
# E-Portfolio

by Rafael Lüdtke

# Content

- What is MongoDB
- Key Features
- Advanced Features
- CRUD Operation

# What is MongoDB

# What is MongoDB

- document-oriented database
- designed for
  - ease of application development
  - scaling

```
{
  name: "sue",                          <-- field: value
  age: 26,                              <-- field: value
  status: "A",                          <-- field: value
  groups: [ "news", "sports" ]          <-- field: value
}
```

# What is MongoDB

| SQL Terms/Concepts | MongoDB Terms/Concepts |
|---|---|
| database | database |
| table | collection |
| row | document or BSON document |
| column | field |
| primary key<br>Specify any unique column or column combination as primary key. | primary key<br>In MongoDB, the primary key is automatically set to the _id field. |

Visit https://www.mongodb.com/docs/manual/reference/sql-comparison/ for an in depth comparison

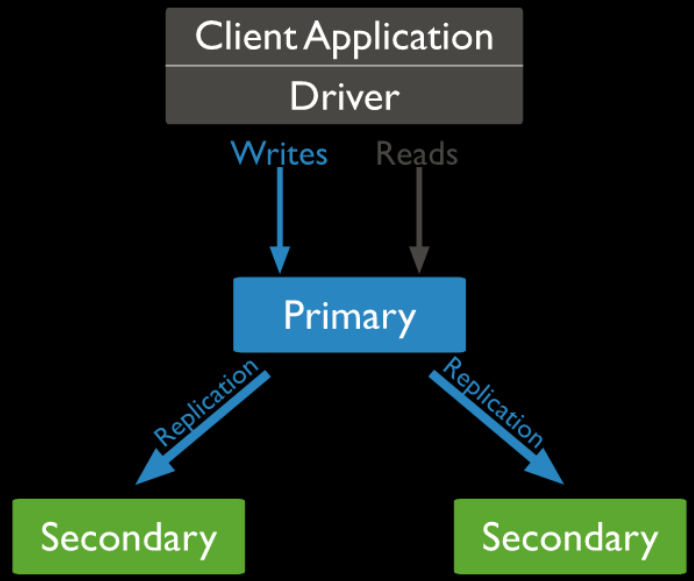
# Key Features

# Key Features

- High Performance
  - embedded data models reduces I/O activity
- Query API
  - Data Aggregation
  - Text Search and Geospatial Queries.
- supports multiple storage engines

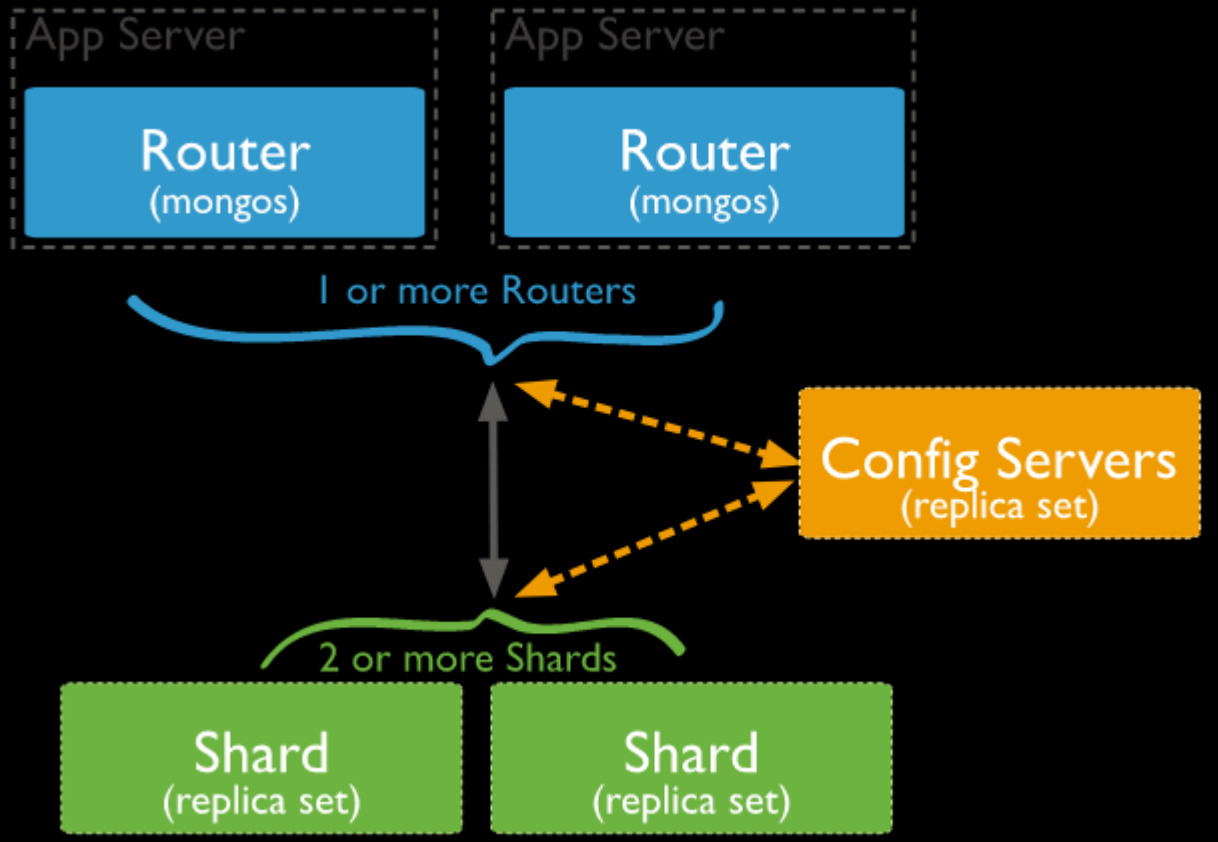
# Key Features

- replication facility, called replica set
  - automatic failover
  - data redundancy
- A replica set is a group of MongoDB servers that maintain the same data set, providing redundancy



# Key Features

- Horizontal Scalability (Sharding) as part of core functionality
  - shard: a subset of the sharded data.
  - mongos: query router, providing an interface between client applications and the sharded cluster.
  - config servers: store metadata and configuration settings



# Validation

# Validation

- validationLevel option: how strictly validation rules apply to existing documents during an update.

- strict (the default): MongoDB applies validation rules to all inserts and updates.
- moderate: to existing documents that already fulfill the validation criteria. Updates that do not fulfill the criteria are not checked.

# Validation

- validationAction option: determines whether MongoDB should

- error and reject documents that violate the validation rules or
- warn about the violations in the log but allow invalid documents.

# Validation

```
db.runCommand( {
   collMod: "contacts",
   validator: { $jsonSchema: {
      bsonType: "object",
      required: [ "phone", "name" ],
      properties: {
         phone: {
            bsonType: "string",
            description: "must be a string and is required"
         },
         name: {
            bsonType: "string",
            description: "must be a string and is required"
         }
      }
   } },
   validationLevel: "moderate“
   validationAction: "warn"
} )
```

# Validation

```
db.createCollection("students", {
   validator: {
      $jsonSchema: {
            ....
```

# Geospatial Queries

# Geospatial Queries

- GeoJSON data
- <field>: { type: <GeoJSON type> , coordinates: <coordinates> }
- GeoJSON object type examples:
  - Point
  - LineString
  - Polygon

longitude first
latitude second

# Geospatial Queries

- geospatial index types to support the geospatial queries:
  - 2dsphere indexes support queries that calculate geometries on an earth-like sphere
  - 2d indexes support queries that calculate geometries on a two-dimensional plane
- db.collection.createIndex( { <location field> : "2dsphere" } )

# Geospatial Queries

- Suppose the user is located at -73.93414657 longitude and 40.82302903
- Find the Current Neighborhood:

- db.neighborhoods.findOne({ geometry: { $geoIntersects: { $geometry: { type: "Point", coordinates: [ -73.93414657, 40.82302903 ] } } } })

# Geospatial Queries

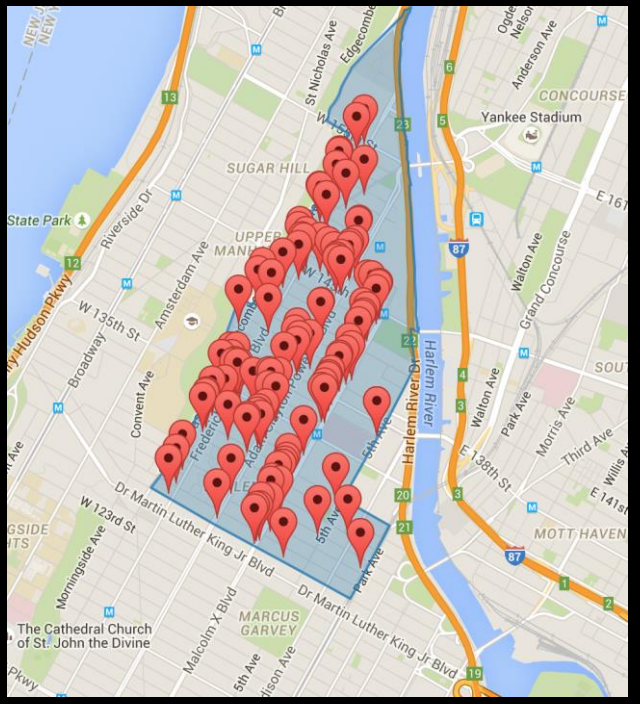
```
{
   "_id" : ObjectId("55cb9c666c522cafdb053a68"),
   "geometry" : {
      "type" : "Polygon",
      "coordinates" : [
         [
            [
               -73.93383000695911,
               40.81949109558767
            ],
            ...
         ]
      ]
   },
   "name" : "Central Harlem North-Polo Grounds"
}
```

# Geospatial Queries

- `var neighborhood = db.neighborhoods.findOne({ geometry: { $geoIntersects: { $geometry: { type: "Point", coordinates: [ -73.93414657, 40.82302903 ] } } } })`

- `db.restaurants.find( { location: { $geoWithin: { $geometry: neighborhood.geometry } } } ).count()`

127 restaurants in the requested neighborhood



# CRUD Operation Live Demo