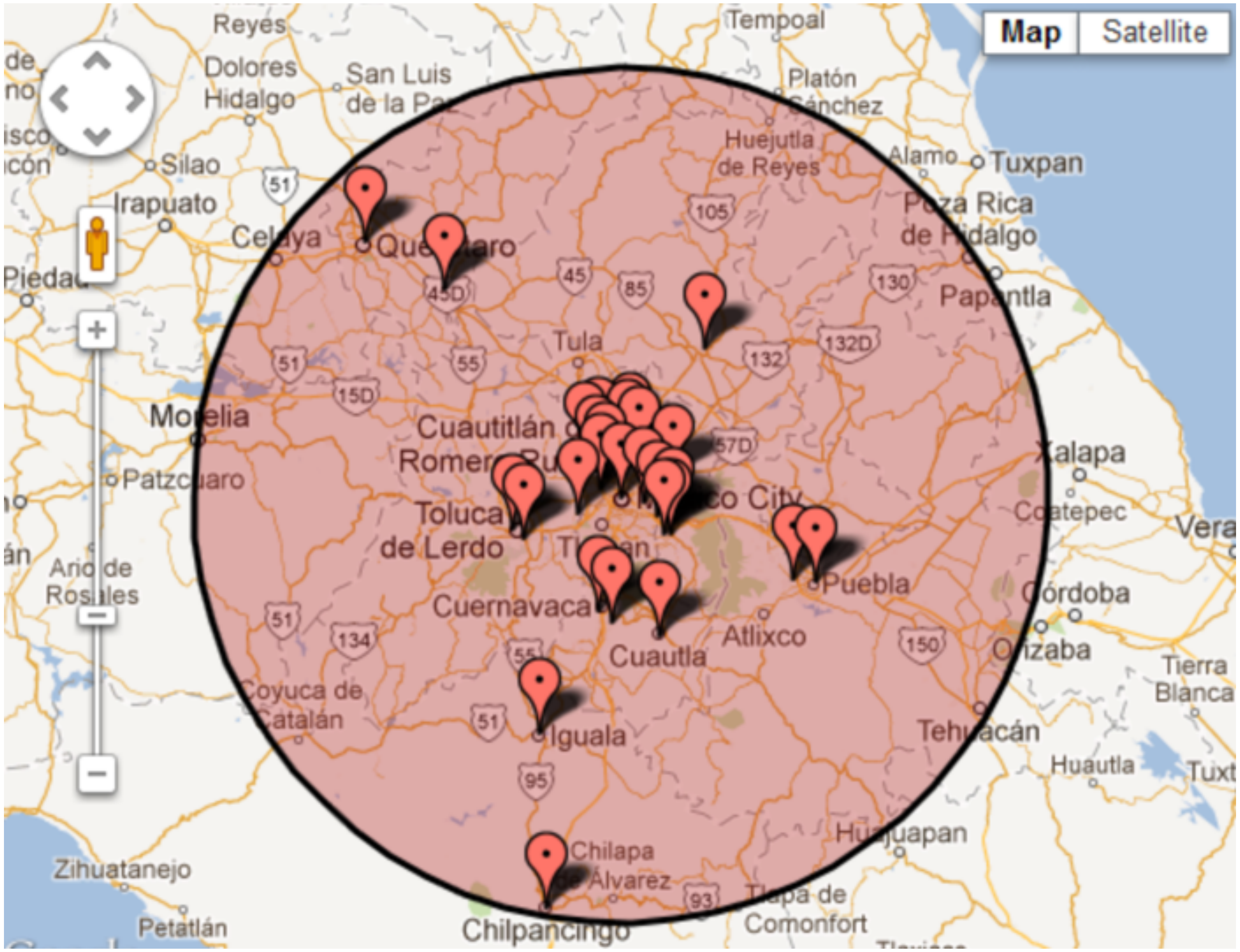
# MongoDB Geospatial Queries

Olivier Liechti & Simon Oulevay
COMEM Web Services 2016

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

# How do I make geospatial queries?

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud



## Storing geospatial data (1)

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

**GeoJSON** is a format for encoding a variety of geographic data structures.

```
{
  "banner": "Starbucks",
  "city": "Lausanne",
  "location": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  }
}
```

GeoJSON supports the following geometry types: **Point**, **LineString**, **Polygon**, **MultiPoint**, **MultiLineString**, and **MultiPolygon**.

## Storing geospatial data (2)

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

To be able to execute **geospatial queries** on this data, you must add a MongoDB **geospatial index**.

```
var ShopSchema = new Schema({
  banner: String,
  city: String,
  location: {
    type: String,
    coordinates: [Number]
  }
});

ShopSchema.index({
  location: '2dsphere'
});
```

Add the **GeoJSON** data to your schema.

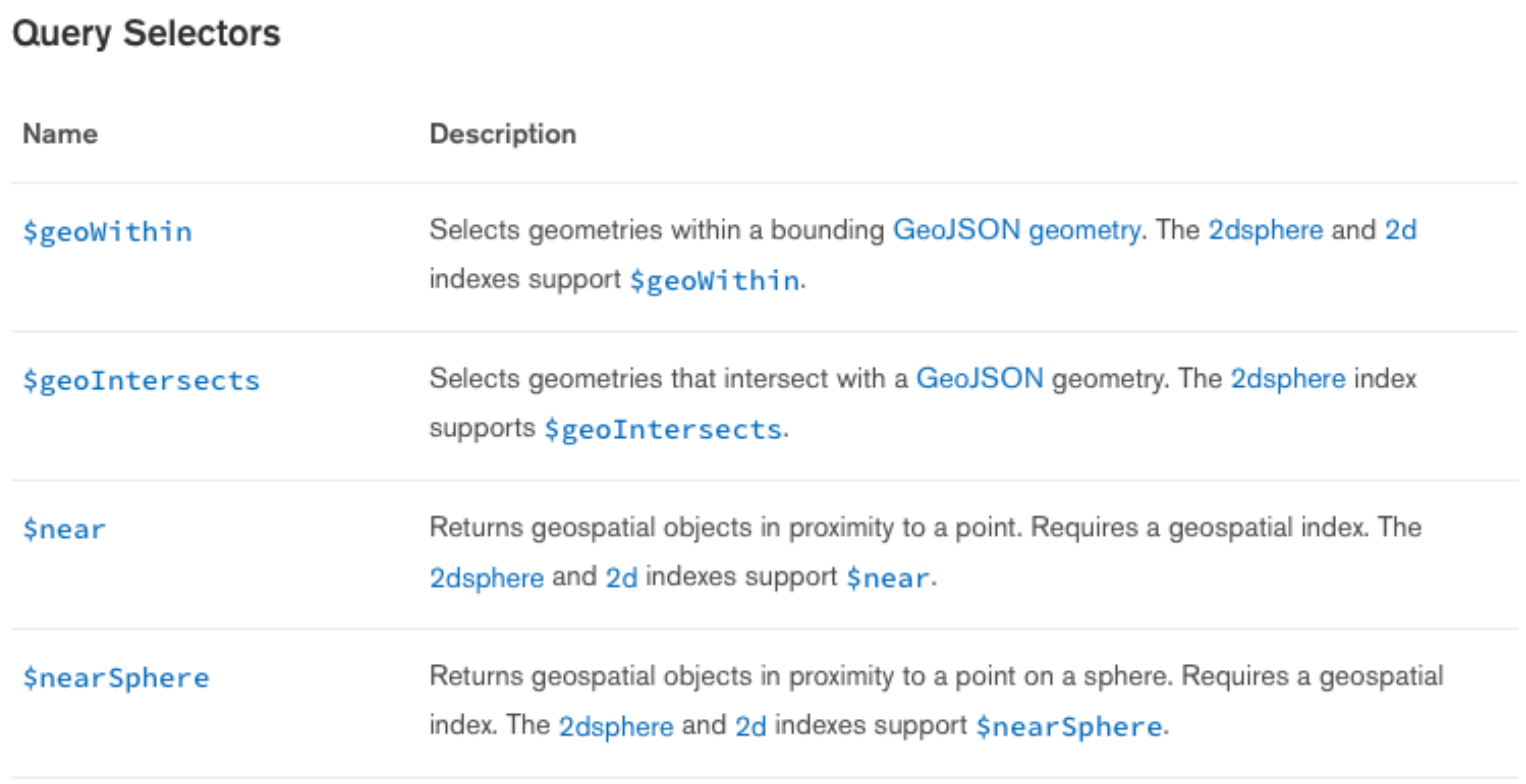
Add a **2dsphere** index on that property.

https://docs.mongodb.org/manual/applications/geospatial-indexes/

# Making geospatial queries (1)

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

MongoDB supports several **geospatial query operators**.



**Query Selectors**

| Name | Description |
|---|---|
| `$geoWithin` | Selects geometries within a bounding GeoJSON geometry. The 2dsphere and 2d indexes support `$geoWithin`. |
| `$geoIntersects` | Selects geometries that intersect with a GeoJSON geometry. The 2dsphere index supports `$geoIntersects`. |
| `$near` | Returns geospatial objects in proximity to a point. Requires a geospatial index. The 2dsphere and 2d indexes support `$near`. |
| `$nearSphere` | Returns geospatial objects in proximity to a point on a sphere. Requires a geospatial index. The 2dsphere and 2d indexes support `$nearSphere`. |

https://docs.mongodb.org/manual/reference/operator/query-geospatial/

# Making geospatial queries (2)

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

```
router.get('/', function(req, res, next) {

  var criteria = {};

  var latitude = req.query.latitude,
      longitude = req.query.longitude,
      distance = req.query.distance;

  if (latitude && longitude && distance) {
    criteria.location = {
      $near: {
        $geometry: {
          type: 'Point',
          coordinates: [
            parseFloat(longitude),
            parseFloat(latitude)
          ]
        },
        $maxDistance: parseInt(distance, 10)
      }
    };
  }

  Shop.find(criteria, function(err, shops) {
    // ...
  });
});
```

Define what query parameters API users should provide. Here we expect a **latitude**, **longitude** and a **distance** (in meters).

Here, we use the **$near** operator to find all shops within the specified distance to the specified point.

Note that the **$geometry** argument to the **$near** operator is itself a **GeoJSON** object.

Then simply pass the criteria to **find**, as we did before.