

////

MONGOOSE -  
MODELING  
RELATIONSHIPS  
BETWEEN  
CONNECTED  
DATA





# Modelling Relationships

To model relationships between connected data, we can either reference a document or embed it in another document



# Referencing documents

Referencing documents (normalization) is a good approach when you want to enforce data consistency. Because there will be a single instance of an object in the database.

But this approach has a negative impact on the performance of your queries because in MongoDB we cannot JOIN documents as we do in relational databases.

So, to get a complete representation of a document with its related documents, we need to send multiple queries to the database



# Embedding documents

Embedding documents (denormalization) solves this issue.

We can read a complete representation of a document with a single query.

All the necessary data is embedded in one document and its children.

But this also means we'll have multiple copies of data in different places.

While storage is not an issue these days, having multiple copies means changes made to the original document may not propagate to all copies.

If the database server dies during an update, some documents will be inconsistent.

For every business, for every problem, you need to ask this question: "can we tolerate data being inconsistent for a short period of time?" If not, you'll have to use references. But again, this means that your queries will be slower



```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/playground')
  .then(() => console.log('Connected to MongoDB...'))
  .catch(err => console.error('Could not connect to MongoDB...', err));
const Author = mongoose.model('Author', new mongoose.Schema({
  name: String,
  bio: String,
  website: String
}));
const Course = mongoose.model('Course', new mongoose.Schema({
  name: String,
}));
async function createAuthor(name, bio, website) {
  const author = new Author({
    name,
    bio,
    website
  });
  const result = await author.save();
  console.log(result);
}
async function createCourse(name, author) {
  const course = new Course({
    name,
    author
  });

  const result = await course.save();
  console.log(result);
}
async function listCourses() {
  const courses = await Course
    .find()
    .select('name');
  console.log(courses);
}

createAuthor('Marizza', 'My bio', 'My Website');
// createCourse('Node Course', 'authorId')
// listCourses();
```

population.js

# Referencing documents



# Referencing documents



playground.authors

DOCUMENTS 1 TOTAL SIZE 89B AVG. SIZE 89B INDEXES 1 TOTAL SIZE 4.0KB AVG. SIZE 4.0KB

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER { field: 'value' } OPTIONS FIND RESET ↺ ...

ADD DATA VIEW [List View] [JSON View] [Table View]

Displaying documents 1 - 1 of 1 < > REFRESH

```
_id: ObjectId("6138b287f6bc38a225221f94")
name: "Marizza"
bio: "My bio"
website: "My Website"
__v: 0
```

← Copy





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

population.js

# Referencing documents

```
mongoose.connect('mongodb://localhost/playground')  
  .then(() => console.log('Connected to MongoDB...'))  
  .catch(err => console.error('Could not connect to MongoDB...', err));
```

```
const Author = mongoose.model('Author', new mongoose.Schema({  
  name: String,  
  bio: String,  
  website: String  
}));
```

```
const Course = mongoose.model('Course', new mongoose.Schema({  
  name: String,  
}));
```

```
async function createAuthor(name, bio, website) {  
  ...  
}
```

```
async function createCourse(name, author) {  
  ...  
}
```

```
async function listCourses() {  
  ...  
}
```

```
// createAuthor('Marizza', 'My bio', 'My Website');
```

```
createCourse('Node Course', '6138b287f6bc38a225221f94')
```

```
// listCourses();
```

Paste



# Referencing documents

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

mongoose.connect('mongodb://localhost/playground')
  .then(() => console.log('Connected to MongoDB...'))
  .catch(err => console.error('Could not connect to MongoDB...', err));

const Author = mongoose.model('Author', new mongoose.Schema({
  name: String,
  bio: String,
  website: String
}));

const Course = mongoose.model('Course', new mongoose.Schema({
  name: String,
  author: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'Author'
  }
}));

async function createAuthor(name, bio, website) {
  ...
}
async function createCourse(name, author) {
  ...
}
async function listCourses() {
  ...
}

// createAuthor('Marizza', 'My bio', 'My Website');
createCourse('Node Course', '6138b287f6bc38a225221f94')
// listCourses();
```

Insert  
code





# Referencing documents



playground.courses Documents

playground.courses

DOCUMENTS 2 TOTAL SIZE 126B AVG. SIZE 63B INDEXES 1 TOTAL SIZE 36.0KB AVG. SIZE 36.0KB

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("6138b3ccf694c7ddcaf06398")
name: "Node Course"
__v: 0
```

```
_id: ObjectId("6138b7025938dfd76dbc134d")
name: "Node Course"
author: ObjectId("6138b287f6bc38a225221f94")
__v: 0
```





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

population.js

# Population

```
mongoose.connect('mongodb://localhost/playground')  
  .then(() => console.log('Connected to MongoDB...'))  
  .catch(err => console.error('Could not connect to MongoDB...', err));
```

```
const Author = mongoose.model('Author', new mongoose.Schema({  
  name: String,  
  bio: String,  
  website: String  
}));
```

```
const Course = mongoose.model('Course', new mongoose.Schema({  
  ...  
}));
```

```
async function createAuthor(name, bio, website) {
```

```
  ...
```

```
  async function createCourse(name, author) {
```

```
    ...
```

```
  }
```

```
  async function listCourses() {
```

```
    const courses = await Course
```

```
      .find()
```

```
      .populate('author', 'name -_id')
```

```
      .select('name author');
```

```
    console.log(courses);
```

```
  }
```

```
// createAuthor('Marizza', 'My bio', 'My Website');
```

```
// createCourse('Node Course', '6138b287f6bc38a225221f94')
```

```
listCourses();
```

Insert  
code





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

mongoose.connect('mongodb://localhost/playground')
  .then(() => console.log('Connected to MongoDB...'))
  .catch(err => console.error('Could not connect to MongoDB...', err));

const authorSchema = new mongoose.Schema({
  name: String,
  bio: String,
  website: String
});

const Author = mongoose.model('Author', authorSchema);

const Course = mongoose.model('Course', new mongoose.Schema({
  name: String,
  author: authorSchema
}));

async function createCourse(name, author) {
  const course = new Course({
    name,
    author
  });

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

async function listCourses() {
  const courses = await Course.find();
  console.log(courses);
}

createCourse('Node Course', new Author({ name: 'Marizza' }));
```

embedding.js

# Embedding documents



# Embedding documents



Drop Database

⚠ To drop **playground** type the database name **playground**.

Terminal

**nodemon** embedding.js

```
PROBLEMS 1 OUTPUT TERMINAL DEBUG CONSOLE

[nodemon] starting `node embedding.js`
Connected to MongoDB...
{
  name: 'Node Course',
  author: { name: 'Marizza', _id: new ObjectId("613b082b7f686444c6b24651") },
  _id: new ObjectId("613b082b7f686444c6b24652"),
  __v: 0
}
```





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

mongoose.connect('mongodb://localhost/playground')
  .then(() => console.log('Connected to MongoDB...'))
  .catch(err => console.error('Could not connect to MongoDB...', err));

const authorSchema = new mongoose.Schema({
  name: String,
  bio: String,
  website: String
});

const Author = mongoose.model('Author', authorSchema);

const Course = mongoose.model('Course', new mongoose.Schema({
  name: String,
  author: authorSchema
}));

async function createCourse(name, author) {
  ...
}

async function listCourses() {
  ...
}

async function updateAuthor(courseId){
  const course = await Course.findById(courseId);
  course.author.name = 'Marizza Mill';
  course.save();
}

updateAuthor('613b082b7f686444c6b24652')
```

embedding.js

# Embedding documents

Insert  
code



# Embedding documents

Terminal

`nodemon embedding.js`



playground.courses

DOCUMENTS	1	TOTAL SIZE	106B	AVG. SIZE	106B
INDEXES	1	TOTAL SIZE	36.0KB	AVG. SIZE	36.0KB

Documents Aggregations Schema Explain Plan Indexes Validation

**FILTER** { field: 'value' } **OPTIONS** **FIND** **RESET** **REFRESH**

**ADD DATA** **VIEW** **REFRESH**

Displaying documents 1 - 1 of 1

```
{
  _id: ObjectId("613b082b7f686444c6b24652")
  name: "Node Course"
  author: Object
    name: "Marizza Mill"
    _id: ObjectId("613b082b7f686444c6b24651")
    __v: 0
}
```





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

embedding.js

# Embedding documents

```
mongoose.connect('mongodb://localhost/playground')  
  .then(() => console.log('Connected to MongoDB...'))  
  .catch(err => console.error('Could not connect to MongoDB...', err));
```

```
const authorSchema = new mongoose.Schema({  
  name: String,  
  bio: String,  
  website: String  
});
```

```
const Author = mongoose.model('Author', authorSchema);
```

```
const Course = mongoose.model('Course', new mongoose.Schema({  
  name: String,  
  author: authorSchema  
}));
```

```
async function createCourse(name, author) {  
  ...  
}  
async function listCourses() {  
  ...  
}
```

```
async function updateAuthor(courseId){  
  const course = await Course.updateOne({_id: courseId}, {  
    $set: {  
      'author.name': 'Mary Smith'  
    }  
  });  
}  
updateAuthor('613b082b7f686444c6b24652')
```

Replace  
code



# Embedding documents

Terminal

`nodemon embedding.js`



playground.courses

DOCUMENTS	1	TOTAL SIZE	106B	AVG. SIZE	106B
INDEXES	1	TOTAL SIZE	36.0KB	AVG. SIZE	36.0KB

**Documents** Aggregations Schema Explain Plan Indexes Validation

**FILTER** { field: 'value' } **OPTIONS** **FIND** **RESET** **REFRESH**

**ADD DATA** **VIEW** **JSON** **YAML** **SQL**

Displaying documents 1 - 1 of 1

```
{
  _id: ObjectId("613b082b7f686444c6b24652")
  name: "Node Course"
  author: {
    name: "Mary Smith"
    _id: ObjectId("613b082b7f686444c6b24651")
    __v: 0
  }
}
```







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

embedding.js

# Embedding documents

```
mongoose.connect('mongodb://localhost/playground')  
  .then(() => console.log('Connected to MongoDB...'))  
  .catch(err => console.error('Could not connect to MongoDB...', err));
```

```
const authorSchema = new mongoose.Schema({  
  name: String,  
  bio: String,  
  website: String  
});
```

```
const Author = mongoose.model('Author', authorSchema);
```

```
const Course = mongoose.model('Course', new mongoose.Schema({  
  name: String,  
  author: authorSchema  
}));
```

```
async function createCourse(name, author) {  
  ...  
}
```

```
async function listCourses() {  
  ...  
}
```

```
async function updateAuthor(courseId){  
  const course = await Course.updateOne({_id: courseId}, {  
    $unset: {  
      'author': ''  
    }  
  });
```

```
}  
updateAuthor('613b082b7f686444c6b24652')
```

← Replace  
code



# Embedding documents

Terminal

`nodemon embedding.js`



playground.courses

	DOCUMENTS	TOTAL SIZE	AVG. SIZE	INDEXES	TOTAL SIZE	AVG. SIZE
	1	106B	106B	1	36.0KB	36.0KB

Documents Aggregations Schema Explain Plan Indexes Validation

**FILTER** { field: 'value' } **OPTIONS** **FIND** **RESET** **REFRESH**

**ADD DATA** **VIEW** **JSON** **YAML** **TABLE**

Displaying documents 1 - 1 of 1

```
{
  "_id": ObjectId("613b082b7f686444c6b24652"),
  "name": "Node Course",
  "__v": 0
}
```





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

embedding.js

# Using an Array of Sub-documents

```
mongoose.connect('mongodb://localhost/playground')
  .then(() => console.log('Connected to MongoDB...'))
  .catch(err => console.error('Could not connect to MongoDB...', err));
```

```
const authorSchema = new mongoose.Schema({
  name: String,
  bio: String,
  website: String
});
const Author = mongoose.model('Author', authorSchema);
const Course = mongoose.model('Course', new mongoose.Schema({
  name: String,
  authors: [authorSchema]
}));
```

```
async function createCourse(name, authors) {
  const course = new Course({
    name,
    authors
  });
  const result = await course.save();
  console.log(result);
}
async function listCourses() {
  ...
}
```

replace  
code

```
createCourse('Angular Course', [
  new Author ({name: 'Marizza'}),
  new Author ({name: 'Alex'}),
])
```

Insert  
code





Terminal

nodemon embedding.js



# Using an Array of Sub-documents

playground.courses

DOCUMENTS 1 TOTAL SIZE 153B AVG. SIZE 153B INDEXES 1 TOTAL SIZE 20.0KB AVG. SIZE 20.0KB

Documents Aggregations Schema Explain Plan Indexes Validation

Displaying documents 1 - 1 of 1

```
_id: ObjectId("613b0f4c28c5d07020279169")
name: "Angular Course"
authors: Array
  0: Object
    name: "Marizza"
    _id: ObjectId("613b0f4c28c5d07020279167")
  1: Object
    name: "Alex"
    _id: ObjectId("613b0f4c28c5d07020279168")
__v: 0
```





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

mongoose.connect('mongodb://localhost/playground')
  .then(() => console.log('Connected to MongoDB...'))
  .catch(err => console.error('Could not connect to MongoDB...', err));

const authorSchema = new mongoose.Schema({
  name: String,
  bio: String,
  website: String
});

const Author = mongoose.model('Author', authorSchema);

const Course = mongoose.model('Course', new mongoose.Schema({
  name: String,
  authors: [authorSchema]
}));

async function createCourse(name, authors) {
  ...
}

async function listCourses() {
  ...
}
```

```
async function addAuthor(courseId, author) {
  const course = await Course.findById(courseId);
  course.authors.push(author);
  course.save();
}

addAuthor('613b0f4c28c5d07020279169', new Author({ name: 'Bob'}));
```

embedding.js

# Using an Array of Sub- documents

← Insert  
code





Terminal

nodemon embedding.js



# Using an Array of Sub-documents

playground.courses

DOCUMENTS 1 TOTAL SIZE 153B AVG. SIZE 153B INDEXES 1 TOTAL SIZE 20.0KB AVG. SIZE 20.0KB

Documents Aggregations Schema Explain Plan Indexes Validation

Displaying documents 1 - 1 of 1

```
_id: ObjectId("613b0f4c28c5d07020279169")
name: "Angular Course"
authors: Array
  0: Object
    name: "Marizza"
    _id: ObjectId("613b0f4c28c5d07020279167")
  1: Object
    name: "Alex"
    _id: ObjectId("613b0f4c28c5d07020279168")
  2: Object
    name: "Bob"
    _id: ObjectId("613b12747bd19275d02ec894")
__v: 1
```



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

embedding.js

# Using an Array of Sub- documents

```
mongoose.connect('mongodb://localhost/playground')  
  .then(() => console.log('Connected to MongoDB...'))  
  .catch(err => console.error('Could not connect to MongoDB...', err));
```

```
const authorSchema = new mongoose.Schema({  
  name: String,  
  bio: String,  
  website: String  
});  
const Author = mongoose.model('Author', authorSchema);  
const Course = mongoose.model('Course', new mongoose.Schema({  
  name: String,  
  authors: [authorSchema]  
}));  
async function createCourse(name, authors) {  
  ...  
}  
async function listCourses() {  
  ...  
}  
async function addAuthor(courseId, author) {  
  ...  
}
```

```
async function removeAuthor(courseId, authorId) {  
  const course = await Course.findById(courseId);  
  const author = course.authors.id(authorId);  
  author.remove();  
  course.save();  
}  
removeAuthor('613b0f4c28c5d07020279169', '613b12747bd19275d02ec894');
```

← Insert  
code





Terminal

**nodemon** embedding.js



# Using an Array of Sub- documents

playground.courses

DOCUMENTS	1	TOTAL SIZE	153B	AVG. SIZE	153B
INDEXES	1	TOTAL SIZE	20.0KB	AVG. SIZE	20.0KB

**Documents** Aggregations Schema Explain Plan Indexes Validation

**FILTER** { field: 'value' } **OPTIONS** **FIND** **RESET** **REFRESH**

**ADD DATA** **VIEW** **JSON** **YAML** **TABLE**

Displaying documents 1 - 1 of 1

```
_id: ObjectId("613b0f4c28c5d07020279169")
name: "Angular Course"
authors: Array
  > 0: Object
  > 1: Object
__v: 2
```

