CARTO at Universitat de Girona SIGTE

June 15th, 11.45AM - 13.45PM (CEST)
Universitat de Girona, Spain

WORKSHOP

Spatial indices: A new paradigm for working with big data

Pedro-Juan Ferrer Matoses
Lead GIS Developer

Cayetano Benavent Viñuales
Head of Data Engineering

https://tinyurl.com/CARTOSIJSLG23

What's the biggest dataset you have ever seen?

- Canada 2nd biggest country in the world
- Canada has 9 984 670 sq km (~20x Spain)
- Covered in 30 m side cells
    - 11 094 077 777 of rows

How do we manage this?

- Why not Geography
  - We CAN store 11 094 077 777 small squares
    - But accessing back is SLOW and EXPENSIVE
  - Using that for Geo Ops is SLOW
    - We get TIMEOUT after 6h top

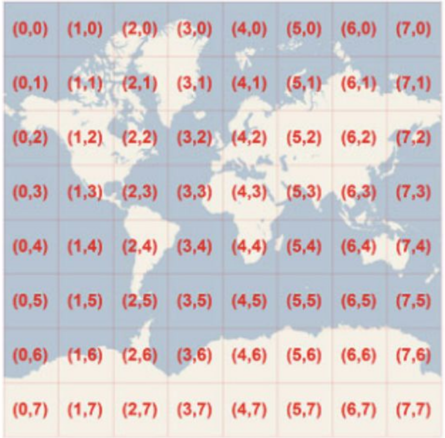
- The answer is changing the paradigm

Discrete Global Grid (DGG)

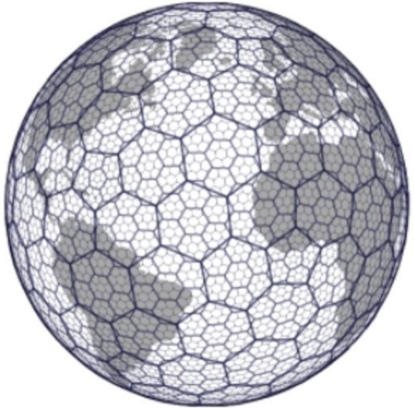
- Mosaic covering the entire Earth's surface.
- DGG is a space partitioning, so subsets are disjoint and no empty space exists.
- Could be regular or irregular. We are interested in regular DGG.

What are Spatial Indexes?

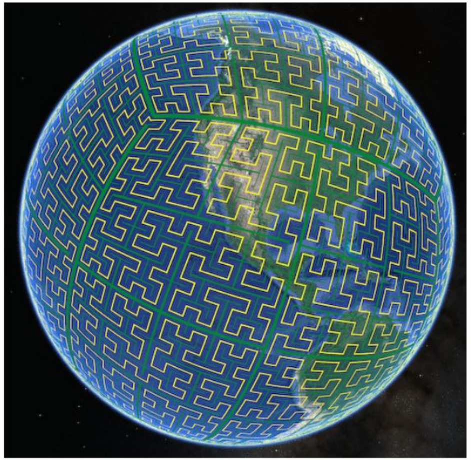
Cover the world with a UNIQUE geometry system



Quadkey



Uber's H3



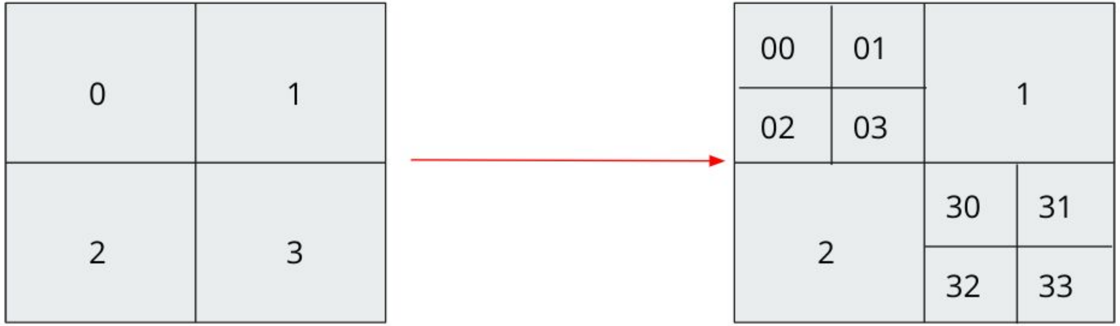
S2

Each and every portion of the earth is covered by a geometry with a Unique ID

Quadkey

- It’s a DGGS using EPSG:3857 as CRS.

- The spatial index, quadkey is built using a **quadtree** structure (each cell has 4 children).

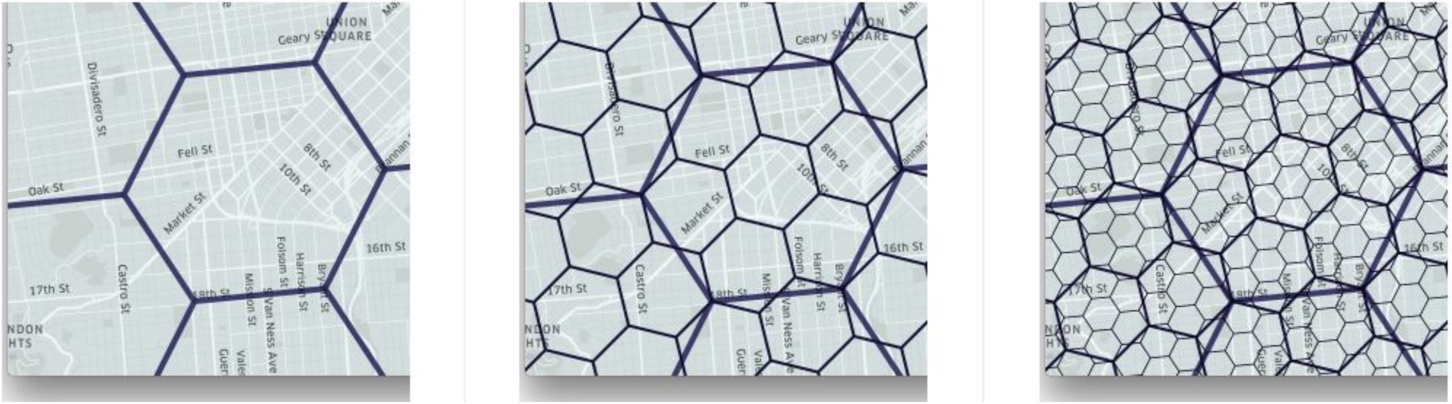


Quadkey


Level 1
0
1
2
3
Level 2
00
01
10
11
02
03
12
13
20
21
30
31
22
23
32
33
Level 3
000
001
010
011
100
101
110
111
002
003
012
013
102
103
112
113
020
021
030
031
120
121
130
131
022
023
032
033
122
123
132
133
200
201
210
211
300
301
310
311
202
203
212
213
302
303
312
313
220
221
230
231
320
321
330
331
222
223
232
233
322
323
332
333

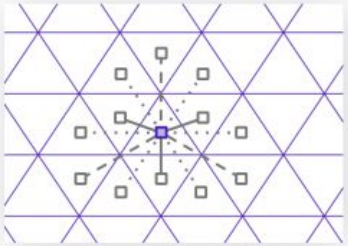
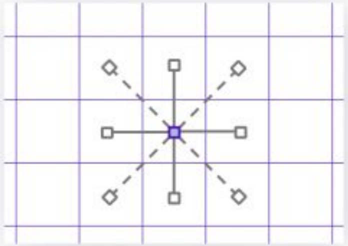
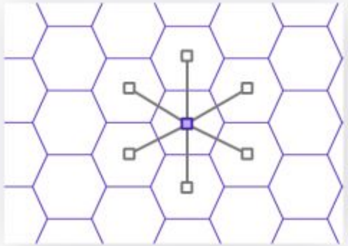
H3

- It's a DGGS using **hexagons** as cells.
- Each cell has 7 children.
- Hexagons cannot be perfectly subdivided into seven hexagons.



H3

- Interesting property for algorithms: distances to all neighbours are the same.

H3

- How H3 is built:

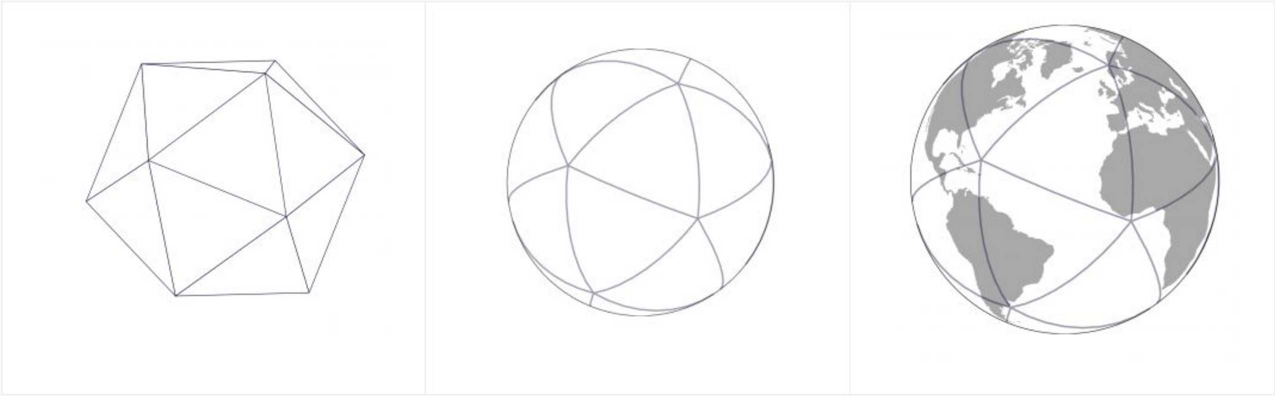
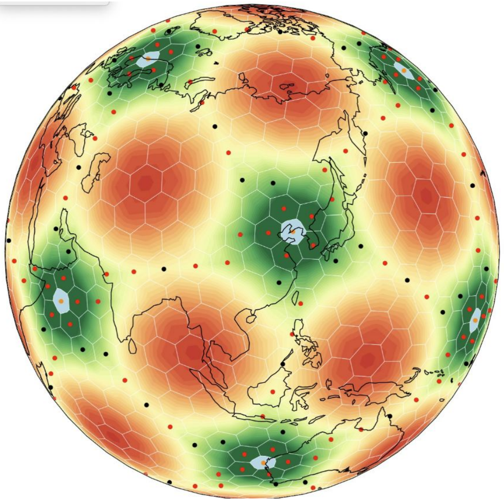


Figure 5. We chose to use gnomonic projections centered on icosahedron faces (left) for H3's map projection, projecting Earth as a spherical icosahedron (right).

H3

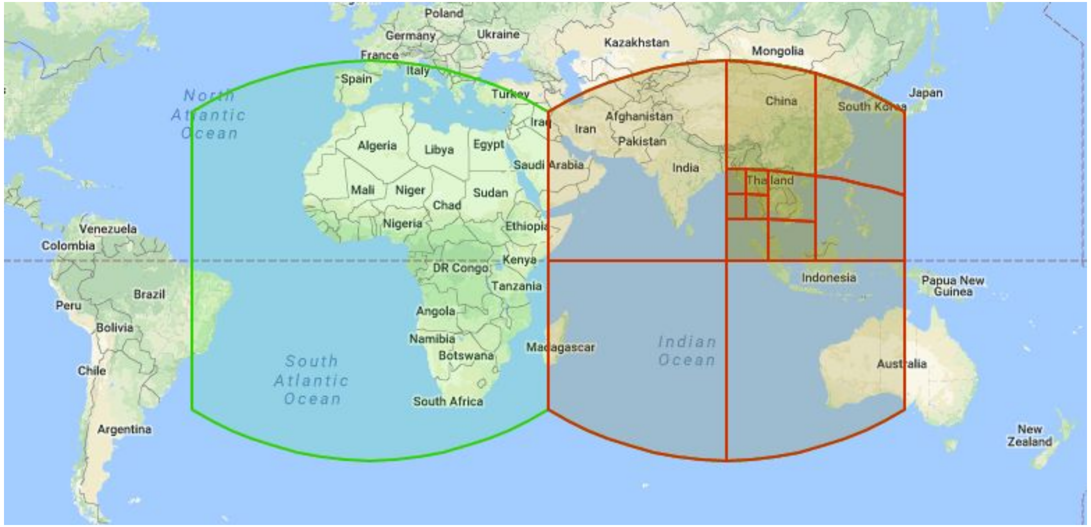
- The grid must include exactly 12 pentagons.



S2

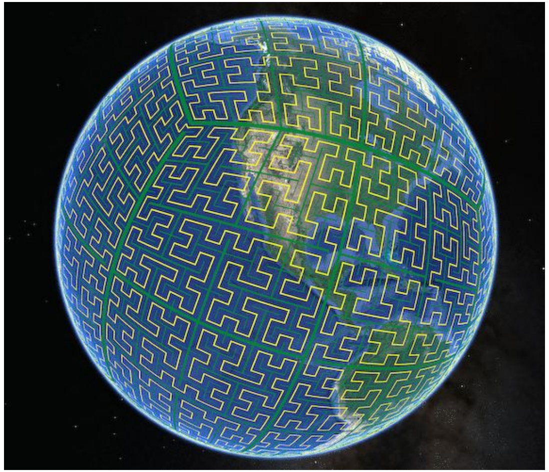
- It’s a DGGS using **squares** (spherical geometry) as cells.

- Each cell has 4 children.



S2

- How S2 is built:



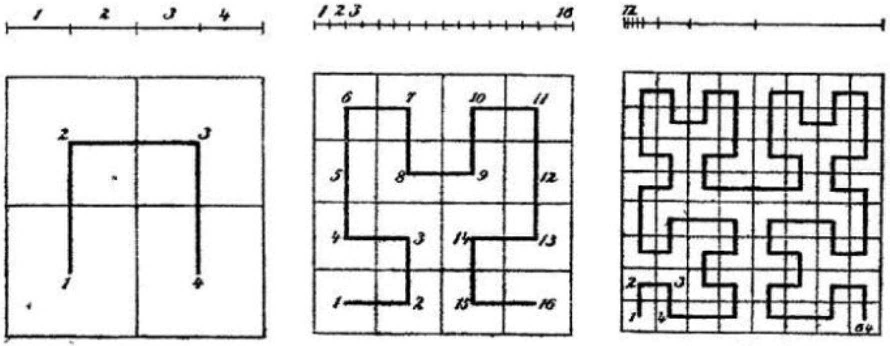
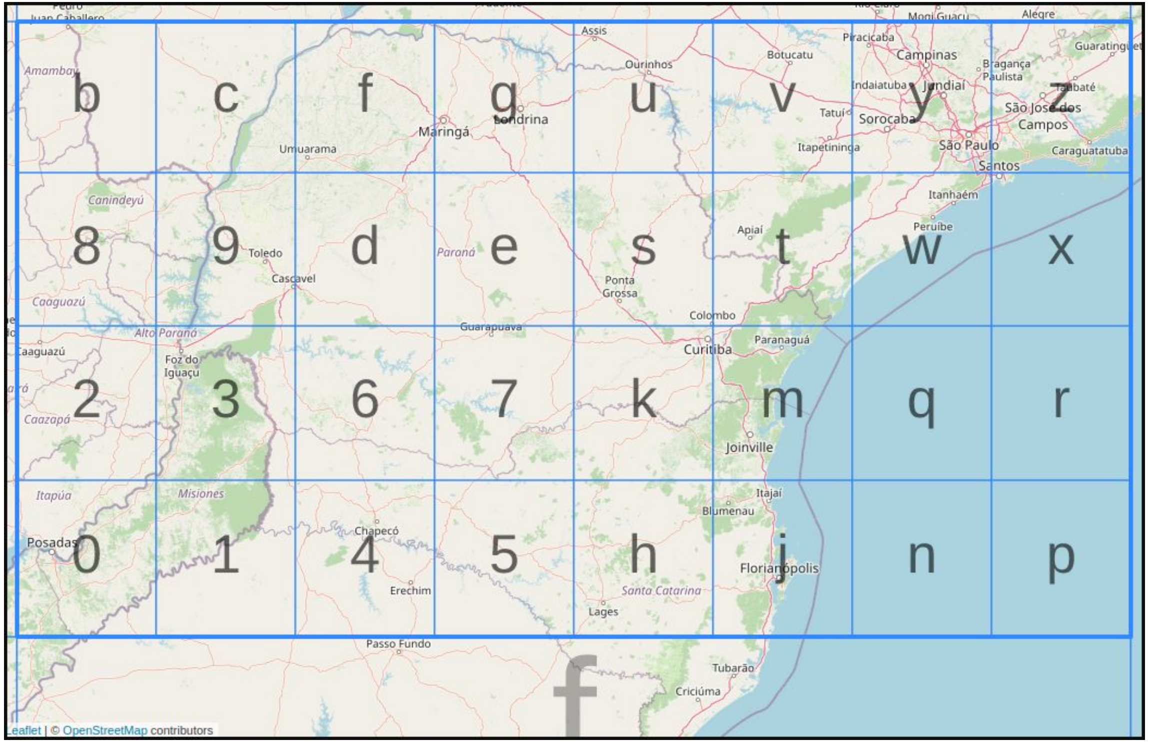
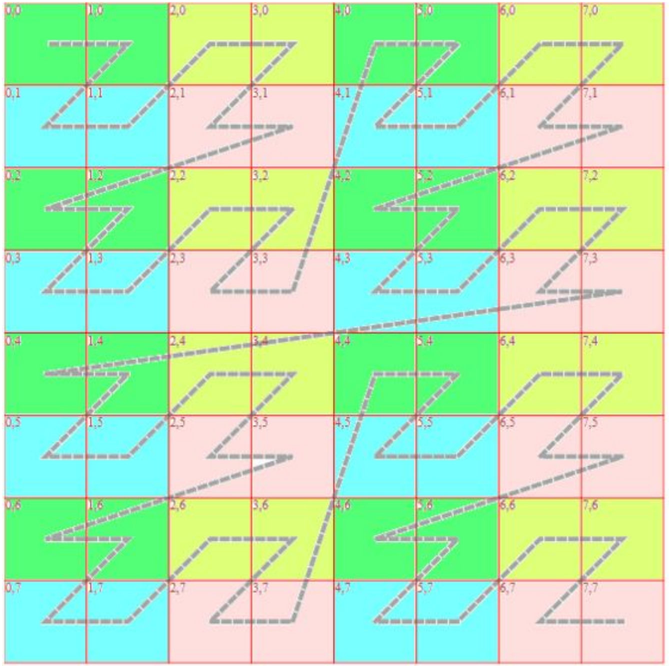


Fig. 1. Fig. 2. Fig. 3.

Geohash


b
c
f
g
u
v
y
z
8
9
d
e
s
t
w
x
2
3
6
7
k
m
q
r
0
1
4
5
h
j
n
p
Maringá
Londrina
Umuarama
Campinas
Sorocaba
São Paulo
Santos
Toledo
Cascavel
Ponta Grossa
Curitiba
Paranaguá
Foz do Iguaçu
Joinville
Blumenau
Chapecó
Posadas
Florianópolis
Lages
Passo Fundo
Tubarão
Criciúma
Leaflet | © OpenStreetMap contributors



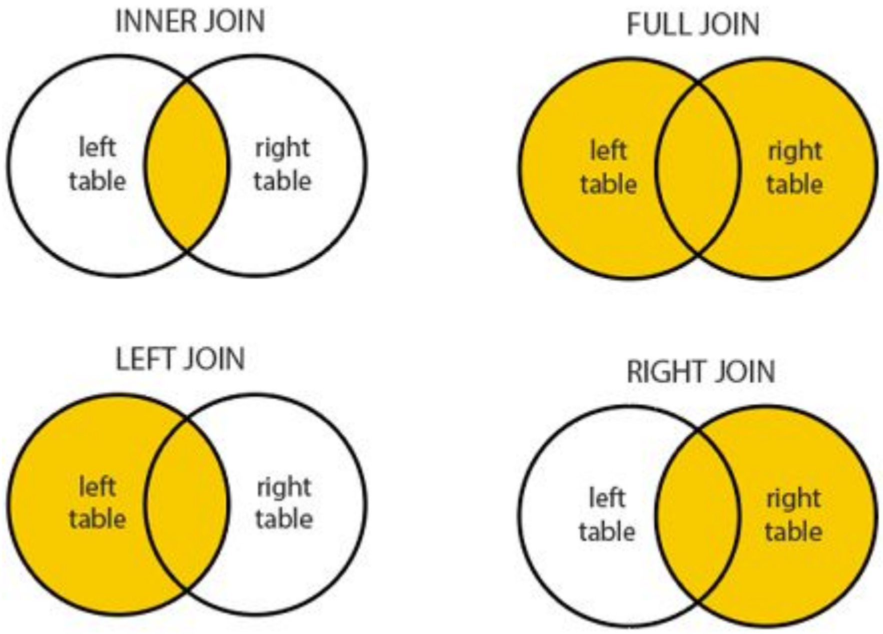
What are spatial indexes?

As every portion of the Earth is covered with a *unique* ***geometry*** with an *unique* ***id***

If you keep the id, you can **forget** about geometry

(If you have the proper tools to rebuild the geom from the id)

Use traditional SQL to perform Queries



Some PROS & CONS

- **H3**
  - Pro:
    - Best for distances
  - Con:
    - Parent/children non-inclusion
- **S2**
  - Pro:
    - Geodetically accurate
  - Con:
    - Computationally hard

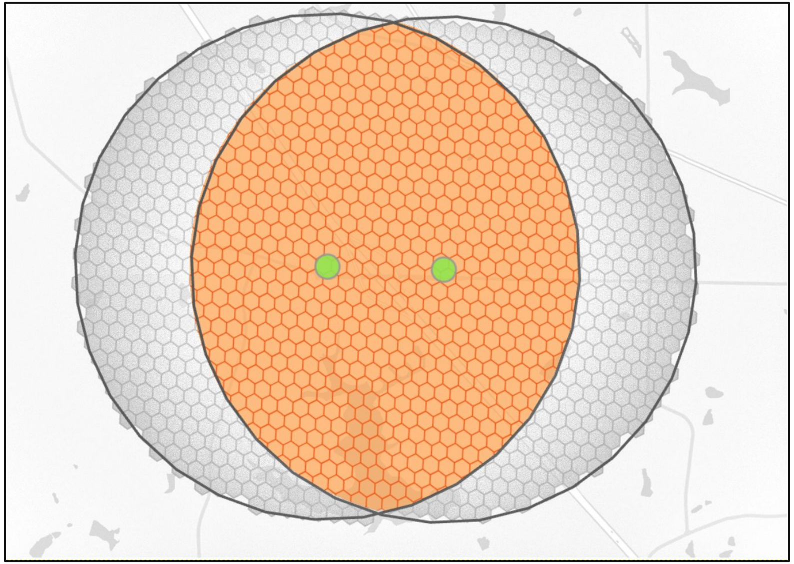
- **Quadkey**
  - Pro:
    - Algorithmically simpler
    - Parent/children inclusion
  - Con:
    - Area variation with Latitude
    - Distance distortion

Geometric intersection

Different approach

Convert the geometries problem into a numeric problem.

It’s faster and cheaper.



Geometric intersection

Q: *What's the city area covered by the anthena?*

A: -- Geometric approach:

```
SELECT
 ST_Intersection(a.geom, c.geom)
FROM city_boundary AS c
JOIN anthena_coverage AS a
ON ST_Intersects(a.geom, c.geom);
```

-- Spatial Indices approach:

```
SELECT
  a.h3
FROM city_boundary AS c
JOIN anthena_coverage AS a
ON a.h3 = c.h3;
```

Spatial join

Spatial Join

Q: *Which subway stations are there in each neighborhood?*

A: -- Geometric approach:

```
SELECT
  s.name AS subway_name,
  n.name AS neighborhood_name
FROM nyc_neighborhoods AS n
JOIN nyc_subway_stations AS s
ON ST_Contains(neighborhoods.geom, subways.geom);
```

-- Spatial Indices approach:

```
SELECT
  s.name AS subway_name,
  n.name AS neighborhood_name
FROM nyc_neighborhoods AS n
JOIN nyc_subway_stations AS s ON n.h3 = s.h3;
```

Point in polygon

Point in Polygon

Q: *Add the neighborhood to the point table*

A: -- Geometric approach:

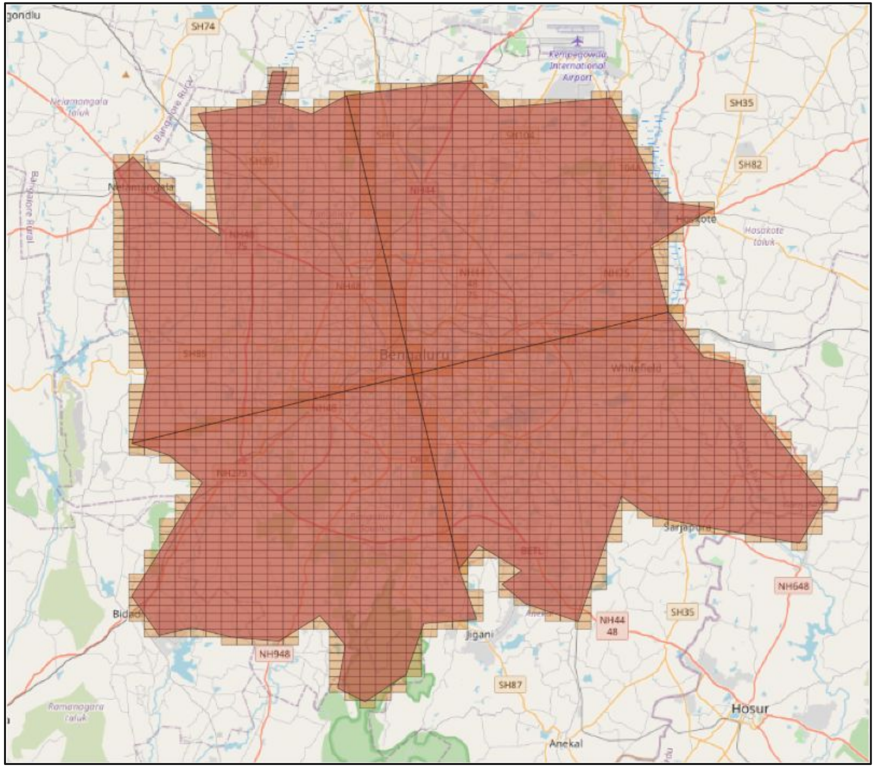
```
SELECT
 p.*,
 n.name AS neighborhood_name
FROM points_table p,nyc_neighborhoods n
WHERE
 n.geom && p.geom AND ST_Contains(n.geom, p.geom);
```

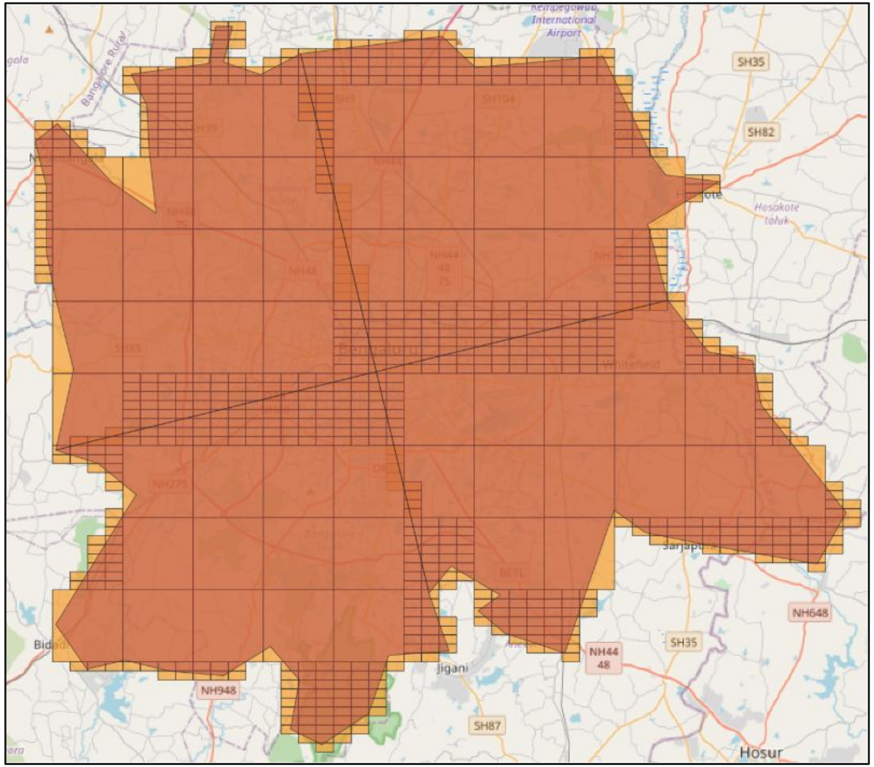
-- Spatial Indices approach:

```
SELECT
 p.*,
 n.name AS neighborhood_name
FROM points_table p,nyc_neighborhoods n
WHERE n.h3 = p.h3;
```

Parent level aggregation

a.k.a. Compactation in geohash



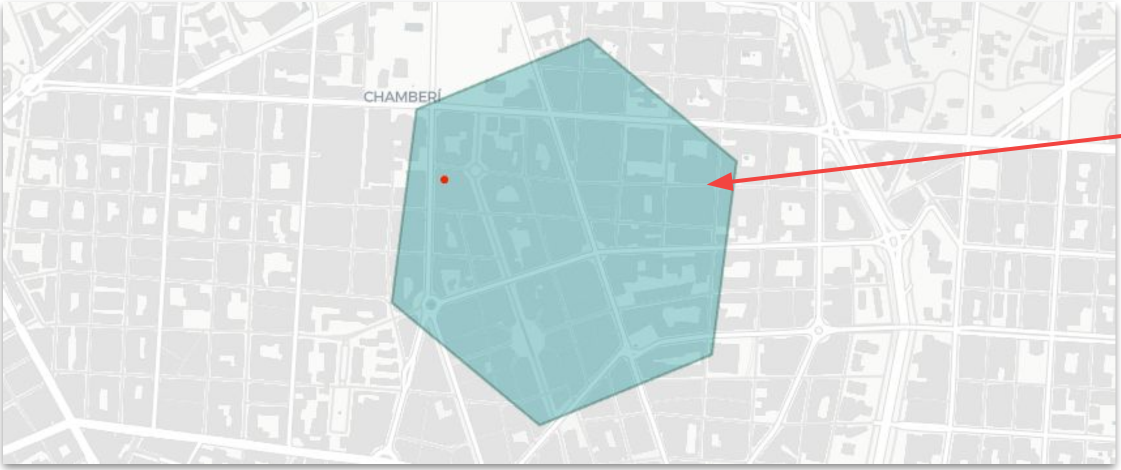


a.k.a. Compactation in H3

Other operations

Get the SI from a Point

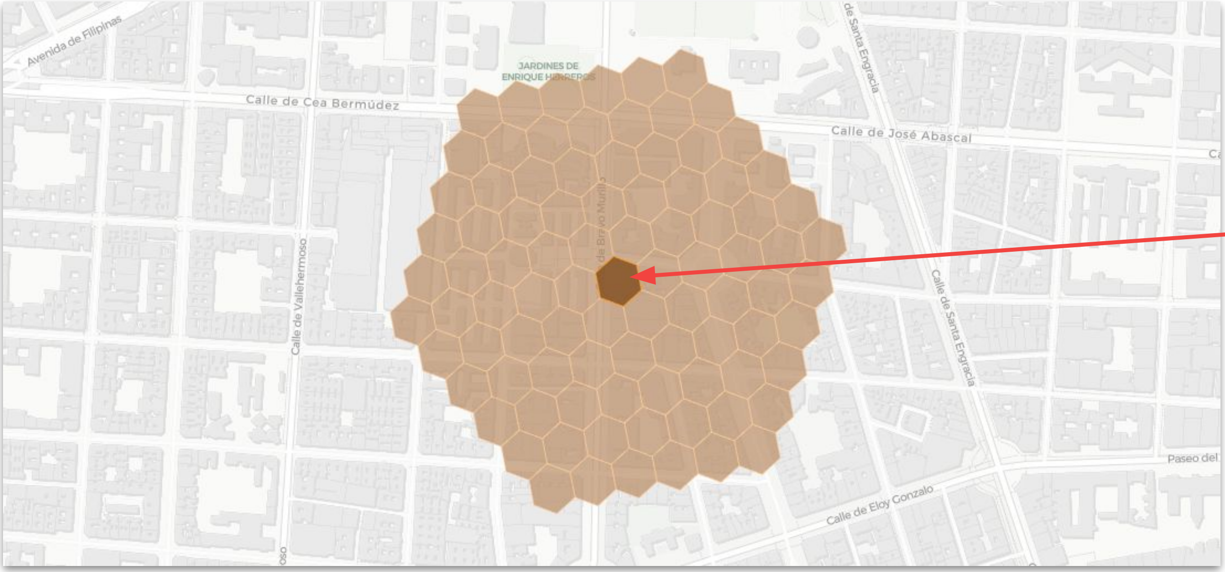
```
SELECT H3_FROMLONGLAT(-3.7038, 40.4368, 8);

SELECT H3_FROMGEOGPOINT('POINT(-3.7038 40.4368)', 8);
```



H3 with resolution 8

K-Rings: the SI buffer

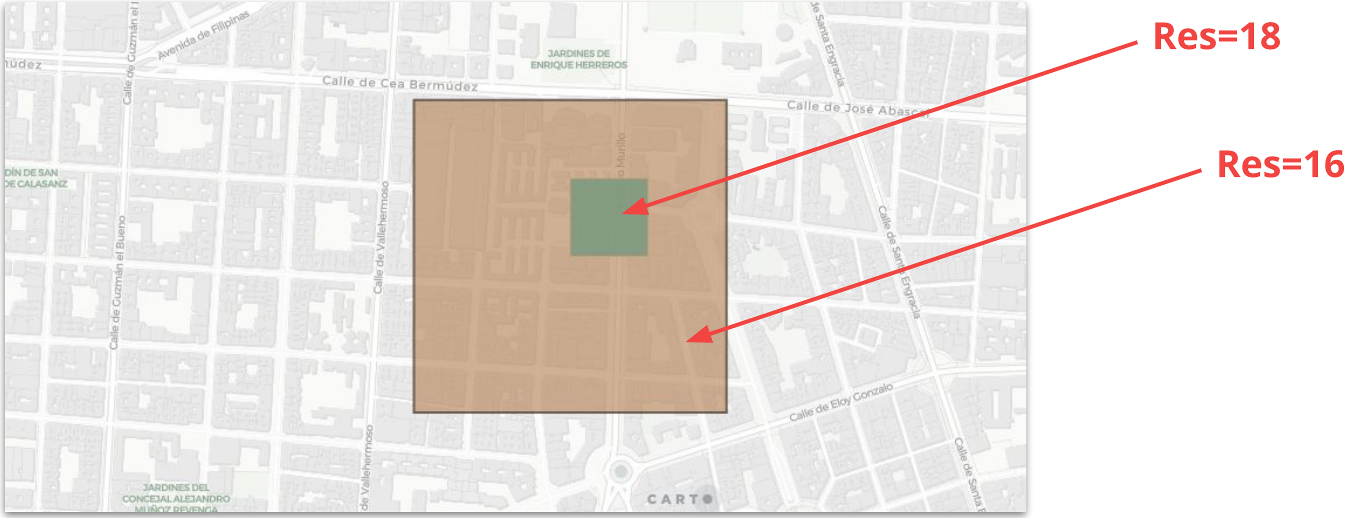
```
SELECT H3_KRING('8b390cb19516fff', 5);
```



5 rings from this H3 cell

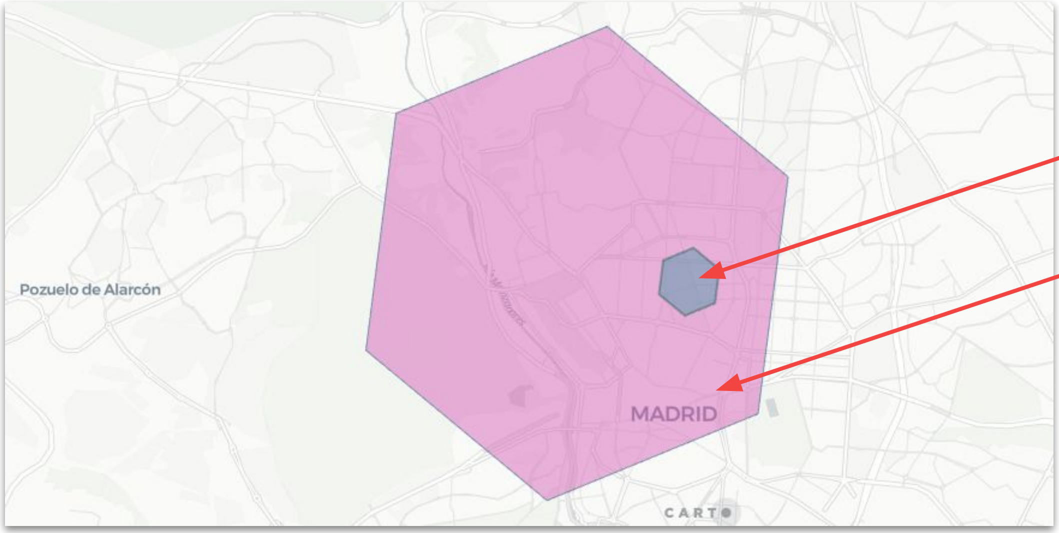
To Parent - To Children

```
SELECT QUADBIN_TOPARENT(5270290292666728447, 16);
```



To Parent - To Children

```
SELECT H3_TOPARENT('8b390cb19516fff', 6) h3;
```

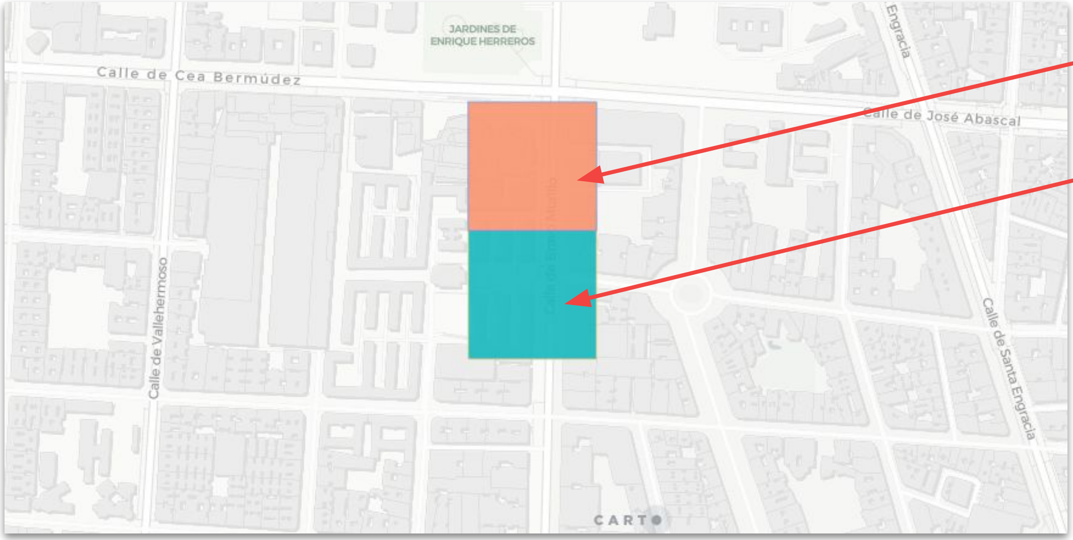


Res=8

Res=6

Neighbours

```
SELECT QUADBIN_SIBLING(5270290292666728447, 'up');
```



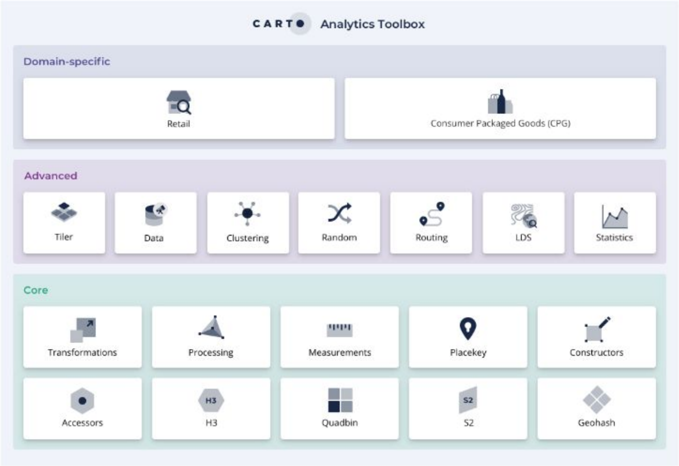
Sibling (Up)

Origin

CARTO Analytics Toolbox

SQL reference for SI functions:

https://docs.carto.com/data-and-analysis/analytics-toolbox-overview

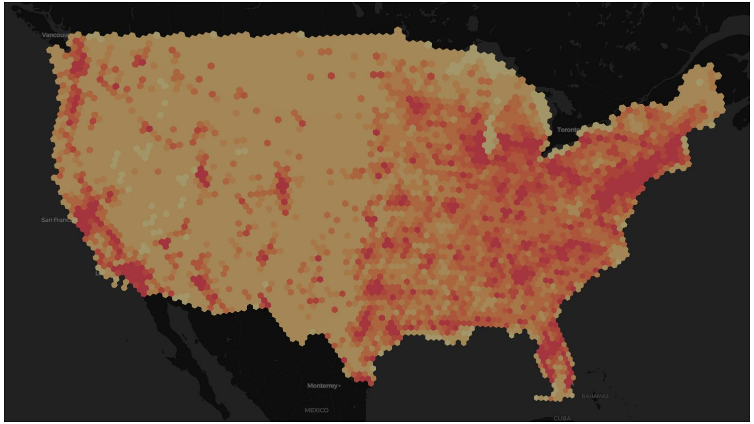


Spatial indices vs Geometries (when handling huge datasets)

	Spatial Indices	Geometries
Scalability	High, pure SQL statements are used most of the time. This harnesses the true power of DW (highly **distributed architecture**)	Low, query **cost** increases exponentially with geometry size and complexity
Query performance	High, queries work with integer/string fields	Medium, i.e. geography type works well in DW, but it's not as fast as using integer/string
Storage performance	High, **no geometries**	Low, we need to store repeated and **complex** geometries
Data Precision	Medium, as some noise may be introduced by regular grids	The original precision is kept

Spatial indices vs Geometries (when handling huge datasets)

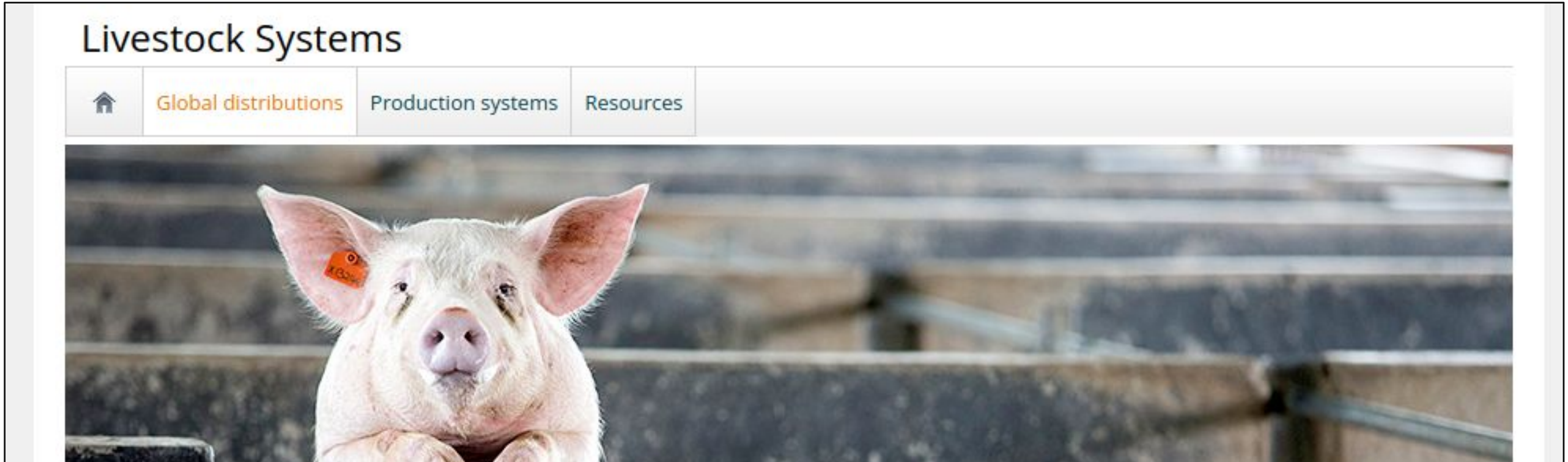
	Spatial Indices	Geometries
Visualization	Fast render. Possibility of dynamic tiling	Computationally hard



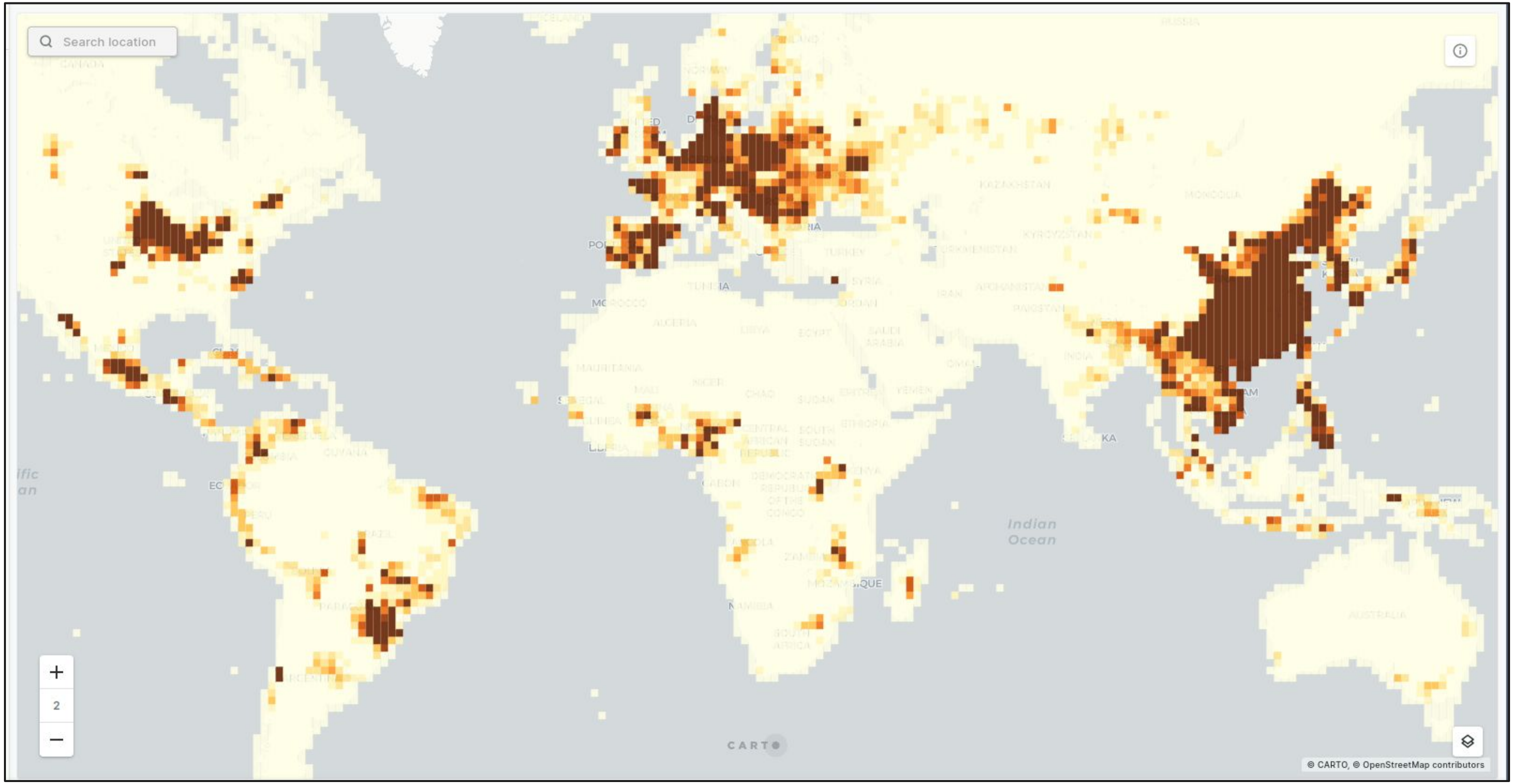
Objective: Get some SI and use them to make basic GIS ops

Get the Data: Global pigs distribution in 2010 (5 minutes of arc)

- Link to data
- CSV File (quadbin level 12)

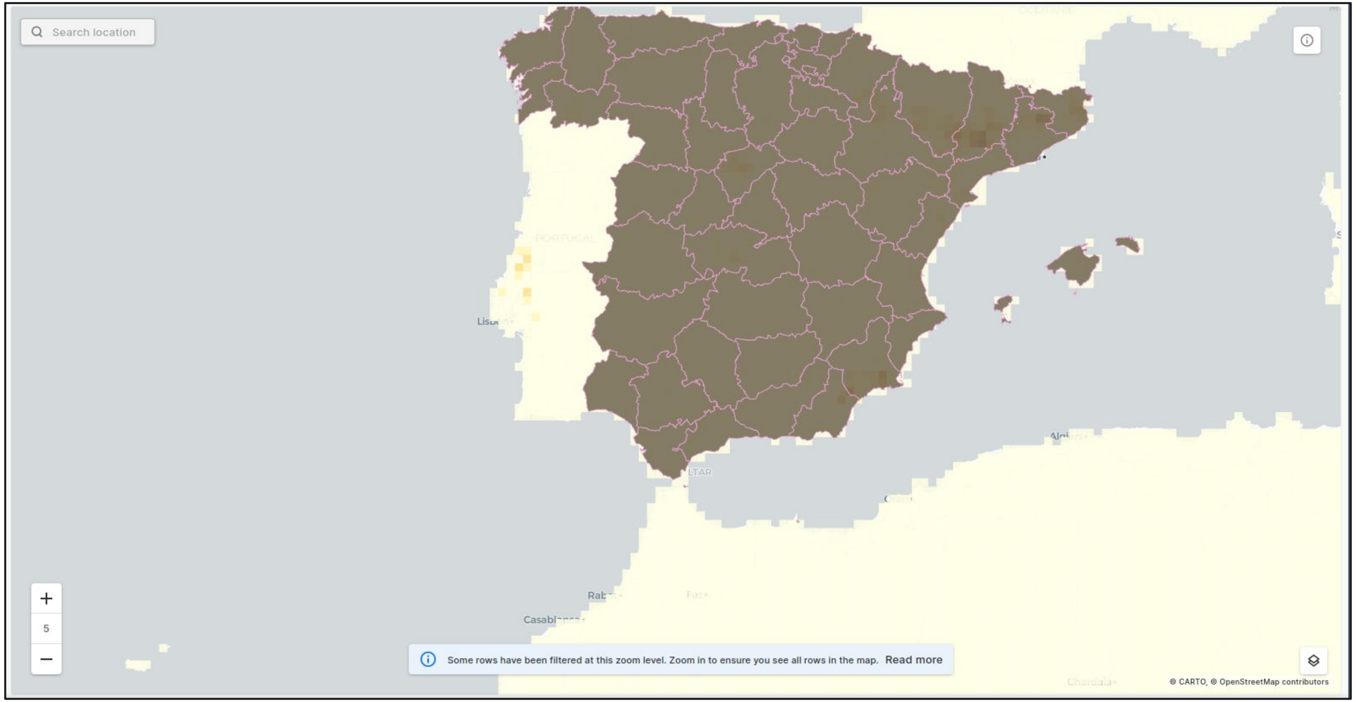


Visualization in CARTO



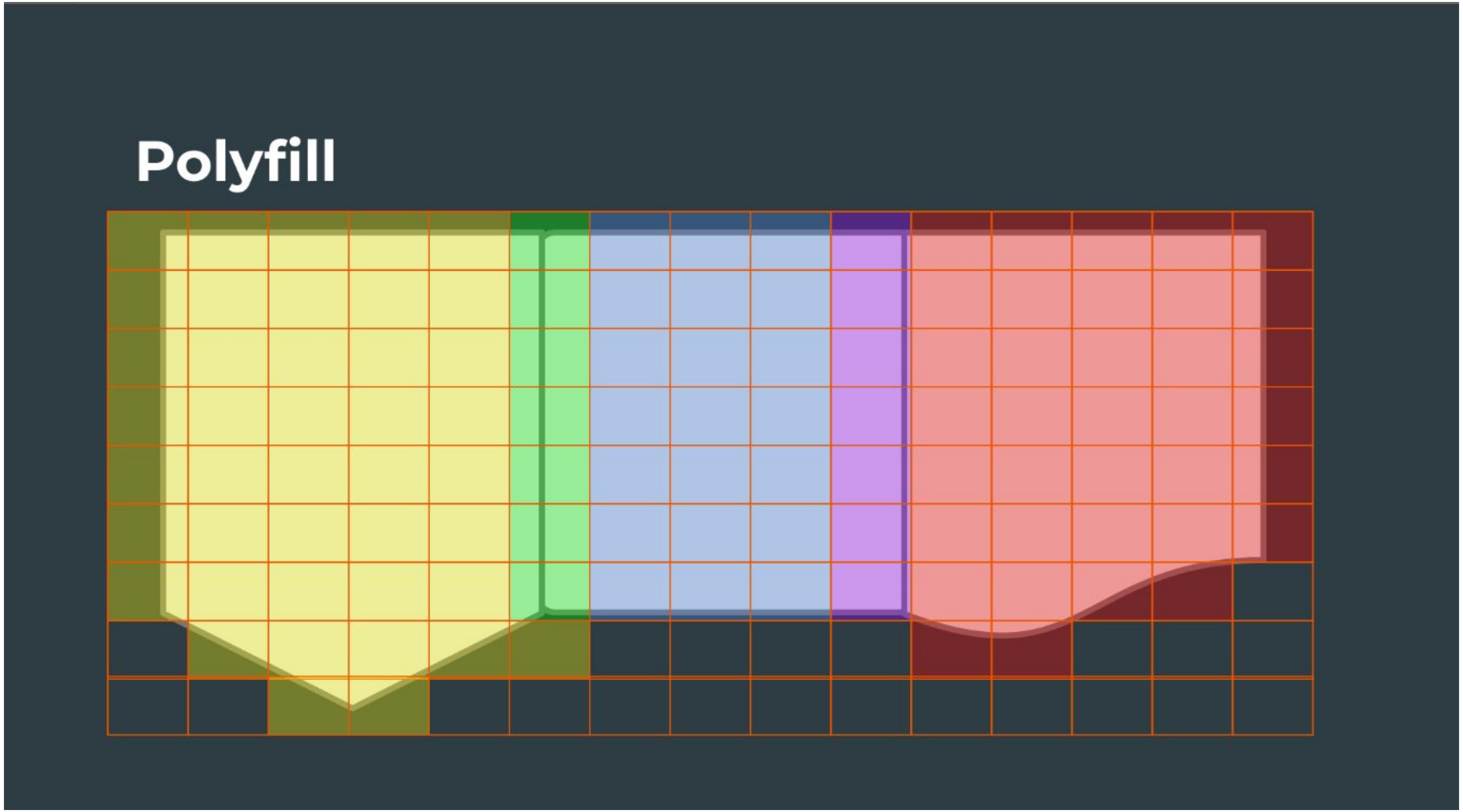
ETL: Provinces of Spain

- [Data Observatory Link to INE provinces]



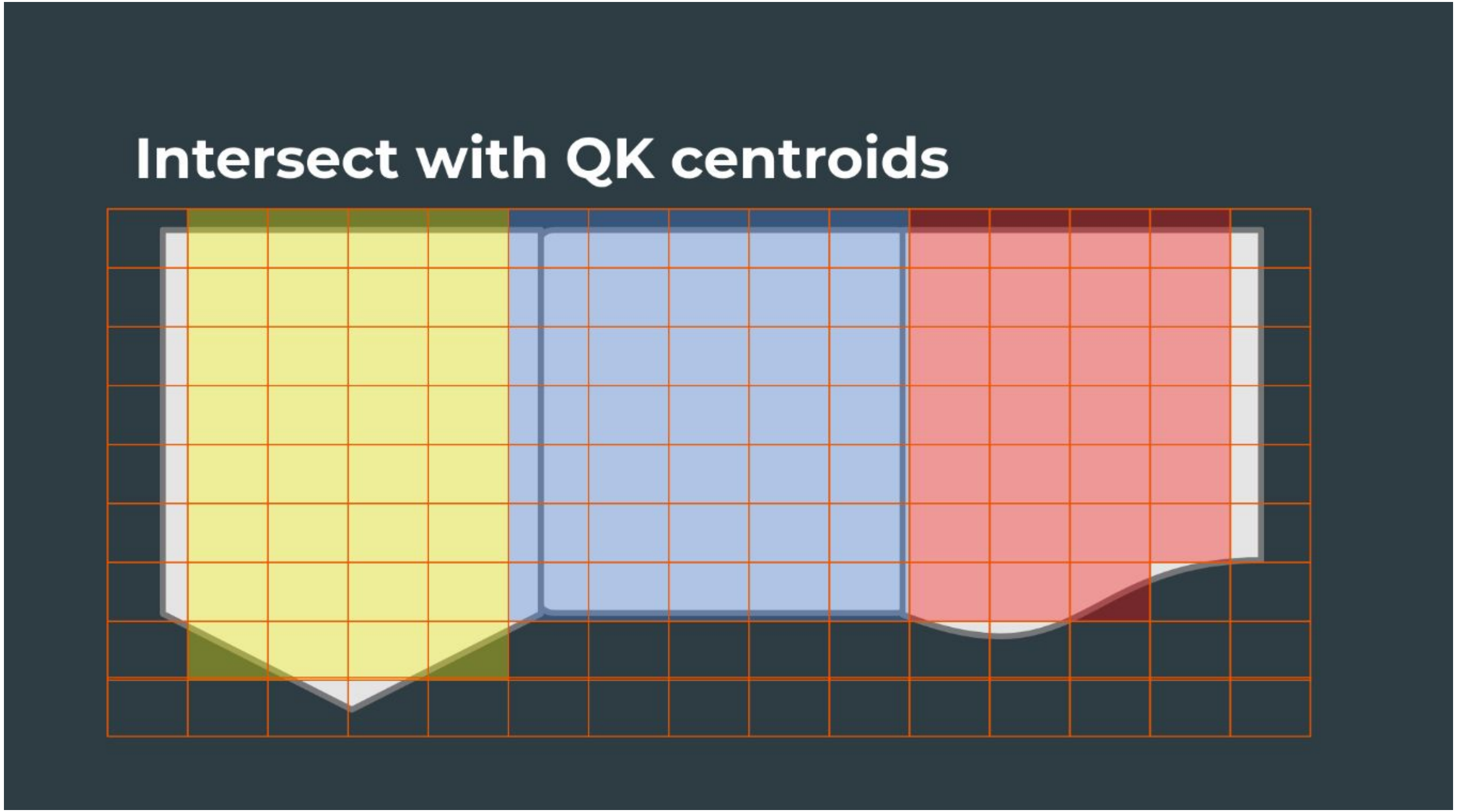
ETL: Provinces to QUADGRID

Polyfill

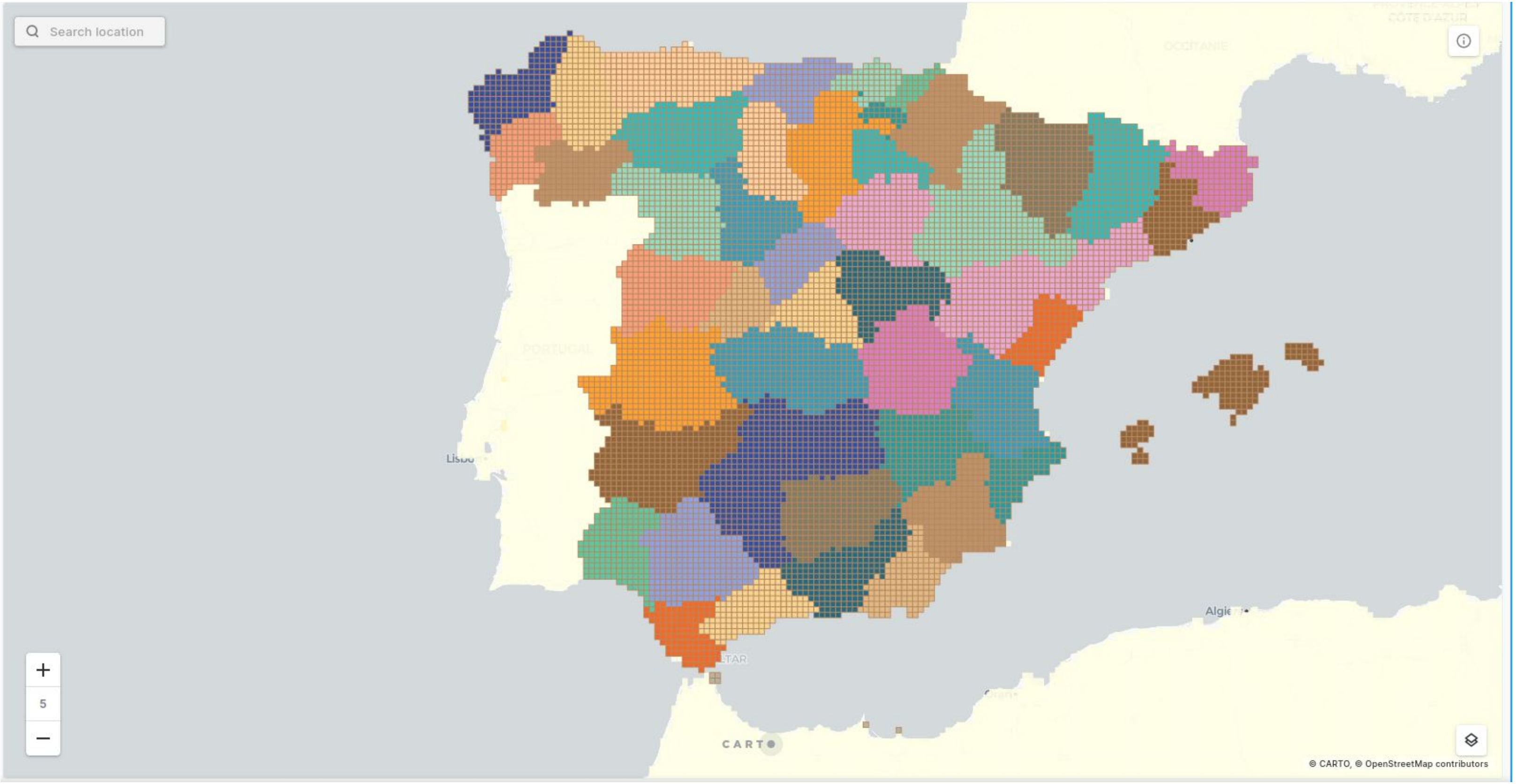


ETL: Provinces to QUADGRID

Centroids



ETL: Provinces to QUADGRID ([CSV file]() quadgrid lvl 12)



SQL to calculate total pig per province

```
SELECT codine, name, sum(value) pigs
FROM `sub_esp_ign_geography_esp_province_2020_quadgrid12_v1`
prov, `fao_pigs` pig

WHERE prov.quadbin = pig.quadbin


GROUP BY codine, name
  ORDER BY 3 DESC
```

SQL to calculate total pig per province

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	codine	name	pigs
1	25	Lleida	4440629
2	22	Huesca	3339350
3	50	Zaragoza	2702194
4	30	Murcia	2334580
5	08	Barcelona	1942871
6	40	Segovia	1376715
7	31	Navarra	1309488
8	17	Girona	1300881
9	44	Teruel	1240662
10	06	Badajoz	1184779

Objective: Rendering SI in a web application

We need a library: Deck.gl

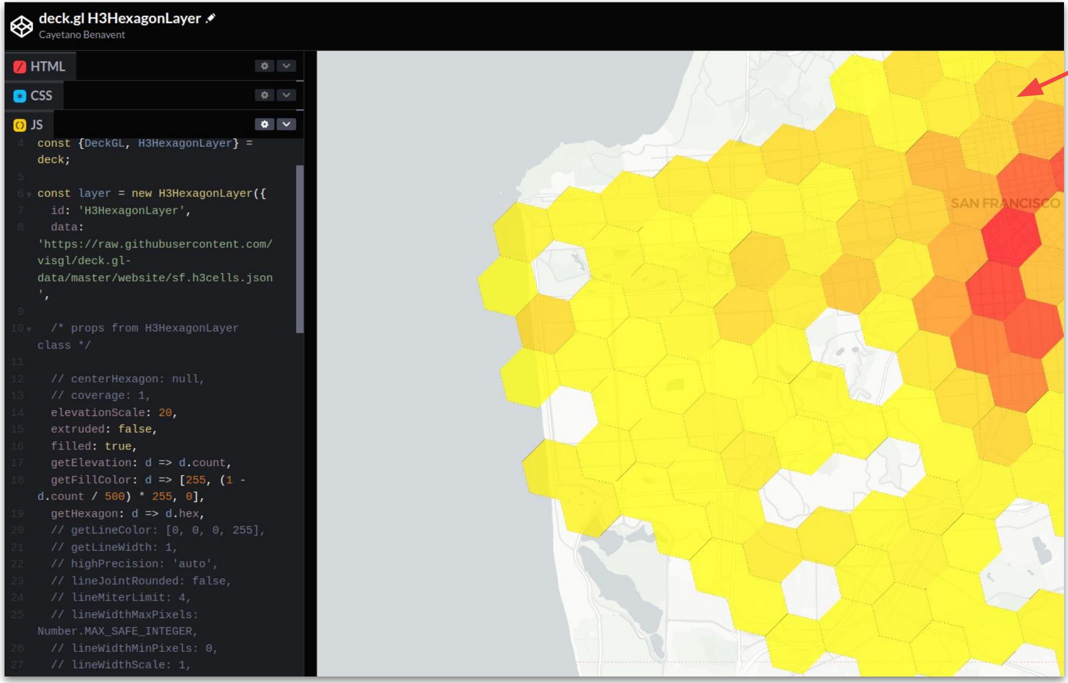
Deck.gl is an Open Source WebGL-powered framework for visual exploratory data analysis of large datasets.

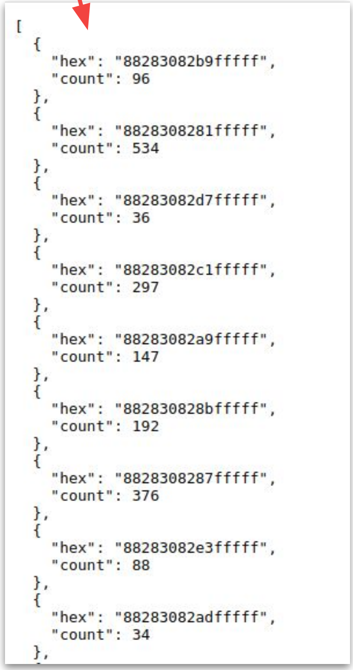
Deck.gl can render directly SI (QuadKey, H3, Geohash and S2):

- https://deck.gl/docs/api-reference/geo-layers/h3-hexagon-layer
- https://deck.gl/docs/api-reference/geo-layers/quadkey-layer
- https://deck.gl/docs/api-reference/geo-layers/s2-layer
- https://deck.gl/docs/api-reference/geo-layers/geohash-layer

Let’s see the code*

H3 data





```
[
  {
    "hex": "88283082b9fffff",
    "count": 96
  },
  {
    "hex": "8828308281fffff",
    "count": 534
  },
  {
    "hex": "88283082d7fffff",
    "count": 36
  },
  {
    "hex": "88283082c1fffff",
    "count": 297
  },
  {
    "hex": "88283082a9fffff",
    "count": 147
  },
  {
    "hex": "882830828bfffff",
    "count": 192
  },
  {
    "hex": "8828308287fffff",
    "count": 376
  },
  {
    "hex": "88283082e3fffff",
    "count": 88
  },
  {
    "hex": "88283082adfffff",
    "count": 34
  },
```


* https://deck.gl/docs/api-reference/geo-layers/h3-hexagon-layer (open codepen link)

Rendering SI from databases

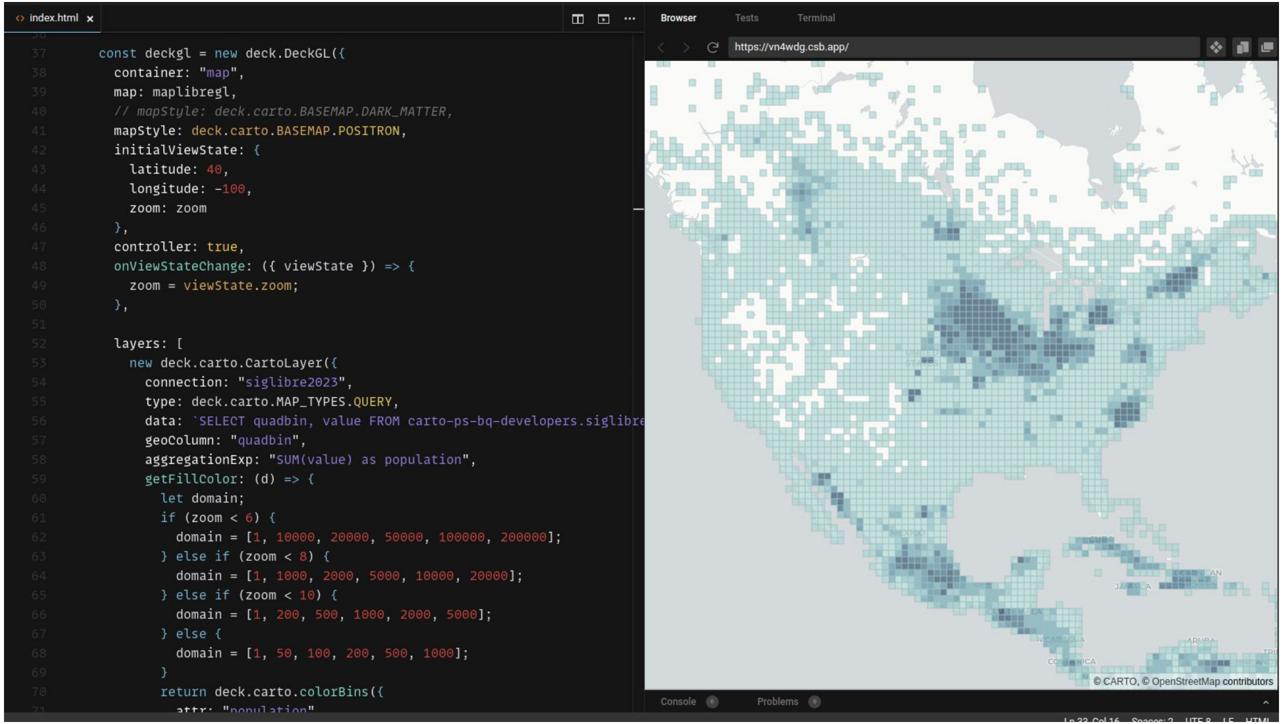
Deck.gl supports SI rendering from different databases*****
through the CARTO layer:

https://deck.gl/docs/api-reference/carto/carto-layer#spatial-index-data

https://docs.carto.com/carto-for-developers/carto-for-deck.gl

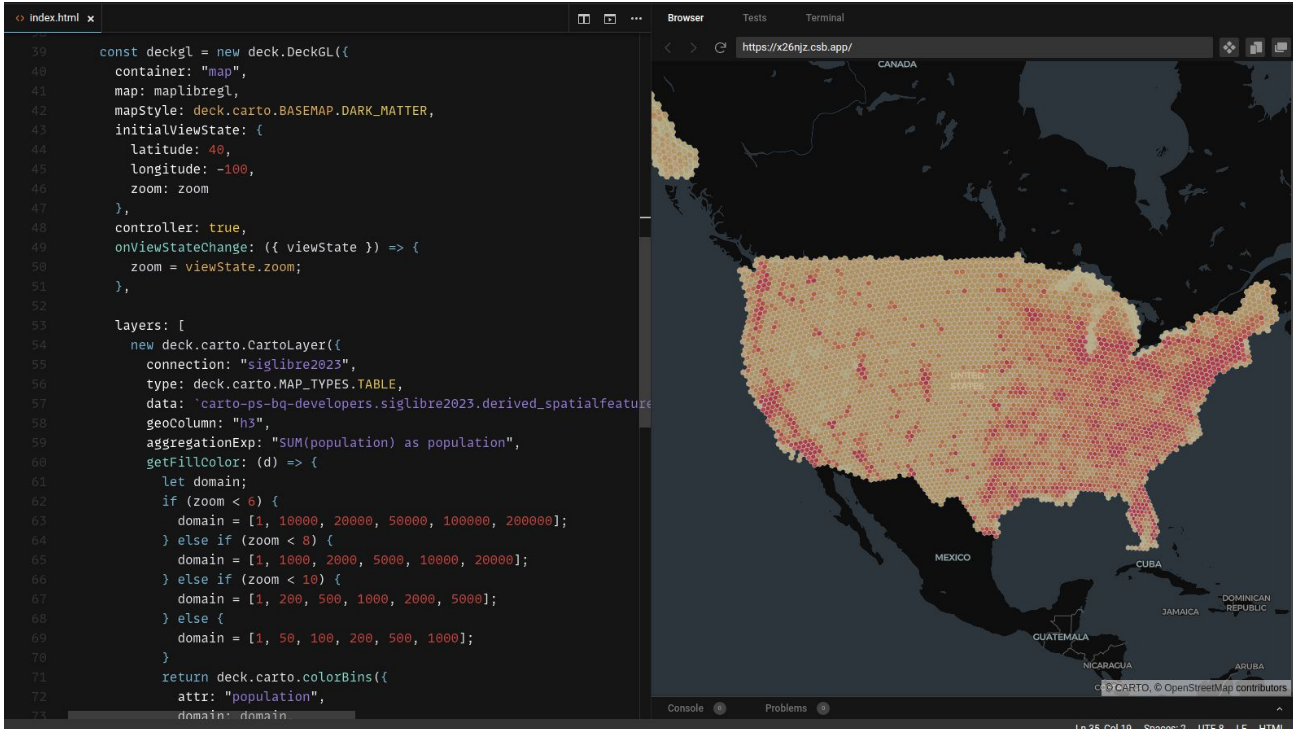
*****_Google BigQuery, AWS Redshift, Snowflake, Databricks and PostgreSQL-compatible databases._

Quadbin web map example (dynamic tiles)



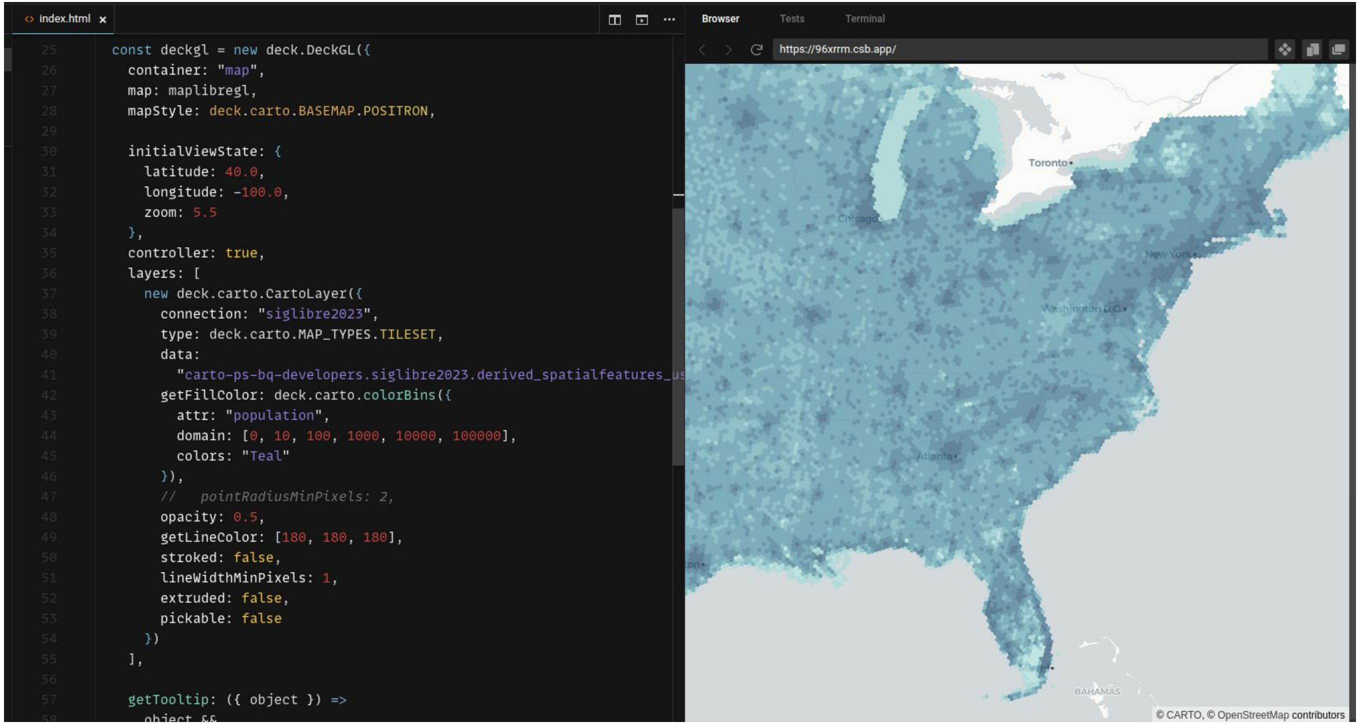
https://codesandbox.io/s/sad-surf-vn4wdg?file=/index.html

H3 web map example (dynamic tiles)



https://codesandbox.io/s/sad-bogdan-x26njz?file=/index.html

H3 web map example (static tiles)



https://codesandbox.io/s/infallible-moon-96xrrm?file=/index.html

dynamic tiles and static tiles (tileset)

Row	h3	population	female	male
1	8a0c0036a49ffff	0.0	0.0	0.0
2	8a0c002e4c0ffff	0.0	0.0	0.0
3	8a0c002e4caffff	0.0	0.0	0.0
4	8a0c002e4d8ffff	0.0	0.0	0.0
5	8a0c00304027fff	0.0	0.0	0.0
6	8a0c002188b7fff	0.0	0.0	0.0
7	8a0c0030454ffff	0.0	0.0	0.0
8	8a0c00205d2ffff	0.0	0.0	0.0
9	8a0c002e1b5ffff	0.0	0.0	0.0
10	8a0c0006b6dffff	0.0	0.0	0.0
11	8a0c002f5697fff	0.0	0.0	0.0
12	8a0c00315a57fff	0.0	0.0	0.0
13	8a0c00264c4ffff	0.0	0.0	0.0
14	8a0c002e426ffff	0.0	0.0	0.0
15	8a0c002e42d7fff	0.0	0.0	0.0
16	8a0c002d94a7fff	0.0	0.0	0.0
17	8a0c002c4157fff	0.0	0.0	0.0
18	8a0c0028d20ffff	0.0	0.0	0.0

Row	tile	data
1	603708299841372159	{"id":"8a0cd8000797fff","properties":{"population":0}}
2	603708299841372159	{"id":"8a0cd80006e7fff","properties":{"population":0}}
3	603708299841372159	{"id":"8a0cd80007a7fff","properties":{"population":0}}
4	603708299841372159	{"id":"8a0cd8003987fff","properties":{"population":0}}
5	603708299841372159	{"id":"8a0cd8003927fff","properties":{"population":0}}
6	603708299841372159	{"id":"8a0cd8000617fff","properties":{"population":0}}
7	603708299841372159	{"id":"8a0cd80039a7fff","properties":{"population":0}}
8	603708299841372159	{"id":"8a0cd8003807fff","properties":{"population":0}}
9	603708299841372159	{"id":"8a0cd8006927fff","properties":{"population":0}}
10	603708299841372159	{"id":"8a0cd800024ffff","properties":{"population":0}}
11	603708299841372159	{"id":"8a0cd800029ffff","properties":{"population":0}}
12	603708299841372159	{"id":"8a0cd800685ffff","properties":{"population":0}}
13	603708299841372159	{"id":"8a0cd8000717fff","properties":{"population":0}}
14	603708299841372159	{"id":"8a0cd8003977fff","properties":{"population":0}}
15	603708299841372159	{"id":"8a0cd80038cffff","properties":{"population":0}}
16	603708299841372159	{"id":"8a0cd800684ffff","properties":{"population":0}}
17	603708299841372159	{"id":"8a0cd80068effff","properties":{"population":0}}
18	603708299841372159	{"id":"8a0cd8000247fff","properties":{"population":0}}

Questions?

References

- BigQuery Geography functions
- Analytics Toolbox for BigQuery
- H3 Resolution table
- Bing Maps Tile System
- Spatial Indexes 101

Thanks !