Storing Geospatial Information

As you know you can store any type of data, but if you want to query them you need to use some coordinates, and create index on them.

GeoJSON is a format for encoding, in JSON, a variety of geographic data structures, and support the following types:

Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon and Geometry.

The GeoJSON format is quite straightforward based, for the simple geometries, on two attributes: type and coordinates.

Point:

latitude & longitude: 48.5917, -2.5469

so this point in GeoJSON will look like:

{

"type": "Point",

"coordinates": [

-2.5469,

48.5917

]

}



#### **let's now for example look at a line, a very nice walk on the beach:**

{

"type": "LineString",

"coordinates": [

[-2.551082,48.5955632],

[-2.551229,48.594312],

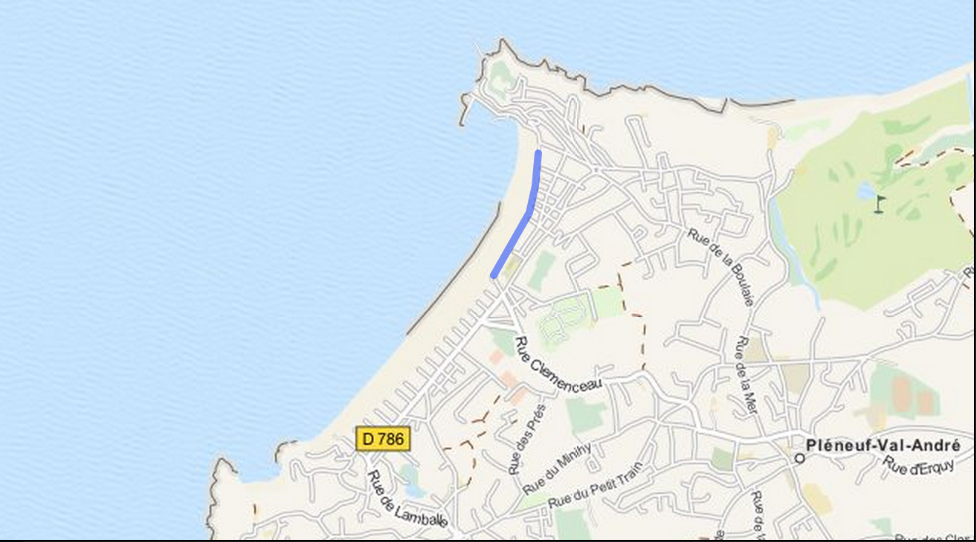
[-2.551550,48.593312],

[-2.552400,48.592312],

[-2.553677, 48.590898]

]

}



#### **So using the same approach you will be able to create MultiPoint, MultiLineString, Polygon, MultiPolygon.**

#### **It is also possible to mix all these in a single document using a GeometryCollection.**

#### **The following example is a Geometry Collection of MultiLineString and Polygon over Central Park:**

{

"type" : "GeometryCollection",

"geometries" : [

{

"type" : "Polygon",

"coordinates" : [

[

[ -73.9580, 40.8003 ],

[ -73.9498, 40.7968 ],

[ -73.9737, 40.7648 ],

[ -73.9814, 40.7681 ],

[ -73.9580, 40.8003 ]

]

]

},

{

"type" : "MultiLineString",

"coordinates" : [

[ [ -73.96943, 40.78519 ], [ -73.96082, 40.78095 ] ],

[ [ -73.96415, 40.79229 ], [ -73.95544, 40.78854 ] ],

[ [ -73.97162, 40.78205 ], [ -73.96374, 40.77715 ] ],

[ [ -73.97880, 40.77247 ], [ -73.97036, 40.76811 ] ]

]

}

]

}



### Querying Geospatial Informations

Now that we have the data stored in MongoDB, it is now possible to use the geospatial information to do some interesting queries.

For this we need a sample dataset. I have created one using some open data found in various places. This dataset contains the following information’s:

* airports collection with the list of US airport (Point)
* states collection with the list of US states (MultiPolygon)

MongoDB allows applications to do the following types of query on geospatial data:

* inclusion
* intersection

Obviously, you will be able to use all the other operator in addition to the geospatial ones. Let's now look at some concrete examples.

#### Inclusion

Find all the airports in California.

For this you need to get the California location (Polygon) and use the command $geoWithin in the query. From the shell it will look like:

var cal = db.states.findOne( {code : "CA"} );

db.airports.find(

{

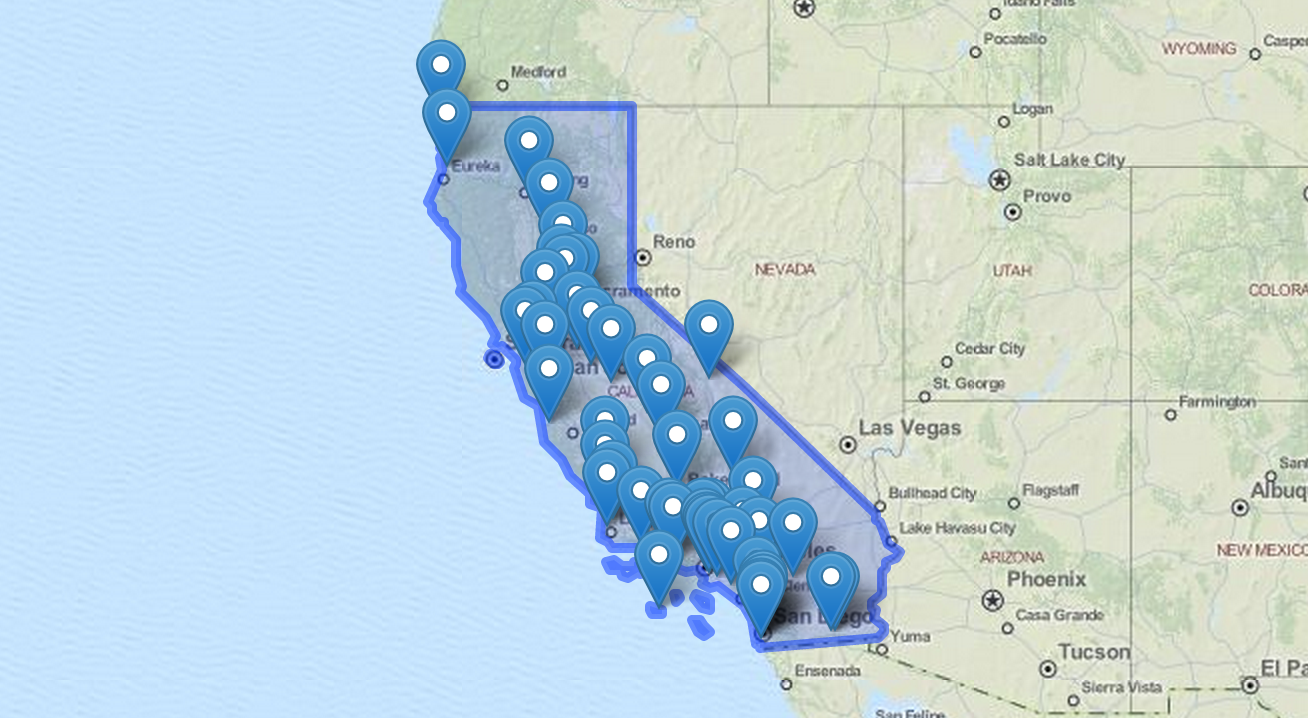
loc : { $geoWithin : { $geometry : cal.loc } }

},

{ name : 1 , type : 1, code : 1, \_id: 0 }

);

So the query is using the "California MultiPolygon" and looks in the airports collection to find all the airports that are in these polygons. This looks like the following image on a map:



You can use any other query features or criteria, for example you can limit the query to international airport only sorted by name :

db.airports.find(

{

loc : { $geoWithin : { $geometry : cal.loc } },

type : "International"

},

{ name : 1 , type : 1, code : 1, \_id: 0 }

).sort({ name : 1 });

#### Intersection

Suppose you want to know what are all the adjacent states to California, for this we just need to search for all the states that have coordinates that "intersects" with California.

This is done with the following query:

var cal = db.states.findOne( {code : "CA"} );

db.states.find(

{

loc : { $geoIntersects : { $geometry : cal.loc } } ,

code : { $ne : "CA" }

},

{ name : 1, code : 1 , \_id : 0 }

);