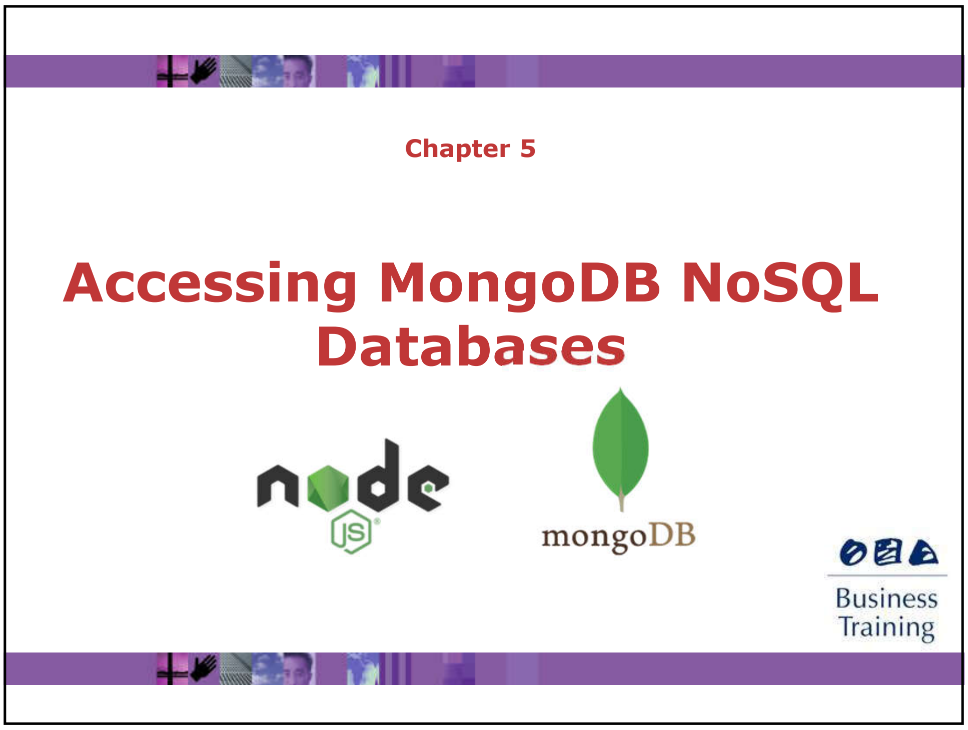


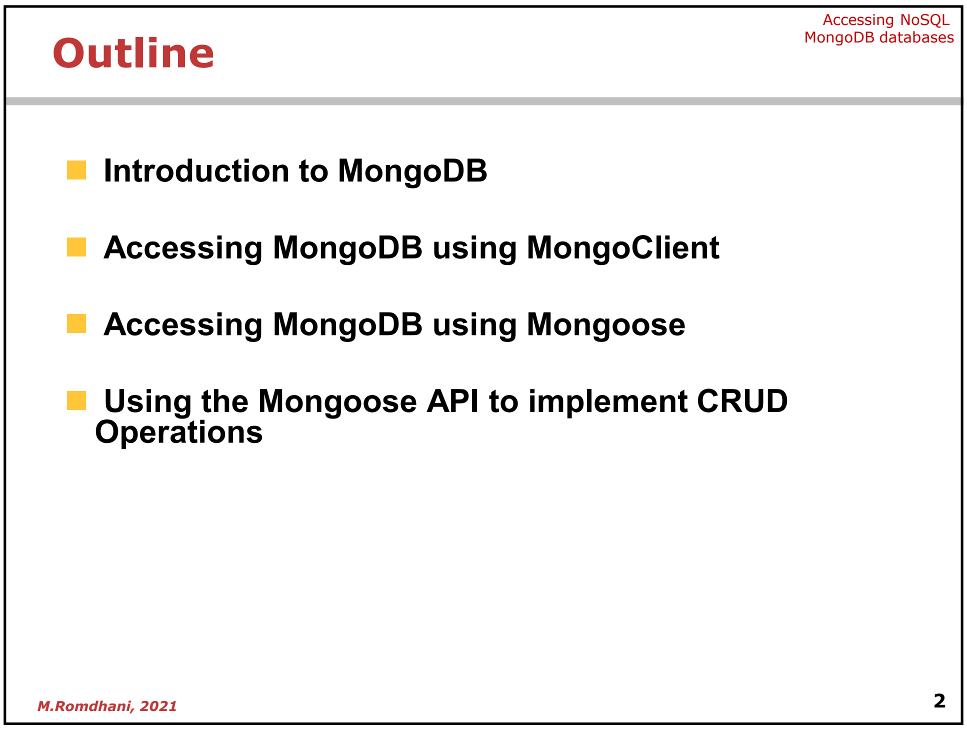
Chapter 5

Accessing MongoDB NoSQL Databases

mongoDB

Business Training

1



Accessing NoSQL MongoDB databases

Outline

- Introduction to MongoDB
- Accessing MongoDB using MongoClient
- Accessing MongoDB using Mongoose
- Using the Mongoose API to implement CRUD Operations

M.Romdhani, 2021

2

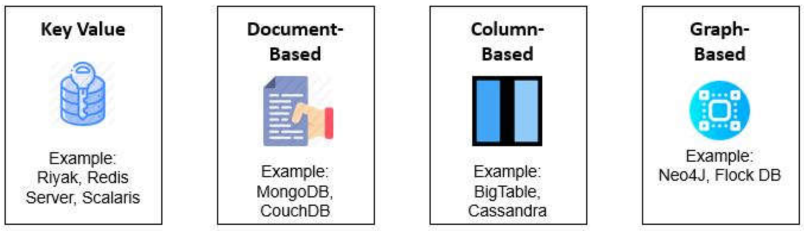
2

Introduction to MongoDB

3

Introduction to NoSQL Paradigm

- **ShortComings of Relational Databases**
  - **Scaling-Up is Expensive**
    - With ever increasing user growth and the huge amounts of data being generated, relational databases cannot provide on demand scalability.
  - **Rigid Data Models**
    - Any change in the data model requires a change in the schema which leads to creating new columns, defining new relations, reflecting the changes in your application, discussing with your database administrators etc.
- **NoSQL data Models**


Key Value
Example: Riyak, Redis Server, Scalaris
Document-Based
Example: MongoDB, CouchDB
Column-Based
Example: BigTable, Cassandra
Graph-Based
Example: Neo4J, Flock DB

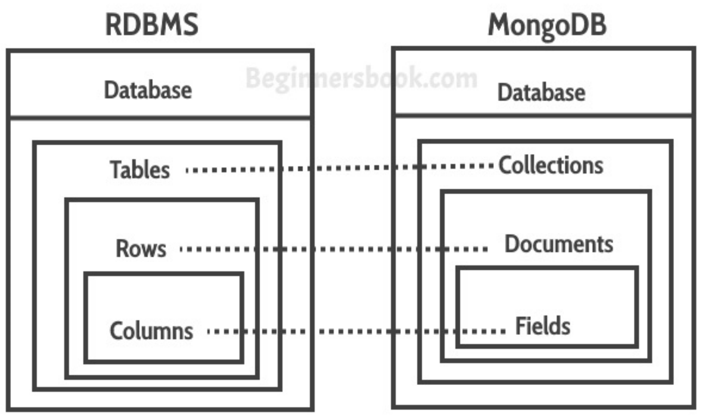
4

What is MongoDB ?

- **MongoDB is a NoSQL database which stores the data as JSON documents.**
- **It is an Open Source, Document Database which provides high performance and scalability along with data modelling and data management of huge sets of data in an enterprise application.**
- **What is Document based storage?**
  - A Document is nothing but a data structure with name-value pairs like in JSON.
  - Example

    {"name":"John", "age":31, "city":"Brussels"}

Relational DB vs MongoDb Concepts



Accessing MongoDB using MongoClient

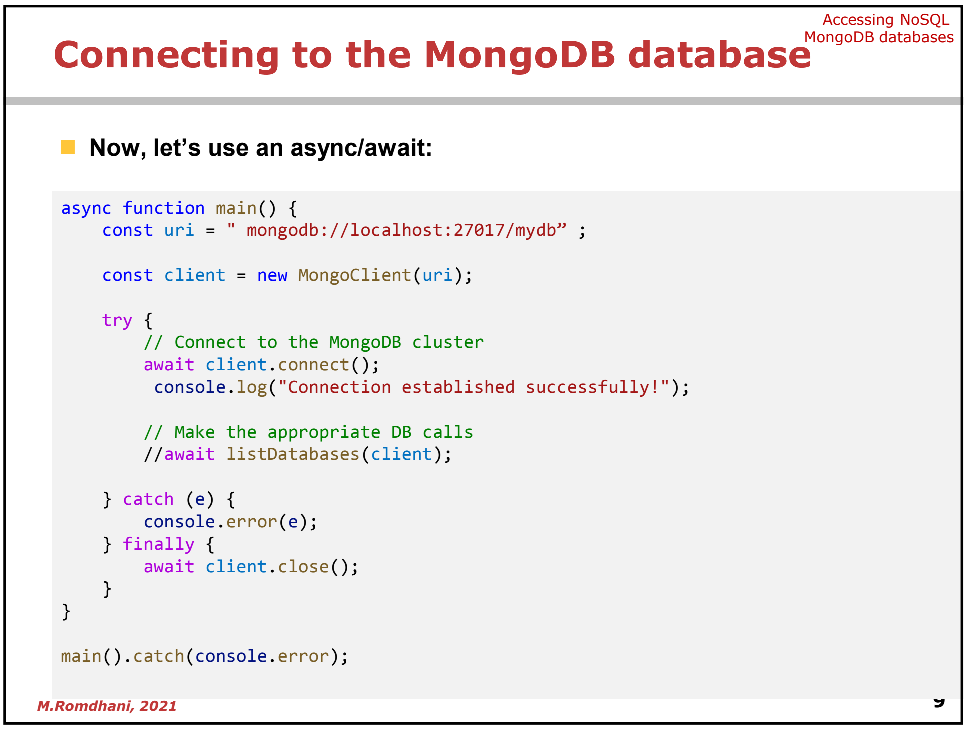
7

Connecting to the MongoDB database

- **Install the MongoDB driver using npm**
  - npm install mongodb –save
- **Connecting Node.js to MongoDB**

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/mydb";
// Connect to the db
MongoClient.connect(url, function (err, db) {
    if (err) throw err;
    console.log("Connection established successfully!");
    // Note: The db will be created if it does not exist
    db.close();
});
```

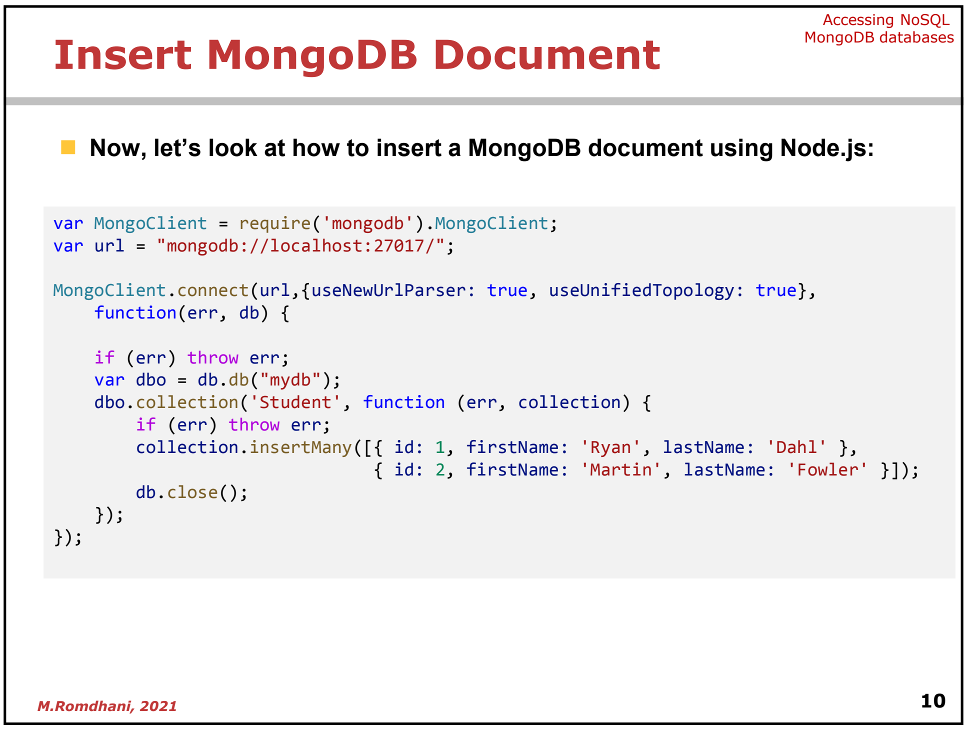
8



Connecting to the MongoDB database

- **Now, let’s use an async/await:**

```
async function main() {
    const uri = " mongodb://localhost:27017/mydb” ;

    const client = new MongoClient(uri);

    try {
        // Connect to the MongoDB cluster
        await client.connect();
         console.log("Connection established successfully!");

        // Make the appropriate DB calls
        //await listDatabases(client);

    } catch (e) {
        console.error(e);
    } finally {
        await client.close();
    }
}

main().catch(console.error);
```



Insert MongoDB Document

- **Now, let’s look at how to insert a MongoDB document using Node.js:**

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url,{useNewUrlParser: true, useUnifiedTopology: true},
    function(err, db) {

    if (err) throw err;
    var dbo = db.db("mydb");
    dbo.collection('Student', function (err, collection) {
        if (err) throw err;
        collection.insertMany([{ id: 1, firstName: 'Ryan', lastName: 'Dahl' },
                               { id: 2, firstName: 'Martin', lastName: 'Fowler' }]);
        db.close();
    });
});
```

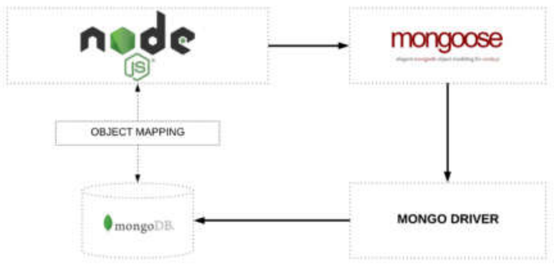
Accessing MongoDB using Mongoose

11

What is Mongoose

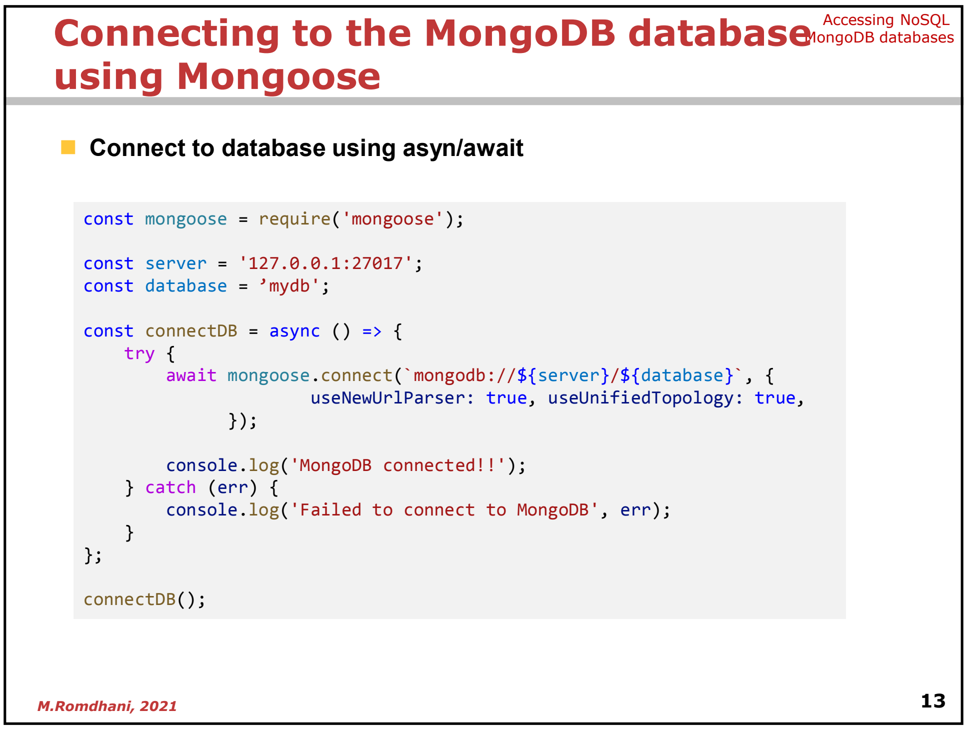
Accessing NoSQL MongoDB databases

- Mongoose: Mongoose is a MongoDB object modeling tool designed to work in an asynchronous environment.
- It is an Object Data Modeling (ODM) library for MongoDB and Node.js. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB.



M.Romdhani, 2021 12

12



Connecting to the MongoDB database using Mongoose

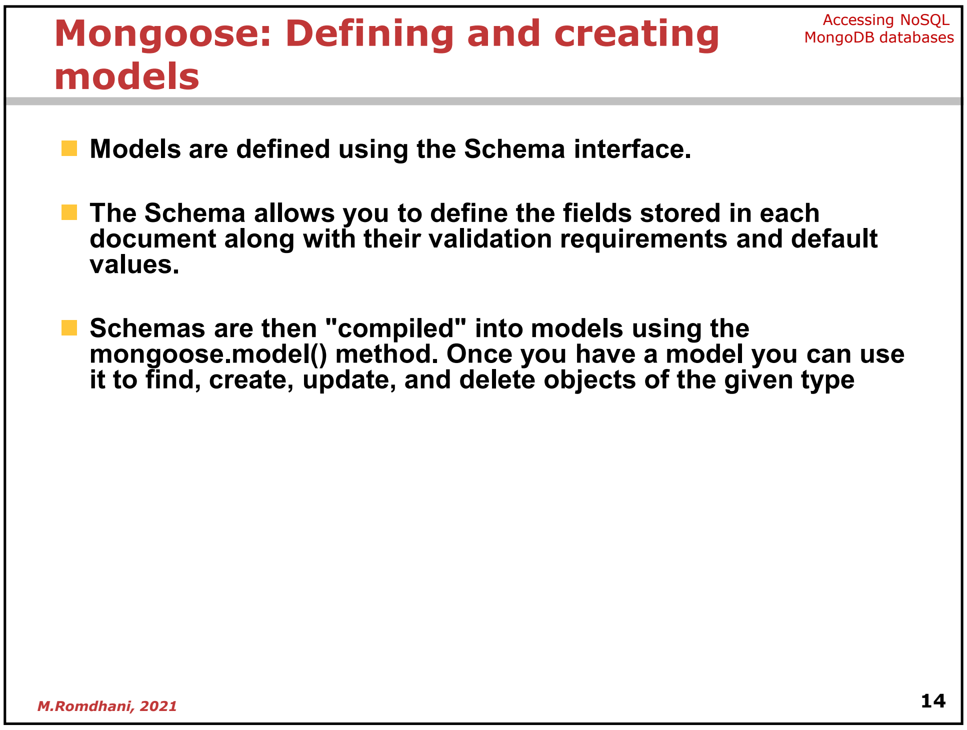
- Connect to database using asyn/await

```
const mongoose = require('mongoose');

const server = '127.0.0.1:27017';
const database = ’mydb';

const connectDB = async () => {
    try {
        await mongoose.connect(`mongodb://${server}/${database}`, {
                      useNewUrlParser: true, useUnifiedTopology: true,
              });

        console.log('MongoDB connected!!');
    } catch (err) {
        console.log('Failed to connect to MongoDB', err);
    }
};

connectDB();
```

M.Romdhani, 2021

13

13



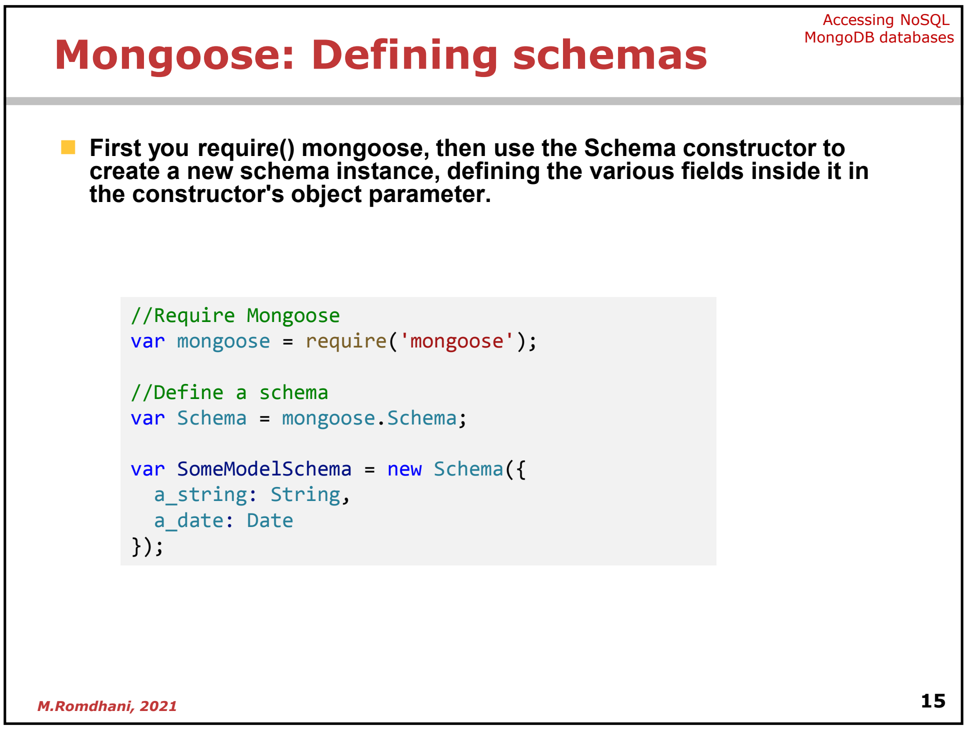
Mongoose: Defining and creating models

- Models are defined using the Schema interface.
- The Schema allows you to define the fields stored in each document along with their validation requirements and default values.
- Schemas are then "compiled" into models using the mongoose.model() method. Once you have a model you can use it to find, create, update, and delete objects of the given type

M.Romdhani, 2021

14

14



Accessing NoSQL
MongoDB databases

Mongoose: Defining schemas

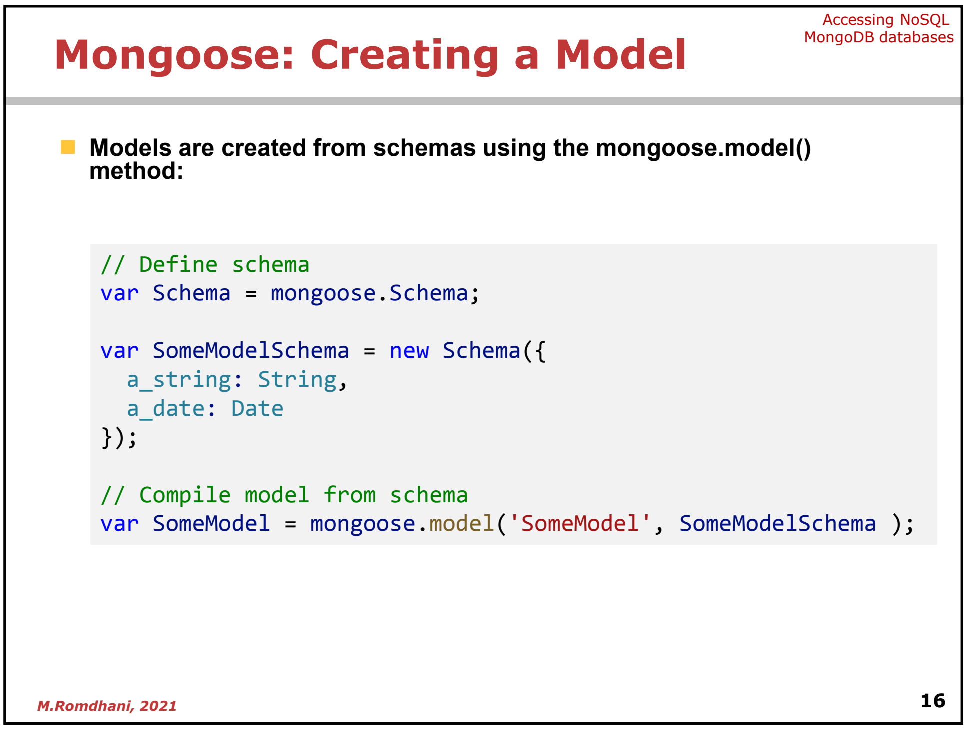
- **First you require() mongoose, then use the Schema constructor to create a new schema instance, defining the various fields inside it in the constructor's object parameter.**

```
//Require Mongoose
var mongoose = require('mongoose');

//Define a schema
var Schema = mongoose.Schema;

var SomeModelSchema = new Schema({
  a_string: String,
  a_date: Date
});
```

M.Romdhani, 2021

15

15



Accessing NoSQL
MongoDB databases

Mongoose: Creating a Model

- **Models are created from schemas using the mongoose.model() method:**

```
// Define schema
var Schema = mongoose.Schema;

var SomeModelSchema = new Schema({
  a_string: String,
  a_date: Date
});

// Compile model from schema
var SomeModel = mongoose.model('SomeModel', SomeModelSchema );
```

M.Romdhani, 2021

16

16

Using the Mongoose API to implement CRUD Operations

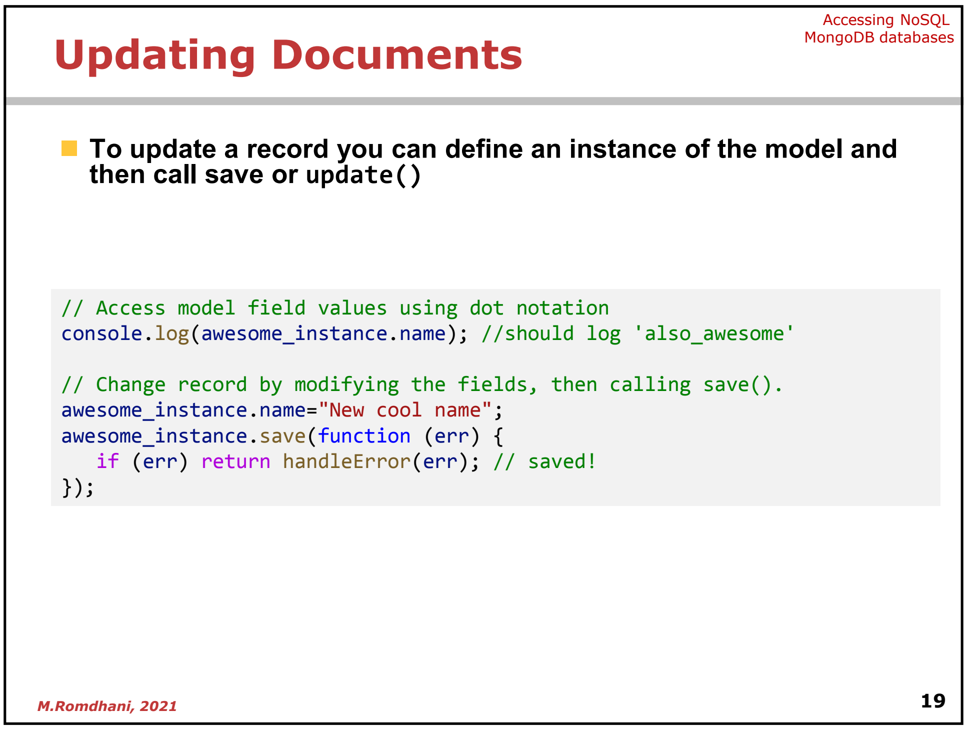
17

Creating Documents

- To create a record you can define an instance of the model and then call `save()`

```
// Create an instance of model SomeModel
var awesome_instance = new SomeModel({ name: 'awesome' });

// Save the new model instance, passing a callback
awesome_instance.save(function (err) {
  if (err) return handleError(err);
  // saved!
});
```

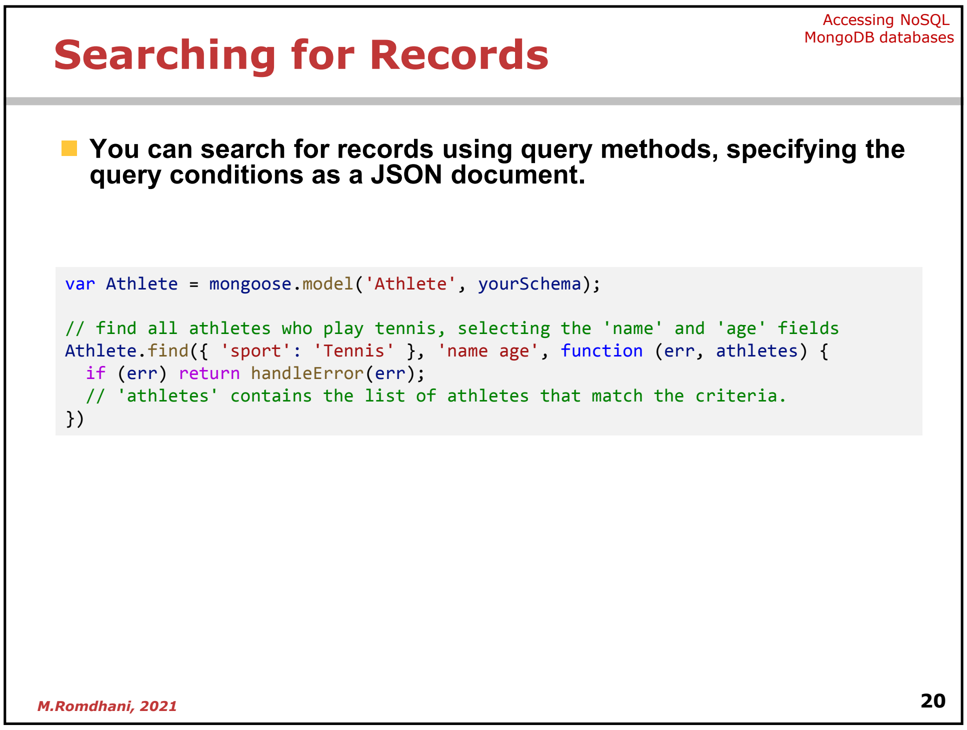
18



Updating Documents

- **To update a record you can define an instance of the model and then call save or `update()`**

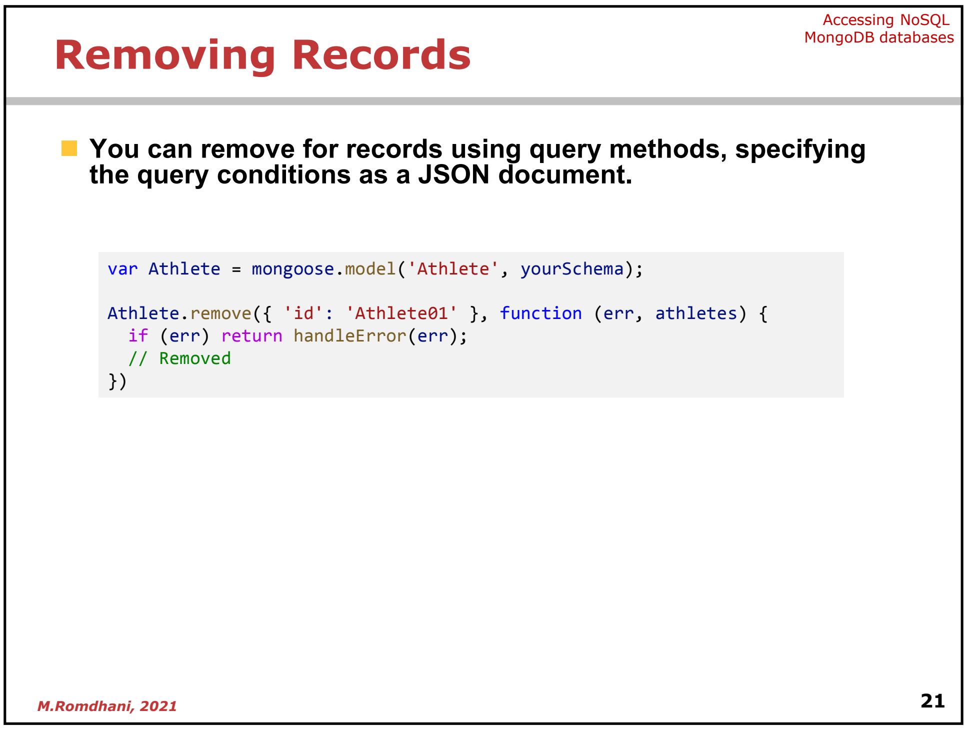
```
// Access model field values using dot notation
console.log(awesome_instance.name); //should log 'also_awesome'

// Change record by modifying the fields, then calling save().
awesome_instance.name="New cool name";
awesome_instance.save(function (err) {
   if (err) return handleError(err); // saved!
});
```



Searching for Records

- **You can search for records using query methods, specifying the query conditions as a JSON document.**

```
var Athlete = mongoose.model('Athlete', yourSchema);

// find all athletes who play tennis, selecting the 'name' and 'age' fields
Athlete.find({ 'sport': 'Tennis' }, 'name age', function (err, athletes) {
  if (err) return handleError(err);
  // 'athletes' contains the list of athletes that match the criteria.
})
```



Removing Records

- **You can remove for records using query methods, specifying the query conditions as a JSON document.**

```
var Athlete = mongoose.model('Athlete', yourSchema);

Athlete.remove({ 'id': 'Athlete01' }, function (err, athletes) {
  if (err) return handleError(err);
  // Removed
})
```

21