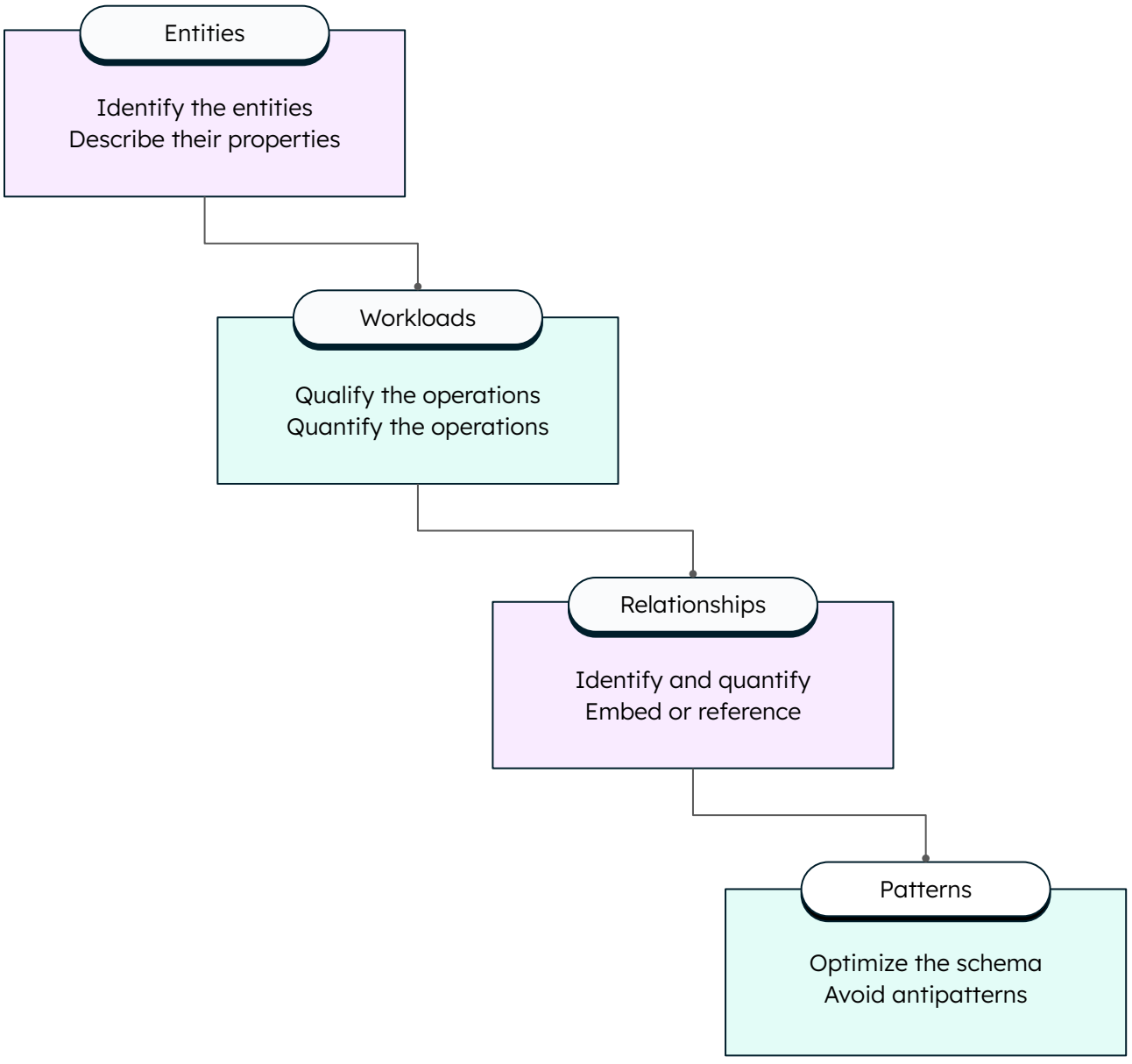
Data Modeling Methodology for MongoDB



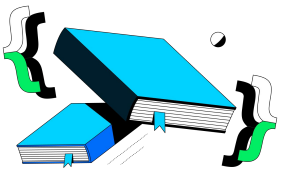
Exercise 1.
Identify the library app entities

Instructions: Which entities would be part of this application?
What are their properties and are they related to other entities?
Refer to the application prototype for guidance: mdb.link/library.
Complete the table below with entity names, their properties, and potential related entities.

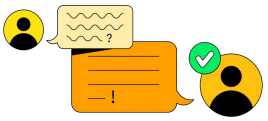
Publisher
name
country
Related entities

biography
aliases
Related entities

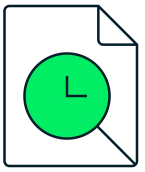
address
email



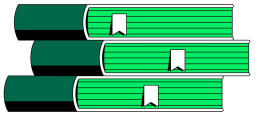
Book
ISBN
title
copies* *number*
cover image *URL*
description
genres
Related entities
Authors (many)



text
rating
date
Related entities
Book



Reservation
reservation date
Related entities
Book



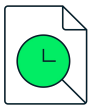
Book loan
_____ date
due date
Related entities
User *loaned by*
User *borrowed by*

*** Advanced exercise to discuss in your group:**

What is an alternative to a number of copies in the book, and what are the advantages and disadvantages of such solution?

Exercise 2. Determine the app workload

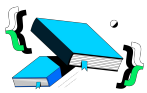
The library has 100,000 active users.



Each user reserves an average of 3 books per month. Users can only reserve a book if copies are available in the library.
How often is the "create a reservation" operation performed?
_____ *users x* ____ *reservations per user / 30 days =* _____ *reservations per day*

Only half of the reservations result in loans, and all unclaimed reservations expire at the end of the day.
How many loan documents are created every day?
_____ reservations per day x _____% converted to loans = _____ loans per day
How many reservations expire every day?
_____ reservations per day x _____% expired = _____ expired reservations per day

Expired reservations are deleted, but book loans are persisted as historical records.
<u>Advanced question:</u> Which type of MongoDB index enables the automatic deletion of expired reservations? _____



Users browse more books than they reserve. Assume that for every reservation, users browse an average of 50 books.
How often is the "fetch book details" operation performed?
_____ *x* _____ *reservations per day =* _____ *"fetch book details" ops per day*

On average, 100 new books are added to the catalogue each month.
What is the 5-year growth in the number of books?
_____ *x 12 months x 5 years =* _____ *new books in 5 years*

The application has functionality showing an authors page with all the books the author has written. Assume that the author details pages are visited 10 times less often than book details pages.

How often is the "fetch author details" operation performed?
_____ ÷ _____ = _____ *"fetch author details" operations per day*

20% of users leave a review after returning a book. Some popular books, which the library has multiple copies of, are read at least 1,000 times per month, though the actual number could be higher.
How many reviews does a popular book get per month?
_____ *x* _____ = _____ *reviews per month*
What is the annual growth in the number of reviews for a popular book?
_____ *x* _____ = _____ *reviews per year*

Application workload insights	
Is the workload read-heavy or write-heavy?	☐ read-heavy ☐ write-heavy
What is the most frequent operation in the application?	☐ fetch book ☐ fetch author
Based on the frequency of operations in the application, should the data model be optimized for books queries or authors queries?	☐ optimized for book queries ☐ optimized for author queries
Does the number of reviews on popular books grow without bound (continue to increase indefinitely)?	☐ yes ☐ no

Exercise 3. Model the relationships

Instructions: For each relationship, consider the guideline questions above the table and then:

1. Determine the type of the relationship: one-to-one, one-to-few (low cardinality), one-to-many, one-to-zillions (high cardinality), many-to-many, etc.
2. Decide whether to use embedding or referencing.
3. Specify which side of the relationship should store the embedded data or references.

Go together: Do the pieces of information express a "has-a," "contains," or similar relationship?
Document size: Would combining these pieces of information result in excessive memory usage or transfer bandwidth?

Relationship	Type	Go together?	Document size?	Embed or reference?	Which side?
User - Contact information (object with email and address)		Yes No	Yes No	Select	
Author - Author alias		Yes No	Yes No	Select	

High cardinality: Does the “many” side of the relationship contain a large or rapidly growing number of unique values?

Relationship	Type	High cardinality?	Embed or reference?	Which side?
Book - Review		Yes No	Select	
User - Loan		Yes No	Select	

Accessed separately: Are the related entities frequently accessed separately?
Based on the requirements, expired reservations are deleted at the end of the day.

Relationship	Type	Accessed separately?	Embed or reference?	Which side?
User - Reservation	one-to-many	Yes No	Select	
Book - Reservation	one-to-many	Yes No	Select	

The app has book and author details pages. You established in the previous exercise that the book page is queried more often than the author page. Based on this, how would you model this relationship?

Relationship	Type	Embed or reference?	Which side?
Book - Author		Select	

The app needs to support filtering books by publisher. Only the publisher's name and country must be stored and no other functionality involves publishers. Based on this, how would you model this relationship?

Relationship	Type	Embed or reference?	Which side?
Book - Publisher		Select	

Embedding vs referencing guidelines

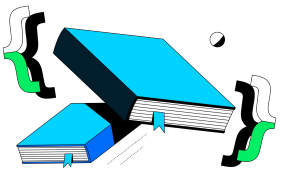
Guideline name	Question	Embed	Reference
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Yes	
Go together	Do the pieces of information express a "has-a," "contains," or similar relationship?	Yes	
Query atomicity	Does the application query the pieces of information together?	Yes	
Update complexity	Are the pieces of information updated together?	Yes	
Archival	Should the pieces of information be archived at the same time?	Yes	
Cardinality	Is there a high cardinality (current or growing) in a "many" side of the relationship?	No	Yes
Data duplication	Would data duplication be too complicated to manage and undesired?	No	Yes
Document size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	No	Yes
Document growth	Would the embedded piece grow without bound?	No	Yes
Workload	Are the pieces of information written at different times in a write-heavy workload?		Yes
Individuality	For the children side of the relationship, can the pieces exist by themselves without a parent?		Yes

Schema design anti-patterns

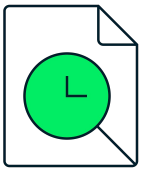
1. Unbounded arrays
2. Bloated documents
3. Massive number of collections
4. Unnecessary indexes
5. Separating data that is accessed together
6. Case-insensitive queries without case-insensitive indexes

Solutions to exercise 1.
Identify the library app entities

Publisher
name
country
Related entities
Books (many)

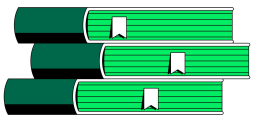


Book
ISBN
title
copies* *number*
cover image *URL*
description
genres
year
language
Related entities
Publisher
Authors (many)

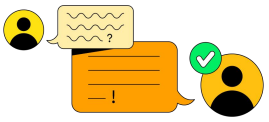


Reservation
reservation date
expiration date
Related entities
User
Book

Author
name
biography
aliases
Related entities
Books (many)



Book loan
start date
due date
Related entities
User *loaned by*
User *borrowed by*
Book



Review
text
rating
Related entities
Book
User *reviewer*

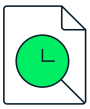
User
name
address
email
role *"reader", "librarian"*

Alternatively, you can model "Book Copy" as a separate entity instead of tracking the number of copies as a property of a book.

This would allow tracking of individual copies, including their condition and lending history. However, this approach may be unnecessary for simpler systems. Whether this is the right approach depends on your system requirements.

Solutions to exercise 2. App workload

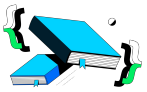
The library has 100,000 active users.



Each user reserves an average of 3 books per month. Users can only reserve a book if copies are available in the library.
How often is the "create a reservation" operation performed?
100,000 users x 3 reservations per user per month / 30 days = 10,000 reservations per day

Only half of the reservations result in loans, and all unclaimed reservations expire at the end of the day.
How many loan documents are created every day?
10,000 reservations per day x 50% converted to loans = 5,000 loans per day
How many reservations expire every day?
10,000 reservations per day x 50% expired = 5,000 expired reservations per day

Expired reservations are deleted, but book loans are persisted as historical records.
Advanced question: Which type of MongoDB index enables the automatic deletion of expired reservations? *TTL (time-to-live) index*



Users browse more books than they reserve. Assume that for every reservation, users browse an average of 50 books.
How often is the "fetch book details" operation performed?
50 browsed books x 10,000 reservations per day = 500,000 "fetch book details" ops per day

On average, 100 new books are added to the catalogue each month.
What is the 5-year growth in the number of books?
100 new books x 12 months x 5 years = 6,000 new books in 5 years

The application has functionality showing an authors page with all the books the author has written. Assume that the author details pages are visited 10 times less often than book details pages.

How often is the "fetch author details" operation performed?
500,000 "fetch book details" / 10 = 50,000 "fetch author details" operations per day

20% of users leave a review after returning a book. Some popular books, which the library has multiple copies of, are read at least 1,000 times per month, though the actual number could be higher.
How many reviews does a popular book get per month?
1,000 reads per month x 20% = 200 reviews per month
What is the annual growth in the number of reviews for a popular book?
200 reviews per month x 12 months = 2,400 reviews per year

Application workload insights	
Is the workload read-heavy or write-heavy?	The workload is read-heavy.
What is the most frequent operation in the application?	"Fetch book details" — 500,000 ops/day.
Based on the frequency of operations in the application, should the data model be optimized for books queries or authors queries?	Book queries because they are more frequent than author queries.
Does the number of reviews on popular books grow without bound (continue to increase indefinitely)?	Yes, while the growth rate may not be very high, there's no upper limit.

Solutions to exercise 3. Model the relationships

Go together: Do the pieces of information express a "has-a," "contains," or similar relationship?
Document size: Would combining these pieces of information result in excessive memory usage or transfer bandwidth?

Relationship	Type	Go together?	Document size?	Embed or reference?	Which side?
User - Contact information (object with email and address)	one-to-one	Yes	No	Embed	User
Author - Author Alias	one-to-few	Yes	No	Embed	Author

High cardinality: Does the “many” side of the relationship contain a large or rapidly growing number of unique values?

Relationship	Type	High cardinality?	Embed or reference?	Which side?
Book - Review	one-to-many	Yes	Reference*	Review
User - Loan	one-to-many	Yes	Reference*	Loan

Accessed separately: Are the related entities frequently accessed separately?
You can assume that expired reservations are deleted daily to optimize storage costs.

Relationship	Type	Accessed separately?	Embed or reference?	Which side?
User - Reservation	one-to-many	Yes	Reference*	Reservation
Book - Reservation	one-to-many	Yes	Reference*	Reservation

The app has book and author details pages. You established in the previous exercise that the book page is queried more often than the author page. Based on this, how would you model this relationship?

Relationship	Type	Embed or reference?	Which side?
Book - Author	many-to-many	Reference*	Book

The app needs to support filtering books by publisher. Only the publisher's name and country must be stored and no other functionality involves publishers. Based on this, how would you model this relationship?

Relationship	Type	Embed or reference?	Which side?
Book - Publisher	many-to-one	Embed	Book

* This relationship can be optimized if needed with a MongoDB schema design pattern.

From Relational to Document Model

Data Modeling for MongoDB

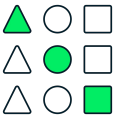
With MongoDB, you can design the most optimal schema for your use case instead of a generic schema for any use case.

Core MongoDB Design Concepts

Embedding

You can store documents within other documents to represent relationships and complex hierarchies.

Embedding allows for storing related data within a single document, unlike relational databases that use separate tables and JOINs to represent relationships.



Arrays

The value of a field in a document can be an array of values, which can themselves be documents.

Arrays enable many data modeling options, particularly for one-to-many and many-to-many relationships, allowing you to store related data together.

Polymorphism

You can store documents with different fields within the same collection.

Polymorphism allows similar types of data to be stored in the same collection and retrieved together.

Polymorphism also allows you to evolve your schema without downtime. Different versions of the schema can coexist.

When modeling data for MongoDB, your workload should drive your data model.

Schema Validation

Highly recommended for maintaining data integrity

Defined at the collection level

Enforced by the database regardless of the connection source

```
{
   bsonType: 'object',
   required: ['name', 'biography'],
   properties: {
       name: { bsonType: 'string' },
       biography: { bsonType: 'string' },
       aliases: {
           bsonType: 'array',
           maxItems: 10
       }
   }
};
```

```
{
   bsonType: 'object',
   required: [
     'rating',
     'reviewer',
     'book'
   ],
   properties: {
     text: { bsonType: 'string' },
     rating: {
       bsonType: 'int',
       minimum: 1,
       maximum: 5
     },
     reviewer: { bsonType: 'objectId' },
     book: { bsonType: 'objectId' },
   }
};
```

Modeling Relationships

One-to-one. Embed in the natural parent entity.
Exception: Avoid embedding if it causes bloated documents, leading to high memory or bandwidth usage.

One-to-few. Embed on the "one" side.
Exceptions:
- If the “few” are frequently accessed on their own.
- Avoid embedding if it causes bloated documents, leading to high memory or bandwidth usage.

One-to-many (medium cardinality). Follow the “Embedding vs referencing guidelines” sheet.
Most importantly, store together if accessed together.

One-to-zillions (unbounded growth). Use references on the "many" side.

Many-to-many. Reference on the more often queried side.
Exception: If the array with references could grow unbounded, reference from the other side instead.

MongoDB.