Lecture 3: Principles of Data Modelling with MongoDB

Quan Nguyen, Ph.D.
Assistant Professor
Department of Computing Science

THOMPSON RIVERS UNIVERSITY

Sep 16, 2025

Reminder

- Sign up for Github student developer pack (https://education.github.com/pack)
- Upload letter of enrollment to verify your student status

Benefits:

- ✅ GitHub Copilot Pro – your AI coding assistant
- ✅ $50 credit for MongoDB Atlas

M10

$0.08/hour

RECOMMENDED

Dedicated Clusters for development environments and low-traffic applications.

STORAGE	RAM	vCPU
10 GB	2 GB	2 vCPUs

Features

- Flexible backups
- Zero-downtime cluster scaling
- Performance diagnostic tools
- Additional search indexes
- Uptime SLA

Flex

From $0.011/hour

Up to $30/month

MINIMUM UPGRADE

For application development and testing, with on-demand burst capacity for unpredictable traffic.

STORAGE	RAM	vCPU
5 GB	Shared	Shared

Features

- Basic backups
- Upgradeable
- Pay As You Go pricing
- On-demand burst capacity

Free

CURRENT

For learning and exploring MongoDB in a cloud environment.

STORAGE	RAM	vCPU
512 MB	Shared	Shared

Features

- Free forever

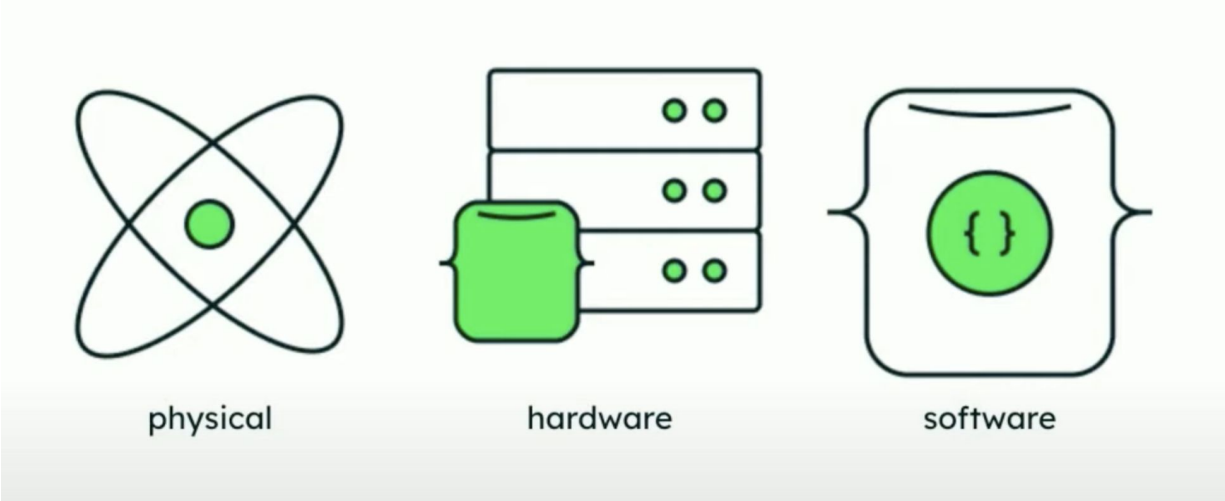
Learning objectives

- Why data modelling is important
- The key principle of data modelling in MongoDB
- Techniques to develop a data model

Why do we model?

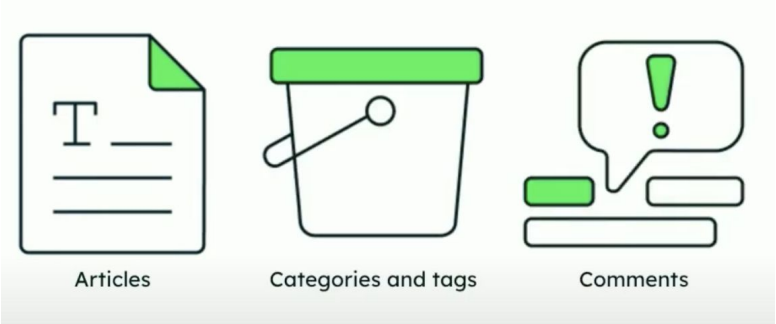
We model to face constraints and answer to constraints

So we can store, query, and use resources optimally!



Case study

You are hired to build a database for a Chef social network applications

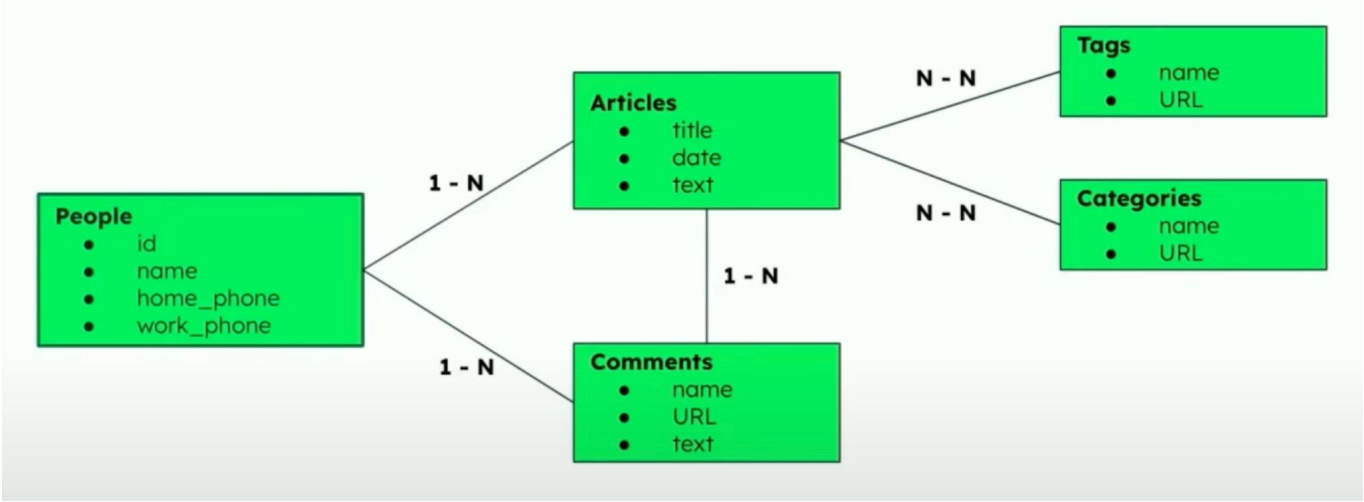


What questions would you ask?

- What does my application do?
- What data will I store?
- What are the relationships between data?
- How do users will access the data?
- What data is the most valuable?

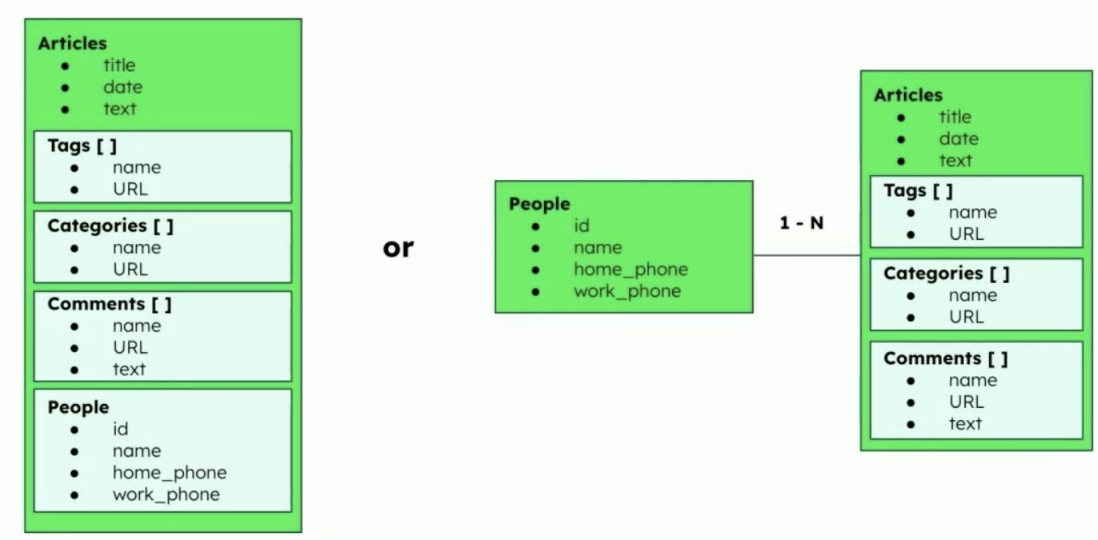
SQL solution ER diagram

In SQL, there will be only a single optimal form that satisfy all the normalization rules



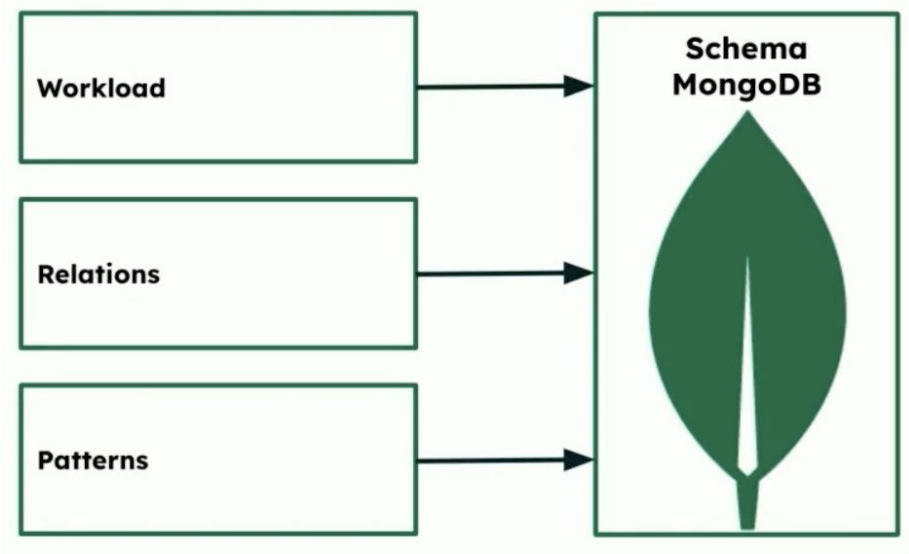
MongoDB: 1, 2, or more collections?

The optimal groupings will be determined by workload!



“Data that are access together should be stored together”

Data modelling methodology



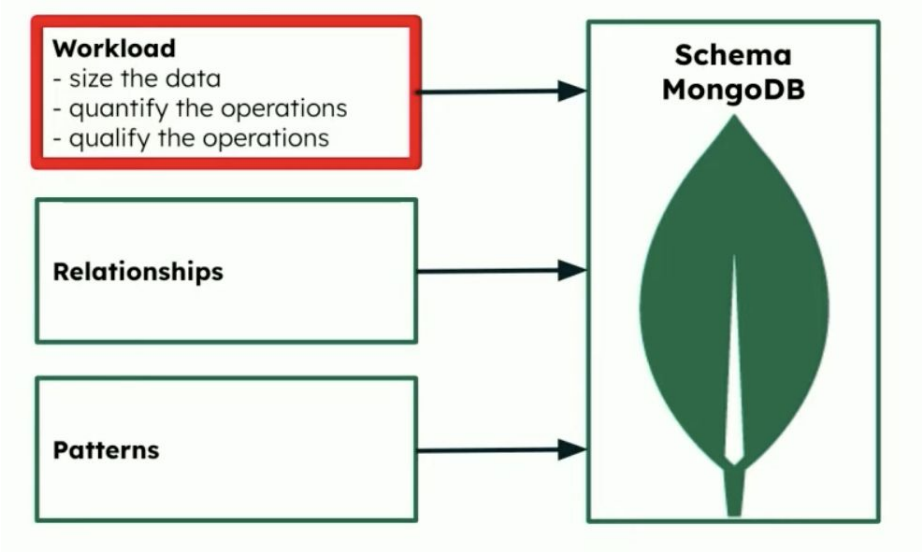
SQL

1. Model
2. Workload identification

MongoDB

1. Workload identification
2. Model

Data modelling methodology



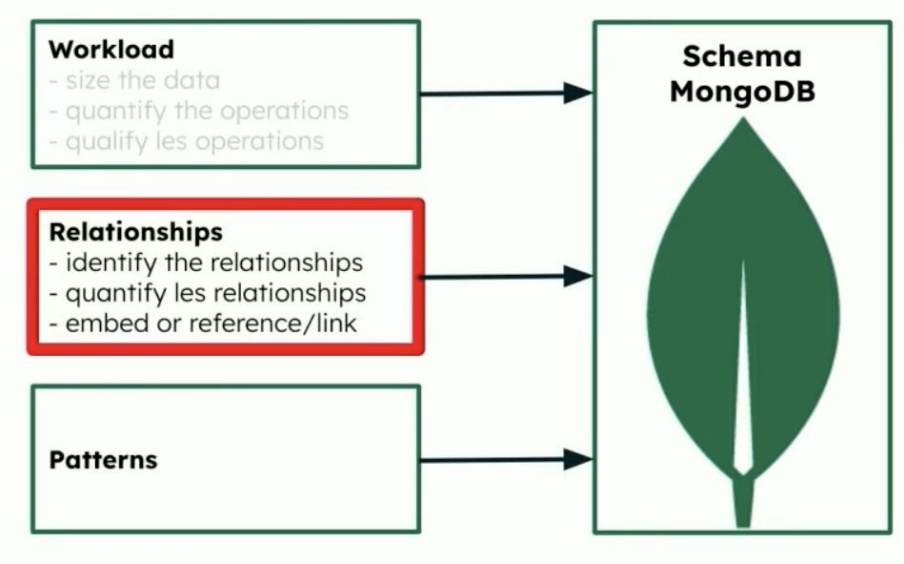
Example: listings operations

Type	Operation	Information	Frequency	Criticality
write	new articles	author, text	10 per day	High
write	comments	user	10 K per day (1000 / article)	Medium
read	reading an article	id, text, comments	10 M per day	Medium
read	analytical requests	articles, comments, clicks, ...	10 per hour	Low

Details of operations

Attribute	Details
Description	read an article and comments
Type	read
Frequency	10 M per day (115 / second)
Latency	10 ms
Size	5000 bytes (article) + 1000 * 1000 bytes (comments)
Data lifespan	3 years fast access (hot data), 10 years archives (cold data)
Security	GDPR?

Relationships



“Data that are access together should be stored together”

Car in SQL

Car in MongoDB

Relationship types

One-to-one
One-to-many
Many-to-many

2 ways to represent relationships

Embedding
Referencing

Relationship types: One-to-one

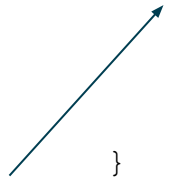
Embedding

```
{
    "_id": ObjectId("customerId1"),
    "name": "Jane Smith",
    "email": "jane.smith@example.com",
    "profile": {
        "address": "123 Main St",
        "phone": "555-1234"
    }
}
```

Relationship types: One-to-one

Referencing option 1: When the app primarily reads the customer collection

```
{
    "_id": ObjectId("customerId1"),
    "name": "Jane Smith",
    "email": "jane.smith@example.com",
    "profile_id": ObjectId("profileId1")
}
```



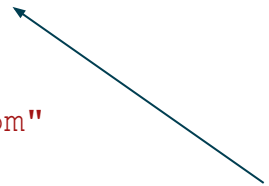
```
{
    "_id": ObjectId("profileId1"),
    "address": "123 Main St",
    "phone": "555-1234"
}
```

Relationship types: One-to-one

Referencing option 2: When the app primarily reads the profile collection

```
{
   "_id": ObjectId("customerId1"),
   "name": "Jane Smith",
   "email": "jane.smith@example.com"
}
```

```
{
   "_id": ObjectId("profileId1"),
   "address": "123 Main St",
   "phone": "555-1234",
   "customer_id": ObjectId("customerId1")
}
```



Relationship types: One-to-many

Embedding

```
{
   "_id": ObjectId("chefId1"),
   "name": "Gordon Ramsay",
   "specialty": "British cuisine",
   "recipes": [
       {
           "title": "Beef Wellington",
           "ingredients": ["beef", "mushrooms", "puff pastry"],
           "instructions": "Wrap the beef in mushrooms and puff
pastry, then bake."
       },
       {
           "title": "Scrambled Eggs",
           "ingredients": ["eggs", "butter", "salt"],
           "instructions": "Whisk eggs with butter and cook
slowly."
       }
   ]
```

Embed the “many” side as an array of subdocuments in the “one” side

Possibility of unbounded document?

Relationship types: One-to-many

Referencing option 1

```
{
   "_id": ObjectId("chefId1"),
   "name": "Gordon Ramsay",
   "specialty": "British cuisine",
   "recipe_ids": [ObjectId("recipeId1"), ObjectId("recipeId2")]
}
```

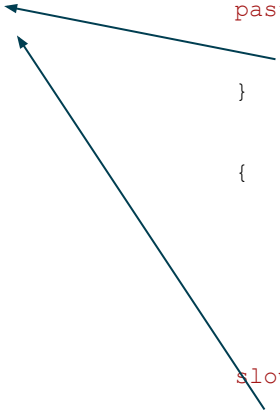
```
{
   "_id": ObjectId("recipeId1"),
   "title": "Beef Wellington",
   "ingredients": ["beef", "mushrooms", "puff pastry"],
   "instructions": "Wrap the beef in mushrooms and puff pastry, then bake."
}
```

```
{
   "_id": ObjectId("recipeId2"),
   "title": "Scrambled Eggs",
   "ingredients": ["eggs", "butter", "salt"],
   "instructions": "Whisk eggs with butter and cook slowly."
}
```

Relationship types: One-to-many

Referencing option 2

```
{
   "_id": ObjectId("chefId1"),
   "name": "Gordon Ramsay",
   "specialty": "British
cuisine"}
```



```
{
   "_id": ObjectId("recipeId1")
   "title": "Beef Wellington",
   "ingredients": ["beef", "mushrooms", "puff pastry"],
   "instructions": "Wrap the beef in mushrooms and puff
pastry, then bake.",
   "chef_id": ObjectId("chefId1")
}

{
   "_id": ObjectId("recipeId2"),
   "title": "Scrambled Eggs",
   "ingredients": ["eggs", "butter", "salt"],
   "instructions": "Whisk eggs with butter and cook
slowly.",
   "chef_id": ObjectId("chefId1")
}
```

Relationship types: Many-to-many

Embedding author info in the book collections

```
{
   _id: "book001",
   title: "Cell Biology",
   authors: [
     {
        author_id: "author124",
        name: "Ellie Smith"
     },
     {
        author_id: "author381",
        name: "John Palmer"
     }
   ]
}
```

```
{
   _id: "book002",
   title: "Organic Chemistry",
   authors: [
     {
        author_id: "author290",
        name: "Jane James"
     },
     {
        author_id: "author381",
        name: "John Palmer"
     }
   ]
}
```

Assumptions:

- Each book was written only by a small number of authors
- The author details tend to remain static
- User want to access author information and book together!

Relationship types: Many-to-many

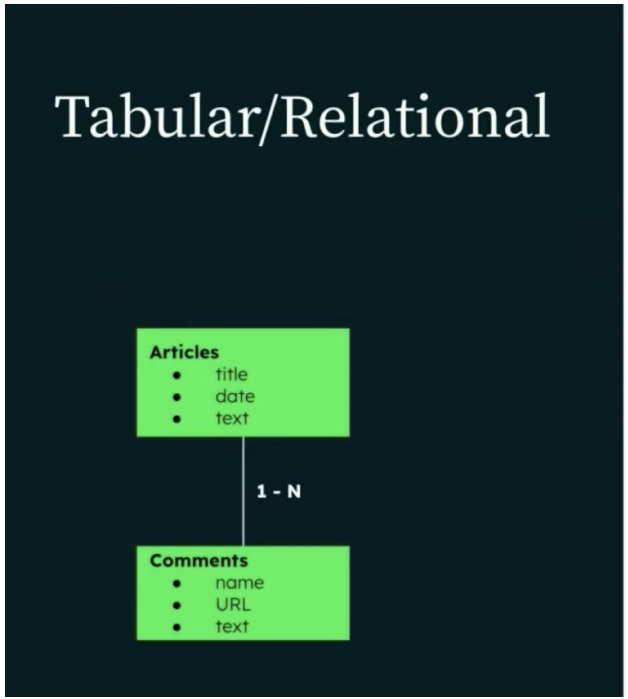
Referencing book IDs in the author collection

```
{
   "_id":
ObjectId("12hv92fse224"),
   "author": "Jone Palmer",
   "book_id":
[ObjectId("sg3b23"),
ObjectId("47m878gfsw"),
...]
}
```

Assumptions:

- Each author could have written a **large** number of books.

Embed vs referencing



Embedding is using one collection in MongoDB

How many is ‘many’?

Reference

1. When the “many” side is a huge number
2. For integrity on write operations on many-to-many
3. When a piece is frequently used, but not the other and memory is an issue

Embed

1. For integrity of read operations
2. For integrity of write operations on one-to-one and one-to-many
3. For data that is deleted or archived together
4. By default

Embedding

Single query to retrieve data

Single operation to update/delete data

Data duplication

Large documents

Referencing

No duplication

Smaller documents

Need to join data from multiple documents

Activity

You are designing a MongoDB database for a university management system. The database needs to store information about students and the courses they are enrolled in. **Each student can enroll in multiple courses, and each course can have multiple students enrolled.**

You can assume some courses could have hundreds of enrolled students. Each student can enroll in a few courses (e.g., up to 5 courses), and the course details are relatively static.

Given the case study above, how would you model the relationship between students and courses in MongoDB? **For each collection (students, courses)**, explain your choice of whether to use embedding or referencing, explicitly state your assumptions

Solution

Many-to-many relationship. Each student can enrol in multiple courses, each course has multiple enrolled students.

- **Students Collection**: **Embed** the course information inside the student document. This makes sense given the small number of courses per student, and it provides fast read performance when retrieving student and course details.
- **Courses Collection**: Use **referencing** to store only student IDs in the course document, since embedding hundreds of students per course would create large documents that are inefficient to manage.

Conclusion

- Data that are accessed together should stay together
- Identify workload before modelling
- Two ways to model relationships: embed & reference