Replica Set - a few connected machines that store the same data to ensure that if something happens to one of the machines the data will remain intact. Comes from the word replicate - to copy something.

Instance - a single machine locally or in the cloud, running a certain software, in our case it is the MongoDB database.

Cluster - group of servers that store your data.

Import Export & Queuing

To learn more about other `mongoimport` supported formats check out this documentation page.

SRV connection string - a specific format used to establish a connection between your application and a MongoDB instance. Click here to learn more.

Code used in this lecture:

```
mongodump --uri "mongodb+srv://<your username>:<your password>@<your cluster>.mongodb.net/sample_supplies"
```

```
mongoexport --uri="mongodb+srv://<your username>:<your password>@<your cluster>.mongodb.net/sample_supplies" --collection=sales --out=sales.json
```

```
mongorestore --uri "mongodb+srv://<your username>:<your password>@<your cluster>.mongodb.net/sample_supplies"  --drop dump
```

```
mongoimport --uri="mongodb+srv://<your username>:<your password>@<your cluster>.mongodb.net/sample_supplies" --drop sales.json
```

Namespace - The concatenation of the database name and collection name is called a namespace.

We looked at the `sample_training.zips` collection and issued the following queries:

- `{"state": "NY"}`
- `{"state": "NY", "city": "ALBANY"}`

Query1

In the sample_training.trips collection a person with birth year 1961 took a trip that started at "Howard St & Centre St". What was the end station name for that trip?

{"start station name" : "Howard St & Centre St", "birth year" : 1961}

`"South End Ave & Liberty St"`

Connect to the Atlas cluster:

mongo "mongodb+srv://<username>:<password>@<cluster>.mongodb.net/admin"

show dbs

use sample_training

show collections

db.zips.find({"state": "NY"})

`it` iterates through the cursor

db.zips.find({"state": "NY"}).count()

db.zips.find({"state": "NY", "city": "ALBANY"})

db.zips.find({"state": "NY", "city": "ALBANY"}).pretty()

Commands

mongo # connects to mongodb://127.0.0.1:27017 by default

mongo --host <host> --port <port> -u <user> -p <pwd> # omit the password if you want a prompt

mongo "mongodb://192.168.1.1:27017"

mongo "mongodb+srv://cluster-name.abcde.mongodb.net/<dbname>" --username <username> # MongoDB Atlas

show dbs

db // prints the current database

use <database_name>

show collections

CRUD operations

Create

db.coll.insertOne({name: "Max"})

db.coll.insert([{name: "Max"}, {name:"Alex"}]) // ordered bulk insert

db.coll.insert([{name: "Max"}, {name:"Alex"}], {ordered: false}) // unordered bulk insert

db.coll.insert({date: ISODate()})

db.coll.insert({name: "Max"}, {"writeConcern": {"w": "majority", "wtimeout": 5000}})

db.coll.findOne() // returns a single document

db.coll.find() // returns a cursor - show 20 results - "it" to display more

db.coll.find().pretty()

db.coll.find({name: "Max", age: 32}) // implicit logical "AND".

db.coll.find({date: ISODate("2020-09-25T13:57:17.180Z")})

db.coll.distinct("name")

```
// Count
db.coll.count({age: 32})         // estimation based on collection metadata
db.coll.estimatedDocumentCount()  // estimation based on collection metadata
db.coll.countDocuments({age: 32}) // alias for an aggregation pipeline - accurate count

// Comparison
db.coll.find({"year": {$gt: 1970}})
db.coll.find({"year": {$gte: 1970}})
db.coll.find({"year": {$lt: 1970}})
db.coll.find({"year": {$lte: 1970}})
db.coll.find({"year": {$ne: 1970}})
db.coll.find({"year": {$in: [1958, 1959]}})
db.coll.find({"year": {$nin: [1958, 1959]}})

// Logical
db.coll.find({name:{$not: {$eq: "Max"}}})
db.coll.find({$or: [{"year" : 1958}, {"year" : 1959}]})
db.coll.find({$nor: [{price: 1.99}, {sale: true}]})
db.coll.find({ $and: [ {$or: [{qty: {$lt :10}}, {qty :{$gt: 50}}]},
   {$or: [{sale: true}, {price: {$lt: 5 }}]}  ]})

// Element
db.coll.find({name: {$exists: true}})
db.coll.find({"zipCode": {$type: 2 }})
db.coll.find({"zipCode": {$type: "string"}})

// Aggregation Pipeline
db.coll.aggregate([
  {$match: {status: "A"}},
  {$group: {_id: "$cust_id", total: {$sum: "$amount"}}},
  {$sort: {total: -1}}
```

```
])

// Array
db.coll.find({tags: {$all: ["Realm", "Charts"]}})
db.coll.find({field: {$size: 2}}) // impossible to index - prefer storing the size of the array & update it
db.coll.find({results: {$elemMatch: {product: "xyz", score: {$gte: 8}}}})

// Projections
db.coll.find({"x": 1}, {"actors": 1})              // actors + _id
db.coll.find({"x": 1}, {"actors": 1, "_id": 0})    // actors
db.coll.find({"x": 1}, {"actors": 0, "summary": 0}) // all but "actors" and "summary"

// Sort, skip, limit
db.coll.find({}).sort({"year": 1, "rating": -1}).skip(10).limit(3)


// FindOneAndUpdate
db.coll.findOneAndUpdate({"name": "Max"}, {$inc: {"points": 5}}, {returnNewDocument: true})
// Upsert
db.coll.update({"_id": 1}, {$set: {item: "apple"}, $setOnInsert: {defaultQty: 100}}, {upsert: true})
// Replace
db.coll.replaceOne({"name": "Max"}, {"firstname": "Maxime", "surname": "Beugnet"})
```