CRUD OPERATIONS USING MONGOOSE

MongoDB

- MongoDB is an open-source document database. It stores data in flexible, JSON-like documents.

- In relational databases we have tables and rows, in MongoDB we have collections and documents. A document can contain sub-documents.

- We don’t have relationships between documents.

mongoDB

Connecting to MongoDB

PowerShell

```
npm init --yes
npm i mongoose
```

index.js

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

mongoose.connect('mongodb://localhost/playground')
    .then(() => console.log('Connected...'))
    .catch(err => console.error('Connection failed...', err));
```

Schemas

index.js

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

// Connecting to MongoDB
mongoose.connect('mongodb://localhost/playground')
    .then(() => console.log('Connected...'))
    .catch(err => console.error('Connection failed...', err));

// Defining a schema
const courseSchema = new mongoose.Schema({
    name: String,
    author: String,
    tags: [String],
    date: { type: Date, default: Date.now },
    isPublished: Boolean
});
```

Models

index.js

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

// Connecting to MongoDB
mongoose.connect('mongodb://localhost/playground')
    .then(() => console.log('Connected...'))
    .catch(err => console.error('Connection failed...', err));

// Defining a schema
const courseSchema = new mongoose.Schema({
    name: String,
    author: String,
    tags: [String],
    date: { type: Date, default: Date.now },
    isPublished: Boolean
});

const Course = mongoose.model('Course', courseSchema);
const course = new Course({
    name: 'Node.js Course',
    author: 'Marizza',
    tags: ['node', 'backend'],
    isPublished: true
});
```

Saving a Document

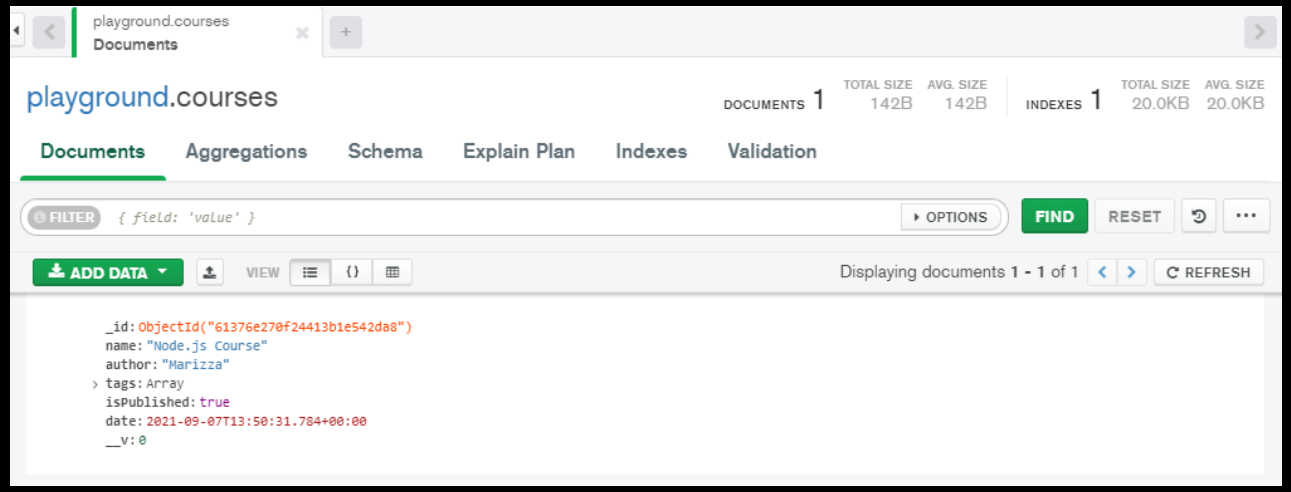
index.js

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

// Connecting to MongoDB
mongoose.connect('mongodb://localhost/playground')
    .then(() => console.log('Connected...'))
    .catch(err => console.error('Connection failed...', err));

// Defining a schema
const courseSchema = new mongoose.Schema({
    name: String,
    author: String,
    tags: [String],
    date: { type: Date, default: Date.now },
    isPublished: Boolean
});

const Course = mongoose.model('Course', courseSchema);

async function createCourse(){
    const course = new Course({
        name: 'Node.js Course',
        author: 'Marizza',
        tags: ['node', 'backend'],
        isPublished: true
    });

    const result = await course.save();
    console.log(result);
}

createCourse();
```

Saving a Document

PowerShell

```
nodemon index.js
```



Saving a Document

index.js

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

// Connecting to MongoDB
mongoose.connect('mongodb://localhost/playground')
    .then(() => console.log('Connected...'))
    .catch(err => console.error('Connection failed...', err));

// Defining a schema
const courseSchema = new mongoose.Schema({
    name: String,
    author: String,
    tags: [String],
    date: { type: Date, default: Date.now },
    isPublished: Boolean
});

const Course = mongoose.model('Course', courseSchema);

async function createCourse(){
    const course = new Course({
        name: 'Angular Course',
        author: 'Marizza',
        tags: ['angular', 'frontend'],
        isPublished: true
    });

    const result = await course.save();
    console.log(result);
}

createCourse();
```

Saving a Document

PowerShell

```
nodemon index.js
```

playground.courses
Documents

playground.courses

DOCUMENTS 1 TOTAL SIZE 142B AVG. SIZE 142B INDEXES 1 TOTAL SIZE 20.0KB AVG. SIZE 20.0KB

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER { field: 'value' } OPTIONS FIND RESET

ADD DATA VIEW Displaying documents 1 - 2 of 2 REFRESH

```
_id: ObjectId("61376e270f24413b1e542da8")
name: "Node.js Course"
author: "Marizza"
> tags: Array
isPublished: true
date: 2021-09-07T13:50:31.784+00:00
__v: 0

_id: ObjectId("61377099e4425ca5506968c6")
name: "Angular Course"
author: "Marizza"
> tags: Array
isPublished: true
date: 2021-09-07T14:00:57.948+00:00
__v: 0
```

Querying Documents

index.js

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

mongoose.connect('mongodb://localhost/playground')
    .then(() => console.log('Connected...'))
    .catch(err => console.error('Connection failed...', err
));

const courseSchema = new mongoose.Schema({
...
});

const Course = mongoose.model('Course', courseSchema);

async function createCourse(){
...
}

createCourse();

async function getCourses(){
    const courses = await Course
        .find({author: 'Marizza',  isPublished: true})
        .limit(10)
        .sort({name: 1});
    console.log(courses);
}

getCourses();
```

Comparison Query Operators

Name	Description
$eq	Matches values that are equal to a specified value.
$gt	Matches values that are greater than a specified value.
$gte	Matches values that are greater than or equal to a specified value.
$in	Matches any of the values specified in an array.
$lt	Matches values that are less than a specified value.
$lte	Matches values that are less than or equal to a specified value.
$ne	Matches all values that are not equal to a specified value.
$nin	Matches none of the values specified in an array.

Logical Query Operators

Name	Description
$and	Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.
$not	Inverts the effect of a query expression and returns documents that do not match the query expression.
$nor	Joins query clauses with a logical NOR returns all documents that fail to match both clauses.
$or	Joins query clauses with a logical OR returns all documents that match the conditions of either clause

Logical Query Operators

index.js

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

// Connecting to MongoDB
mongoose.connect('mongodb://localhost/playground')
    .then(() => console.log('Connected...'))
    .catch(err => console.error('Connection failed...', err));

// Defining a schema
const courseSchema = new mongoose.Schema({
...
});

const Course = mongoose.model('Course', courseSchema);

async function createCourse(){
...
}

createCourse();

async function getCourses(){
    const courses = await Course
        .find()
        .or([{author: 'Marizza'},  {isPublished: true}])
        .limit(10)
        .sort({name: 1});
    console.log(courses);
}

getCourses();
```

Regular Expressions

index.js

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

mongoose.connect('mongodb://localhost/playground')
    .then(() => console.log('Connected...'))
    .catch(err => console.error('Connection failed...', err));

const courseSchema = new mongoose.Schema({
...
});

const Course = mongoose.model('Course', courseSchema);

async function createCourse(){
...
}

createCourse();

async function getCourses(){
    const courses = await Course

    //Starts with Marizza
        .find({author: /^Marizza/})
    //Ebds with Mill
        .find({author: /Mill$/i})
    //Contains Marizza
        .find({author: /.*Mill.*/})

        .limit(10)
        .sort({name: 1});
    console.log(courses);
}

getCourses();
```

Counting

index.js

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

mongoose.connect('mongodb://localhost/playground')
    .then(() => console.log('Connected...'))
    .catch(err => console.error('Connection failed...', err));

const courseSchema = new mongoose.Schema({
    name: String,
    author: String,
    tags: [String],
    date: { type: Date, default: Date.now },
    isPublished: Boolean
});

const Course = mongoose.model('Course', courseSchema);

async function createCourse(){
...
}

createCourse();

async function getCourses(){
    const courses = await Course
        .find({author: 'Marizza',  isPublished: true})
        .limit(10)
        .sort({name: 1})
        .count();
    console.log(courses);
}

getCourses();
```

Pagination

index.js

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

mongoose.connect('mongodb://localhost/playground')
    .then(() => console.log('Connected...'))
    .catch(err => console.error('Connection failed...', err))
;

const courseSchema = new mongoose.Schema({
...
});

const Course = mongoose.model('Course', courseSchema);

async function createCourse(){
...
}
createCourse();

async function getCourses(){
    const pageNumber = 2;
    const pageSize = 10;
    const courses = await Course
        .find({author: 'MarizzaMil',  isPublished: true})
        .skip((pageNumber - 1) * pageSize)
        .limit(pageSize)
        .sort({name: 1})
        .select({name: 1, tags: 1})
    console.log(courses);
}

getCourses();
```

Updating a Document - Query First

index.js

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

mongoose.connect('mongodb://localhost/playground')
    .then(() => console.log('Connected...'))
    .catch(err => console.error('Connection failed...', err));

const courseSchema = new mongoose.Schema({
    name: String,
    author: String,
    tags: [String],
    date: { type: Date, default: Date.now },
    isPublished: Boolean
});

const Course = mongoose.model('Course', courseSchema);

async function createCourse(){
...
}

async function getCourses(){
...
}

async function updateCourse(id){
    const course = await Course.findById(id);
    if (!course) return;

    course.isPublished = true;
    course.author = 'Another Author';

    const result = await course.save();
    console.log(result);
}
updateCourse('61376e270f24413b1e542da8');
```

Updating a Document - Update First

index.js

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

// Connecting to MongoDB
mongoose.connect('mongodb://localhost/playground')
    .then(() => console.log('Connected...'))
    .catch(err => console.error('Connection failed...', err));

// Defining a schema
const courseSchema = new mongoose.Schema({
...
});

const Course = mongoose.model('Course', courseSchema);

async function createCourse(){
...
}

async function getCourses(){
...
}

async function updateCourse(id){
    const result = await Course.update({ _id: id}, {
        $set: {
            author: 'Marizza',
            isPublished: false
        }
    });
    console.log(result);
}

updateCourse('61376e270f24413b1e542da8');
```

Updating a Document - Update First

index.js

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

// Connecting to MongoDB
mongoose.connect('mongodb://localhost/playground')
    .then(() => console.log('Connected...'))
    .catch(err => console.error('Connection failed...', err));

// Defining a schema
const courseSchema = new mongoose.Schema({
...
});

const Course = mongoose.model('Course', courseSchema);

async function createCourse(){
...
}

async function getCourses(){
...
}

async function updateCourse(id){
    const course = await Course.findByIdAndUpdate(id, {
        $set: {
            author: 'Jack',
            isPublished: true
        }
    });
    console.log(course);
}

updateCourse('61376e270f24413b1e542da8');
```

Updating a Document - Update First

index.js

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

mongoose.connect('mongodb://localhost/playground')
    .then(() => console.log('Connected...'))
    .catch(err => console.error('Connection failed...', err));

const courseSchema = new mongoose.Schema({
...
});

const Course = mongoose.model('Course', courseSchema);
...
}

async function getCourses(){
...
}

async function updateCourse(id){
    const course = await Course.findByIdAndUpdate(id, {
        $set: {
            author: 'Jason',
            isPublished: false
        }
    }, {new: true});
    console.log(course);
}

updateCourse('61376e270f24413b1e542da8');
```

Removing Documents

index.js

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

mongoose.connect('mongodb://localhost/playground')
    .then(() => console.log('Connected...'))
    .catch(err => console.error('Connection failed...', err));

const courseSchema = new mongoose.Schema({
...
});

const Course = mongoose.model('Course', courseSchema);

async function createCourse(){
...
}

async function getCourses(){
...
}

async function updateCourse(id){
...
}

async function removeCourse(id){
    const course = await Course.findByIdAndRemove(id);
    console.log(course);
}

removeCourse('61376e270f24413b1e542da8');
```