Geospatial and MongoDB
Agenda

- MongoDB Geospatial Features
- Query Examples
- Optimizations
Norberto Leite

Developer Advocate
Curriculum Engineer

Twitter: @nleite
norberto@mongodb.com
The Basics
[Longitude, Latitude]

Lat: 45°
Lon: 60°

[0, 0] Positive
[60, 45] Positive

Diagram showing the relationship between longitude and latitude.
Quiz Time!

Which of these shapes is the most similar with Planet Earth?
Stripped version of Earth: Geoid

http://www.ngs.noaa.gov/GEOID/
Surface Types

Flat

Spherical

2d Indexes

2dsphere Indexes
2D Indexes

var place = {
    type: "building",
    name: "AW1 - Building"
    location: [4.380717873573303, 50.81219570880462]
}

var checkin = {
    type: "checkin",
    message: "this place is awesome!"
    location: {
        lng: 4.348099529743194,
        lat: 50.850980167854615
    }
}
2D Indexes

//index creation
db.col.createIndex( {'location': '2d'})
db.col.createIndex( {'location': '2d'}, {'sparse': true})
var place = {
  type: "building",
  name: "AW1",
  location: {
    type: "Point",
    coordinates: [4.380717873573303, 50.81219570880462]
  }
}
Spherical Surface

```javascript
var place = {
  type: "building",
  name: "AW1",
  location: {
    type: "Polygon",
    coordinates: [
      [
        4.380406737327576, 50.812253331704625,
      ],
      [
        4.380889534950256, 50.81239569385869,
      ],
      [
        4.381093382835388, 50.812134696244804,
      ],
      [
        4.380605220794678, 50.81198894369594,
      ],
      [
        4.380406737327576, 50.812253331704625,
      ]
    ]
  }
};
```

http://geojson.org/
2dsphere Indexes

//index creation
db.collection.createIndex( { location : "2dsphere" } )

//compound with more than 2 members
db.collection.createIndex( { location : "2dsphere", name: 1, type: 1 } )
<table>
<thead>
<tr>
<th></th>
<th>2d index</th>
<th>2dsphere</th>
</tr>
</thead>
<tbody>
<tr>
<td>Legacy</td>
<td>Legacy</td>
<td>Spatial Support</td>
</tr>
<tr>
<td>Coordinates Pair</td>
<td>Coordinates Pair</td>
<td>GeoJson Format</td>
</tr>
<tr>
<td>Manual Earth-like geometry calculations</td>
<td>Manual Earth-like geometry calculations</td>
<td>WGS84 Datum</td>
</tr>
<tr>
<td>Single extra field for compound indexes</td>
<td>Single extra field for compound indexes</td>
<td>Multiple fields</td>
</tr>
</tbody>
</table>
Indexation
How does MongoDB generate index keys?

```
var place = {
  name: "Starbucks",
  loc: {
    type: "Point",
    coordinates: [1.3, 45]
  }
}

db.places.insert(place)

db.places.createIndex({loc: "2dsphere"})
```

How does MongoDB generate index keys?
Points to Index Keys

Project spherical point to bounding cube

Each face is a Quadtree
Points to Index Keys

Face 5

Key 5F  5F1  5F12  5F120

Every cm² can be represented with 30 levels
## S2 Library

<table>
<thead>
<tr>
<th>Level</th>
<th>Min Area</th>
<th>Max Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>85,011,012 km²</td>
<td>85,011,012 km²</td>
</tr>
<tr>
<td>1</td>
<td>21,252,753 km²</td>
<td>21,252,753 km²</td>
</tr>
<tr>
<td>12</td>
<td>3.31 km²</td>
<td>6.38 km²</td>
</tr>
<tr>
<td>30</td>
<td>0.48 cm²</td>
<td>0.93 cm²</td>
</tr>
</tbody>
</table>

↑

smallest cell
A key is a prefix of another iff it is a parent cell
Query Examples
Geospatial operators

$\text{geoWithin}$

$\text{geoIntersects}$

$\text{near/nearSphere}$
$geoWithin
`{ location: {
    $geoWithin: {
        $geometry: {
            'type': 'Polygon',
            'coordinates': [
                [
                    [-73.975181, 40.758494],
                    [-73.973336, 40.760965],
                    [-73.974924, 40.761663],
                    [-73.976748, 40.759160],
                    [-73.975181, 40.758494]
                ]
            ]
        }
    }
} }`
$geoIntersects
db.coll.find({ location: {
  $geoIntersects: {
    $geometry: {
      "type": "LineString",
      "coordinates": [
        [-73.979543, 40.761132],
        [-73.974715, 40.759127],
        [-73.973363, 40.760969],
        [-73.970059, 40.759600]
      ]
    }
  }
})
Stage 1

Stage 2

Stage 3

$near
Polygons
Define a polygon by specifying 4 points

But what’s the inside of the polygon?
Convention for deciding “inside”: Winding Order + Right Hand Rule
Positive = Inside of Polygon
But how does MongoDB pick the inside of the Polygon?
MongoDB 2.6 behavior
Polygon

```javascript
var polygon = {
  "type": "Polygon",
  "coordinates": [
    [-73.969581, 40.760331],
    [-73.974487, 40.762245],
    [-73.977692, 40.763598],
    [-73.979508, 40.761269],
    [-73.982364, 40.762358],
    [-73.983692, 40.760497],
    [-73.972821, 40.755861],
    [-73.969581, 40.760331]
  ]
};
```

Defines a business service area
Big Polygon
I am an airplane at [0, 0]
What airports are within my flight range?

Start with a plane with a medium sized flight range polygon
I am an airplane at [0, 0]
What airports are within my flight range?

If it’s a longer range plane, that polygon gets bigger
I am an airplane at [0, 0]
What airports are within my flight range?

Eventually polygon get so big it covers more than 50% of the planet
urn:x-mongodb:crs:strictwinding:EPSG:4326

- **urn**
  - Uniform resource name
- **x-mongodb**
  - MongoDB extension
- **strictwinding**
  - Enforces explicit “counter-clockwise” winding
  - a.k.a.
    - anticlockwise,
    - right hand rule
    - the correct way
- **EPSG:4326**
  - Another name for WGS84
    (the standard web geo coordinate system)
How do I do that?

// build some geo JSON
var crs = "urn:x-mongodb:crs:strictwinding:EPSG:4326"
var bigCRS = { type : "name", properties : { name : crs } };
var bigPoly = {  type : "Polygon",
    coordinates : [
        [[-10.0, -10.0],
        [10.0, -10.0],
        [10.0, 10.0],
        [-10.0, 10.0],
        [-10.0, -10.0]]],
    crs : bigCRS
};

var cursor = db.<collection>.find({
    loc : { $geoWithin : { $geometry : bigPoly } } }
});

var cursor = db.<collection>.find({
    loc : { $geoIntersects : { $geometry : bigPoly } } }
});
Complex Polygons
Complex Polygons

```json
{ "type": "Polygon", "coordinates": [ [ -73.969581, 40.760331 ], [ -73.974487, 40.762245 ], [ -73.977692, 40.763598 ], ... ], [ [ -73.975181, 40.758494 ], [ -73.973336, 40.760965 ], [ -73.974924, 40.761663 ], ... ], [ [ -73.979437, 40.755390 ], [ -73.976953, 40.754362 ], [ -73.978364, 40.752448 ], ... ] ]
```
Complex Polygons

```json
{
    "type": "Polygon",
    "coordinates": [
        [ -73.969581, 40.760331 ],
        [ -73.974487, 40.762245 ],
        [ -73.977692, 40.763598 ],
        ...
    ],
    [ -73.975181, 40.758494 ],
    [ -73.973336, 40.760965 ],
    [ -73.974924, 40.761663 ],
    ...
],
[ -73.979437, 40.755390 ],
[ -73.976953, 40.754362 ],
[ -73.978364, 40.752448 ],
...
}
```

$err : Can't canonicalize query
BadValue Secondary loops not contained by first exterior loop - secondary loops must be holes
Complex Polygons

```json
$or: [{
  geometry: {
    $geoWithin: {
      $geometry: {
        "type": "Polygon",
        "coordinates": [
          ...
        ]
      }
    }
  }
}, {
  geometry: {
    $geoWithin: {
      $geometry: {
        "type": "Polygon",
        "coordinates": [
          ...
        ]
      }
    }
  }
}]
```
Optimizations
$\text{geoNear Algorithm}$

Series of $\text{geoWithin} + \text{sort}$
Problem 1: Repeated Scans

Stage 2

Stage 3
Buffer every document in covering

Original Algorithm

New Algorithm
Avoid repeated index scans

Covering

Visited Cells

Difference
Avoid repeated index scans

Stage 2

Stage 3
Problem 1.1: Unnecessary fetches

Index Scan

Filter Out Disjoint Keys

Fetch

Last Interval
Original $\text{geoNear Algorithm}$

Filter out disjoint keys then filter out disjoint docs
Problem 2: Large Initial Radius

Initial radius is over 1km

Finding one document 1cm away
Determining the Radius

The minimum distance to find documents
Indexing Points to Finest Level

Original indexed level

New indexed level

Bounded by cell size
Why were Points Indexed Coarsely?

Polygons have a tradeoff in storage size
Problem 2.1: New Index Version

- Finer index level means different index keys
- 2dSphere index version 3 introduced
Problem 2.2: Larger Index Size

String (v2)  NumberLong (v3)

1F12031 → 00101100011011

Size on 500,000 Documents
Problem 2.3: More intervals

Because we no longer repeat index scans, there is little to no performance hit.
Problem 3: Query Level still Coarse

Covering of radius is constrained by index covering levels
Split Index and Query Constraints

Set query maximum to finest level
Before and After

Finding one document 1cm away
## Results

<table>
<thead>
<tr>
<th></th>
<th>MongoDB 3.0 with 2dsphere Index Version 2</th>
<th>MongoDB 3.1.6 with Index Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>time (ms)</td>
<td>229,550</td>
<td>1571 (0.7% of original)</td>
</tr>
<tr>
<td>Keys</td>
<td>183,994,438</td>
<td>429,739 (0.25% of original)</td>
</tr>
<tr>
<td>entries</td>
<td>19,108,349</td>
<td>422,046 (2.2% of original)</td>
</tr>
<tr>
<td>Test Case</td>
<td>Ops per second before change</td>
<td>Ops per second after change</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>$geoWithin queries</td>
<td>1,733</td>
<td>20,198</td>
</tr>
<tr>
<td>$geoNear queries</td>
<td>533</td>
<td>6,783</td>
</tr>
<tr>
<td>$geoNear queries limiting to one document</td>
<td>111</td>
<td>17,211</td>
</tr>
</tbody>
</table>
More info on Optimization

Obrigado!

Norberto Leite
Engineer
norberto@mongodb.com
@nleite