



Leveraging the Power of Uber H3 Indexing Library in Postgres for Geospatial Data Processing

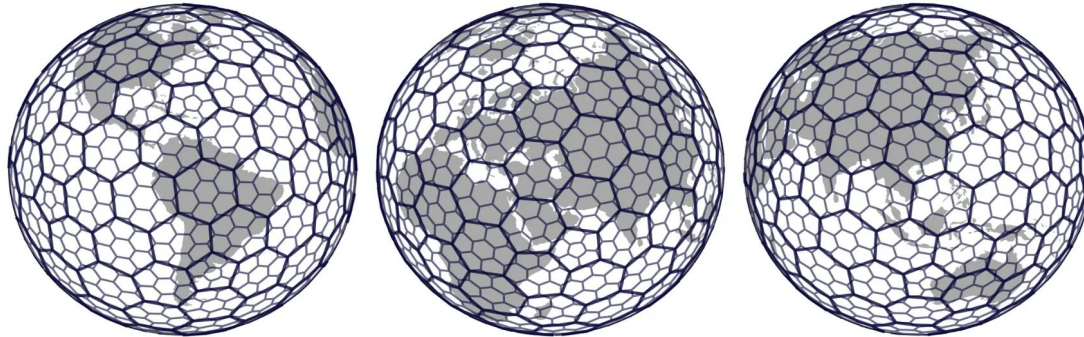


What's uber h3?

H3 is a hierarchical grid system designed to efficiently index and organize geospatial data. It divides the Earth's surface into hexagonal cells of various sizes, creating a hierarchical structure that allows for spatial indexing and querying.

The H3 grid system offers advantages such as spatial indexing, fast neighbor searches, and efficient spatial aggregation. It can be used in various applications involving geospatial data analysis, routing, visualization, and more. The H3 library provides APIs and tools for working with the grid system in different programming languages, making it accessible for developers.

H3 indexing



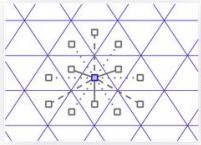
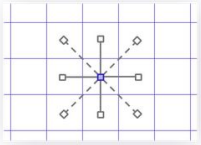
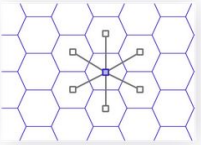
Dymaxion Projection





Aggregation

Analysis of location data, such as locations of cars in a city, can be done by bucketing locations. ([Sahr et al., 2003](#)) Using a regular grid provides smooth gradients and the ability to measure differences between cells.

The cell shape of that grid system is an important consideration. For simplicity, it should be a polygon that tiles regularly: the triangle, the square, or the hexagon. Of these, triangles and squares have neighbors with different distances. Triangles have three different distances, and squares have two different distances. For hexagons, all neighbors are equidistant.

Triangle	Square	Hexagon
		
Triangles have 12 neighbors	Squares have 8 neighbors	Hexagons have 6 neighbors

Indexing

Uncompacted (dense)	Compacted (sparse)
	
California represented by 10633 uncompacted cells	California represented by 901 compacted cells



Choosing a resolution

Res	Average <u>Hexagon</u> Area (km ²)	Pentagon Area* (km ²)	Ratio (P/H)
0	4,357,449.416078381	2,562,182.162955496	0.5880
1	609,788.441794133	328,434.586246469	0.5386
2	86,801.780398997	44,930.898497879	0.5176
3	12,393.434655088	6,315.472267516	0.5096
4	1,770.347654491	896.582383141	0.5064
5	252.903858182	127.785583023	0.5053
6	36.129062164	18.238749548	0.5048
7	5.161293360	2.604669397	0.5047
8	0.737327598	0.372048038	0.5046

8	0.737327598	0.372048038	0.5046
9	0.105332513	0.053147195	0.5046
10	0.015047502	0.007592318	0.5046
11	0.002149643	0.001084609	0.5046
12	0.000307092	0.000154944	0.5046
13	0.000043870	0.000022135	0.5046
14	0.000006267	0.000003162	0.5046
15	0.000000895	0.000000452	0.5046



Getting things done in postgis!

Github: <https://github.com/jashanbhullar/foss4g-2023-spatial-analysis-in-postgres-using-uber-h3>

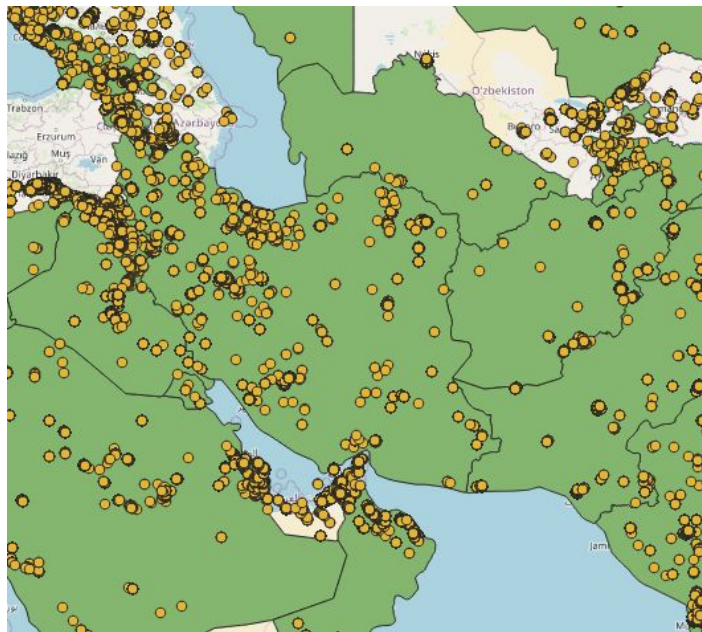
H3 use cases:

- Point in Polygon using ST_Within vs h3index
- Representing shapes with h3index
- Compacting h3index to represent shapes with lower number of h3index
- Aggregating points to coarser resolution

Point in Polygon ST_Within

```
select
  b.osm_id,
  count(a.osm_id)
from
  planet_osm_trees a
  right join
planet_osm_polygon_admin_2 b on
ST_Within(a.way, b.way)
group by
  b.osm_id
order by
  count;
```

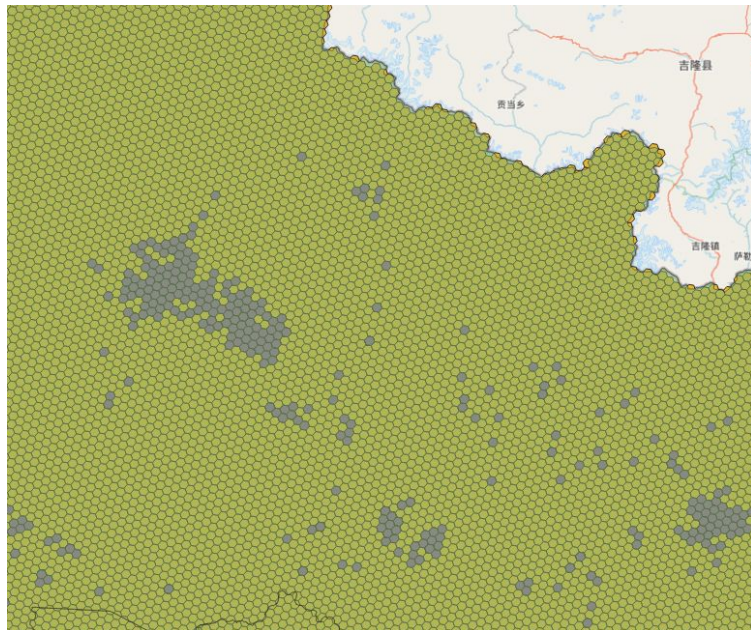
– Takes 38 seconds



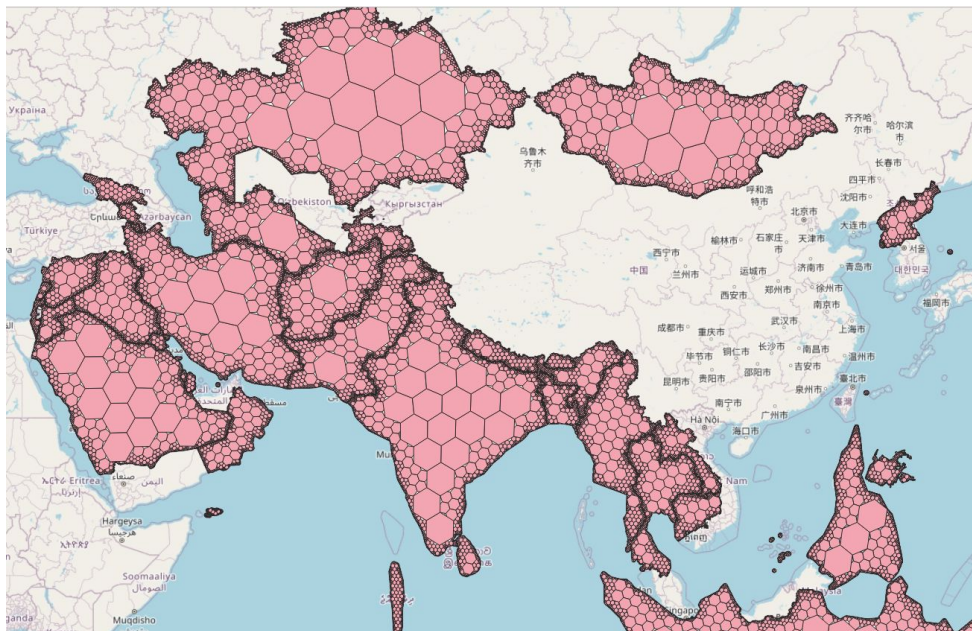
Point in Polygon with h3Index

```
select
  b.osm_id,
  count(a.osm_id)
from
  planet_osm_trees a
join
  planet_osm_polygon_admin_2_flat b on
a.h3_index = b.h3_index
group by
  b.osm_id
order by
  count;
```

– takes 3 secs



Compact Polygons



Aggregation



More things to try:

- Nearest neighbours (kRing), helpful especially in ML analysis
- Edge function, moving from one cell to the next creating a path with h3index
- Optimizing queries using postgres techniques such as indexing and partitioning



Applications Using H3

Visualization

- kepler.gl - An open source geospatial analysis tool
- [pydeck](https://pydeck.gl) - High-scale spatial rendering in Python, powered by deck.gl



Sources

- https://www.youtube.com/watch?v=ay2uwtRO3QE&t=1044s&ab_channel=UberEngineering
- <https://www.uber.com/en-IN/blog/h3/>
- <https://h3geo.org/docs/>
- <https://github.com/zachasme/h3-pg>
- <https://hub.docker.com/r/postgis/postgis>
- <https://h3geo.org/docs/community/tutorials>
- <https://github.com/jashanbhullar/foss4g-2023-spatial-analysis-in-postgres-using-uber-h3>



Thank You

You can reach me out on:



https://twitter.com/json_singh



<https://www.linkedin.com/in/jonsingh/>

All the code is at:

<https://github.com/jashanbhullar/foss4g-2023-spatial-analysis-in-postgres-using-uber-h3>

Look for `json singh` !