Database Interaction II

MongoDB and Node.js

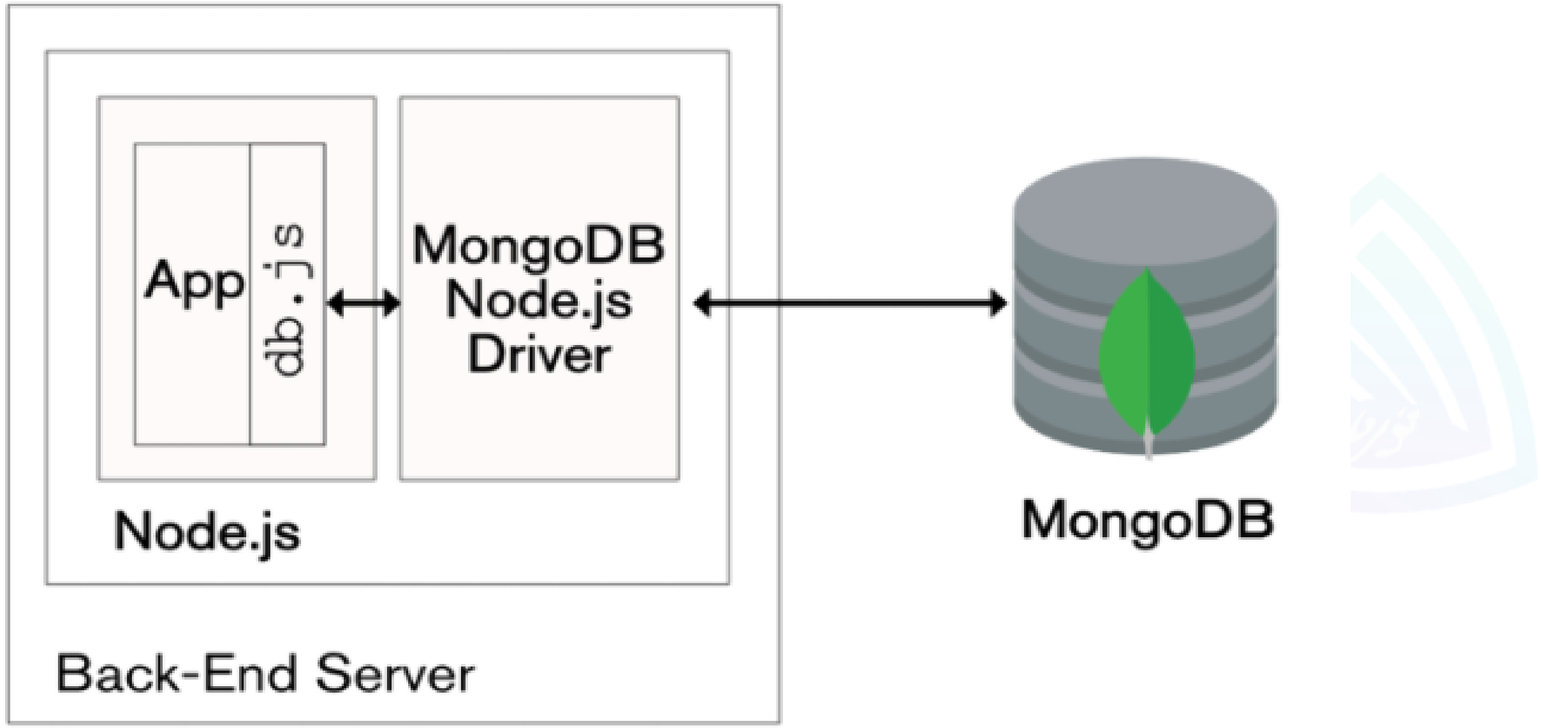


Image source: https://www.mongodb.com/blog/post/the-modern-application-stack-part-2-using-mongodb-with-nodejs

Mongoose

- Mongoose:
  - Library that sits on top of the MongoDB driver and abstracts some of the boilerplate code for you
  - Includes built-in typecasting, validation, query building and business logic hooks
  - Object Data Model (ODM: a tool that allows the programmer to treat documents stored in databases as JavaScript objects)

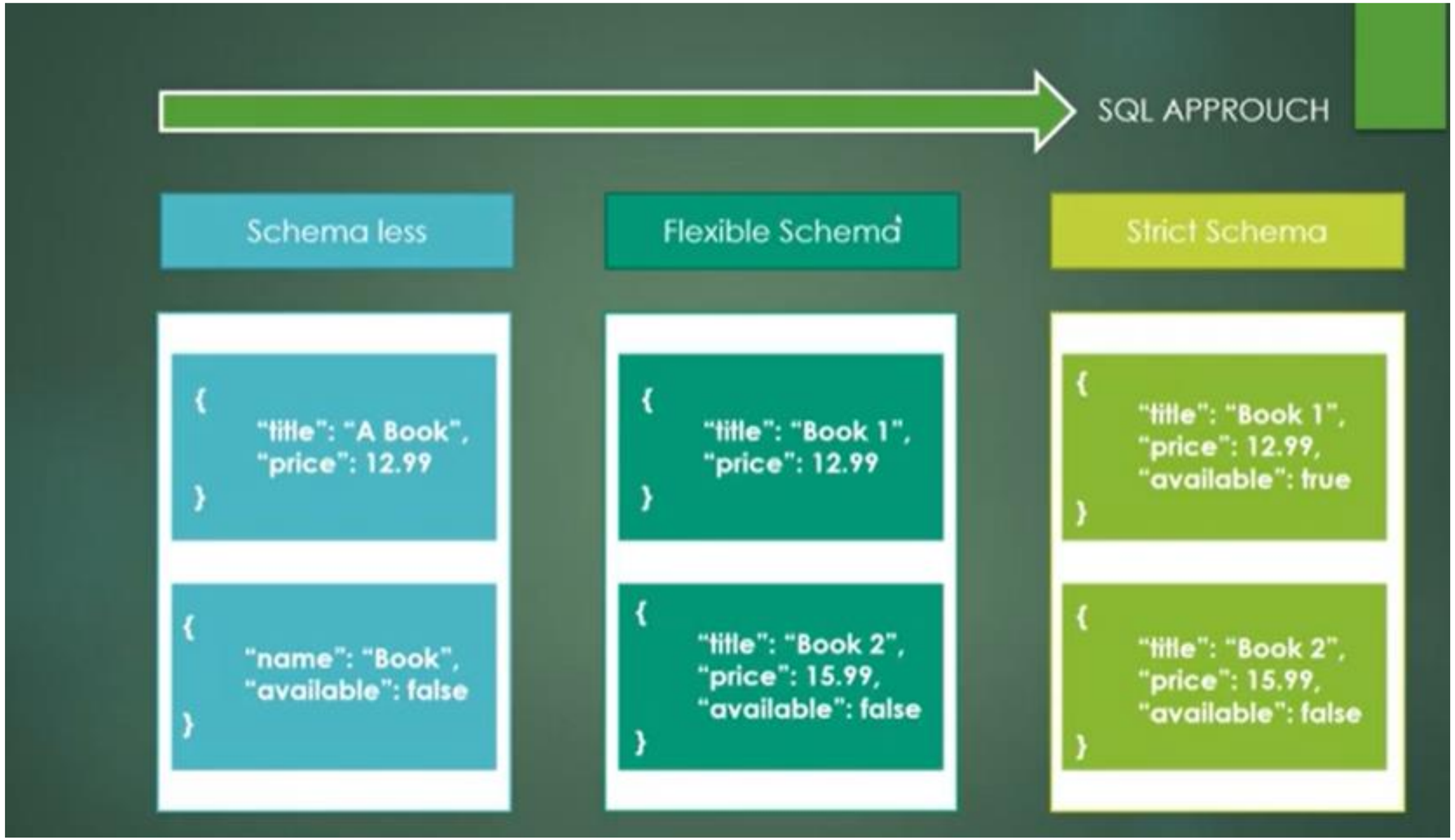
Schemas

What is a Schema?

- A schema is a JSON object that defines the the structure and contents of your data.
- Schemas represent types of data rather than specific values. supports many built-in schema types. These include primitives, like strings and numbers, as well as structural types, like objects and arrays, which you can combine to create schemas that represent custom object types.

```
author.js
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

// Create Schema and Model

const BookSchema = new Schema({
    title: String,
    pages: Number
});

const AuthorSchema = new Schema({
    name: String,
    age: Number,
    books: [BookSchema]
});

const Author = mongoose.model('author', AuthorSchema);

module.exports = Author;
```

Different Schema Approache



Data type in MongoDB

Text	"Jeff", "Some description"		
Boolean	True , False		
Number	NumberInt	NumberLong	NumberDecimal
	2,147,483,647	9,223,372,036,854,775,807	3456.89
ObjectId	ObjectId("5f8075a15104f46744383ae9")		
ISODate	ISODate("2012-12-19T06:01:17.171Z")		

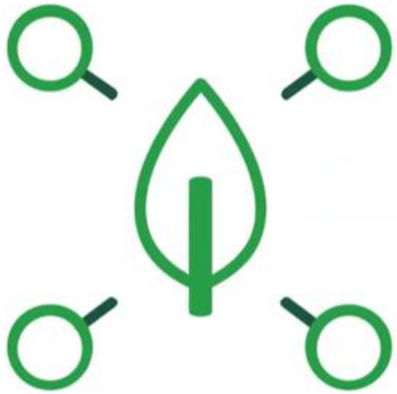
Embedded Document

```
{
    details: {
        title: "something"
    }
}
```

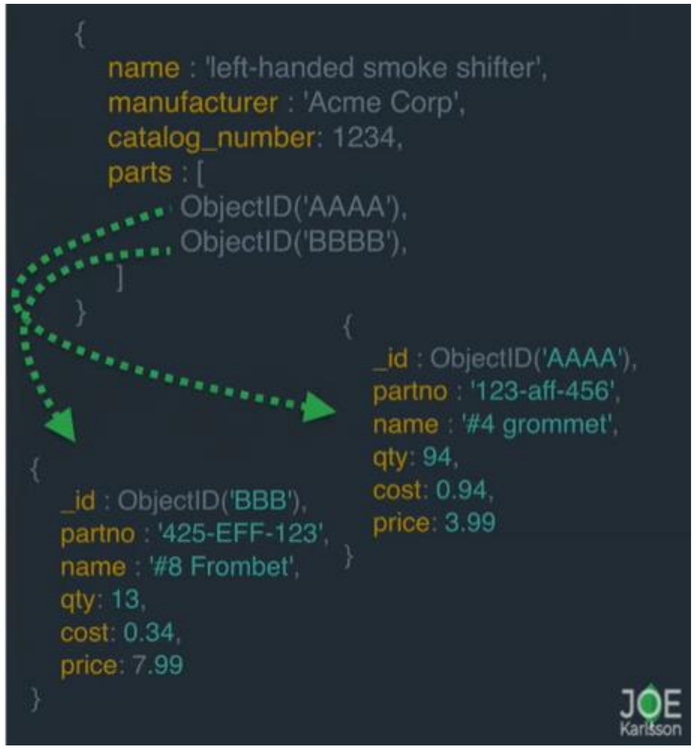
Array

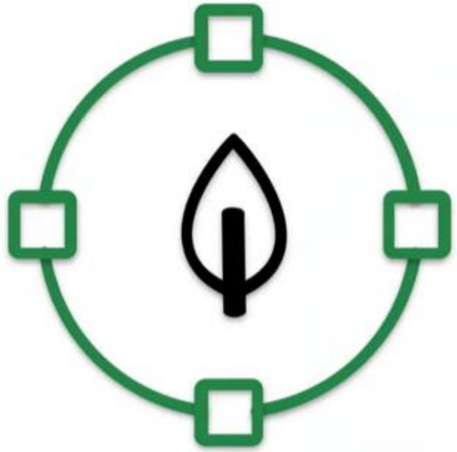
```
{
    tags: ["tag1", "tag2"]
}
```

Relationship in MongoDB



Referencing





Embedding

Relationship in MongoDB

Embedding

Pro

- Retrieve all data with a single query
- Avoids expense JOINs or $lookup
- Update all data with a single atomic operation

Con

- Large docs === more overhead
- 16 MB Document size limit

Relationship in MongoDB

Referencing

Pro

- Smaller documents
- Less likely to reach 16 MB limit
- No duplication of data
- Infrequently accessed data not accessed on every query

Con

- Two queries or $lookup required to retrieve all data

Relationship in MongoDB

One-to-One Embedded

```
Book = {
    "_id": 1,
    "title": "Harry Potter and the Methods of Rationality",
    "slug": "9781857150193-hpmor",
    "author": {
        "firstName": "Eliezer",
        "lastName": "Yudkowsky"
    },
    // more fields follow…
}
```

One-to-One Linked

```
Book = {                                    // either side can track
    "_id": 1,
    "title": "Harry Potter and the Methods of Rationality",
    "slug": "9781857150193-hpmor",
    "author": 1,                            // more fields follow…
}

Author = {
    "_id": 1,
    "firstName": "Eliezer",
    "lastName": "Yudkowsky"
    "book": 1,                              // more fields follow…
}
```

Relationship in MongoDB

One-to-Many: Scalar in Child

```
Book1= {
    "_id": 1,
    "title": "Harry Potter and the Methods of Rationality",
    "slug": "9781857150193-hpmor",
    "author": 1,                // more fields follow…
}

Book2= {
    "_id": 5,
    "title": "How to Actually Change Your Mind",
    "slug": "1939311179490-how-to-change",
    "author": 1,                // more fields follow…
}
```

One-to-Many: Array in Parent

```
Author= {
    "_id": 1,
    "firstName": "Eliezer",
    "lastName": "Yudkowsky",
    "books": [1, 5, 17],
    // more fields follow…
}
```

Relationship in MongoDB

Many-to-Many: Arrays on either side

```
Book = {                                    //either side can track
    "_id": 5,
    "title": "Harry Potter and the Methods of Rationality",
    "slug": "9781857150193-hpmor",
    "authors": [1, 3],                      // more fields follow…
}

Author = {
    "_id": 1,
    "firstName": "Eliezer",
    "lastName": "Yudkowsky",
    "books": [5, 7],                        // more fields follow…

}
```

Create Code Using Mongoose

- Step 1: Install Mongoose npm install mongoose
- Step 2: Create a Models Folder
- Step 3: Create a schema

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

let TodoSchema = mongoose.Schema({
  description :{
     type:String,
     required:true
  },
  title:{
     type:String,
     required:true
  },
```

Create Code Using Mongoose

- How to create a model using model() method:
  - Two arguments passed:
    1. Name of the model
    2. Schema object you created

```
let Todos= mongoose.model('Todos', TodoSchema);
```

- Since you are inside the models folder, you will need to export this module/Schema so you can use it elsewhere

Create Code Using Mongoose

- Step 3: Create a controller file to perform CRUD operations
  - Create: save()

```
app.post('/todos', async (req,res)=> {
    // const todo = new todo(req.body)
    const todo = new Todo({
        description: req.body.description,
        title: req.body.title,
        author: req.body.author
    })
    console.log(todo)
    // create an instance of the todos
    try {
        await todo.save()
        res.status(201).send(todo)
    }
    catch(e){
        console.error(e)
    }
    console.log('Added');
})
```

Create Code Using Mongoose

- Read: find()

```
app.get('/todos', async (req,res)=> {
    try{
        const todos = await Todo.find()
        res.send(todos)
    } catch(e){
        res.status(500).send()
        console.error(e)
    }
})
```

Create Code Using Mongoose

- Read: findOne()

```
app.get('/todos/:id', async (req,res)=> {
    // grab specific ID
   const _id = req.params.id
    try{
        const todo = await Todo.findOne({_id});
        res.send(todo);
    }
        // If there is no error
   catch(e) {
        res.status(500).send()
        // any errors being sent back
        console.error(e)
   }
})
```

Create Code Using Mongoose

- Update: update(), updateOne(), updateMany() or findOneAndUpdate()

```
app.patch('/todos/:id', async (req,res)=> {
   const allowedUpdates = ['description', 'title', 'author'];
   const updates = Object.keys(req.body)
   const isValidOperation  = updates.every((update)=> allowedUpdates.includes(update))

   if(!isValidOperation) {
       return res.status(400).send({erro: 'Invalid updates'});
       // anything not expected from the todos
   }
   try{
       const todo = await Todo.findOne({_id: req.params.id});
       // find id passed into the function
       if(!todo) {return res.status(404).send(404).send()}
       updates.forEach((update)=> {
           todo[update] = req.body[update]
           // update the values and keys dynamically
       })
       await todo.save()
       res.status(200).send(todo)

   } catch(e){
       res.status(400).send(e)
       console.error(e)
   }
})
```

Create Code Using Mongoose

- Delete: remove()

```
app.delete('/todos/:id', async (req,res)=> {
    try{
        const todo = await Todo.findByIdAndDelete({_id: req.params.id});
        // grab current todos
        if(!todo) {
            return res.status(404).send()
        }
        // If all is well
        res.send(todo)
        // send the deleted todos
    } catch(e){
        res.status(500).send()
        console.error(e)
        // if all fails
    }
})
```

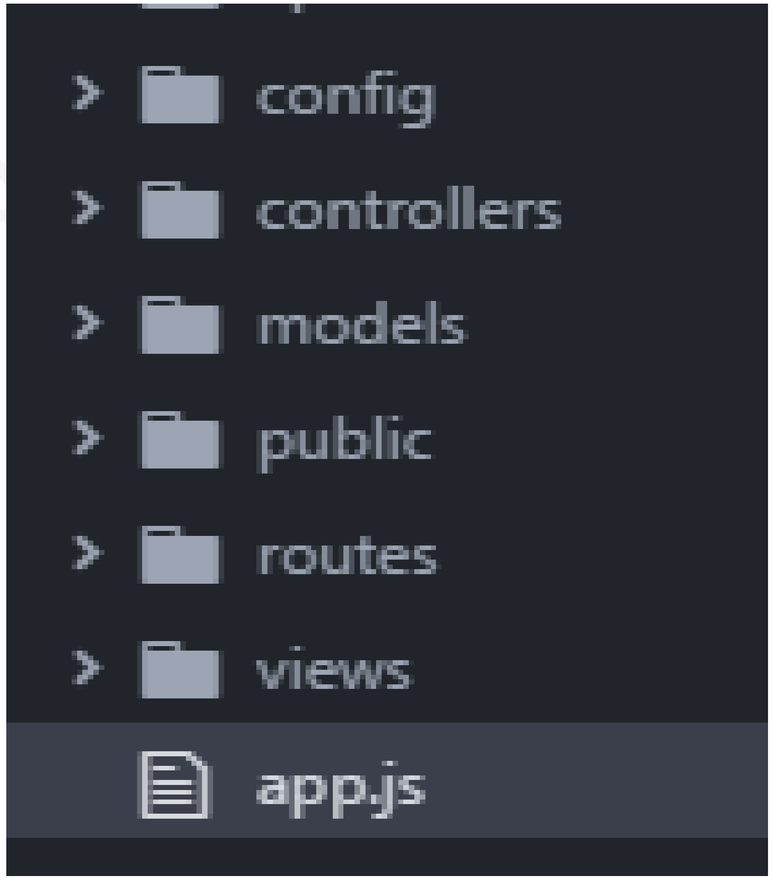
Create Code Using Mongoose

- Step 4: Connect to the database and execute appropriate CRUD operations

```
let mongoose = require('mongoose');
const uri = 'mongodb://hyperionDB:password@hyperion-shard-00-00-
f78fc.m...';

mongoose.connect(uri, {
useNewUrlParser: true, useUnifiedTopology: true
});
const connection = mongoose.connection
connection.once('open', () => console.log('Connected to DB')
connection.on('error', err => {console.log('connection error', err)}
```

MongoDB and Express

- It is recommended that your project is based on a recognised software architecture pattern
- A project structure that is based on the MVC architecture pattern:



Resources

- MongoDB Schema Design Best Practices: https://www.youtube.com/watch?v=leNCfU5SYR8
- MongoDB Crash Course - Schemas and Relationships: https://www.youtube.com/watch?v=DdvhZj7SsEM
- Data Modeling with MongoDB: https://www.youtube.com/watch?v=3GHZd0zv170
- MongoDB Tutorial : https://www.youtube.com/watch?v=9JZJsChpwKs
- Create MongoDB database models in Node.js: https://javascript.plainenglish.io/node-js-models-and-database-3836f0c7f2da