What is MongoDB?

- MongoDB is a document storage and retrieval engine
- It requires almost no configuration to set-up a high available and high performant cluster of database servers
- Each document is stored into a collection, which is stored into a database, which is stored in a database server.
- It is a NoSQL schema-free database
Terminology

- **Document**: the data (row)
- **Collection**: contains documents (table, view)
- **Index**
- **Embedded Document** (~join)
Documents

- Can have embedded documents
- Are schemaless

Document with embedded documents:

```json
{
  "id": "derickr",
  "name": "Derick Rethans",
  "talks": [
    {"title": "Profiling PHP Applications",
      "url": "http://derickrethans.nl/talks/profiling-phptour.pdf",
    },
    {"title": "Xdebug",
    }
  ]
}
```
Geospatial indexes

MongoDB supports indexes on two dimensional fields:

```javascript
db.cities.ensureIndex( { location: '2d' } )
```

and

MongoDB supports indexes on **geospatial fields**:

```javascript
db.cities.ensureIndex( { location: '2dsphere' } )
```
2d index vs. 2dsphere index
GeoJSON

Example:

```
{
  loc: {
    type: "LineString",
    coordinates: [[ -0.09, 51.49 ], [ 2.35, 48.86 ]] 
  }
}
```

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

- Used by various other HTML APIs as well: OpenLayers, Leaflet and through GDAL also PostGIS.

- Point, MultiPoint, LineString, MultiLineString, Polygon, MultiPolygon and GeometryCollection.
GeoJSON: Point

```json
{
    "id": "n356277759",
    "ty": NumberLong(1),
    "l": {
        "type": "Point",
        "coordinates": [
            -0.1607376,
            51.5138662
        ]
    },
    "ts": [
        "amenity=pub",
        "food=yes",
        "name=The Tyburn",
        "operator=wetherspoons",
        "real_ale=yes"
    ]
}
```
GeoJSON: LineString

```
{
  "id": "w4373934",
  "ty": NumberLong(2),
  "l": {
    "type": "LineString",
    "coordinates": [
      [-0.1286202, 51.50752],
      [-0.1283857, 51.5075115],
      [-0.1282389, 51.5075112],
      [-0.1278265, 51.5075453]
    ]
  },
  "ts": [
    "highway=primary",
    "lit=yes",
    "name=Trafalgar Square",
    "oneway=yes",
    "postal_code=WC2",
    "sidewalk=left"
  ]
}
```
GeoJSON: Polygon

```json
{
    "id": "w161490892",
    "ty": NumberLong(2),
    "l": {
        "type": "Polygon",
        "coordinates": [
            [ -0.1356425, 51.497671 ],
            [ -0.1355737, 51.4975662 ],
            [ -0.1354639, 51.4975815 ],
            [ -0.1354536, 51.497694 ],
            [ -0.1356425, 51.497671 ]
        ]
    },
    "ts": [
        "amenity=pub",
        "building=yes",
        "name=The Albert",
        "real_ale=yes"
    ]
}
```
Geo Operators

$\text{geoNear}$
- find stuff near a point
- index required

$\text{geoWithin}$
- find stuff within a polygon/circle
- index not required

$\text{geoIntersects}$
- find stuff that intersects with other stuff
- index not required
Example with `$near`

db.poiConcat.find({
  ts: "amenity=pub",
  l: {
    $near: {
      $geometry: {
        type: 'Point',
        coordinates: [ -0.1204, 51.5168 ]
      },
      $maxDistance: 500
    }
  }
}).limit(5).pretty();

{ 
  "id": "n26848690",
  "l": {
    "type": "Point",
    "coordinates": [ -0.119473, 51.516787 ]
  },
  "ts": [
    "addr:housenumber=64-68",
    "addr:street=Kingsway",
    "amenity=pub",
    "name=The Shakespeare's Head",
    "wifi=free"
  ]
}
Example with $geoWithin

```javascript
hydepark = db.poiConcat.findOne( {
    ts: { $all: [
        "name=Hyde Park", "leisure=park"
    ] }
} );

db.poiConcat.find( {
    l: { $geoWithin: {
        $geometry: hydepark.l
    } },
    ts: "amenity=cafe"
} );
```

```json
{
    "_id": "w19851241",
    "ty": NumberLong(2),
    "l": {
        "type": "Polygon",
        "coordinates": [
            [ [ -0.1549378, 51.508331 ], ... [ -0.1549378, 51.508331 ] ]
        ],
    "ts": [
        "access=yes", "leisure=park", "name=Hyde Park",
    ]
}
```
$geoWithin - example
$geoWithin - example
Example with $geoIntersects

```javascript
building = db.poiConcat.findOne( {
    _id: "w30734457"
});

db.poiConcat.find( {
    l: {
        $geoIntersects: {
            $geometry: building.l
        }
    },
    ts: {
        $exists: true
    }
});
```

```
{
    "_id": "w5059478", "ts": [
        "branch=Charing Cross", "electrified=rail", "frequency=0",
    ],
    "_id": "w139389296", "ts": [
        "branch=Charing Cross", "electrified=rail", "frequency=10",
    ],
    "_id": "n595696911", "ts": [
        "disused=yes", "disused:amenity=bar", "name=Kudos", "toilet=WC"
    ],
    "_id": "n595696974", "ts": [
        "amenity=cafe", "name=Costa", "wheelchair=yes"
    ],
    "_id": "n653124873", "ts": [
        "addr:city=London", "addr:housenumber=441", "addr:street=Strand"
    ],
    "_id": "n1163880380", "ts": [
        "addr:housenumber=430", "addr:street=Strand", "name=Ryman"
    ],
    "_id": "n1571982051", "ts": [
        "name=Charing Cross", "railway=subway_entrance", "source=OpenStreetMap"
    ],
    "_id": "n1571982070", "ts": [
        "name=Charing Cross", "railway=subway_entrance", "source=OpenStreetMap"
    ],
    "_id": "n2066862842", "ts": [
        "addr:housenumber=440", "addr:street=Strand", "amenity=hotel"
    ],
    "_id": "w166702178", "ts": [
        "layer=-3", "line=Jubilee", "name=Jubilee Line (disused)"
    ],
    "_id": "w166707824", "ts": [
        "layer=-2", "name=Northern Line Southbound", "railway=planned"
    ],
    "_id": "w166707825", "ts": [
        "layer=-2", "name=Northern Line Northbound", "railway=planned"
    ]
}
```
Aggregation: $geoNear

- MongoDB's aggregation framework has as $geoNear pipeline operator
- Also adds the distance as a new field to the document
- Has to be the first operator in the pipeline
- Only works if there is one 2dsphere index
Aggregation: $geoNear example

```javascript
res = db.poiConcat.aggregate([{
    'geoNear': {
        'near': {
            'type': 'Point',
            'coordinates': [-0.12160, 51.51065]
        },
        'distanceField': 'distance',
        'distanceMultiplier': 1,
        'maxDistance': 500,
        'spherical': true,
        'query': {
            '$or': [
                { ts: 'amenity=pub' },
                { ts: 'amenity=bar' },
                { ts: 'amenity=restaurant' }
            ]
        }
    }
}])
```
Conclusion

MongoDB for Geospatial use:

- NoSQL database with support for replication and sharding
- GeoJSON support for geospatial data storage
- Can mix geospatial queries with other predicates
derick@mongodb.com – @derickr