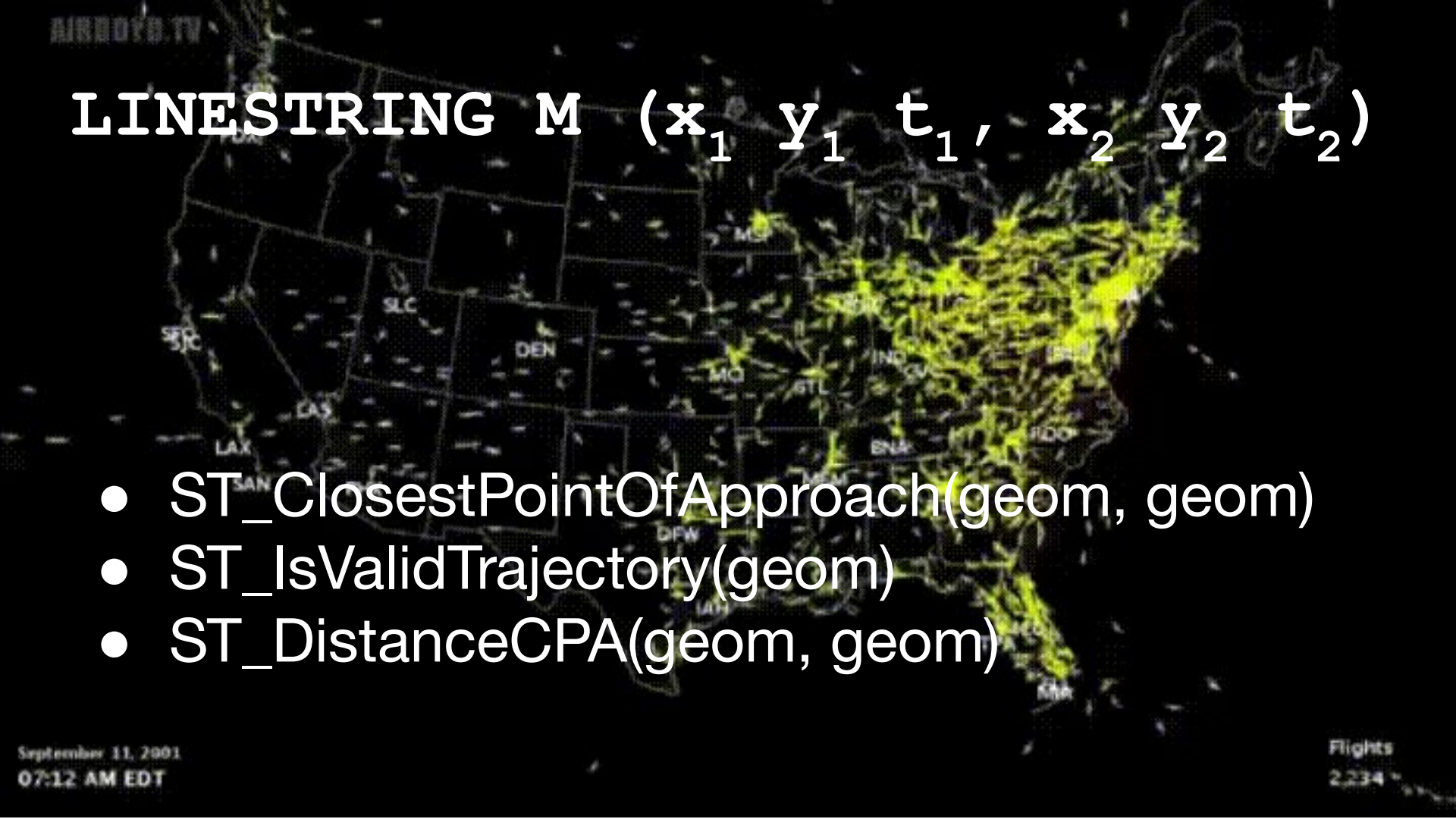
ST_LocateBetween(geom, float, float)

ST_LineSubstring(geom, float, float)

Spatio-Temporal Functionality

LINESTRING M (x_1 y_1 t_1, x_2 y_2 t_2)

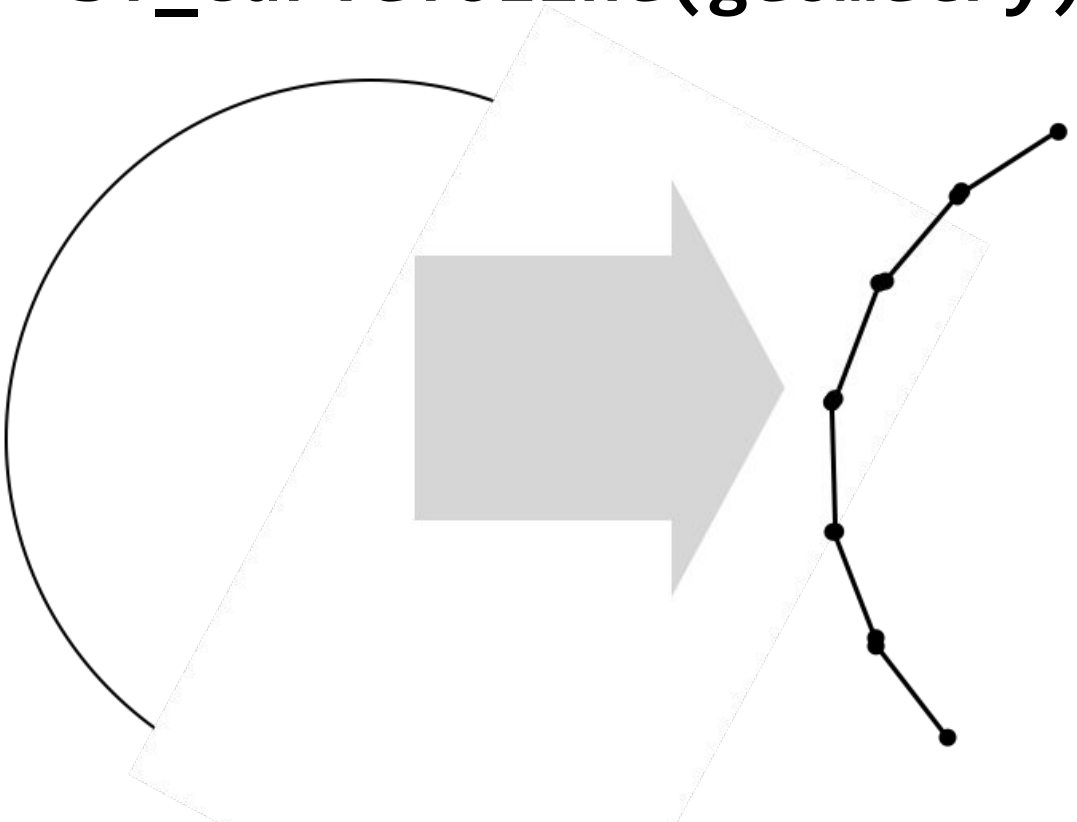




- ST_ClosestPointOfApproach(geom, geom)
- ST_IsValidTrajectory(geom)
- ST_DistanceCPA(geom, geom)

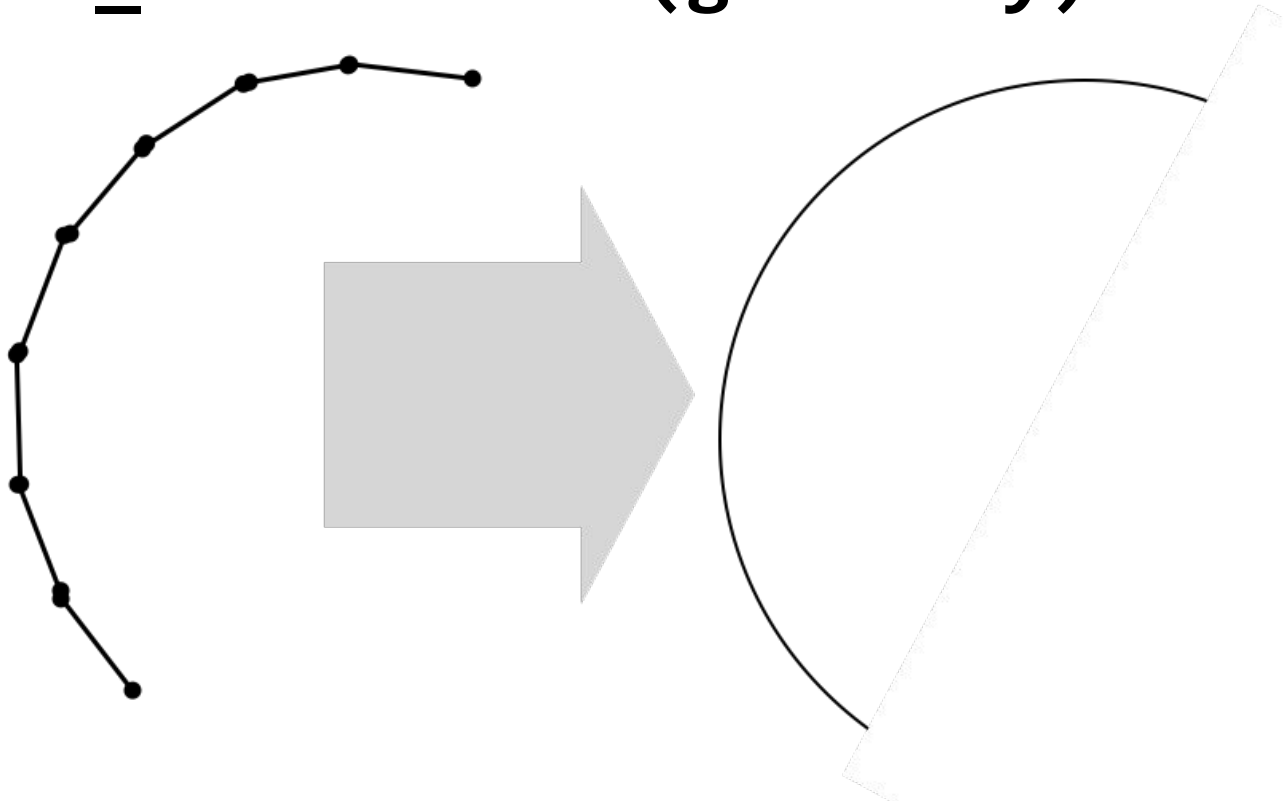
Curve Processing

```
ST_CurveToLine(geometry)
```

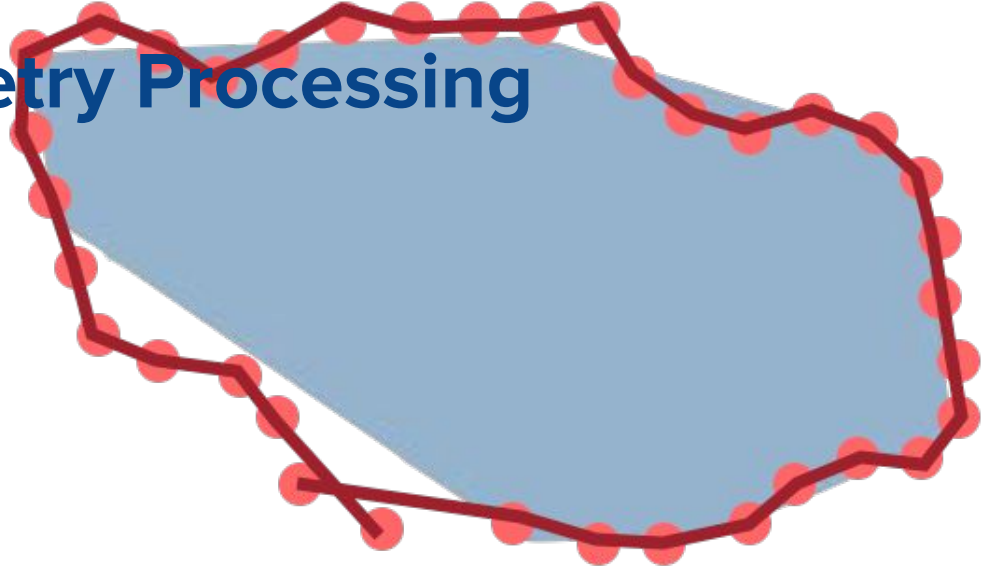


Curve Processing

ST_LineToCurve(geometry)

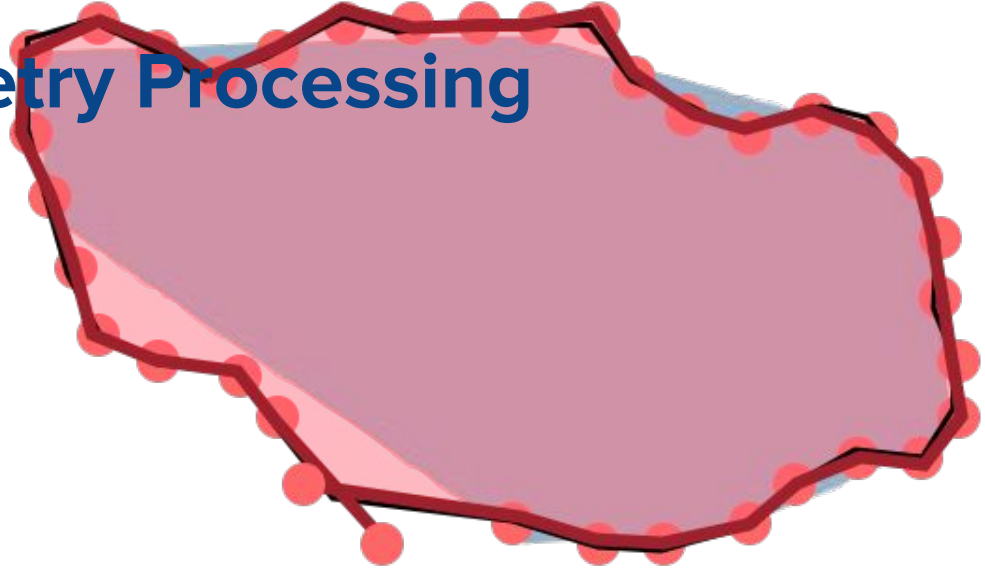


Geometry Processing



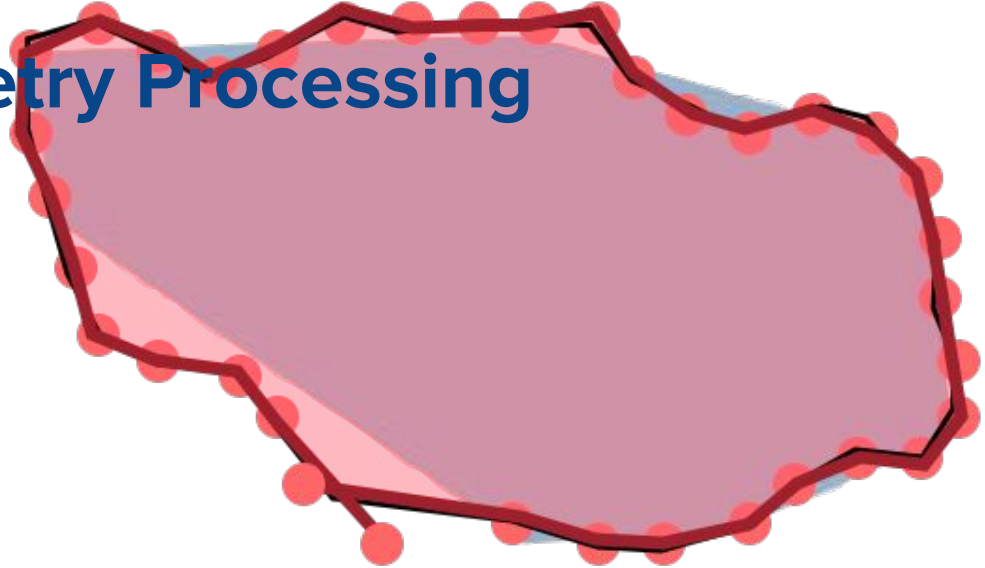
```
ST_MakeLine({point})
```

Geometry Processing



```
ST_BuildArea(multilinestring)
```

Geometry Processing



```
ST_BuildArea(multilinestring)
```

Geometry Processing

OGC Standard
Slow

- **ST_GeometryN**(geometry, integer)
- **ST_PointN**(geometry, integer)
- **ST_ExteriorRing**(geometry)

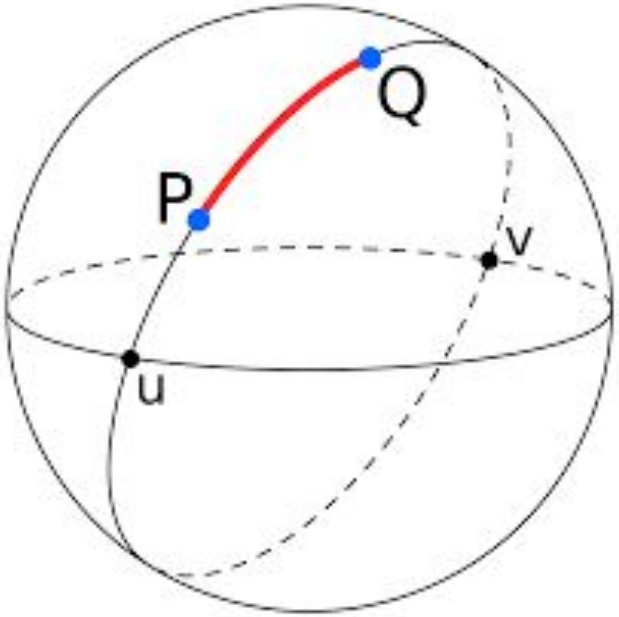
  ST_InteriorRingN(geometry, integer)

PostGIS
Fast

- **ST_Dump**(geometrycollection) => setof(geometry)
- **ST_DumpRings**(polygon) => setof(geometry)
- **ST_DumpPoints**(geometry) => setof(geometry)

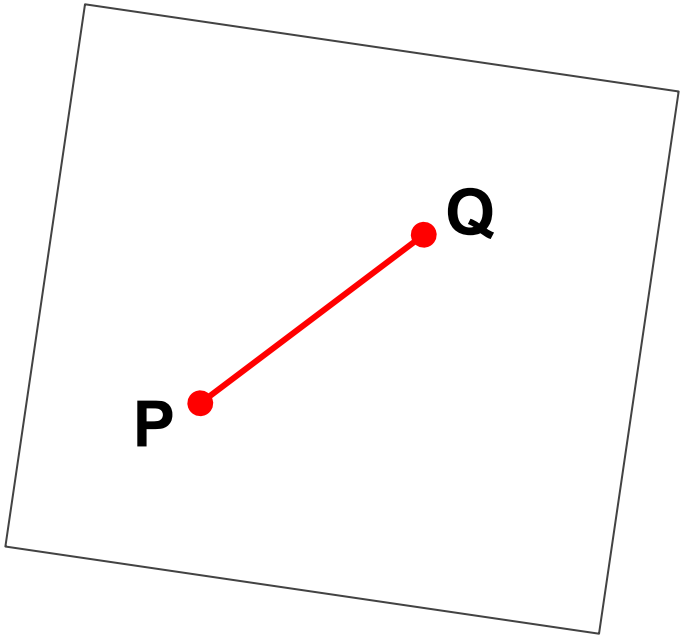

Geography

S_2



geography

R_2



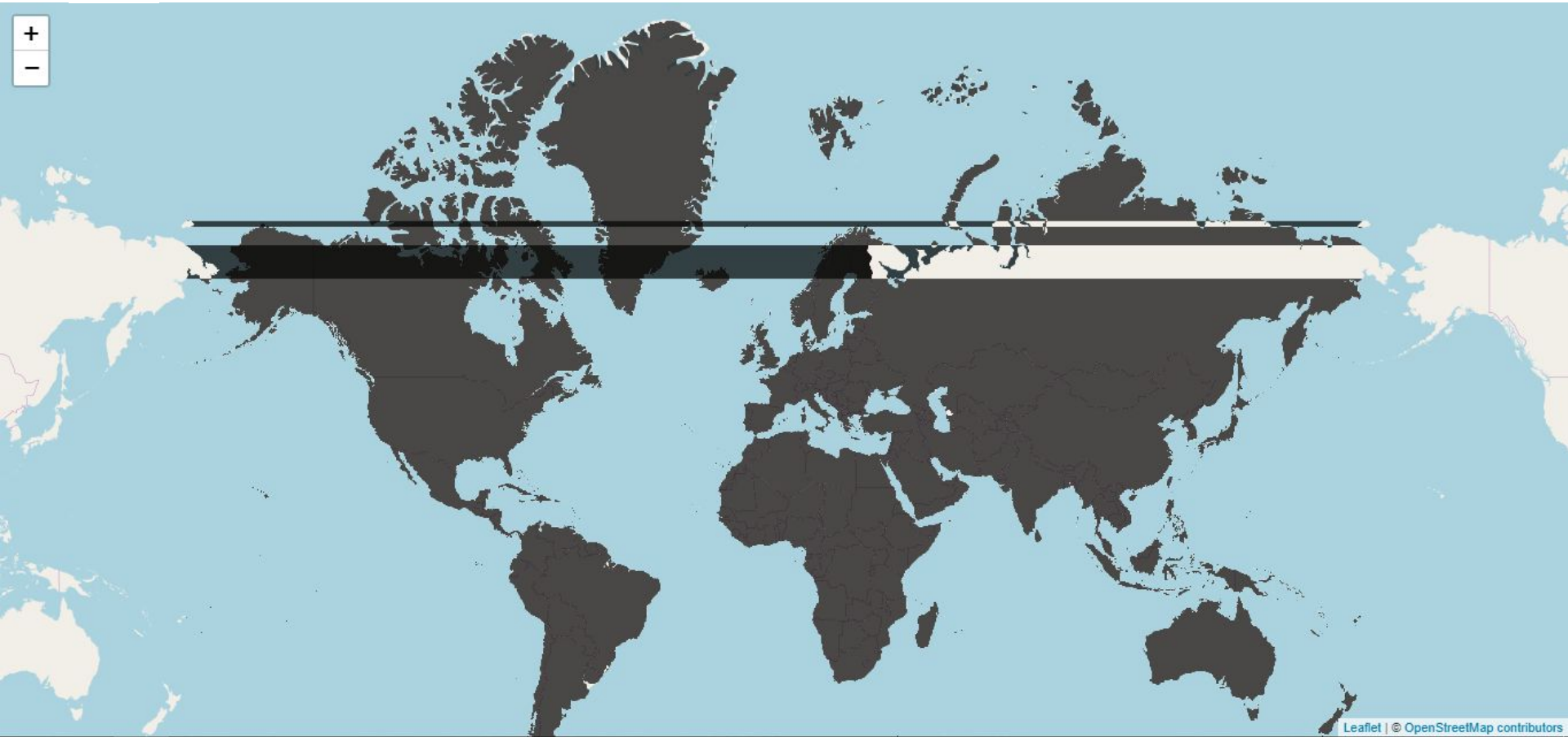
geometry

Geography is for...

GeoNewbies

“I want to find all the address points within one mile! My data is in lat/lon! Google Maps rocks!”

“GIS sucks, and I want to go home now.”



Geography is for...

GeoHugies

"Yeah, I own a freaking satellite, you got a problem with that?"


FREE
LUNCH
$10.00
SCHWADRON

Geography supports...

geography

```
double R = 6371000; /* meters */
double d_lat = lat2-lat1; /* radians */
double d_lon = lon2-lon1; /* radians */
double sin_lat = sin(d_lat/2);
double sin_lon = sin(d_lon/2);
double a = sin_lat * sin_lat +
           cos(lat1) * cos(lat2) *
           sin_lon * sin_lon;
double c = 2 * atan2(sqrt(a),
                     sqrt(1-a));
double d = R * c;
```

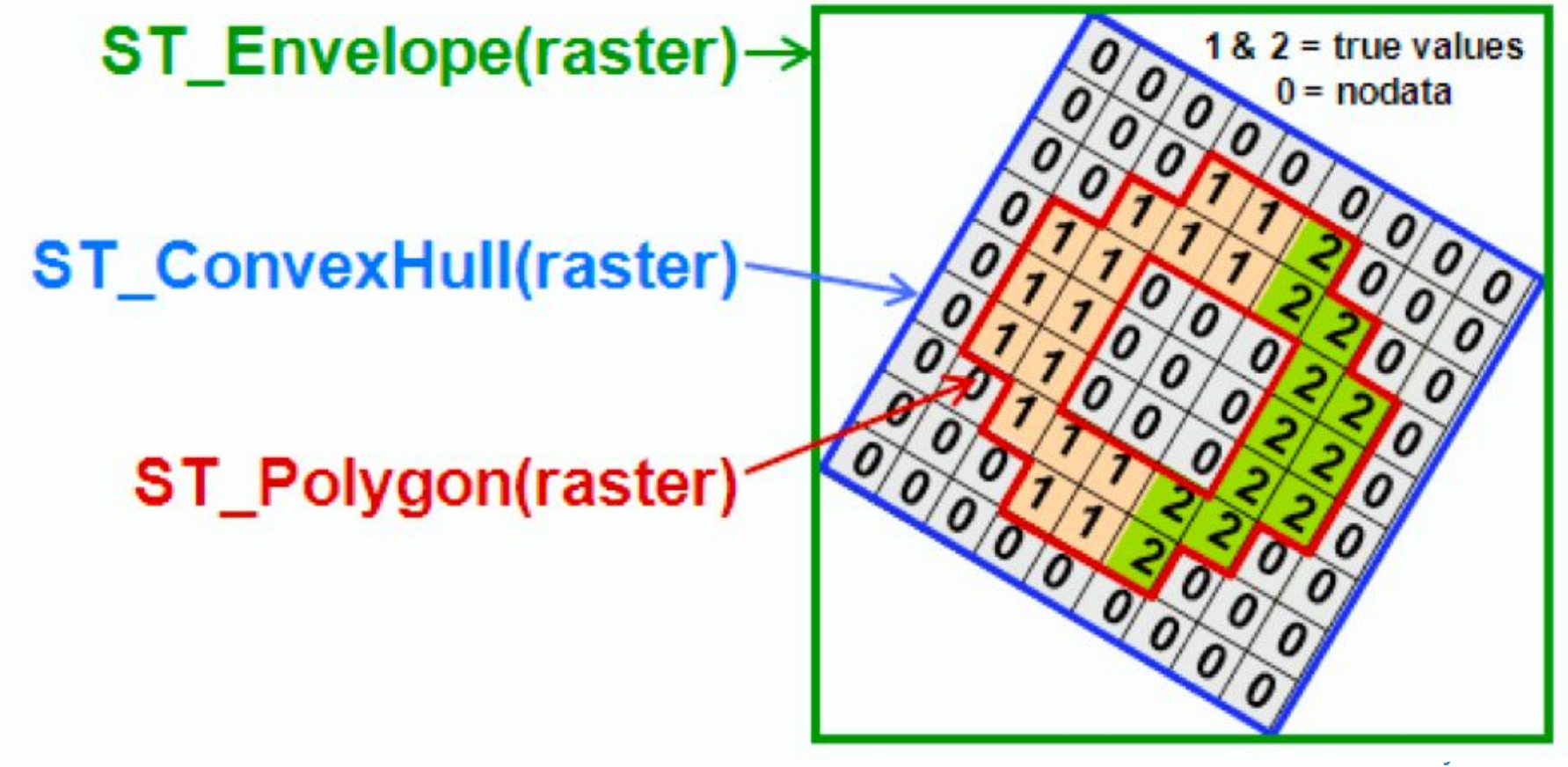
geometry

```
double dx = x2 - x1;
double dy = y2 - y1;
double d2 = dx * dx +
            dy * dy;
double d = sqrt(d2);
```

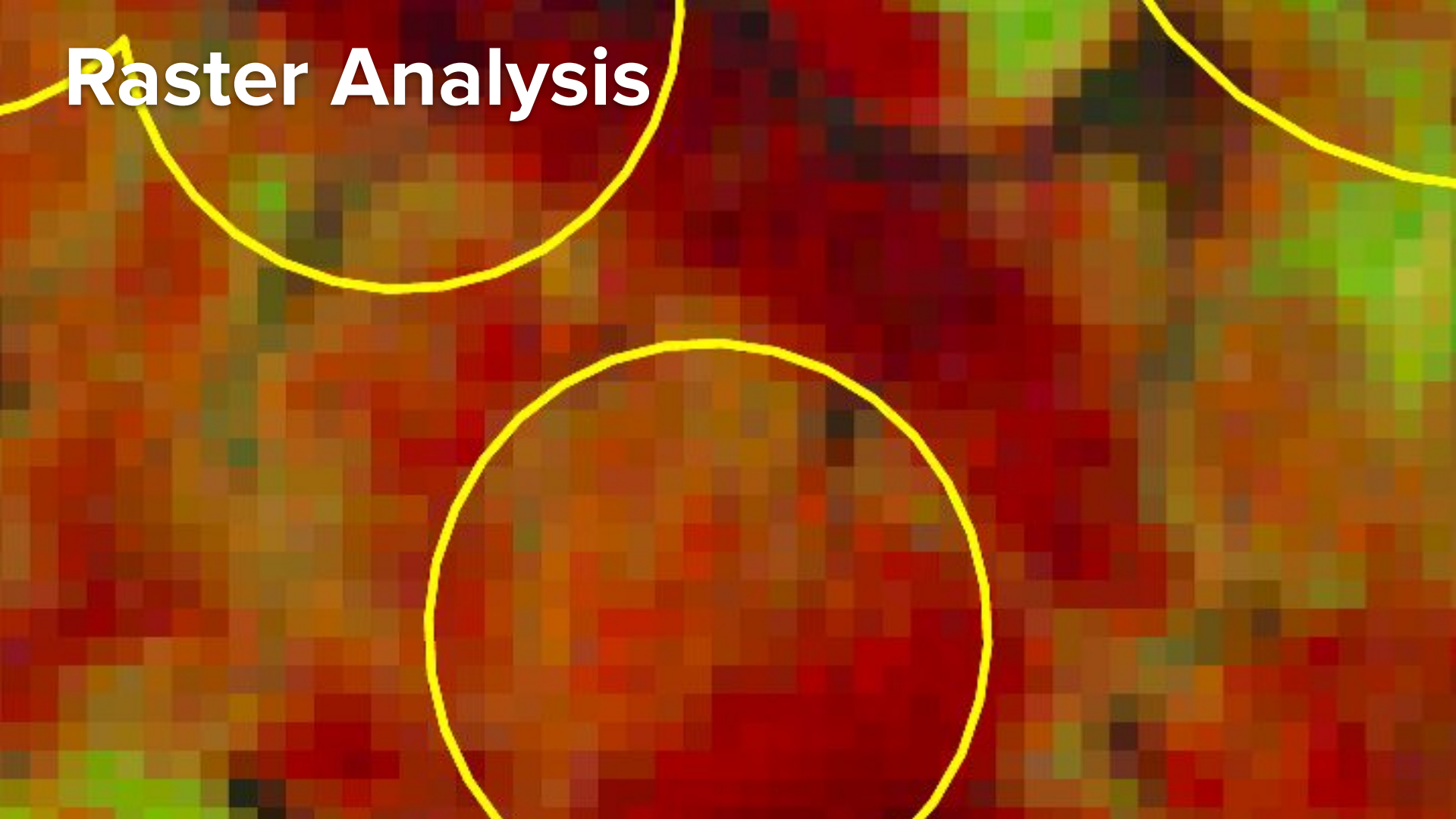
Geography supports...

- Indexes on spherical data
- Nearest neighbor searches
- ST_Intersects()
- ST_Distance()
- ST_DWithin()
- ST_Area()
- Casts to/from GEOMETRY

Raster Analysis

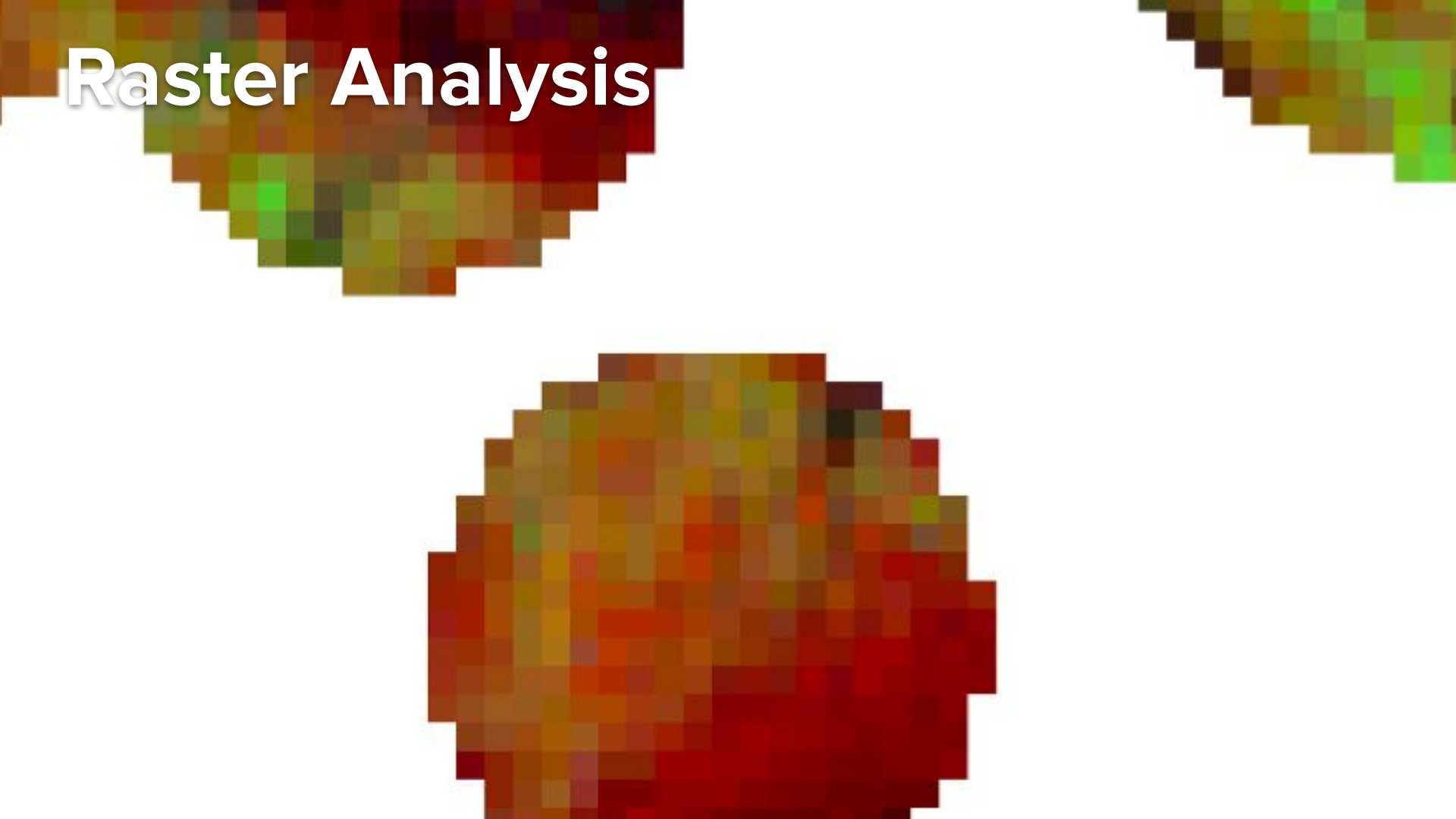


Raster Analysis



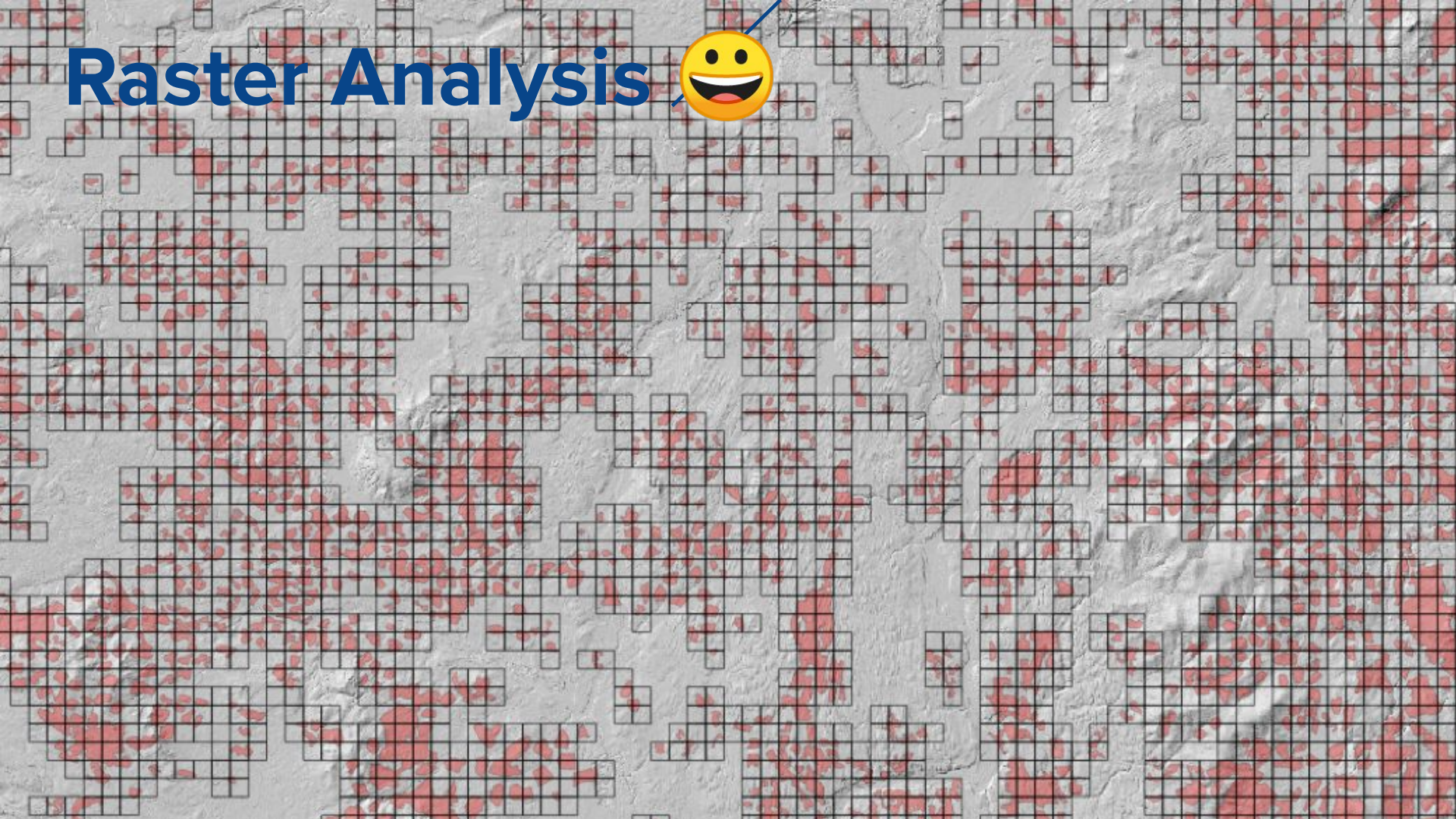
Raster Analysis

Raster Analysis

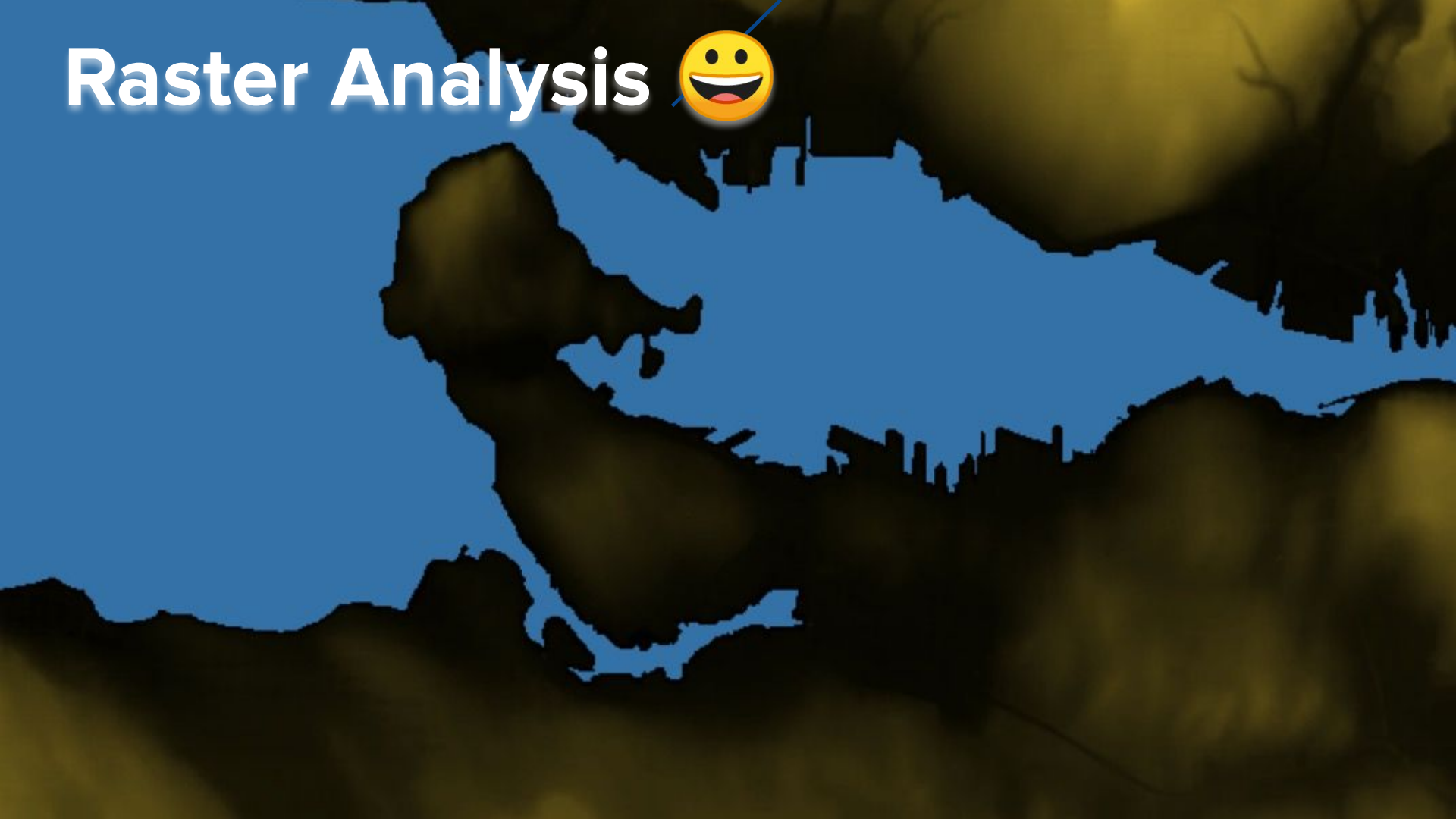


Raster Analysis 😀

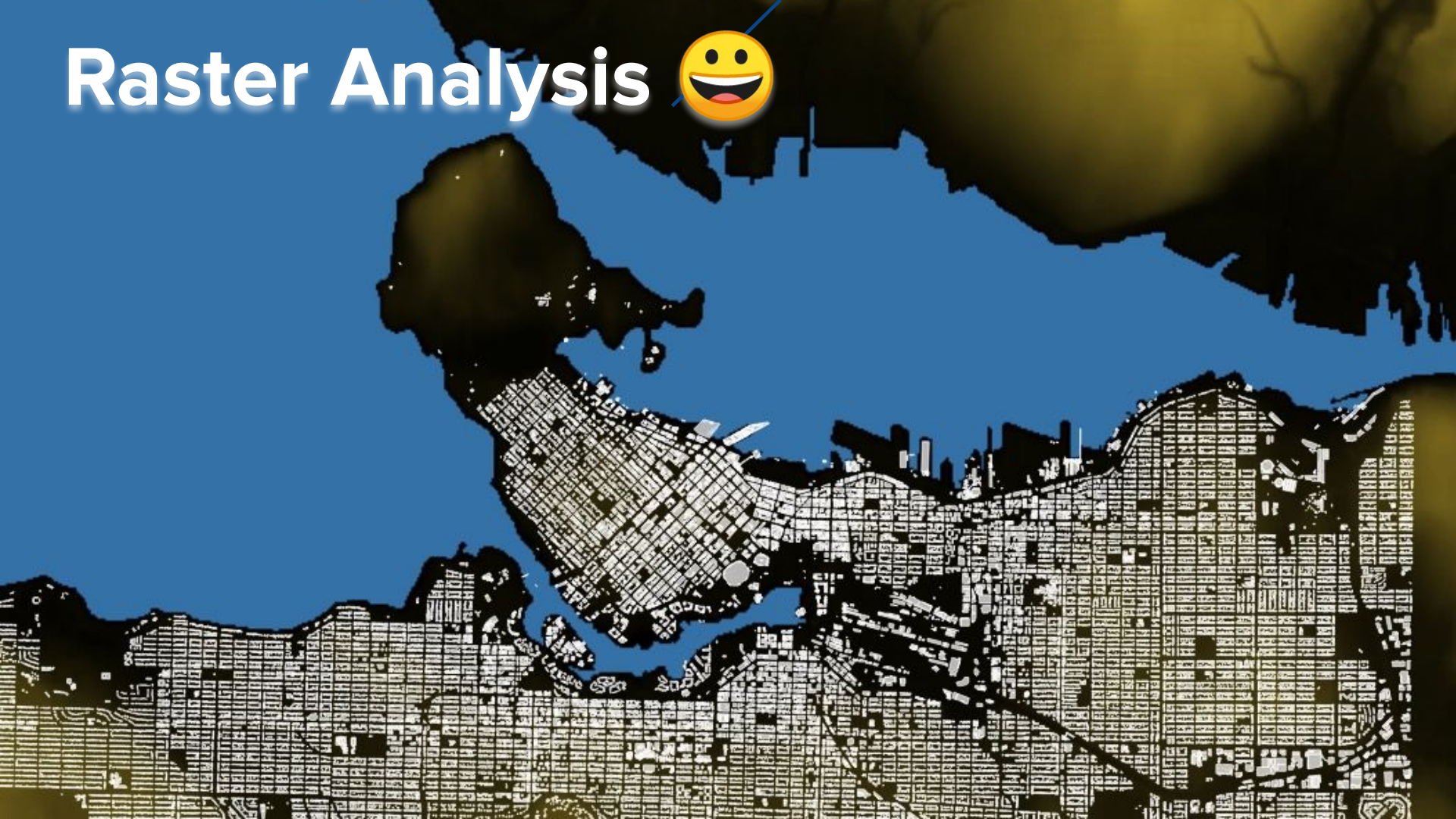
Raster Analysis 😀



Raster Analysis 😀



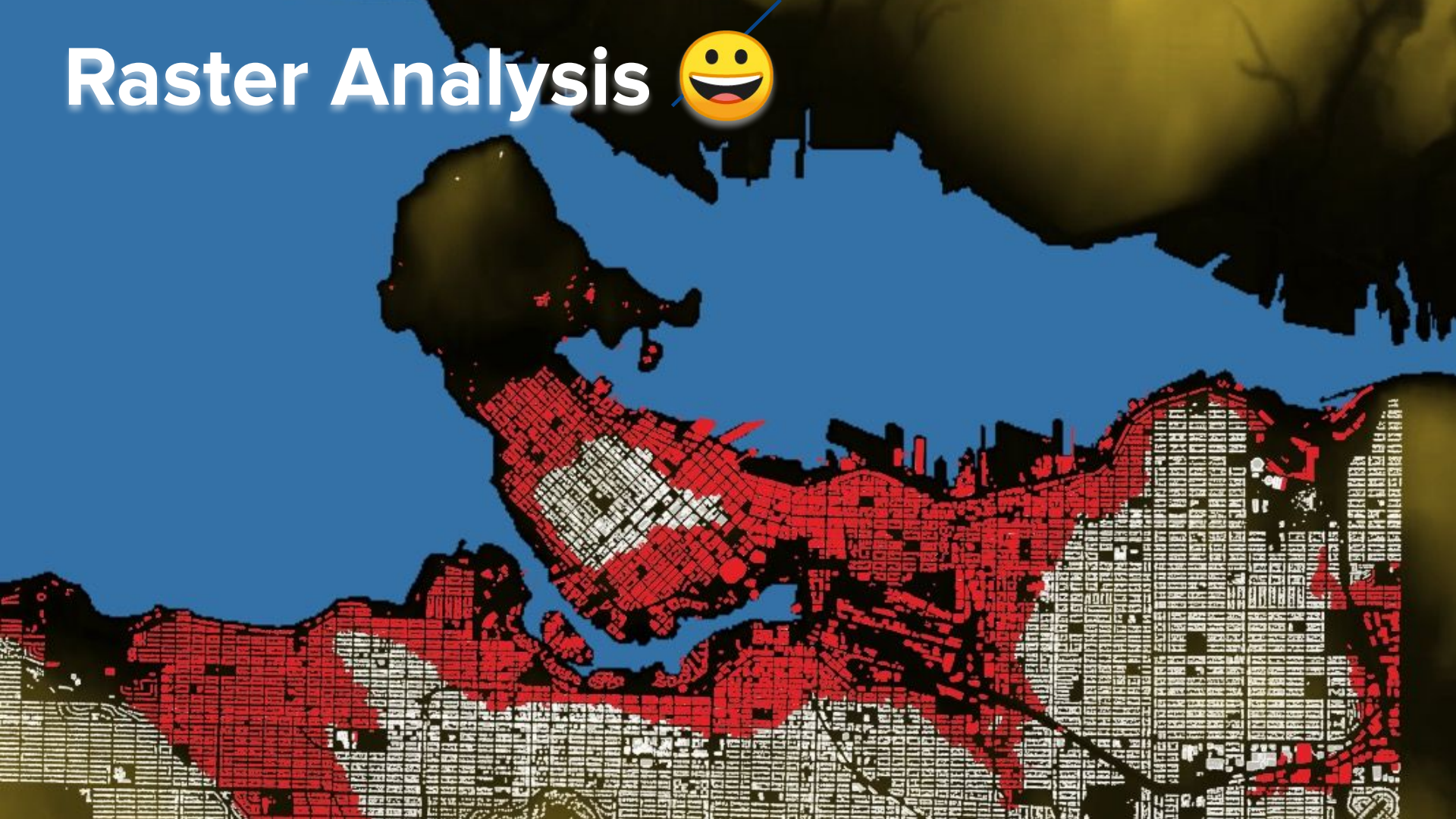
Raster Analysis 😀



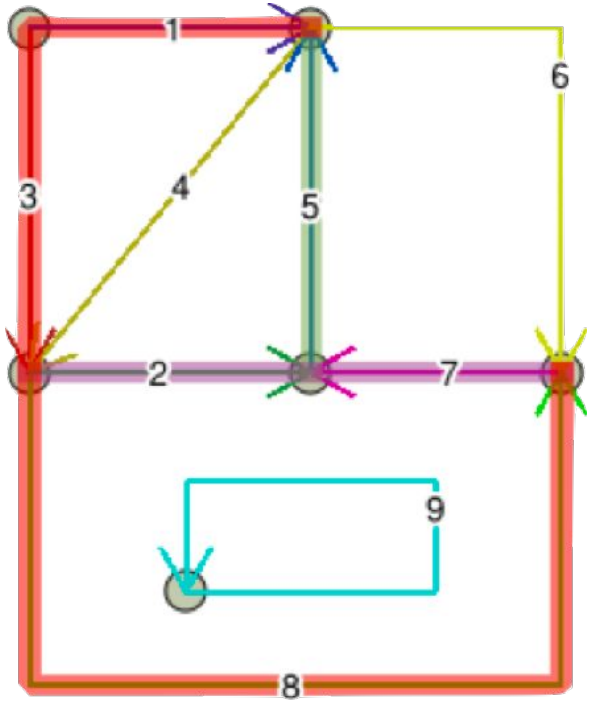
“What buildings will flood at 30 metre sea level rise?”

```
SELECT b.*
FROM buildings b
JOIN dem
ON ST_Intersects(b.geom,
                 dem.rast)
WHERE ST_Value(
   dem.rast,
   ST_Centroid(b.geom)) < 30;
```

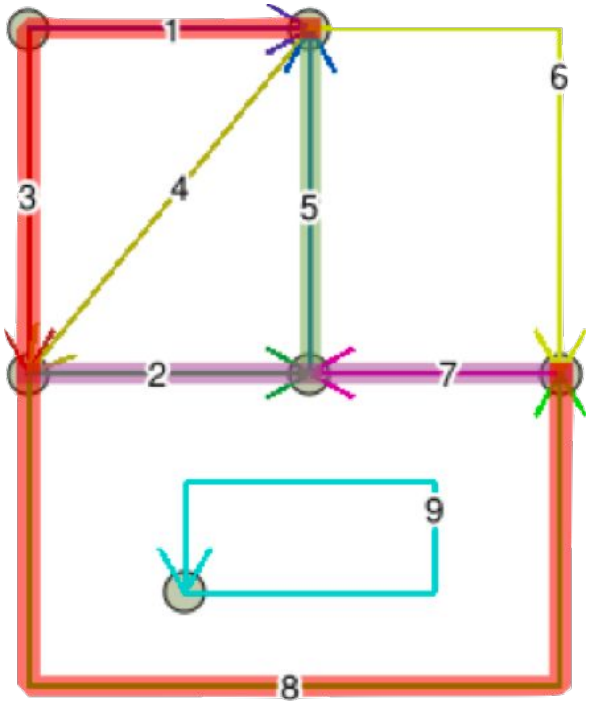
Raster Analysis 😀



Topology Model



Topology Model

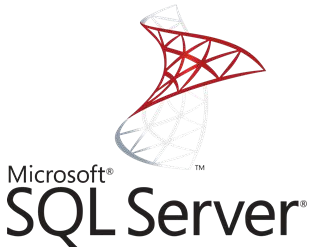


- For coverages & shared boundaries
- Parcels, admin boundaries
- Good tooling required

  eg QGIS PostGIS Topology Editor

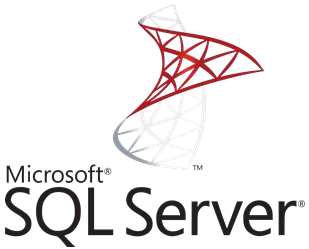

Performance

PostGIS is **faster** than...



PostGIS is **slower** than...

PostGIS is **better** than...



PostGIS is **worse** than...

Performance Tweaks

Smallification

ST_Subdivide()

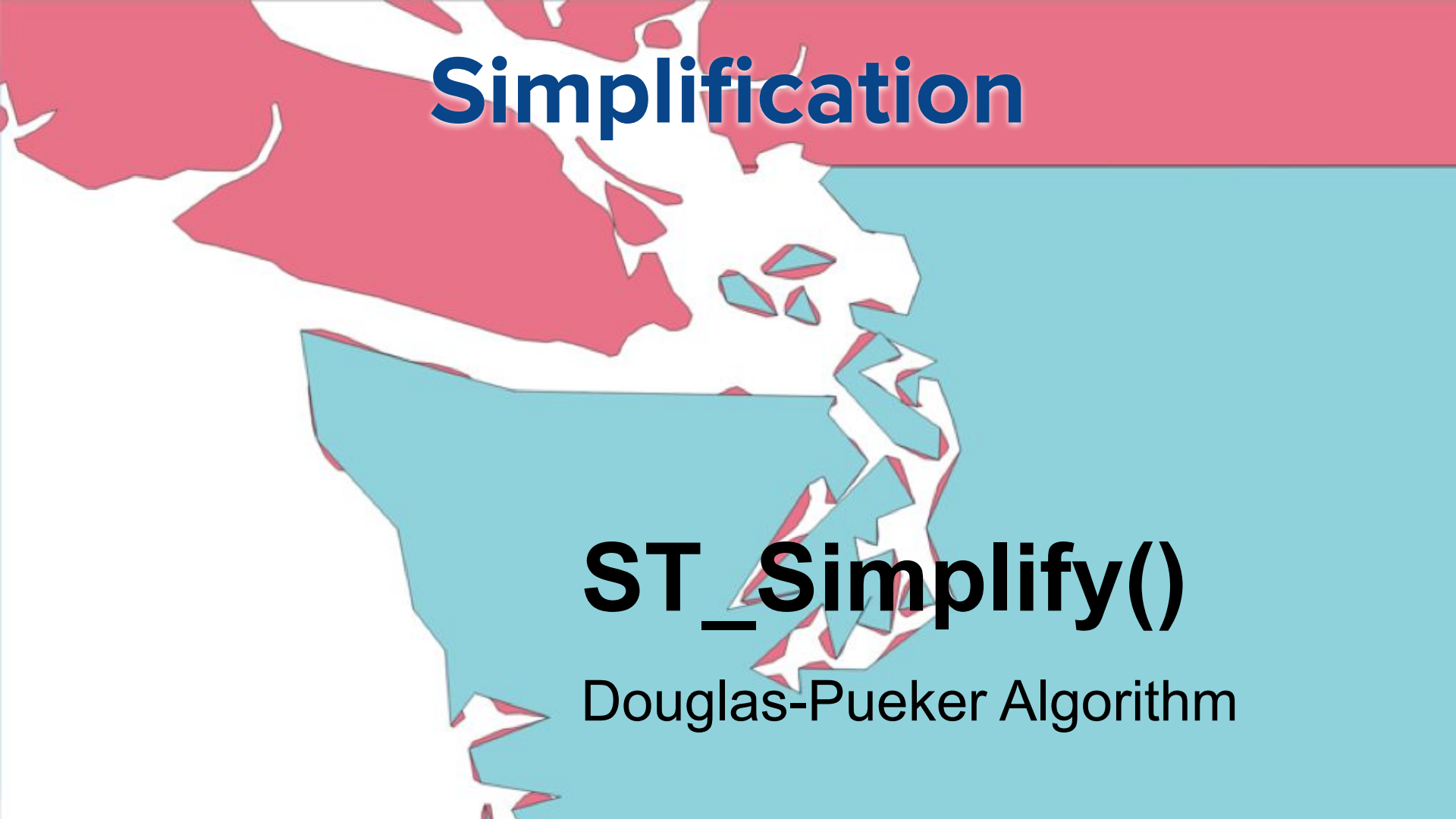
Recursive subdivision

Polygon with
90,000 vertices


CREATE TABLE land_subdivided AS
SELECT
ST_SubDivide(geom) AS geom,
base250_id
FROM land
Polygons with
no more than
250 vertices

Simplification

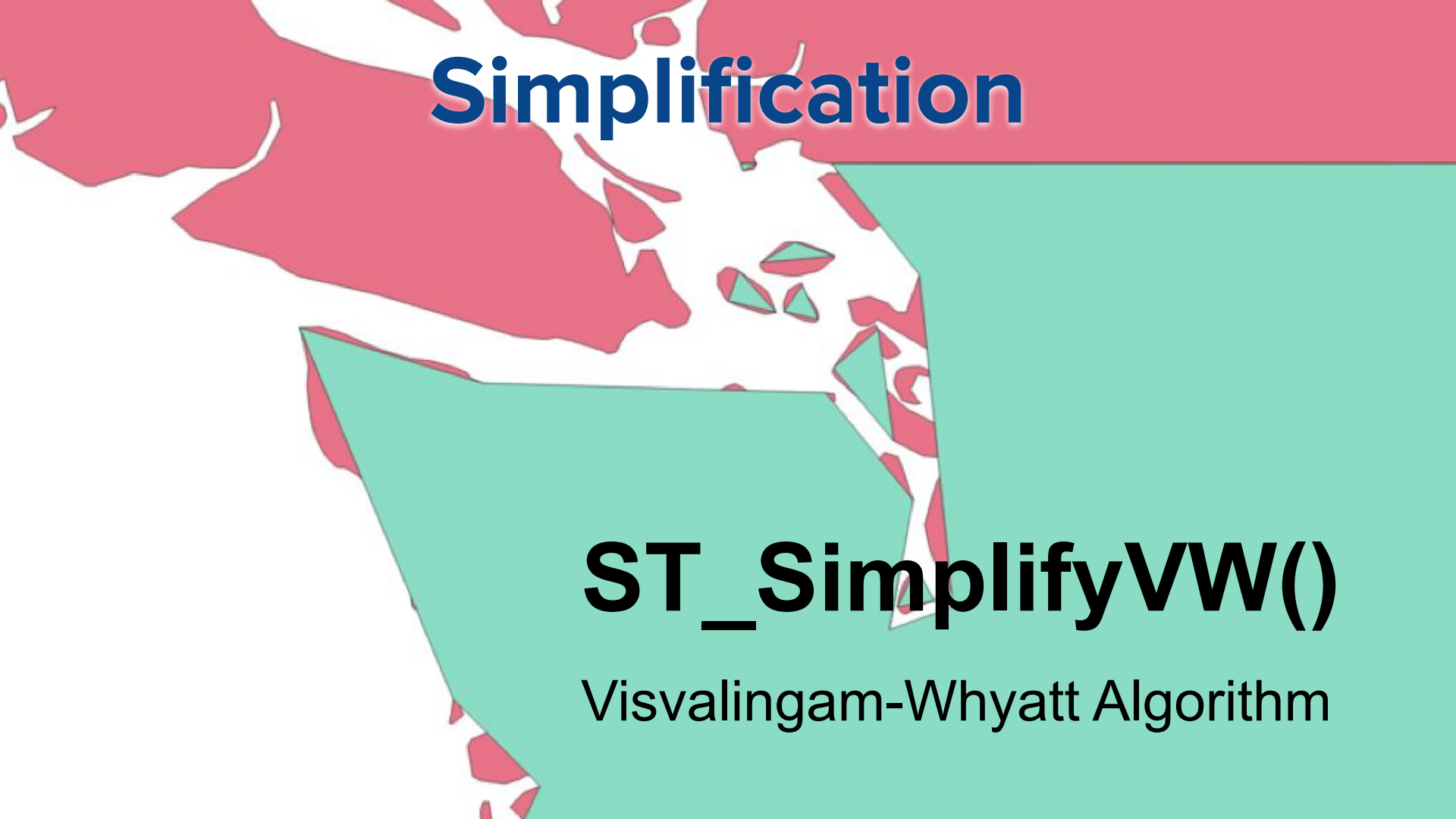
Simplification



ST_Simplify()

Douglas-Pueker Algorithm

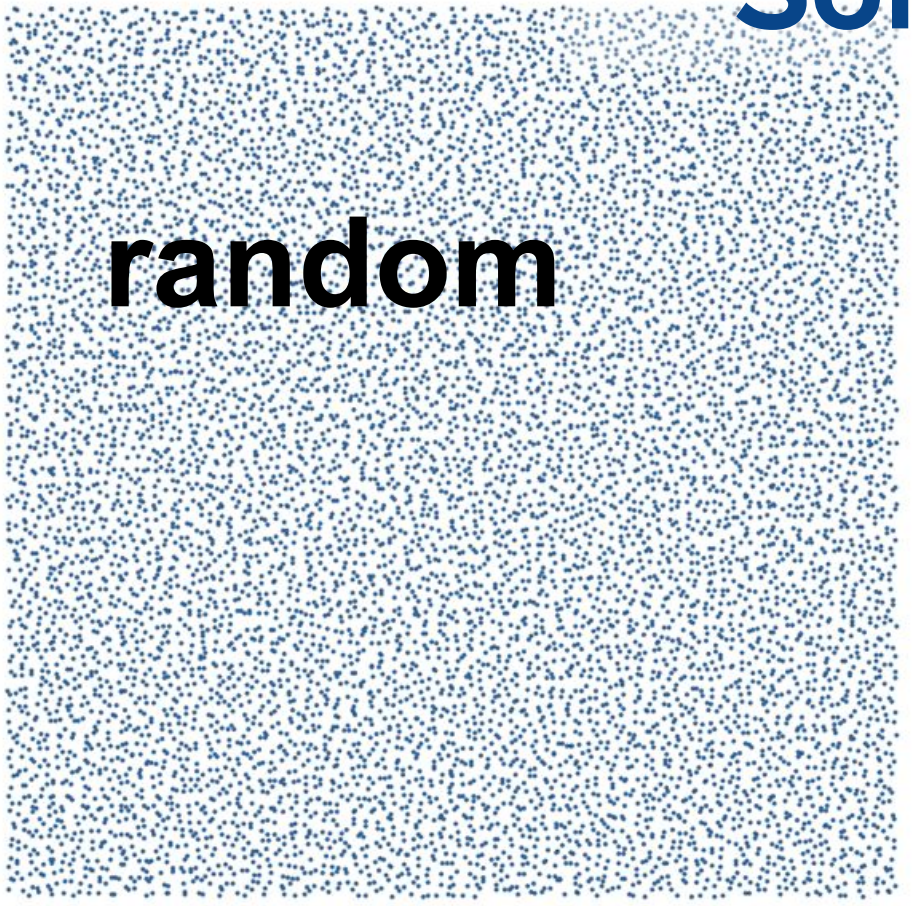
Simplification



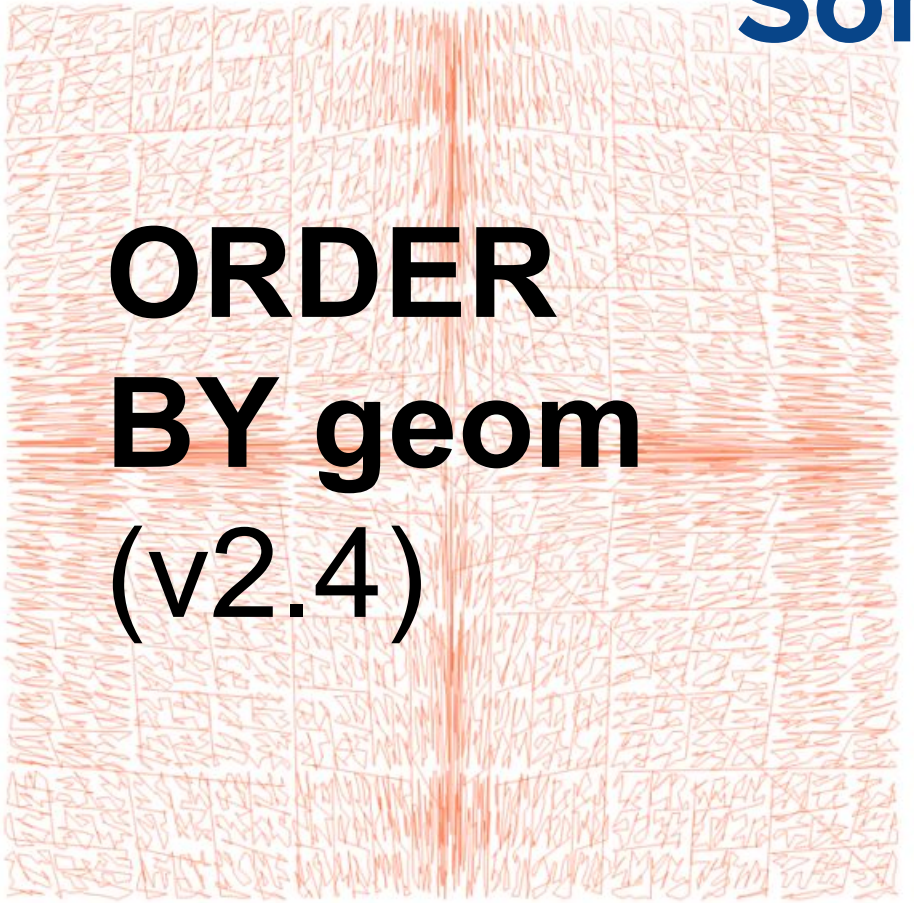
ST_SimplifyVW()

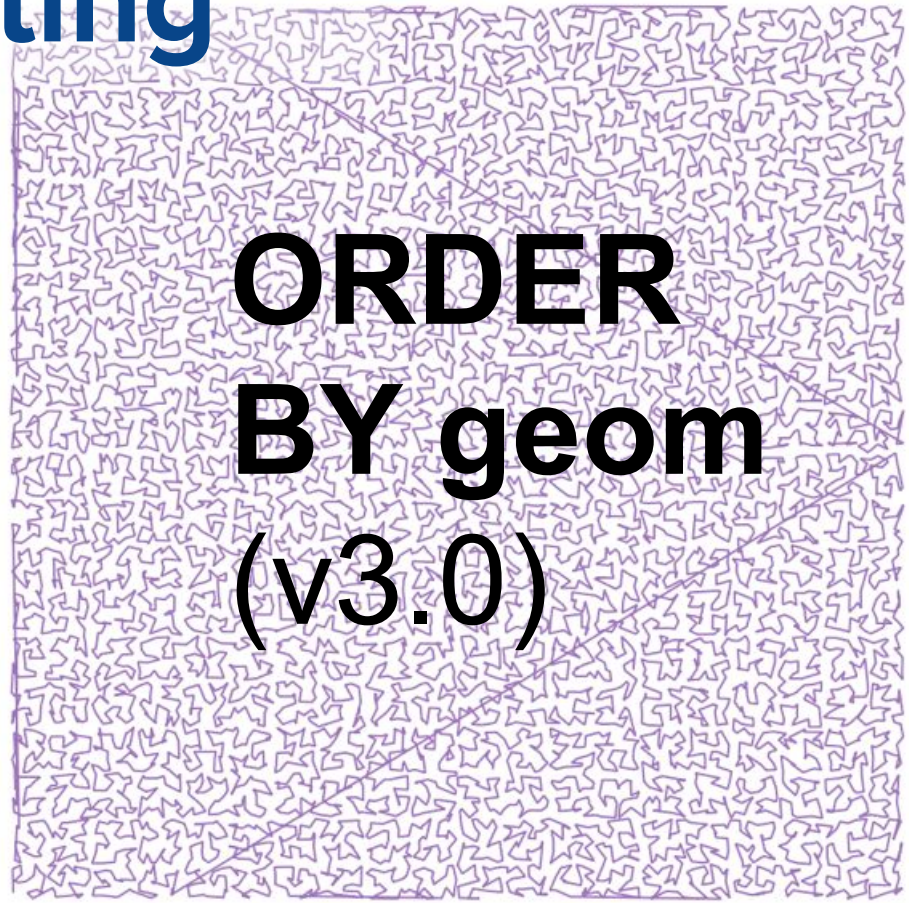
Visvalingam-Whyatt Algorithm

Sorting



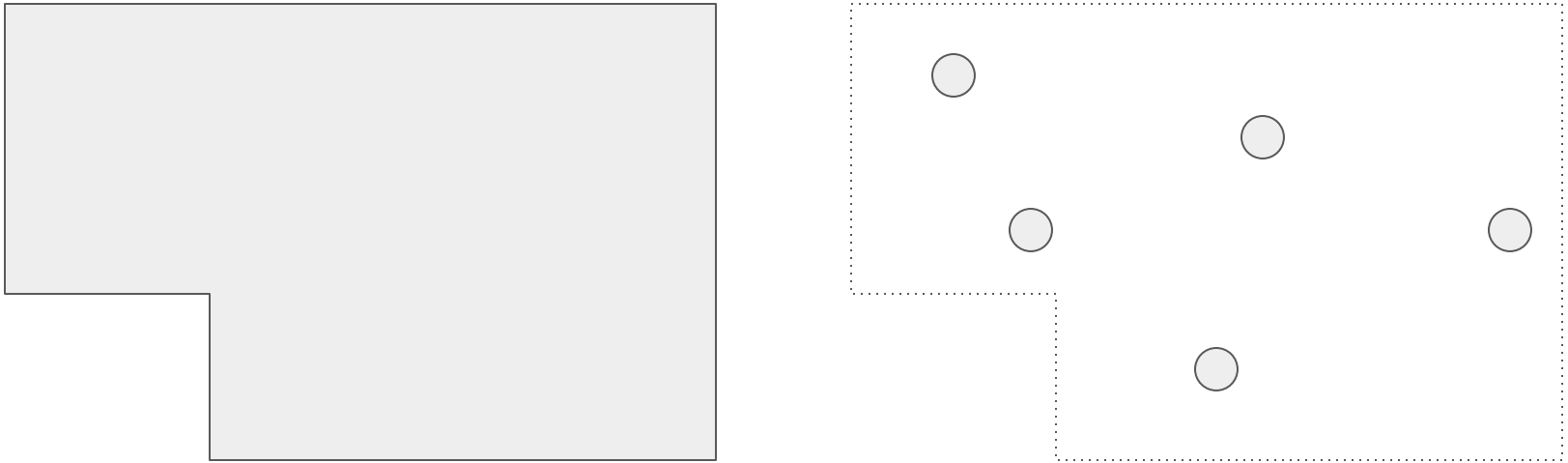
Sorting

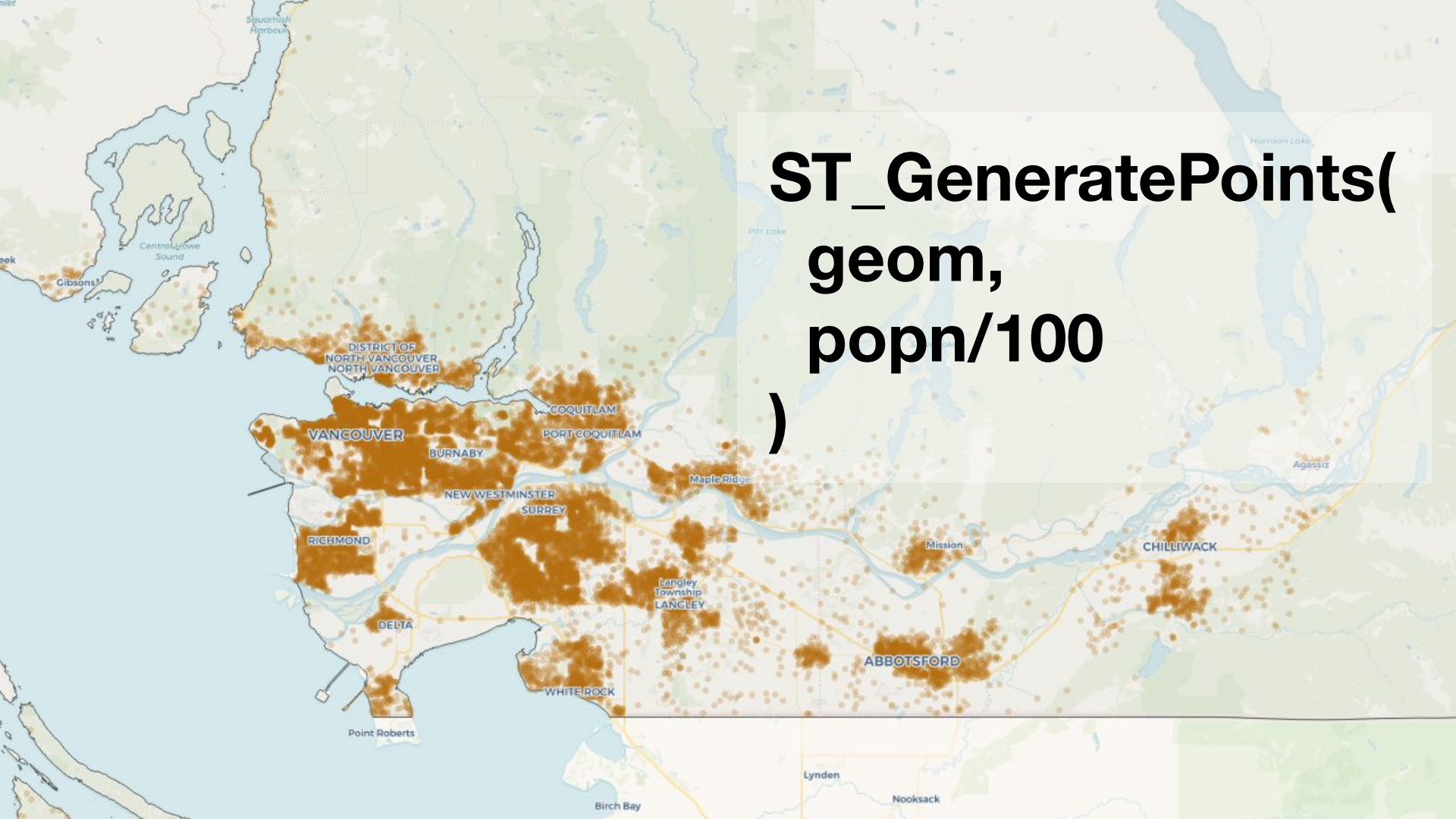


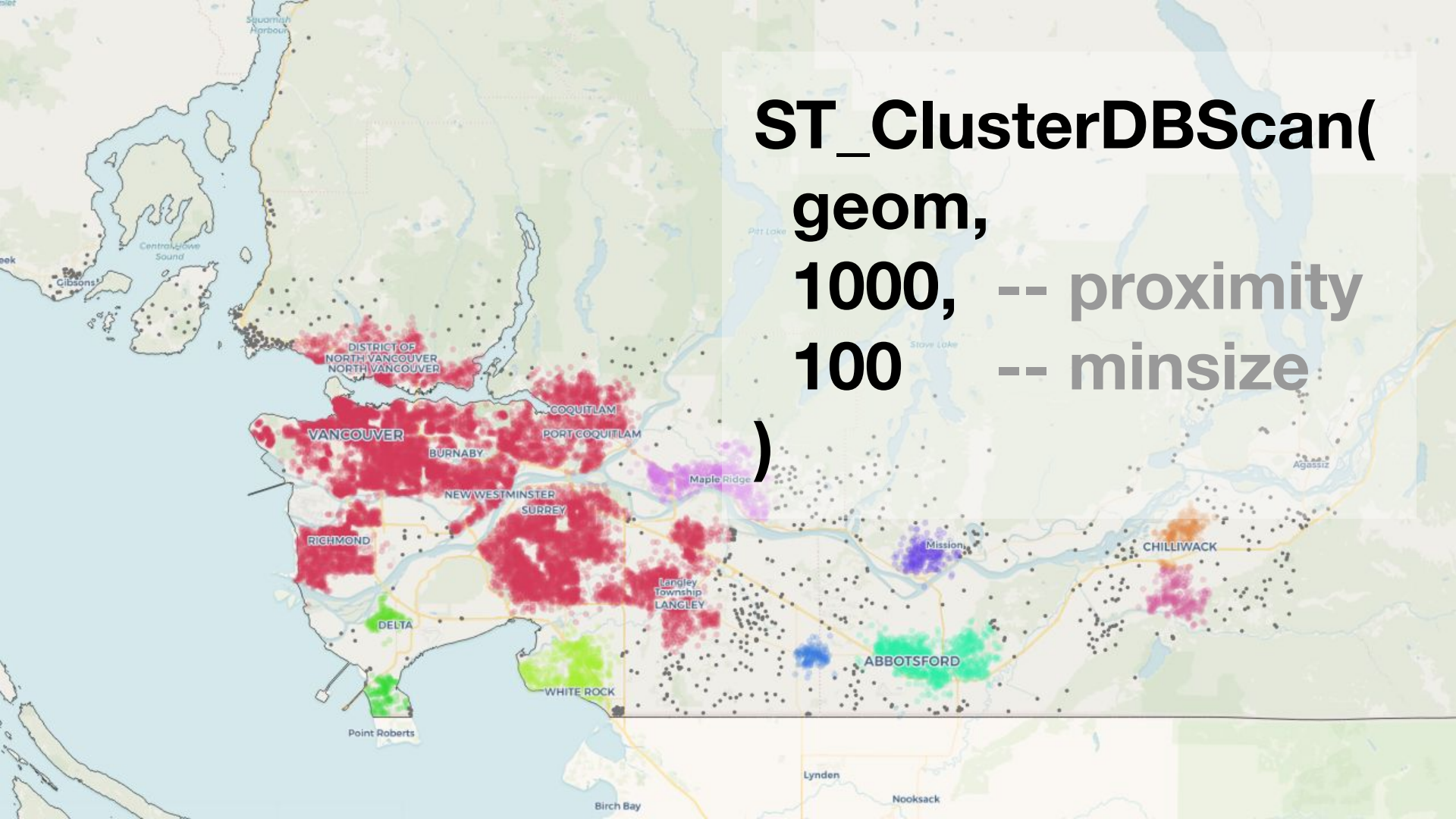


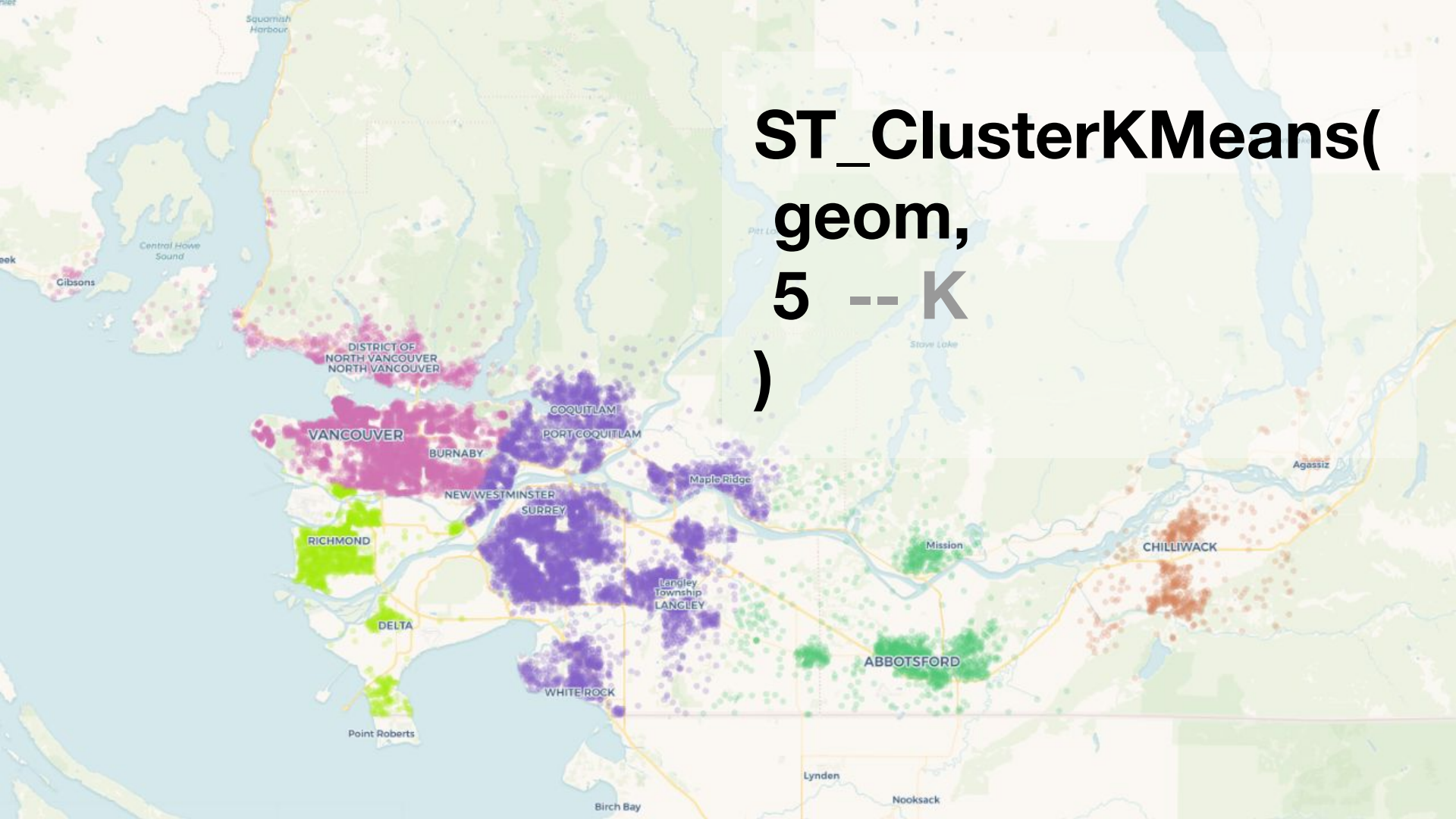
clustering!
(not nodes!)

ST_GeneratePoints(geom, n)



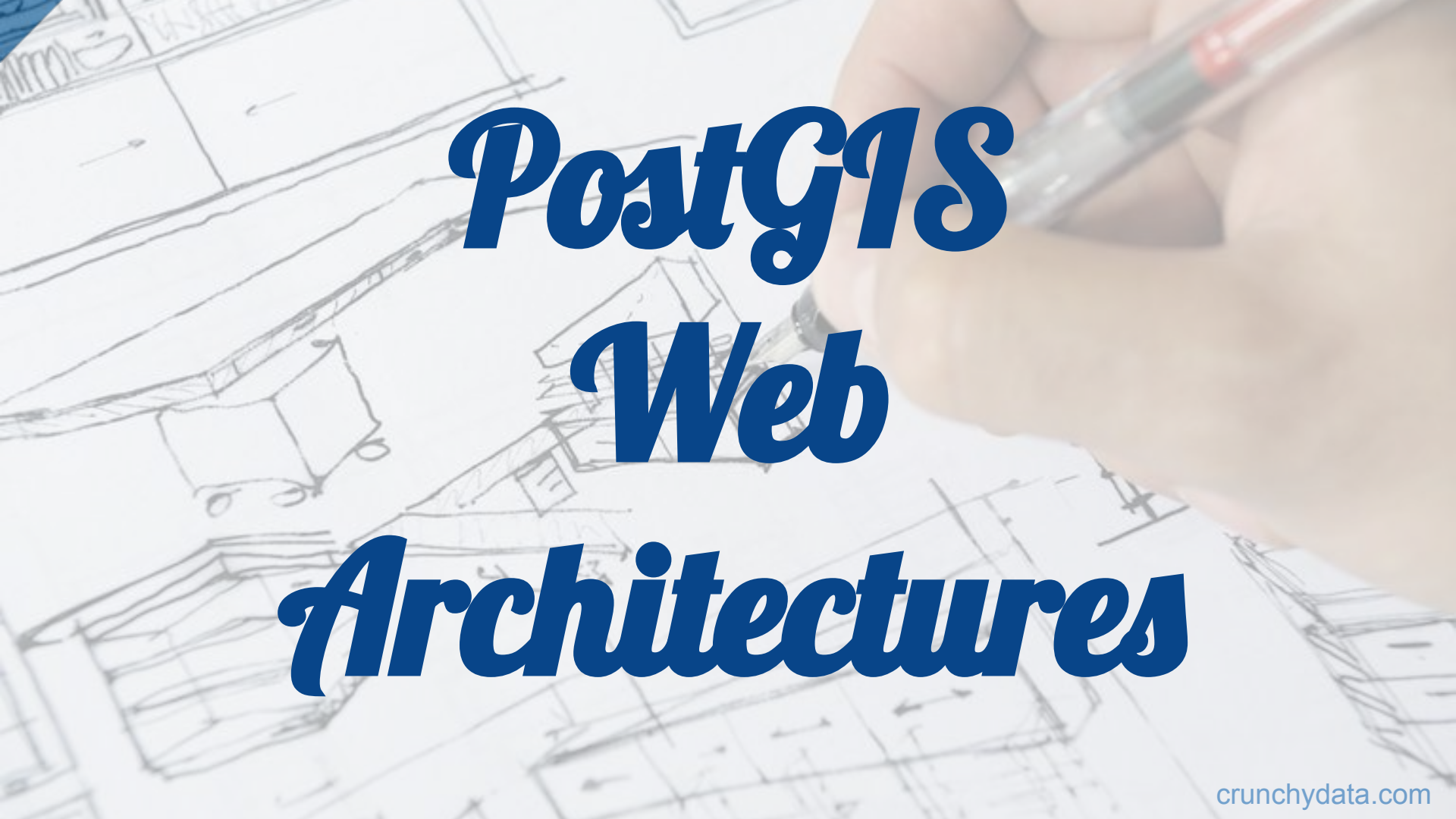

ST_GeneratePoints(
geom,
popn/100
)
DISTRICT OF NORTH VANCOUVER
NORTH VANCOUVER
COQUITLAM
VANCOUVER
PORT COQUITLAM
BURNABY
Maple Ridge
NEW WESTMINSTER
SURREY
RICHMOND
Mission
CHILLIWACK
Langley Township
LANGLEY
DELTA
ABBOTSFORD
WHITE ROCK
Point Roberts
Lynden
Nooksack
Birch Bay
Agassiz
Gibsons
Squamish Harbour
Pitt Lake


ST_ClusterDBScan(
geom,
1000, -- proximity
100 -- minsize
)
DISTRICT OF NORTH VANCOUVER
NORTH VANCOUVER
VANCOUVER
BURNABY
COQUITLAM
PORT COQUITLAM
NEW WESTMINSTER
SURREY
RICHMOND
DELTA
WHITE ROCK
Point Roberts
Langley Township
LANGLEY
Maple Ridge
Mission
CHILLIWACK
ABBOTSFORD
Gibsons
Lynden
Nooksack
Birch Bay


ST_ClusterKMeans(
geom,
5 -- K
)
Squamish Harbour
Central Howe Sound
Gibsons
DISTRICT OF NORTH VANCOUVER
NORTH VANCOUVER
VANCOUVER
BURNABY
COQUITLAM
PORT COQUITLAM
Maple Ridge
NEW WESTMINSTER
SURREY
RICHMOND
DELTA
Langley Township
LANGLEY
WHITE ROCK
Point Roberts
ABBOTSFORD
Mission
CHILLIWACK
Agassiz
Stave Lake
Lynden
Nooksack
Birch Bay

GIS without the GIS

PostGIS Web Architectures



crunchydata.com


“Where’s the middleware?!”

Input / Output Formats

ST_AsText(geometry)

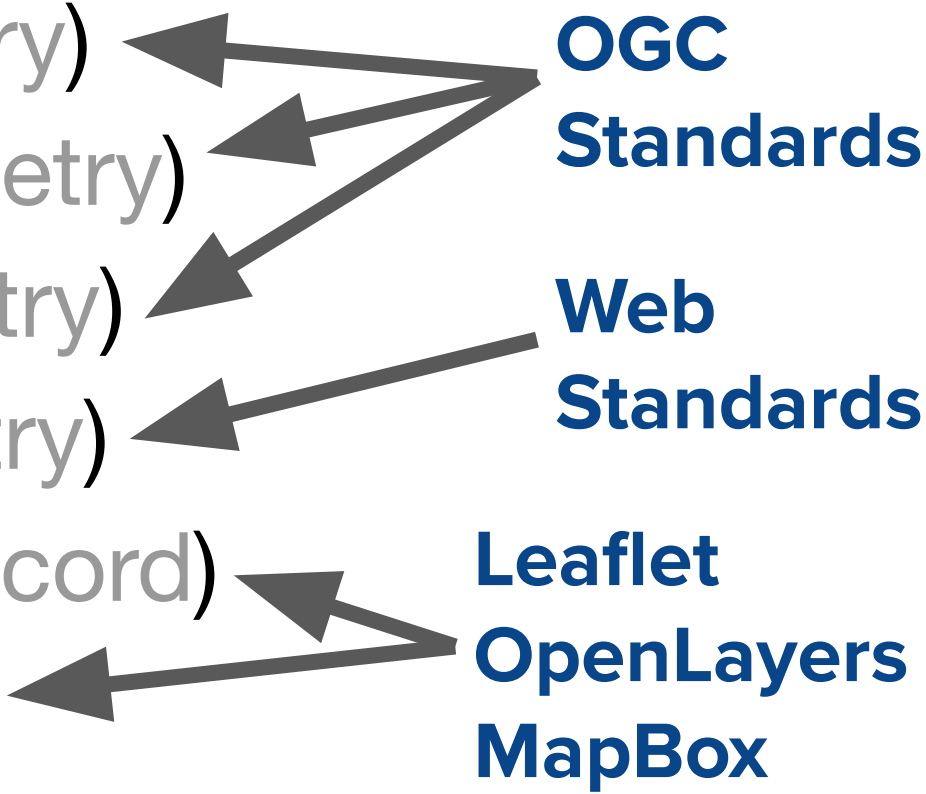
ST_AsBinary(geometry)

ST_AsGML(geometry)

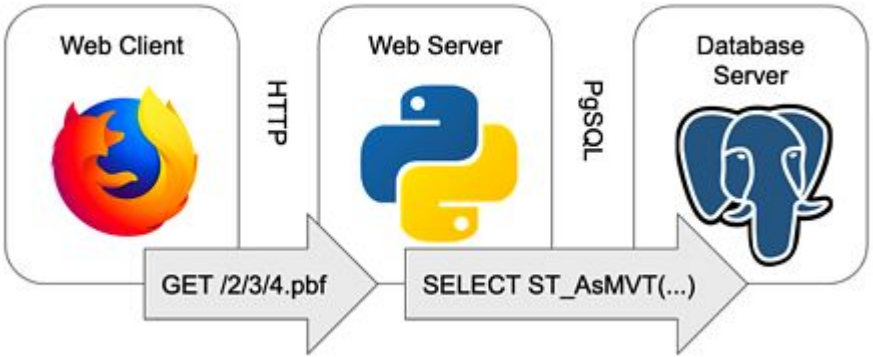
ST_AsSVG(geometry)

ST_AsGeoJSON(record)

ST_AsMVT(record)



Super Lightweight Architecture



Web Maps → **MVT**

Web Queries → Geo**JSON**

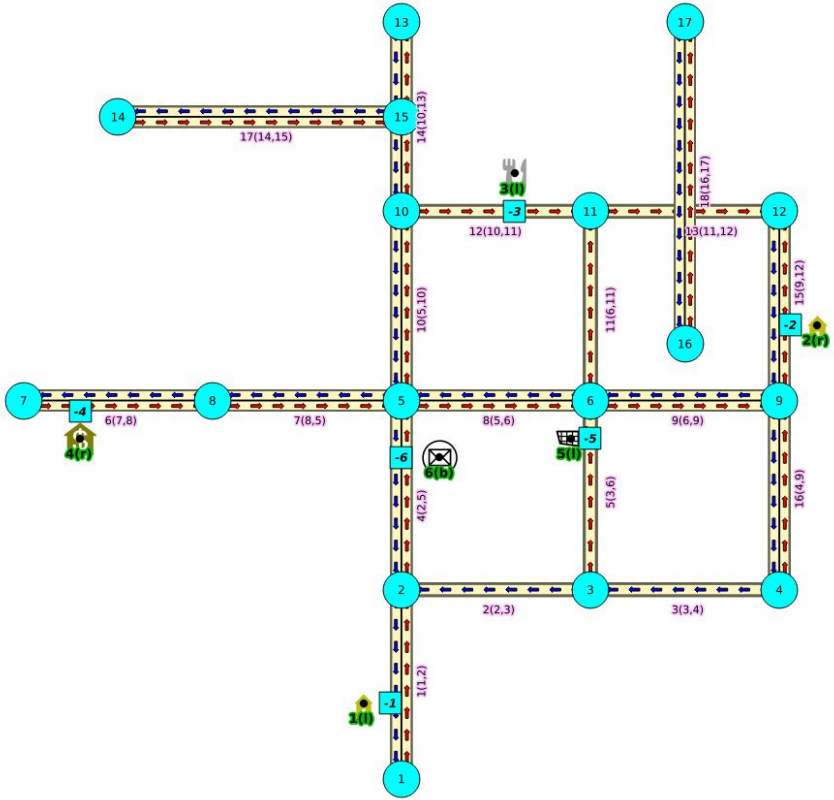
https://info.crunchydata.com/blog/dynamic-vector-tiles-from-postgis

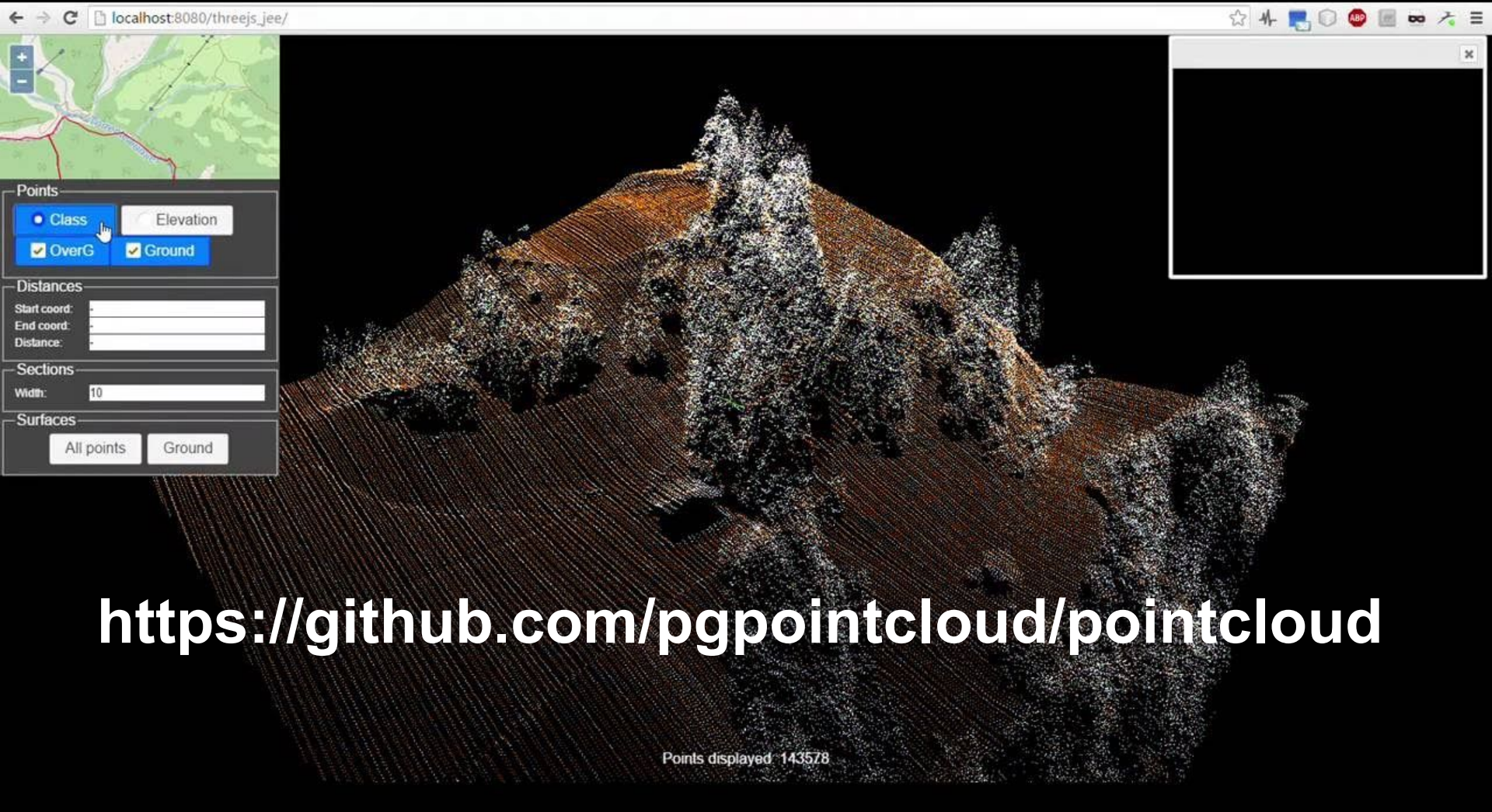
GIS without the GIS


PostGIS
PostgreSQL
Ecosystem

pgRouting

- Network **traversals**
  - Djikstra
  - A*, C*, Shooting*
  - Travelling salesman
  - Trade areas
- Build **any network** you like
  - Utility network?
  - Edge weights, dynamic edge removal, etc



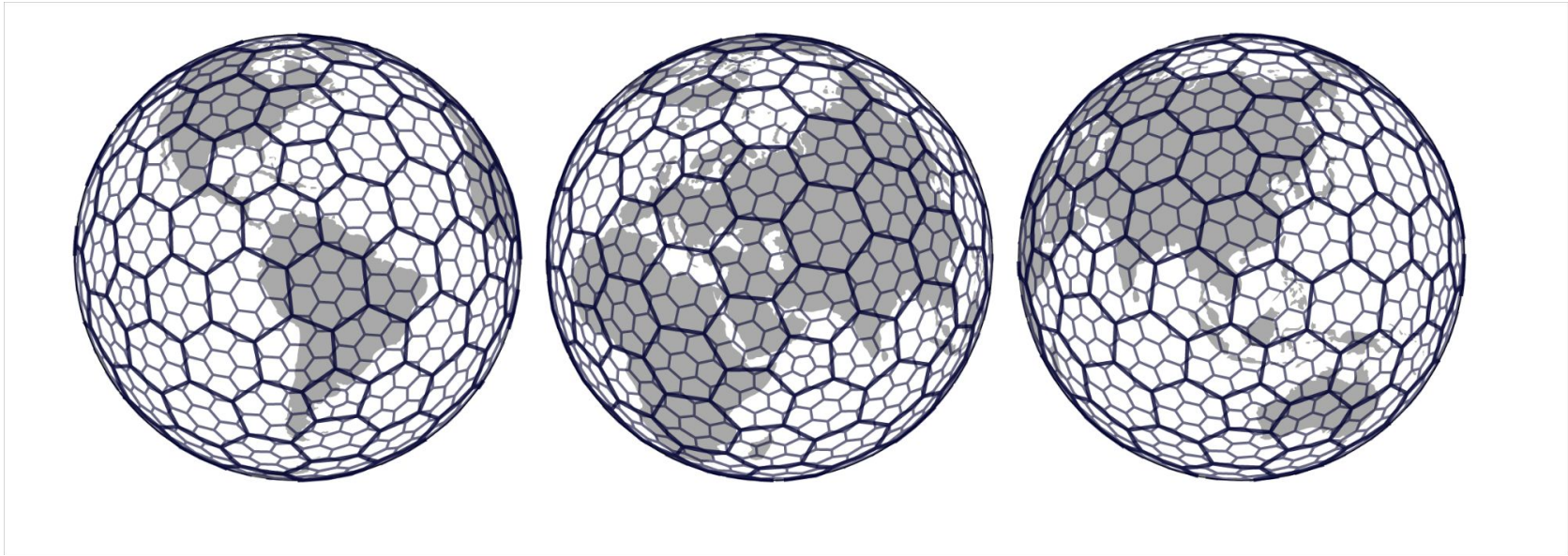


https://github.com/pgpointcloud/pointcloud

PostGIS TIGER Geocoder

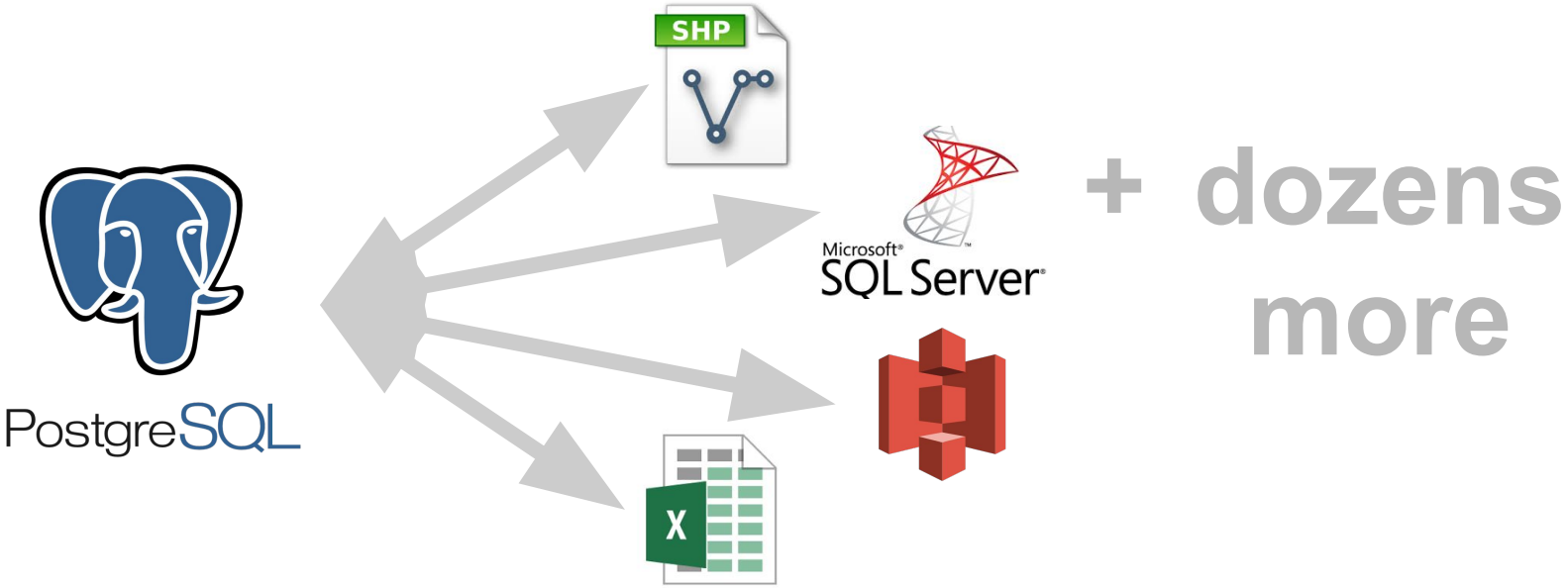
- Forward **Geocoder**
  - Input is an address **string**
  - Output is a address **location** point
- Uses **US Census** data
  - Limited to US states and territories
- **Included** in PostGIS
  - `create extension postgis_tiger_geocoder`
  - `create extension address_standardizer`

Uber H3



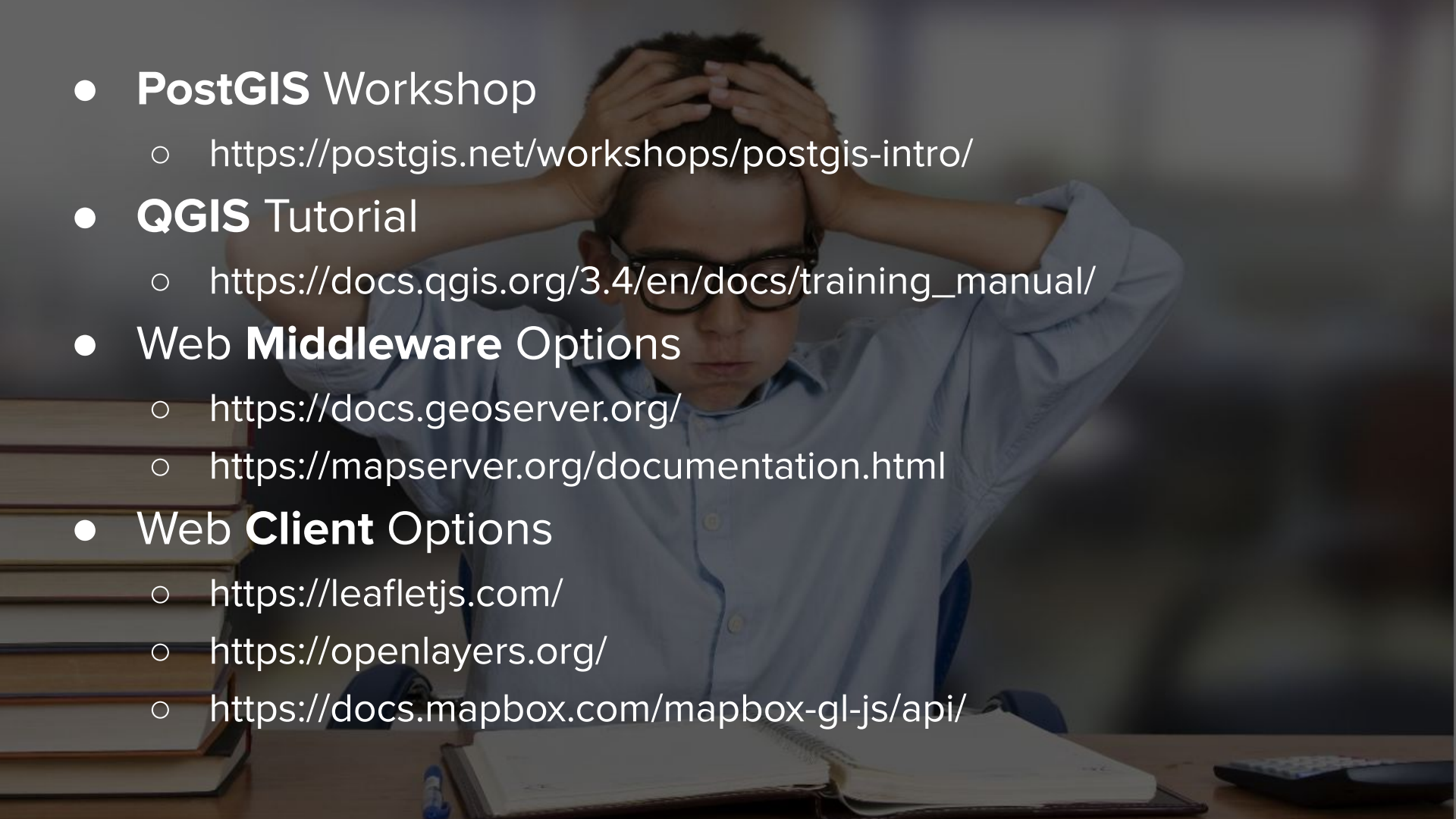
https://github.com/dlr-eoc/pgh3

ogr_fdw



https://github.com/pramsey/pgsql-ogr-fdw


Learning
More



- **PostGIS** Workshop
  - https://postgis.net/workshops/postgis-intro/
- **QGIS** Tutorial
  - https://docs.qgis.org/3.4/en/docs/training_manual/
- Web **Middleware** Options
  - https://docs.geoserver.org/
  - https://mapserver.org/documentation.html
- Web **Client** Options
  - https://leafletjs.com/
  - https://openlayers.org/
  - https://docs.mapbox.com/mapbox-gl-js/api/

https://bit.ly/2JYfqCA

Everything About PostGIS

paul.ramsey@crunchydata.ca