Geospatial Indexes

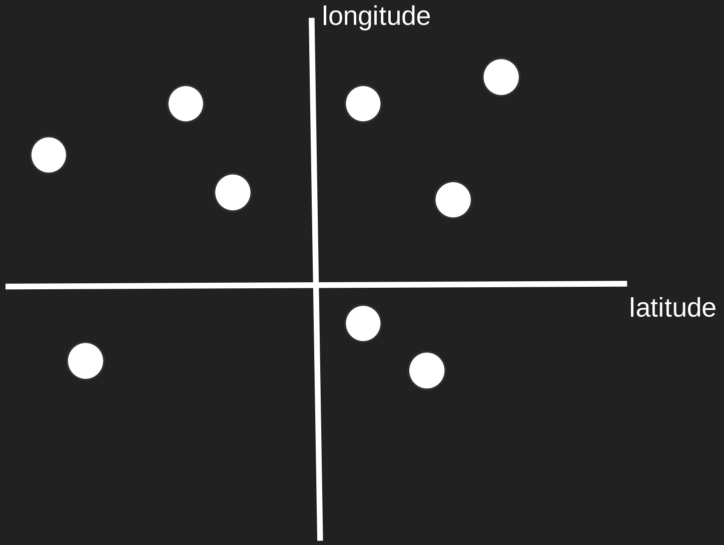
Background

In many applications, it is useful to be able to perform queries based on the location of a given point - for example, finding all objects within a certain distance (radius) of the original point. See Uber, Yelp, Tinder, etc.

As we know, quick querying in a database can be accomplished by creating an index - but in order to index geolocation data, we need to do some more sophisticated work. This is where geospatial indexes come in!

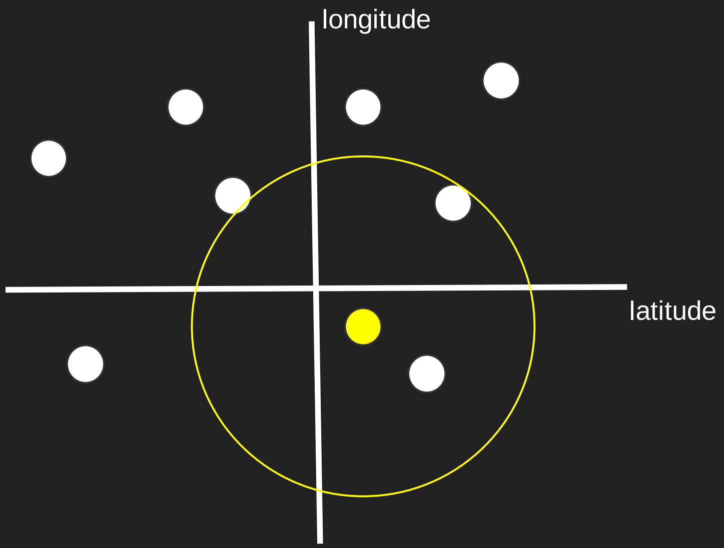
Coordinate Data - The Problem

Coordinates are expressed by a 2D tuple containing latitude and longitude.



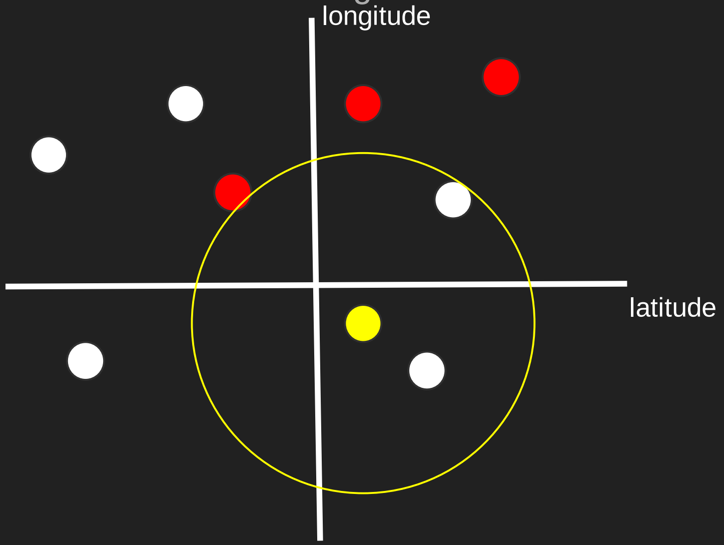
Coordinate Data - The Problem

Let’s find all the points within some distance x of the highlighted point!



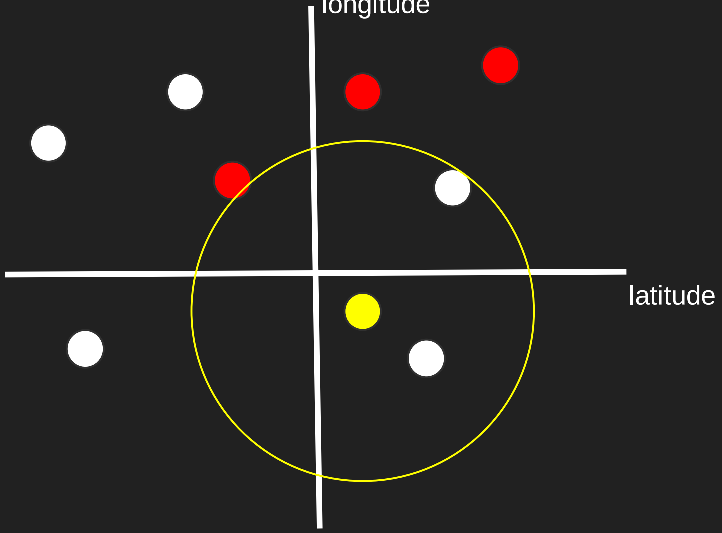
Coordinate Data - The Problem

Since LatLng is 2 dimensions, we can only have an index on one of them - imagine we first selected points based on latitude - we would then after have to check the longitude still in order to see if it falls in the circle.



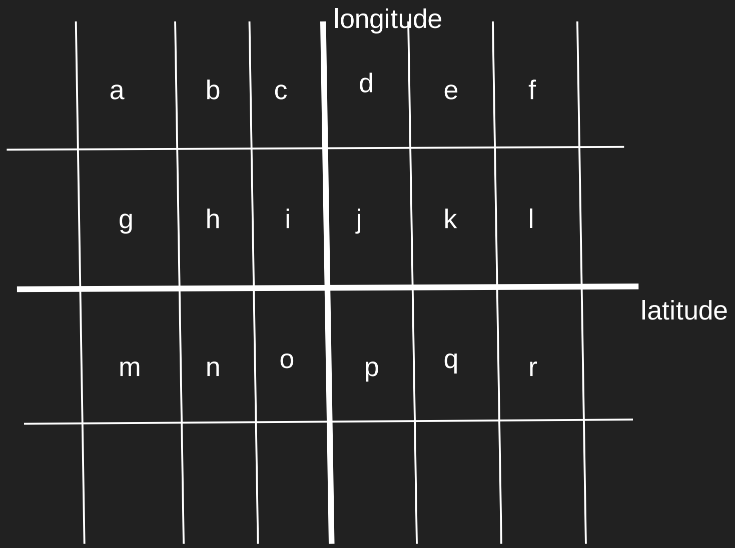
Coordinate Data - The Problem

Ultimately, a traditional index is not really useful to us as in the worst case, we basically still have to perform a scan over most of the points to check manually if they are in the circle - instead let’s find a better solution!



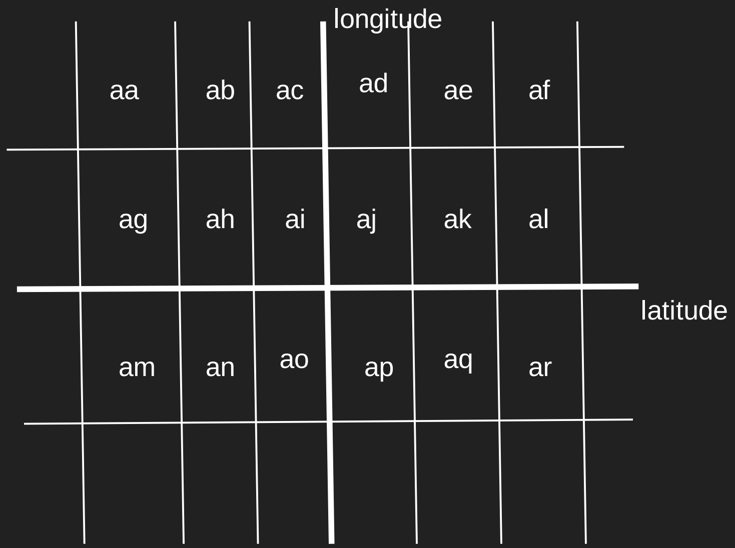
Geohashing

In comes geohashing, which maps areas of a 2d plane to a single string value!



Geohashing

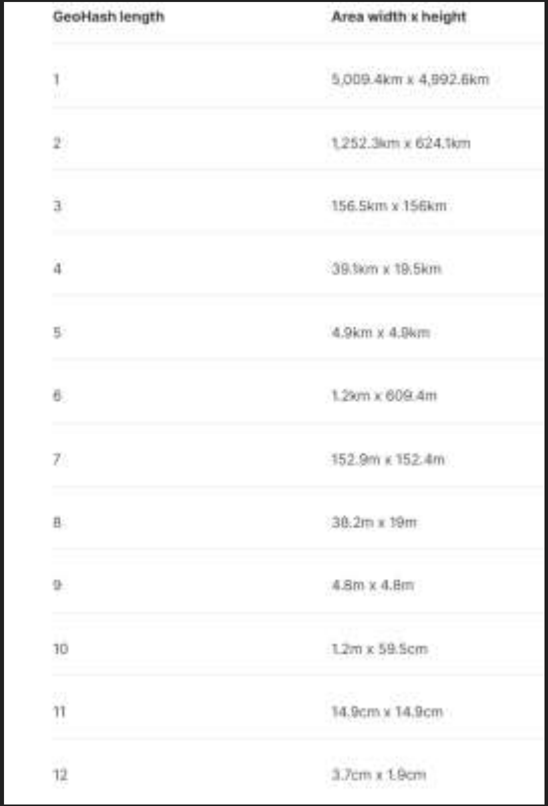
Each region is split into subregions, which have an additional character per depth of subregion.



Geohashing Continued

As the length of the GeoHash increases, the size of the bounding box of the GeoHash decreases!

Now that we have discussed what a GeoHash is, how can we use it in an index?



GeoHash length	Area width x height
1	5,009.4km x 4,992.6km
2	1,252.3km x 624.1km
3	156.5km x 156km
4	39.1km x 19.5km
5	4.9km x 4.9km
6	1.2km x 609.4m
7	152.9m x 152.4m
8	38.2m x 19m
9	4.8m x 4.8m
10	1.2m x 59.5cm
11	14.9cm x 14.9cm
12	3.7cm x 1.9cm

Geospatial Indexing

Recall: Both B-trees and LSM Trees + SSTables allow us to execute fast range queries on the indexed field.

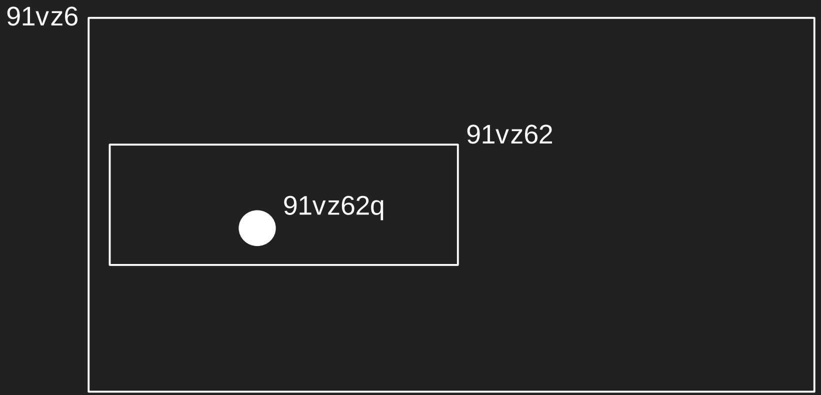
So if we want to find all points within a certain distance of a given point:

- Convert original point to geohash
- Figure out the size of the geohash depth that encapsulates the distance from the point that we want (using the table from the previous slide)
- Truncate the geohash of the original point to the length figured out in the previous step
- Use the index to figure out all of the points in the larger bounding box, and check which ones are actually within the proper radius

Geospatial Indexing Continued

Example: Find all points < 1km from (42.167, 12)

1) Get geohash: 91vz62q
2) Get proper bounding box:
    a) Geohash of length 5 has size 4.9km^2
    b) So we want to find all points in 91vz6
3) Use index to find all points p where 91vz6 < p < 91vz7
    a) This is a range query, super quick!
4) May have to filter down the result of the range query a bit by calculating the actual distance between the original point and the found points



Note: in reality the point may be towards the border of the bounding box, and as a result we may have to check the ranges ofthe bordering bounding boxes as well (so could be upto 9 separate range queries!)

Distributed GeoSpatial Indexes

How can we actually scale a geospatial index over multiple nodes? By partitioning!

Simply assign large chunks of an index (such as the box “a” and all of its children to one node, do the same to box “b” and all of its children). That way all nearby points will be on the same node.

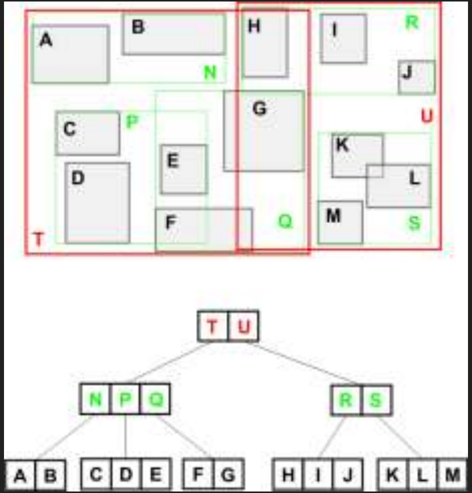
GeoSpatial Indexes in Practice

Lyft uses Redis’ geospatial indexes which use geohashes in order to keep them in memory for great performance.

Uber has created their own type of geospatial index called H3, which is very similar to the geohashing approach that we have spoken about, but uses hexagons instead of rectangles and squares in order to help better figure out what other cells should be queried when finding all points in a given radius.

GeoSpatial Indexes for Shapes?

Note: If you see databases that are indexing full polygon shapes (as opposed to just points), these are unlikely to be using geohashes. Instead, they use another similar but more complicated type of structure known as R-trees!



Conclusion

GeoSpatial indexing is a must have for any application that heavily relies on finding objects with similar location. If you are ever asked to build some sort of real estate service, hotel finder, or localized dating app, you now know what to say!

By converting each two dimensional coordinate to a one dimensional hash with a hierarchical structure, we can quickly perform range queries on GeoSpatial data via an index.

https://gis.stackexchange.com/questions/18330/using-geohash-for-proximity-searches

https://www.youtube.com/watch?v=tu6QKpV7GiI&ab_channel=SuccessinTech

https://eng.uber.com/h3/