Introduction to MongoDB

Naveen Pete

Tuesday, April 10, 2018

Agenda

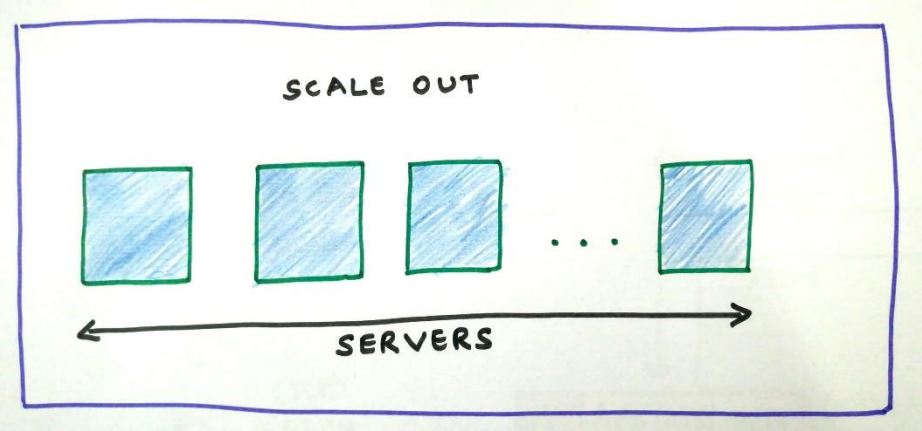
- NoSQL – What & Why?
- Overview of MongoDB
- Installing MongoDB
- JSON
- BSON
- CRUD using Mongo Shell
- MongoDB Node.js Driver
- Q & A

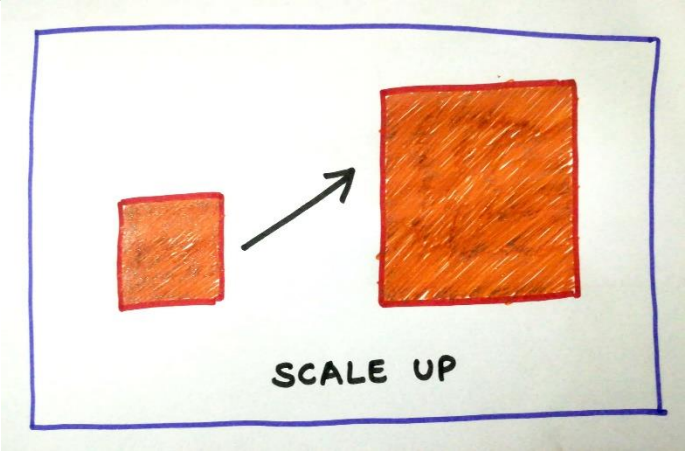
NoSQL – What & Why?

- “Not only SQL”
- Alternative to traditional relational databases
- Useful for working with large sets of distributed data
- Benefits
  - Better performance
  - Large volumes of rapidly changing data
    - Structured, semi-structured and unstructured
  - Agile sprints, quick schema iteration, frequent code pushes
  - OOP that is easy to use and flexible
  - Geographically distributed scale-out architecture instead of expensive, monolithic (scale-up) architecture

NoSQL – What & Why?

- Scalability





Overview of MongoDB

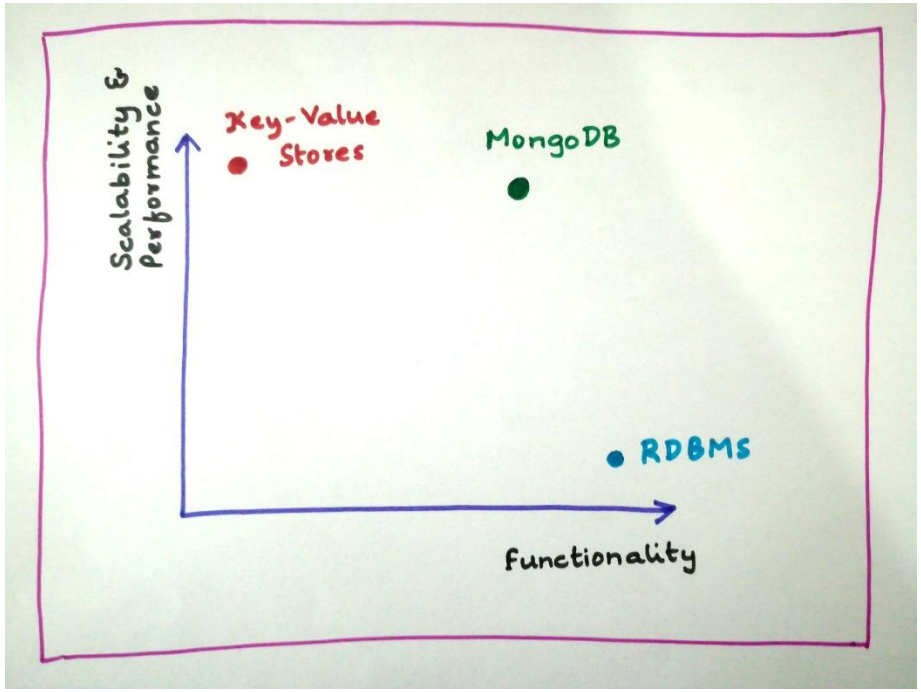
- MongoDB
  - Is an open-source document database
  - Provides high performance, high availability, and automatic scaling
- Documents
  - A record in MongoDB is a document
    - Document is a data structure composed of field and value pairs
  - Are similar to JSON objects
  - Field values may include other documents, arrays, and arrays of documents
- Collections
  - Documents are stored in collections
  - Are analogous to tables in Relational databases
  - A collection does not require its documents to have same schema

Overview of MongoDB

- Key Features
  - High Performance
    - Support for embedded data models reduces I/O activity
    - Indexes support faster queries
  - Rich Query Language
    - CRUD operations
    - Data aggregation
    - Text search & Geospatial queries
  - High Availability
    - Using replica set, provides
      - Automatic failover
      - Data redundancy
  - Horizontal Scalability
    - Sharding distributes data across a cluster of machines
    - Provided as part of its core functionality

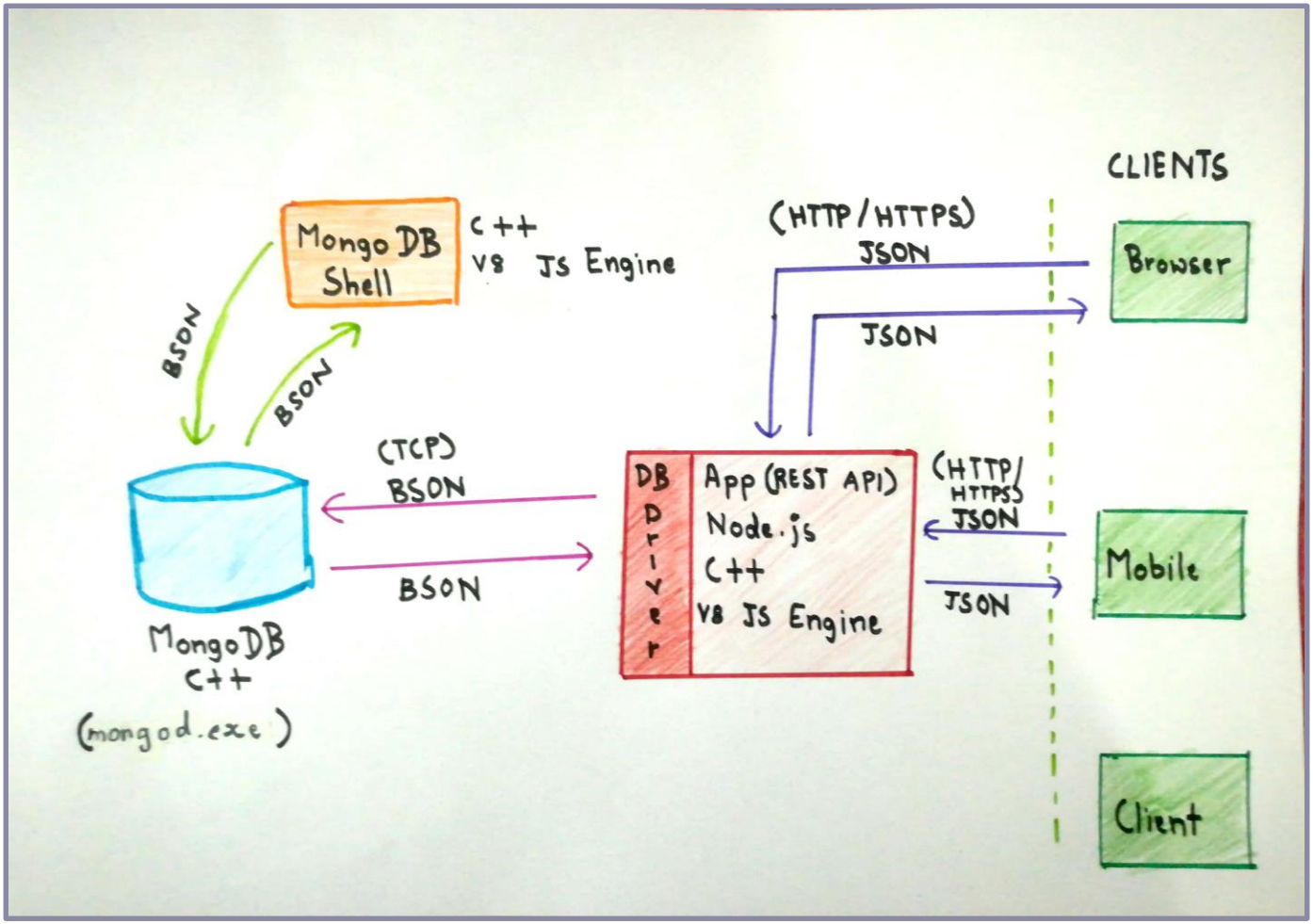
Overview of MongoDB

- Key Features (Continued)
  - Support for Multiple Storage Engines
    - WiredTiger, In-Memory, MMAPv1
    - Pluggable storage engine API that allows third parties to develop storage engines



Overview of MongoDB

- Application Architecture



Installing MongoDB

- Install on Windows
  - Download MongoDB Community Edition .msi file from MongoDB Download Center
    - https://www.mongodb.com/download-center#community
  - Double-click the msi file. Follow the steps in the installation wizard
  - MongoDB is installed in 'C:\Program Files\MongoDB\Server\<version>\bin' directory
    - <version> is the version number, for e.g., '3.6'
  - Two executables within this directory are important
    - mongod.exe – This is the MongoDB server
    - mongo.exe – This is the MongoDB shell
  - Before working with MongoDB,
    - Include the path to installation directory within 'Path' environment variable
    - Make sure that the data directory ('\data\db') is created in 'C' drive. MongoDB stores data in this directory

Installing MongoDB

- Install on Linux
  - For setup instructions on Linux, visit
    - https://docs.mongodb.com/manual/administration/install-on-linux/

- Install on macOS
  - For setup instructions on macOS, visit
    - https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/

JSON

- JSON objects
  - Are composed of key-value pairs
  - Keys are strings
  - Keys and values are separated from one another using a colon
  - Fields are separated using commas
  - Are enclosed within curly braces
  - Support following value types
    - String
    - Date
    - Number
    - Boolean
    - Array
    - Object
  - One object can be nested within another
  - For more information on JSON, visit: http://json.org/

JSON

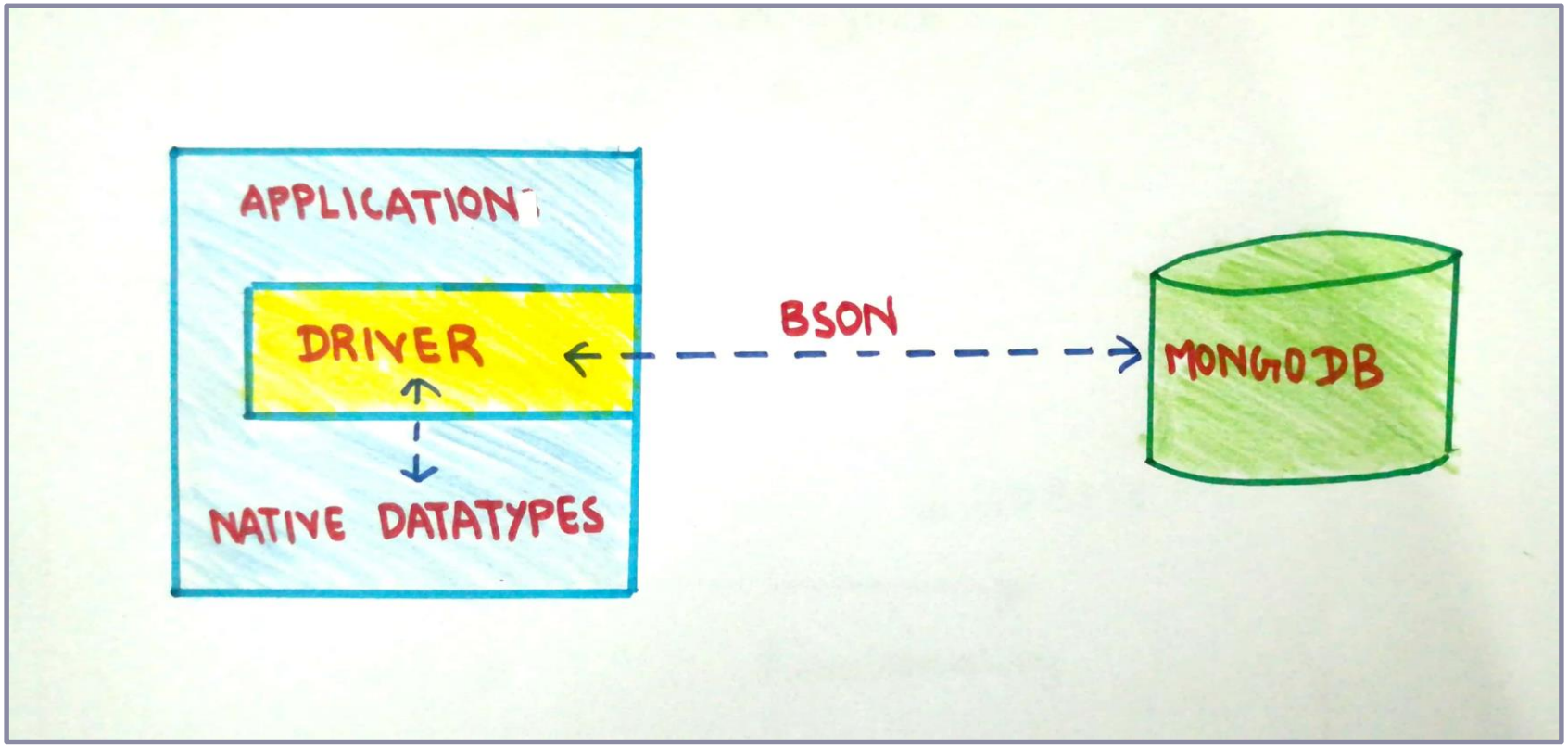
- JSON Example

```
{
  "headline" : "Apple Reported Fourth Quarter Revenue Today",
  "date" : "2015-10-27T22:35:21.908Z",
  "views" : 1132,
  "author" : {
     "name" : "Bob Walker",
     "title" : "Lead Business Editor“
  },
  "published" : true,
  "tags" : [
     "AAPL",
     { "name" : "city", "value" : "Cupertino" },
     [ "Electronics", "Computers" ]
  ]
}
```

BSON

- MongoDB stores data as BSON – Binary JSON
- BSON is
  - *Lightweight* – spare required to represent data is kept to minimum
  - *Traversible* – to support the variety of operations necessary for writing, reading and indexing MongoDB documents
  - *Efficient* – encoding data to BSON and decoding data from BSON can be performed quickly
- BSON extends JSON value types to include
  - Integers, doubles, dates and binary data (to support images), etc.
- MongoDB drivers
  - Send and receive data as BSON
  - Map BSON to native data types appropriate for a given programming language
- For more information on BSON, visit: http://bsonspec.org/

BSON



CRUD using Mongo Shell

- The Mongo Shell
  - Fully functional MongoDB client application
  - An interactive JavaScript interface to MongoDB
  - Used to
    - Query and update data
    - Perform administrative operations

- In MongoDB
  - Documents are stored in collections
  - Collections are organized into databases
  - A collection and the database that contains it form a namespace.
    - For e.g., the namespace ‘video.movies’ would specify the ‘movies’ collection in the ‘video’ database

CRUD using Mongo Shell

- Running MongoDB Server
  - Open Windows Command Prompt
  - Enter ‘mongod’ command to start the server
  - By default, the server runs and listens on port #27017

- Running Mongo Shell (MongoDB client app)
  - Make sure that the server is started and running
  - Open a new Command Prompt window
  - Enter ‘mongo’ command to start the shell
  - The MongoDB shell displays the prompt where you can execute MongoDB commands to interact with the database server

CRUD using Mongo Shell

- CRUD in MongoDB

Operation	MongoDB	SQL
Create	Insert	Insert
Read	Find	Select
Update	Update	Update
Delete	Remove / Delete	Delete

  - Global variable ‘db’ holds a reference to the database that we are currently using

CRUD using Mongo Shell

- Mongo Shell Commands
  - help
    - Displays a list of commands available in the shell
  - show dbs
    - Lists database names
  - use <database-name>
    - Sets current database
    - For example, ‘use video’ sets ‘video’ as current database
  - show collections
    - Lists collections in current database

CRUD using Mongo Shell

- Inserting a document

```
use video;
db.movies.insertOne({
   "title": "Rocky", "year": "1976", "imdb": "tt0075148"
});

db.movies.insertOne({
   "_id": "tt0075148", "title": "Rocky", "year": "1976"
});
```

  - All documents must contain an _id field
    - The _id field
      - acts as a primary key
      - contains an unique identifier for a document within the collection
      - if not mentioned, MongoDB will automatically add it with a unique value

CRUD using Mongo Shell

- Inserting multiple documents

```
db.movies.insertMany([
   {
      "_id" : "tt0084726", "title" : "Star Trek II: The Wrath of Khan",
      "year" : 1982, "type" : "movie"
   },
   {
      "_id" : "tt0796366", "title" : "Star Trek",
      "year" : 2009, "type" : "movie"
   },
   {
      "_id" : "tt1408101", "title" : "Star Trek Into Darkness",
      "year" : 2013, "type" : "movie"
   },
   {
      "_id" : "tt0117731", "title" : "Star Trek: First Contact",
      "year" : 1996, "type" : "movie"
   }
]);
```

CRUD using Mongo Shell

- Reading documents
  - To query all the documents of ‘movies’ collection within video database, use these commands

```
use video;
db.movies.find();
db.movies.find().pretty()        // format the output neatly
```

  - To query a document by title

```
db.movies.find({ title: “Jaws” }).pretty();
```

  - To query documents by year

```
db.movies.find({ year: 1981 }).pretty();
```

CRUD using Mongo Shell

- Reading documents with conditions & operators
  - Retrieve all documents from the inventory collection where status equals either "A" or "D"

```
db.inventory.find( { status: { $in: [ "A", "D" ] } } )
```

  - Retrieve all documents in the inventory collection where the status equals "A" and qty is less than 30

```
db.inventory.find( { status: "A", qty: { $lt: 30 } } )
```

  - Retrieve all documents in the collection where the status equals "A" or qty is less than 30

```
db.inventory.find( { $or: [
   { status: "A" }, { qty: { $lt: 30 } }
] } )
```

CRUD using Mongo Shell

- Reading documents with field selection
  - Return all documents that match the query, also return only ‘item’, ‘status’ and ‘_id’ fields in the result

```
db.inventory.find( { status: "A" }, { item: 1, status: 1 } );
```

  - Suppress _id field

```
db.inventory.find(
   { status: "A" },
   { item: 1, status: 1, _id: 0 }
);
```

  - Return all but the excluded fields

```
db.inventory.find( { status: "A" }, { status: 0, instock: 0 } )
```

CRUD using Mongo Shell

- Reading documents
  - The return value of find() method is not an array of documents, it is instead a cursor object

```
var c = db.movies.find();
c.hasNext();
c.next();
```

  - Cursor objects' 'hasNext()' and 'next()' methods can be used to iterate through the returned documents

CRUD using Mongo Shell

- Updating a document

```
db.movies.updateOne({
  title: "The Martian"}, {
  $set: {
    poster: "http://ia.media-
imdb.com/images/M/MV5BMTc2MTQ3MDA1Nl5BMl5BanBnXkFtZTgwODA3OTI4
NjE@._V1_SX300.jpg"
  }
});
```

  - Here $set operator has been used
    - MongoDB will update the first document matching the selector
    - If the ‘poster’ field did not exist on the document, it will be added to the document. If it exists, it will be updated with the specified value

CRUD using Mongo Shell

- Updating a document

```
db.movies.updateOne({
  title: "The Martian“
}, {
  $set: {
    "awards": {
      "wins": 8,
      "nominations": 14,
      "text": "Nominated for 3 Golden Globes.
Another 8 wins & 14 nominations.“
    }
  }
});
```

CRUD using Mongo Shell

- Updating a document

```
db.movies.updateOne({
  title: "The Martian"}, {
  $inc: {
    "reviews": 3
  }
});
```

  - Here ‘$inc’ operator increments ‘reviews’ field by 3 for the document matching ‘The Martian’ title

CRUD using Mongo Shell

- Updating a document

```
db.movies.updateOne({
  title: "The Martian“
}, {
  $push: {
    reviews: {
      rating: 4.5,
      date: ISODate("2016-01-12T09:00:00Z"),
      reviewer: "Spencer H.",
      text: "Good movie to watch with the family!"
    }
  }
})
```

  - Here $push operator will add a new review document into the ‘reviews’ field. This field represents an array of reviews for the movie

CRUD using Mongo Shell

- Updating multiple documents

```
db.inventory.updateMany(
   { "qty": { $lt: 50 } },
   {
     $set: { "size.uom": "in", status: "P" },
     $currentDate: { lastModified: true }
   }
);
```

  - The update operation:
    - uses the $set operator to update the value of the ‘size.uom’ field to "in" and the value of the ‘status’ field to "P",
    - uses the $currentDate operator to update the value of the ‘lastModified’ field to the current date.

CRUD using Mongo Shell

- Updating a document with ‘upsert’ option

```
db.movies.updateOne({
  "imdb.id": “tt3659388”
}, {
  $set: {
    "title": "The Martian", "year": 2015, "rated": "PG-13",
    "released": ISODate("2015-10-02T04:00:00Z"),
    "runtime": 144, "countries": [ "USA", "UK“ ],
    "genres": [ "Adventure", "Drama", "Sci-Fi“ ],
    "director": "Ridley Scott",
    "writers": [ "Drew Goddard", "Andy Weir" ],
    "actors": [ "Matt Damon", "Jessica Chastain",
       "Kristen Wiig", "Jeff Daniels“ ]
  }
}, {
  upsert: true
});
```

CRUD using Mongo Shell

- Updating a document with ‘upsert’ option
  - Upsert is an operation for which
    - If no document is found matching the filter, a new document is inserted
    - If the document is found, it is updated

CRUD using Mongo Shell

- Replacing a document

```
db.inventory.replaceOne({ item: "paper" },
   {
      item: "paper",
      instock: [
         { warehouse: "A", qty: 60 },
         { warehouse: "B", qty: 40 }
      ]
   }
);
```

  - To replace the entire content of a document except for the _id field, pass an entirely new document as the second argument to replaceOne() method

CRUD using Mongo Shell

- Deleting documents
  - To delete all documents from a collection, pass an empty filter document {} to the deleteMany() method
    - Delete all documents from the inventory collection

```
db.inventory.deleteMany({})
```

    - Remove all documents from the inventory collection where the status field equals "A“

```
db.inventory.deleteMany({ status : "A" })
```

CRUD using Mongo Shell

- Deleting a document
  - Delete the first document where status is "D"

```
db.inventory.deleteOne( { status: "D" } )
```

MongoDB Node.js Driver

- The MongoDB Node.js driver
  - Is a Node.js module
  - Communicates with the MongoDB server using the wired protocol
  - Provides both callback-based and Promise-based interaction with MongoDB
  - Handles things like opening sockets, detecting errors, managing connections to replica sets etc.

MongoDB Node.js Driver

- Installing the driver
  - Open Windows Command Prompt
  - Create a directory for the Node.js app, for e.g. ‘video-app’
  - Change the directory to ‘video-app’
  - Run ‘npm init’ command to create the Node.js project
  - Run the following command in command prompt
    - npm install --save mongodb

MongoDB Node.js Driver

- Connecting to a MongoDB database in Node.js

```
const MongoClient = require('mongodb').MongoClient;

const url = 'mongodb://localhost:27017';
const dbName = 'video';
const collectionName = 'movieDetails';
const filter = { director: 'Steven Spielberg' };

MongoClient.connect(url, function(err, client) {
  if (err) {
    console.log('Connection failed:', err);
    return;
  }
  console.log('Successfully connected to server');
  const db = client.db(dbName);

  // Use db object to perform the necessary operation
  // ...

});
```

MongoDB Node.js Driver

- Displaying data from a collection in Node.js

```
// Find some documents in our collection
 console.log(
   `List of movies that match the filter
      '${JSON.stringify(filter)}'`
 );
 db
   .collection(collectionName)
   .find(filter)
   .toArray(function(err, docs) {
     // Print the documents returned
     docs.forEach(function(doc, ctr) {
       console.log(`   ${ctr + 1}. ${doc.title}`);
     });

     // Close the connection
     client.close();
   });
```

Q & A

- Thank you!