Geospatial Queries in MongoDB

1 Introduction

Geospatial queries in MongoDB allow users to store, index, and query location-based data efficiently. These queries are widely used in applications such as mapping systems, ride-sharing platforms, delivery services, and location-based recommendations.

MongoDB supports geospatial data using:

- GeoJSON objects
- Legacy coordinate pairs

2 Geospatial Index Types

2.1 2dsphere Index

- Used for spherical (Earth-like) calculations
- Supports GeoJSON objects
- Most commonly used index

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

2.2 2d Index

- Used for flat (Cartesian) geometry
- Supports legacy coordinate pairs

```
db.points.createIndex({ loc: "2d" })
```

3 GeoJSON Format

GeoJSON is the standard format used for geospatial data.

```
{
  type: "Point",
  coordinates: [longitude, latitude]
}
```

Note: Coordinates are always stored as [longitude, latitude].

4 Sample Dataset

```
db.places.insertMany([
  {
    name: "Cafe A",
    location: { type: "Point", coordinates: [88.3639, 22.5726] }
  },
  {
    name: "Library B",
    location: { type: "Point", coordinates: [88.3930, 22.5600] }
  }
])
```

5 Geospatial Query Operators

5.1 $near

Finds documents near a specific point.

```
db.places.find({
  location: {
    $near: {
      $geometry: {
        type: "Point",
        coordinates: [88.3639, 22.5726]
      },
      $maxDistance: 5000
    }
  }
})
```

5.2 $nearSphere

Similar to $near but uses spherical calculations.

```
db.places.find({
  location: {
    $nearSphere: {
      $geometry: {
        type: "Point",
        coordinates: [88.3639, 22.5726]
      },
      $maxDistance: 3000
    }
  }
})
```

5.3 $geoWithin

Finds documents within a specified region.

5.3.1 Polygon Example

```
db.places.find({
  location: {
    $geoWithin: {
      $geometry: {
        type: "Polygon",
        coordinates: [[
          [88.30, 22.55],
          [88.45, 22.55],
          [88.45, 22.62],
          [88.30, 22.62],
          [88.30, 22.55]
        ]]
      }
    }
  }
})
```

5.3.2 Box Example

```
{ $geoWithin: { $box: [[0,0],[10,10]] } }
```

5.3.3 Circle Example

```
{ $geoWithin: { $center: [[5,5],10] } }
```

5.3.4 Spherical Circle

```
{ $geoWithin: { $centerSphere: [[lng,lat], radius] } }
```

5.4 $geoIntersects

Finds documents whose geometry intersects a given geometry.

```
db.areas.find({
  region: {
    $geoIntersects: {
      $geometry: {
        type: "Point",
        coordinates: [88.3639, 22.5726]
      }
    }
  }
})
```

5.5 $geoNear (Aggregation)

Returns documents sorted by distance.

```
db.places.aggregate([
  {
    $geoNear: {
      near: {
        type: "Point",
        coordinates: [88.3639, 22.5726]
      },
      distanceField: "distance",
      maxDistance: 5000,
      spherical: true
    }
  }
])
```

6 Operations Possible with Geospatial Queries

- Find nearest locations (e.g., ATM, hospital)
- Find all points inside a region
- Perform radius-based searches
- Check intersection of geometries
- Distance-based sorting
- Geofencing (area-based filtering)
- Combine location queries with filters

7 Comparison of Operators

Operator	Purpose	Sorting
$near	Nearest search	Yes
$nearSphere	Spherical nearest	Yes
$geoWithin	Inside region	No
$geoIntersects	Overlapping geometry	No
$geoNear	Aggregation + distance	Yes

8 Important Rules

- $geoNear must be the first stage in aggregation
- 2dsphere index is required for GeoJSON queries
- Coordinates must be in [longitude, latitude]
- Geospatial indexes improve performance significantly

9 Advantages

- Efficient location-based querying
- Supports real-world mapping applications
- Works with complex geometries
- Integrated with aggregation framework

10 Disadvantages

- Requires indexing for optimal performance
- Limited use as shard key
- Slightly complex syntax for beginners

11 Conclusion

Geospatial queries in MongoDB provide powerful tools for handling location-based data. With proper indexing and understanding of operators such as $near, $geoWithin, and $geoNear, developers can build highly efficient and scalable location-aware applications.