NoSql –Databases MongoDB

COURSE 2: Databases

NoSql vs Relational

NoSql vs Relational

- Optimizez reads vs optimizez writes. → normalized vs denormalized data.

- Transactions → ACID vs BASE

- SQL vs no-standard

- Consistency vs Availability and Performance

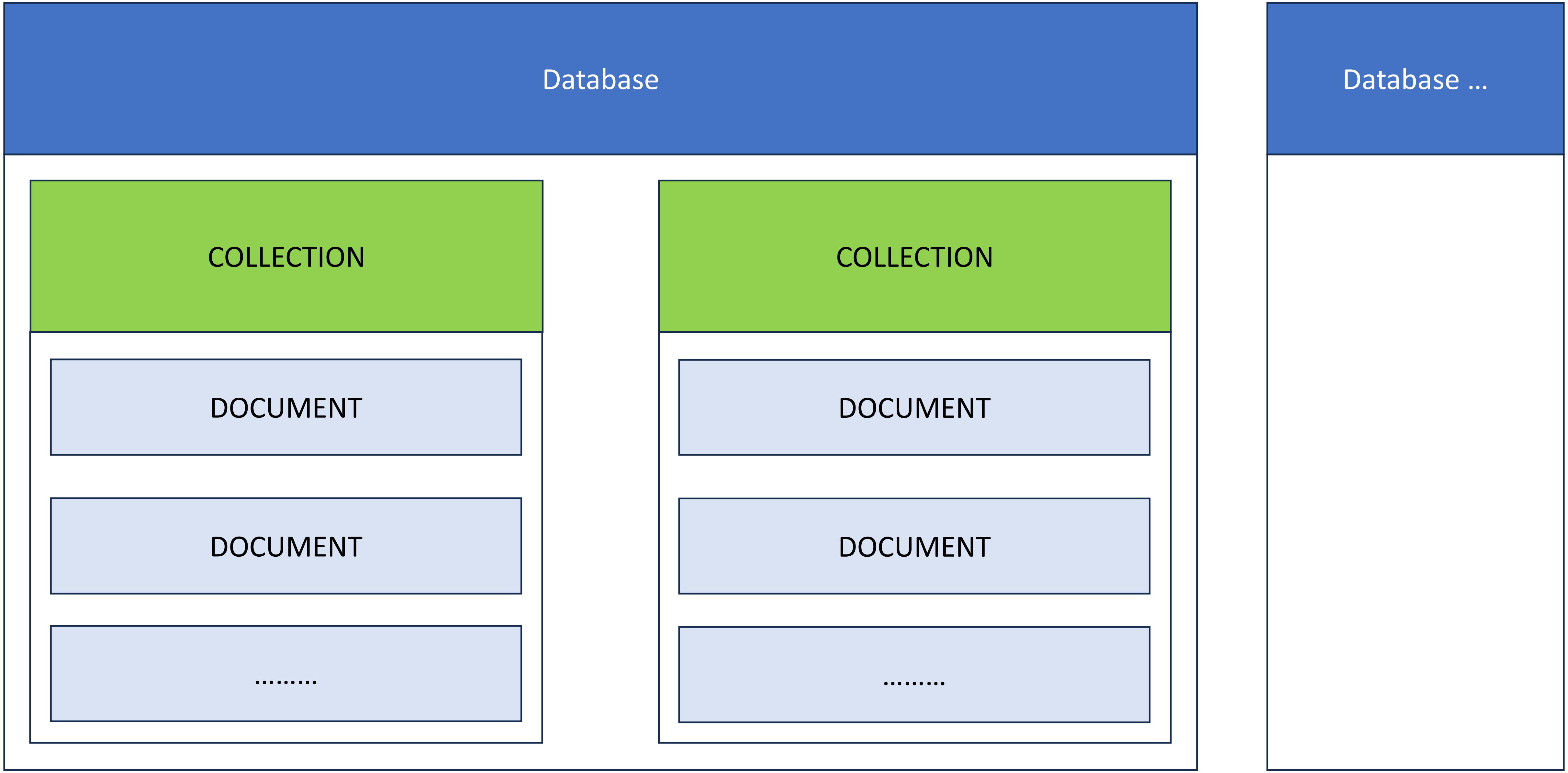
MongoDB-Introduction

MongoDB -- Humongous

- Flexible document model, schemaless. --> flexibility
- Suitable for large datasets. --> scalability
- Application oriented (document structure similar to JSON, Mongo shell allow interacts with the database via Java Script) – BSON format.
- Supports multiple indices (only one for sharding), geospatial indices optimize geospatial queries --> high performance.
- Built-in aggregation capabilities (Hadoop like architecture).

MongoDB

Mongo	**RDBMS**
document: set of **key-value pairs**, similar to JSON objects	row in a table
collection: set of documents, documents in a collection may have different sets of fields	table
field in JSON/BSON document	column
$lookup and embedded documents direct access to nested information, rapid read/write operations.	Join information is distributed into multiple tables, we need joins to regather normalized data.
primary key	Primary key GUID



Database

COLLECTION

DOCUMENT

DOCUMENT

………

COLLECTION

DOCUMENT

DOCUMENT

………

Database …

MongoDB-Datatypes

MongoDB - datatypes

- Numeric types
- Strings
- Boolean
- Date and Timestamp
- Embedded objects
- Arrays

- Data is presented in as unstructured-denormalized form.
- Data types like arrays and embedded objects may represent hierarchical relationships
  - one-to-one, one-to-many ore many-to-many relationships.

```
{
        "_id": { "$oid": "573a1391f29313caabcd7626"},

        "available" : true
        "plot":   "A nobleman becomes the vigilante Robin Hood who protects the oppressed English
                  people from the tyrannical Prince John.",
        "title": "Robin Hood",
        "genres": [
                  "Adventure",
                  "Romance",
                  "Family"
        ],
        "runtime": 143,
        "year": 1922,
        "released": {
                  "$date": {
                            "$numberLong": "-1489708800000"
                  }
        }
}
```

BSON, Unique identifier, can be omitted
“” can also be omitted for the keys.

```
{
        "_id": {"$oid": "573a1391f29313caabcd7626"},

        "tomatoes": {
                        "viewer": {
                                "rating": 3.6,
                                "numReviews": 659,
                                "meter": 70
                        },
                        "critic": {
                                "rating": 8.4,
                                "numReviews": 8,
                                "meter": 100
                        },
        }
}
```

Nesting documents: up to 100 levels/16mb

MongoDB - Interaction with applications

MongoDB -- products

- Local database.
- Mobile database.
- mongo import, mongo shell …
- Cloud database – Atlas.
- Compass – MongoDb client.

- Stitch
  - Query API to be used in clients like Node.js backend applications.
  - Serverless Functions: execute JavaScript code on the cloud without backend applications.
  - Database triggers.
  - Sync options.

- BI Connections and Charts.

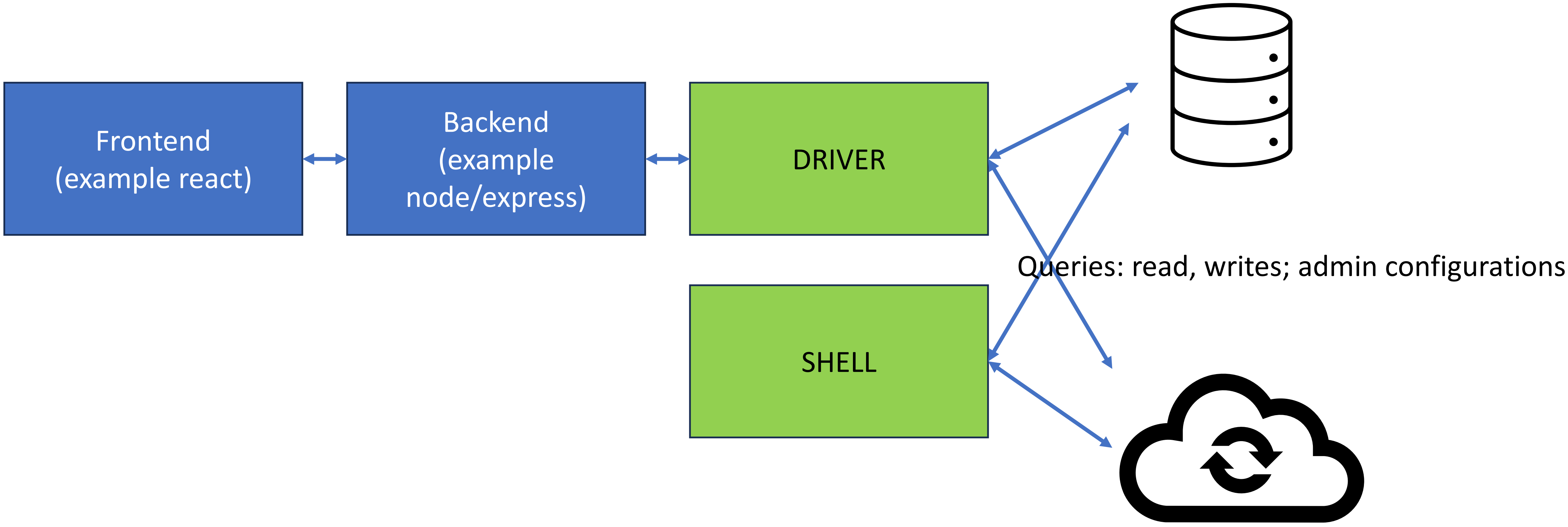
MongoDB - drivers

- **Driver**: bridge between the database server and other programming languages.

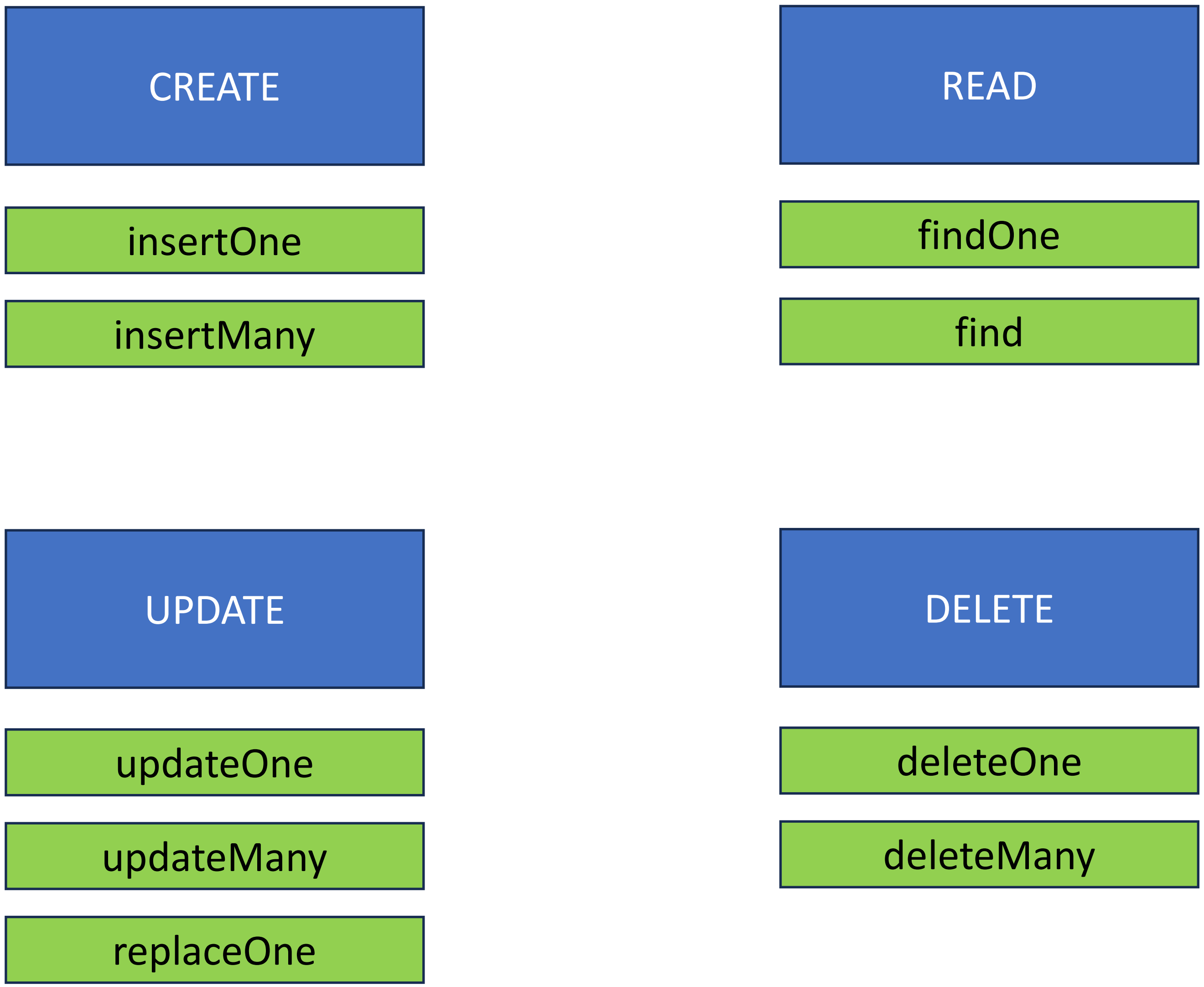
- programming language code (can be integrated in applications)
  →**driver** →
  database command (can be testes in mongo shell).

- https://www.mongodb.com/docs/drivers/

- Python driver **pymongo**
- Node.js package **mongodb**


Frontend
(example react)
Backend
(example
node/express)
DRIVER
SHELL
Queries: read, writes; admin configurations

MongoDB - CRUD



CREATE

insertOne

insertMany

READ

findOne

find

UPDATE

updateOne

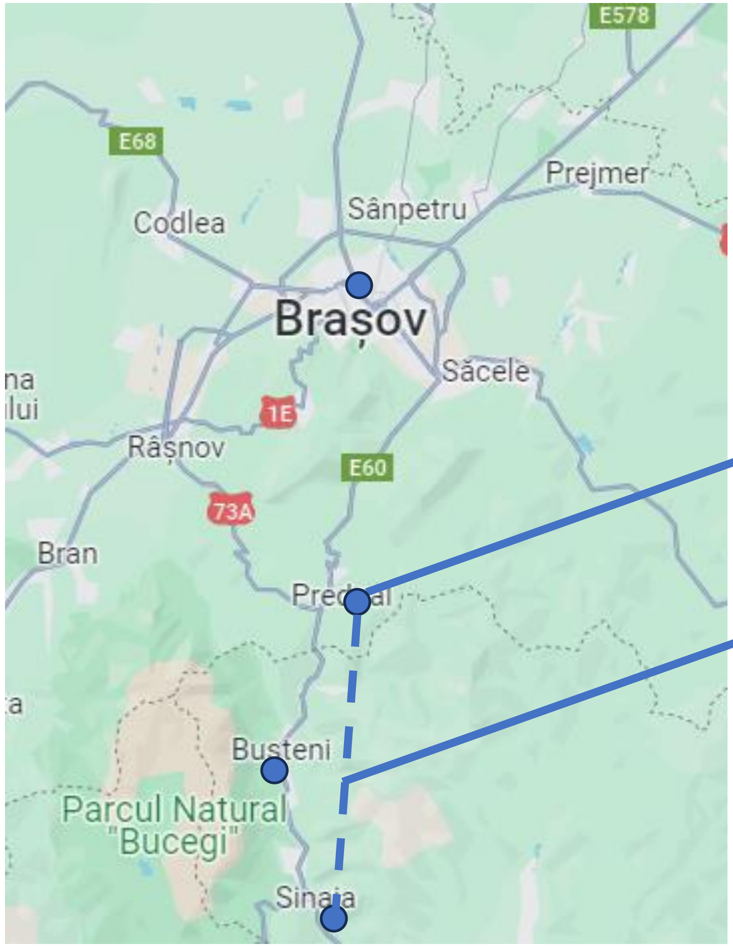
updateMany

replaceOne

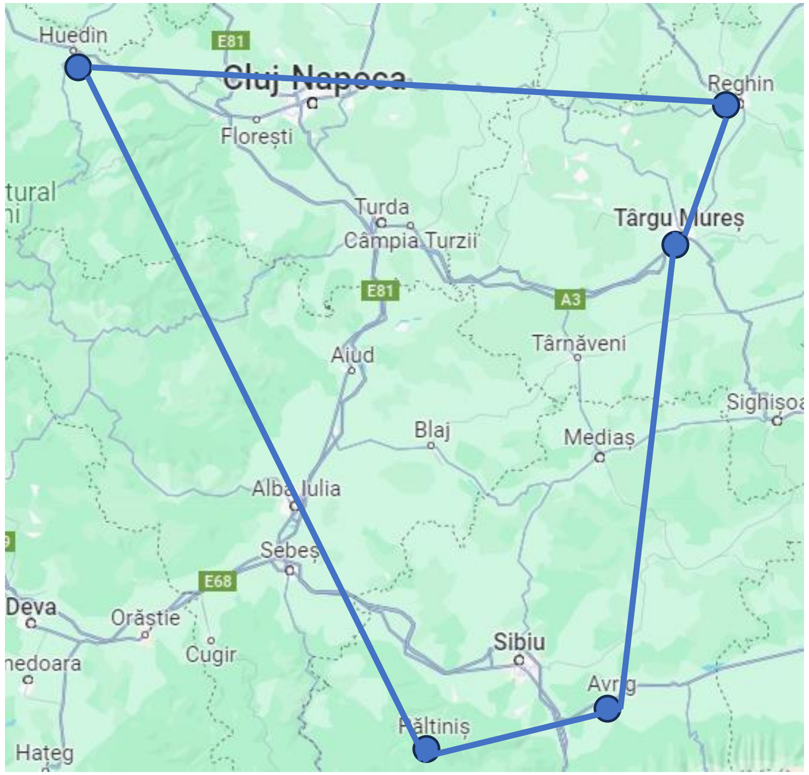
DELETE

deleteOne

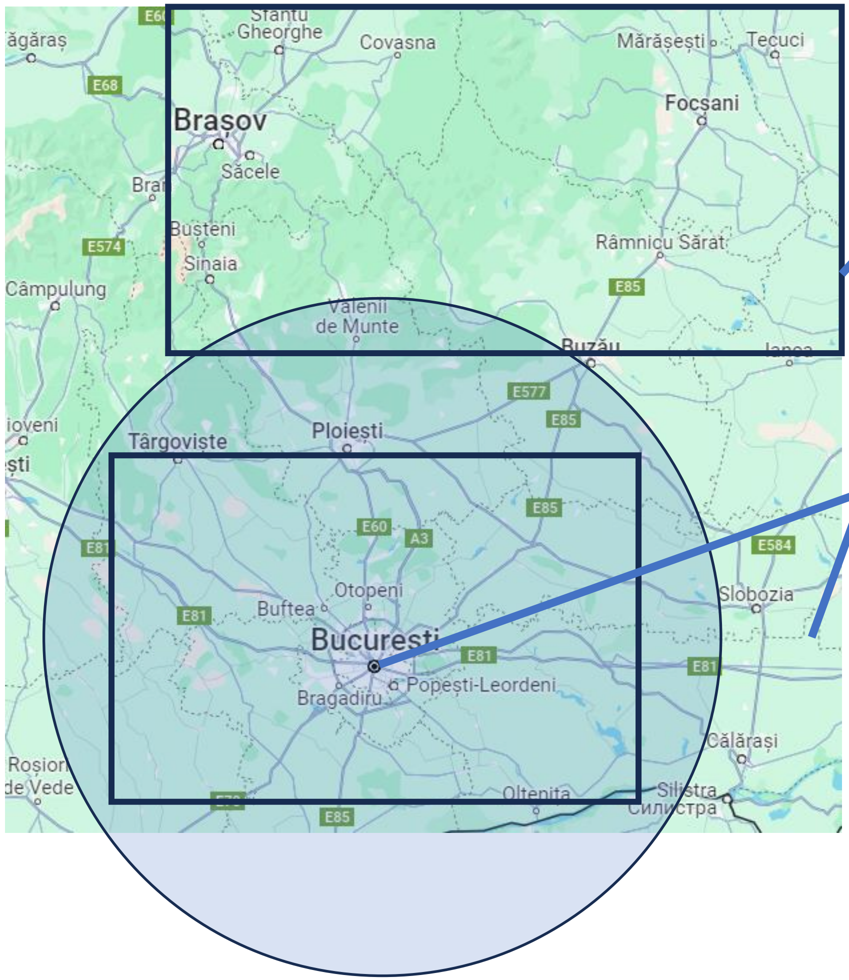
deleteMany



```
db.stations.find({
    location: {
      $nearSphere: {
        $geometry: {
           type: "Point" ,
           coordinates: [ 45.49953016353493, 25.598530301384987 ]
        },
        $maxDistance: 50000,
        $minDistance: 500
      }
    }
    }
 )
```



```
db.stations.find({
    location: {
       $geoWithin: {
          $geometry: {
             type: "Polygon",
             coordinates: [
                            [
                             [46.867002591202414, 23.023213129823013],
                             [46.775339331357955, 24.704595333123272],
                             [46.55026258621749, 24.54292396742133],
                             [45.727709238210856, 24.372014237964983],
                             [45.65995352460454, 23.937811141508323],
                             [46.867002591202414, 23.023213129823013]
                           ]
                        ]
          }
       }
    }
 }
)
```



```
db.zones.find(
{
      "geometry": {
         $geoIntersects:
            { $centerSphere: [ [ 44.42015468729325, 26.11338537573116 ],
                               200 / 3963.2 ]
            }
      }
   }
)
```