INDUSTRIAL UNIVERSITY OF HOCHIMINH CITY

mongoDB

INTRODUCTION
MONGODB DATA MODELING
(Giới Thiệu Mô Hình Hóa Dữ Liệu MongoDB)

Lecturer: Trần Thế Trung

What is Data Modeling?

- **Data modeling** is the process of creating a data model for the data to be stored in a database. This **data model** is a conceptual representation of Data objects, the associations between different data objects, and the rules.

- **Data modeling** helps in the visual representation of data and enforces business rules, regulatory compliances, and government policies on the data.

- **Data Models** ensure consistency in naming conventions, *default values*, *semantics*, *security* while ensuring **quality of the data**.

Why use Data Model?

- **Ensures that all data objects** required by the database are accurately represented. Omission of data will lead to creation of faulty reports and produce incorrect results.

- A **data model** helps design the database at the **conceptual**, **physical** and **logical** levels. It provides a clear picture of the base data and can be used by database developers to create a *physical database*. It is also helpful to identify missing and redundant data.

- Though the **creation of data model** is **labor** and **time** consuming, in the long run, it makes IT infrastructure **upgrade** and **maintenance** cheaper and faster.

Types of Data Models

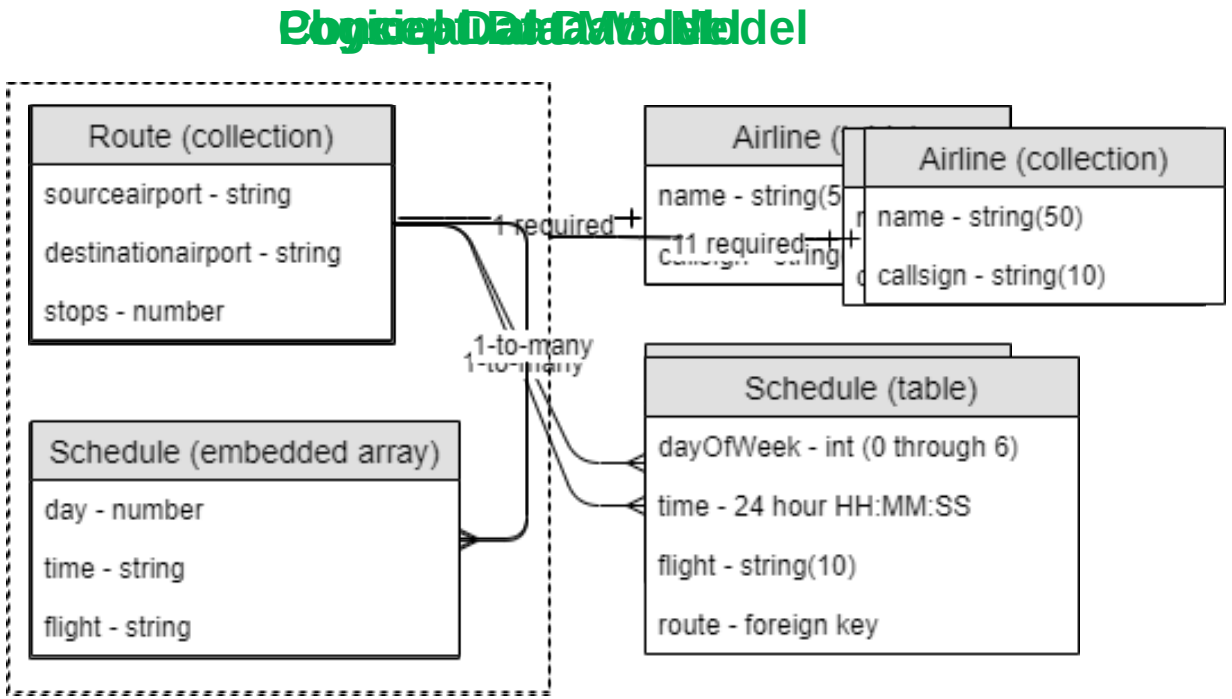
There are three types of data models.

- **Conceptual Data Model**: defines WHAT the system contains. This model is typically created by Business stakeholders and Data Architects. The purpose is to organize, scope and define business concepts and rules.

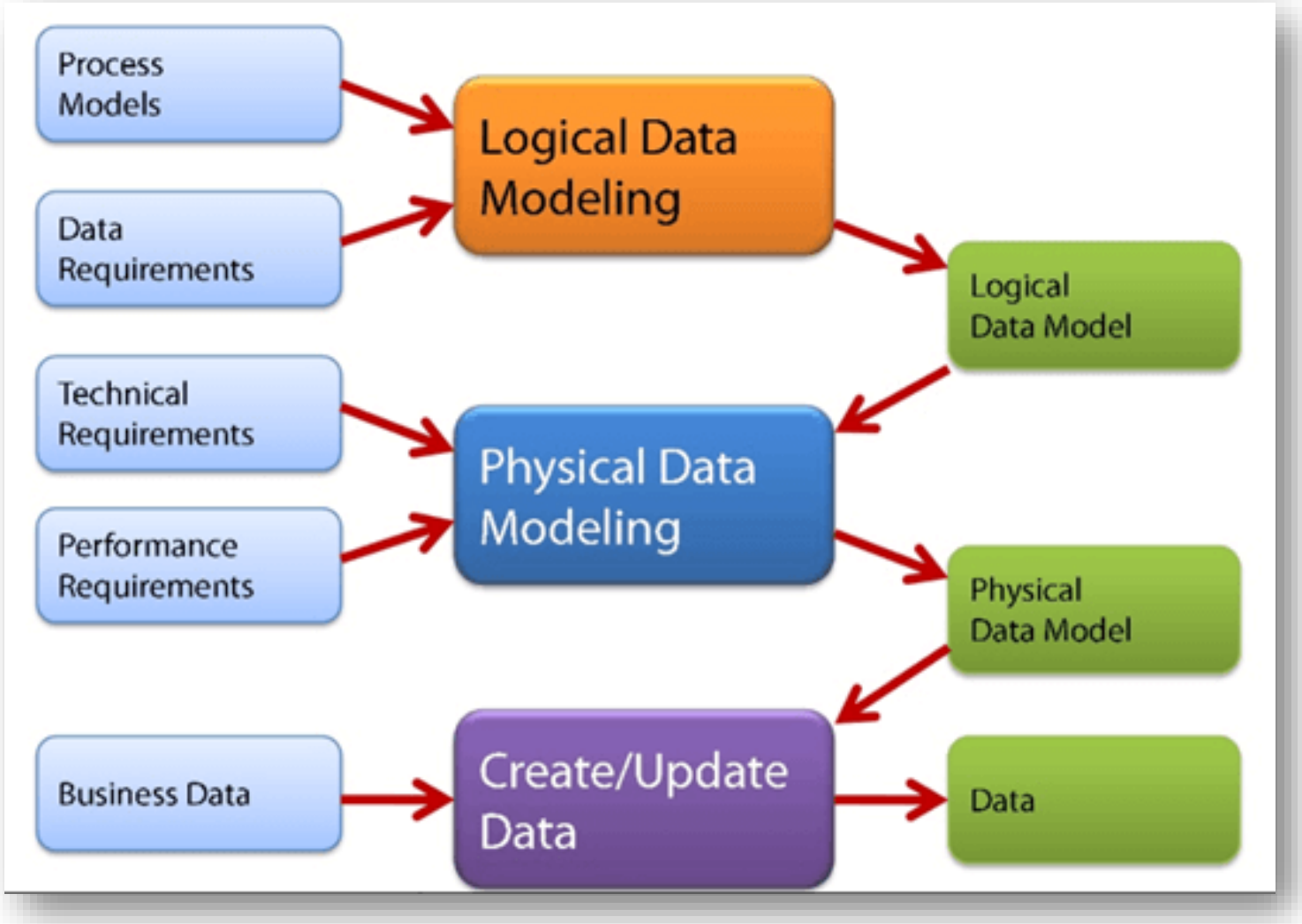
- **Logical Data Model**: Defines HOW the system should be implemented regardless of the DBMS. This model is typically created by Data Architects and Business Analysts. The purpose is to developed technical map of rules and data structures.

- **Physical Data Model**: describes HOW the system will be implemented using a specific DBMS system.
    - With this model, you would establish primary and secondary keys in a relational database or decide whether to embed or link your data in a document database such as MongoDB.
    - You will also establish the data types for each of fields. This will provide you with database schema.

Types of Data Models

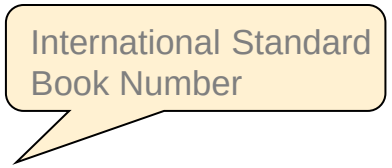


Types of Data Models



Data Models Components

- **Entity**
  - An independent object and is a logical component in the system.
  - In document databases, each document is an entity. In tabular databases, each row is an entity.
- **Entity types**
  - The categories used to **group entities**.
  - For example, the book entity with the title “Alice in Wonderland” belongs to the entity type “book.”
- **Attributes**
  - The characteristics of an entity.
  - For example, the entity “book” has the attributes ISBN (String) and title (String).



- **Relationships**
  - Define the connections between the entities.
  - For example, one user can borrow many books at a time. The relationship between the entities "users" and "books" is one to many.

Features of a NoSQL Database

- **Schema flexibility**: Data are stored in **json** documents. **Json** provides a rich data model that seamlessly maps to native programming languages types, and the dynamic schema makes it easier to evolve your data model

- **Horizontal scaling**: refers to dividing your database into chunks and **stores them on multiple servers**. The main advantage of this approach is that you can add additional servers on the fly to increase your database performance with zero downtime.

- **Ad-Hoc Queries**: Optimizing the way in which ad-hoc queries are handled can make a significant difference at scale, when thousands to millions of variables may need to be considered

- **Indexing**: MongoDB offers a broad range of indexes and features with language-specific sort orders that support complex access patterns to datasets.

- **Load Balancing**: via horizontal scaling features like replication and sharding, MongoDB supports large-scale load balancing. The platform can handle multiple concurrent read and write requests for the same data

Goal of data modeling

- ✓ To identify all the data components of a system
- ✓ How they are connected
- ✓ What are the best ways to represent these relationships.
- ✓ Data modeling allows you to identify the possible relationships between different pieces of information, which will determine what type of queries can be run against that data.

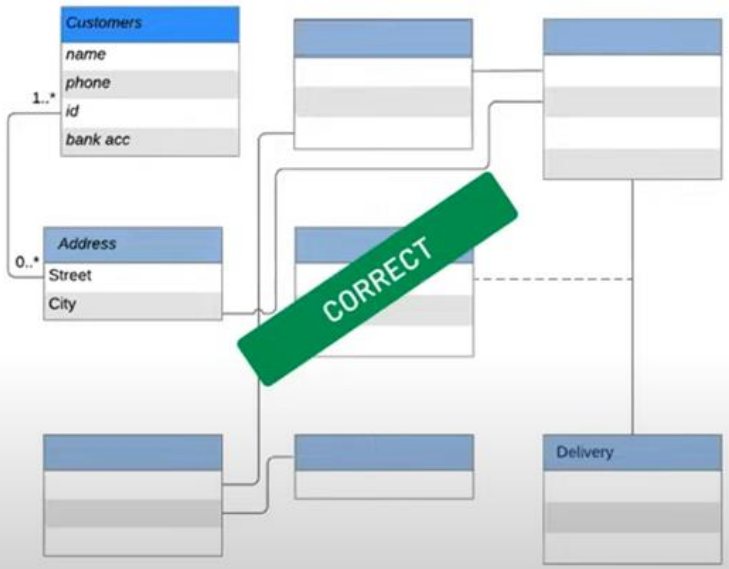
Data Modeling for RDBMS

Modeling for RDBMS

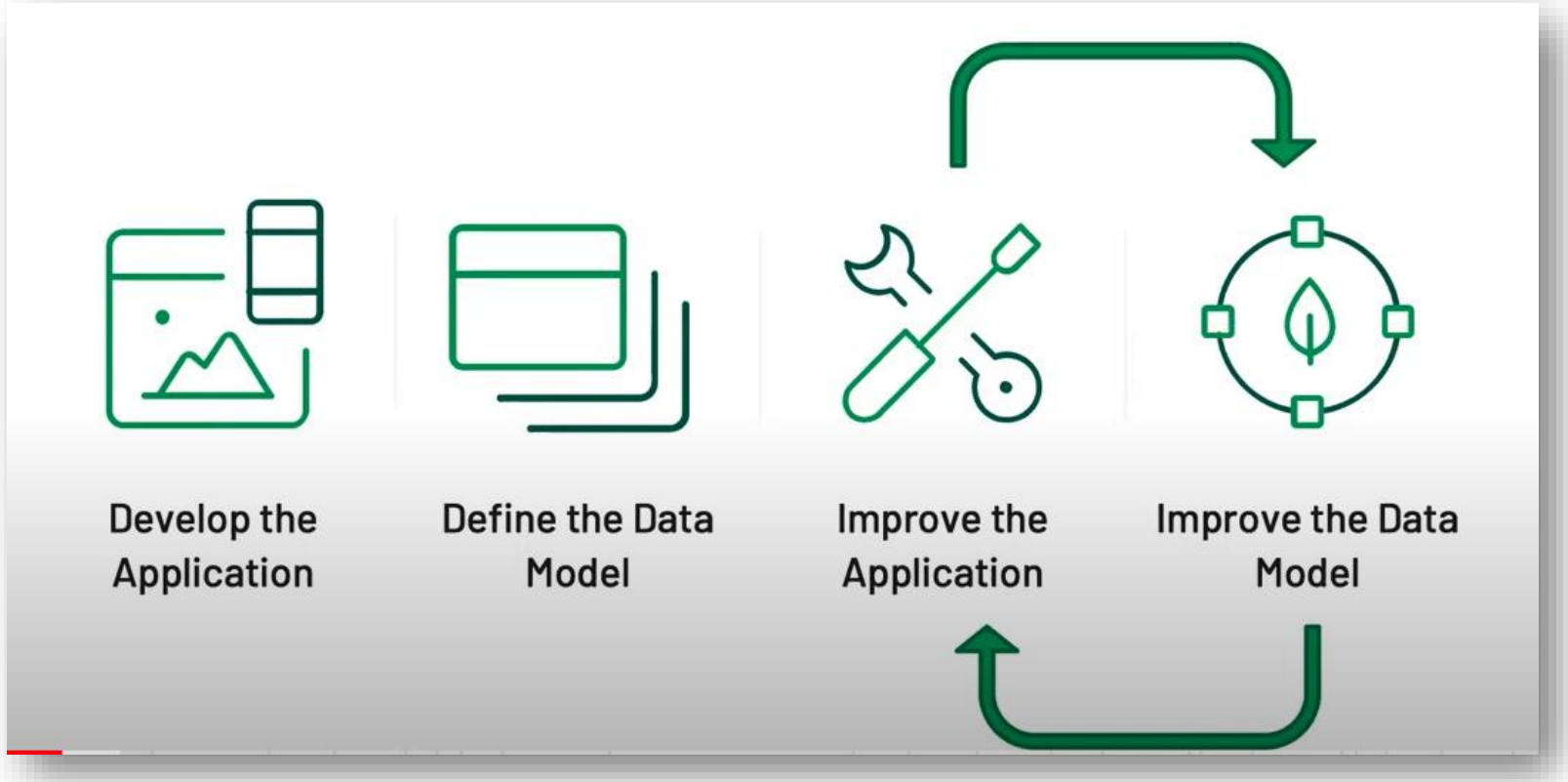
Step 1: Define the Schema

Step 2: Develop the application and queries

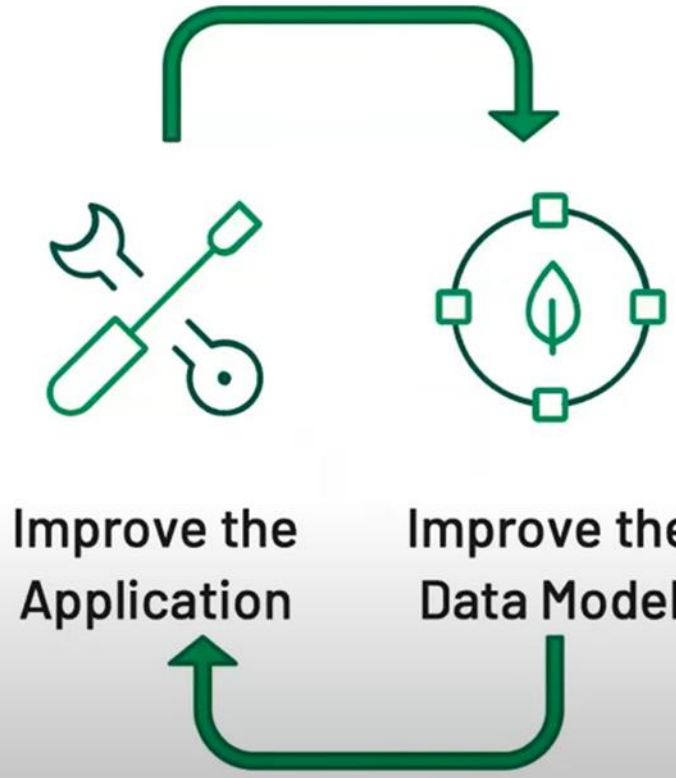
Concerns



Data Modeling with MongoDB



Data Modeling with MongoDB



Many design options

Designed for the usage pattern

Data model evolution is easy

Can evolve without any downtime

Data Modeling with MongoDB

There Is No Magic Formula, but There Is A Method

Data model is defined at the application level

Design is part of each phase of the application lifetime

What affects the data model:

- The data that your application needs
- Application's read and write usage of the data

Example of a Data Model

Build an application for the users of a library

First, you will speak with the **business analysts** to understand the entities that need to be part of your system.

- ***Books****: The library has millions of books, and they all have a unique International Standard Book Number (ISBN). The users will need to search books by title or by author.*
- ***Users****: This library has thousands of users, and each user has a name, along with an address. The library will assign them a unique number that they can find on their library card.*

Second, You need to understand how the various entities will interact with each other. These interactions will give you the relationships in your model. In the case of the library example, interactions might look like:

- ***Users will borrow books****: the library will need to know which books have been borrowed by which user.*
- *Each user is entitled to five borrowed books at a time.*

These business rules will let you organize the information to build your conceptual model. By now, you understand the data necessary to build the first iteration of your software.

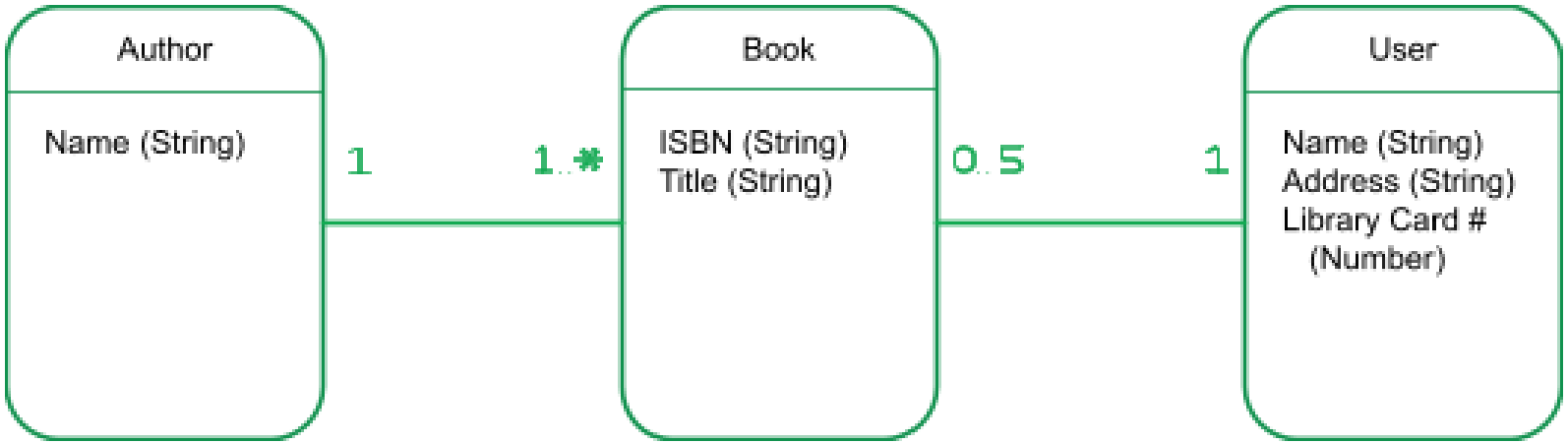
Example of a Data Model

The conceptual model

- **These Business Rules** will let you organize the information to build your **conceptual model**.

The logical model

- You may find that some data structures are more complex and require new entities.
- For example, authors would be better represented as their own entity to allow searching for books by authors. Assume that there is a single author for each book.



Example of a Data Model

It is now time to choose your DBMS and build your physical data model.

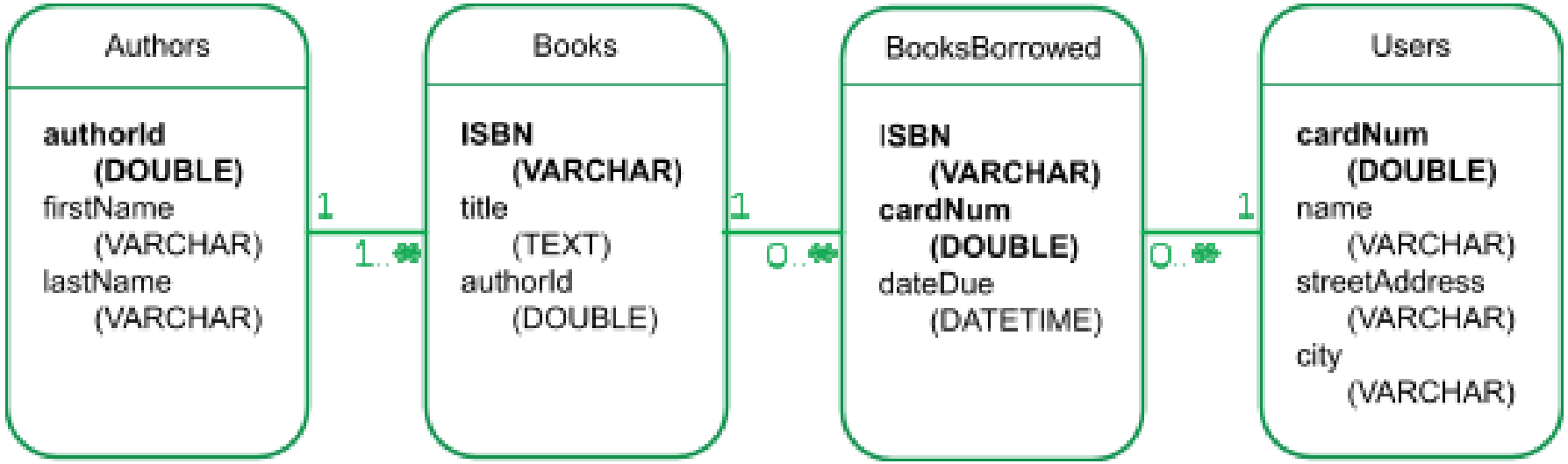
At this point, you will start thinking in terms of the database you picked.

The type of database you choose will determine how you will store the data.

Example of a Data Model

The physical data model for a Relational Database

- In this example, the authors and books table are linked through a one-to-many relationship.
- The authorId field is the primary key in the authors' table, and the authorId field would be the foreign key in the books table.
- A joint table is added to keep track of the borrowed books along with the due dates.



Example of a Data Model

The physical data model for a Document Database

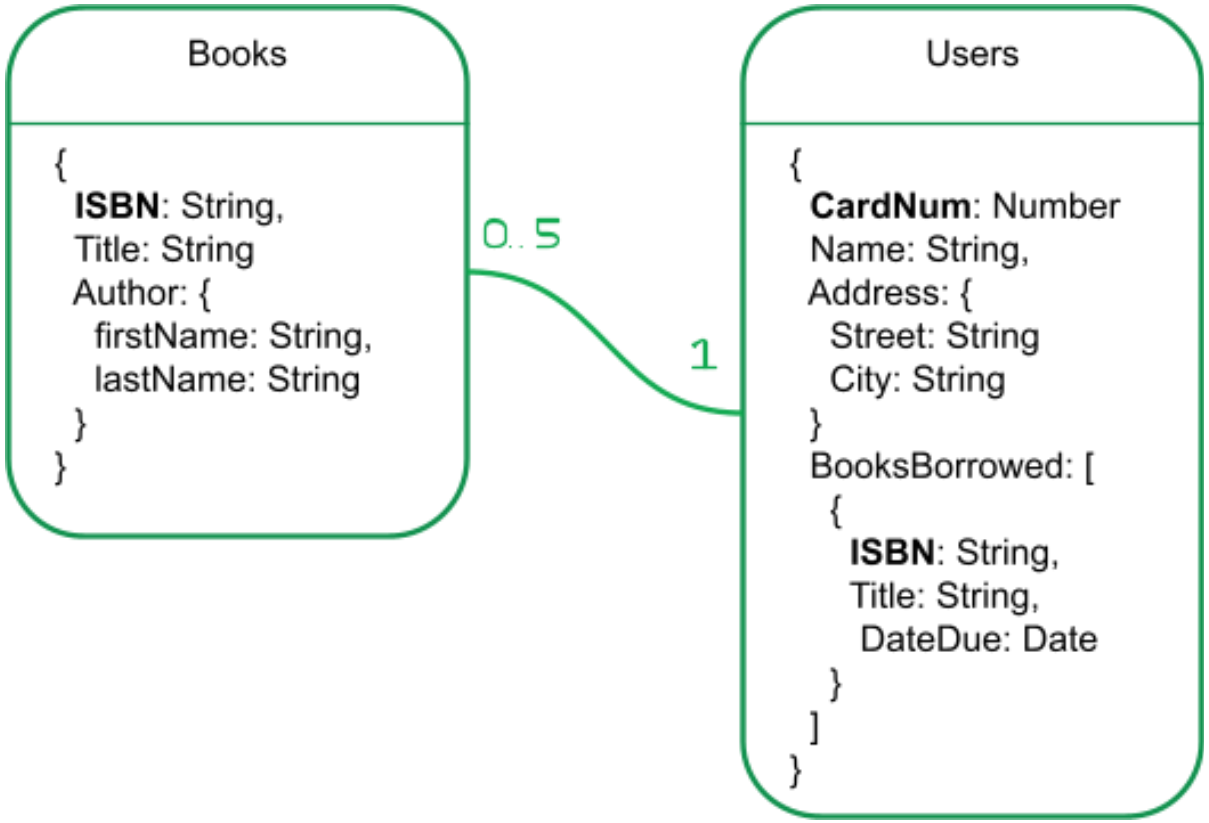
- You will model relationships using embedding or document references.
- As you establish the relationships between various objects, you will also find your IDs and unique values representing your items.
- To create indexes to enable the full-text search capabilities of MongoDB Atlas Search.
- The books borrowed are listed as an array in the user document because this information will be generally retrieved all at once on the application's main page.

Example of a Data Model

The physical data model for a Document Database

A different use case with this same library data might have called for a different physical data model.

- The ISBN and CardNum fields are unique for the documents and could be used as the ID field.
- You could also use them as a shading key if you need to scale to multiple clusters.

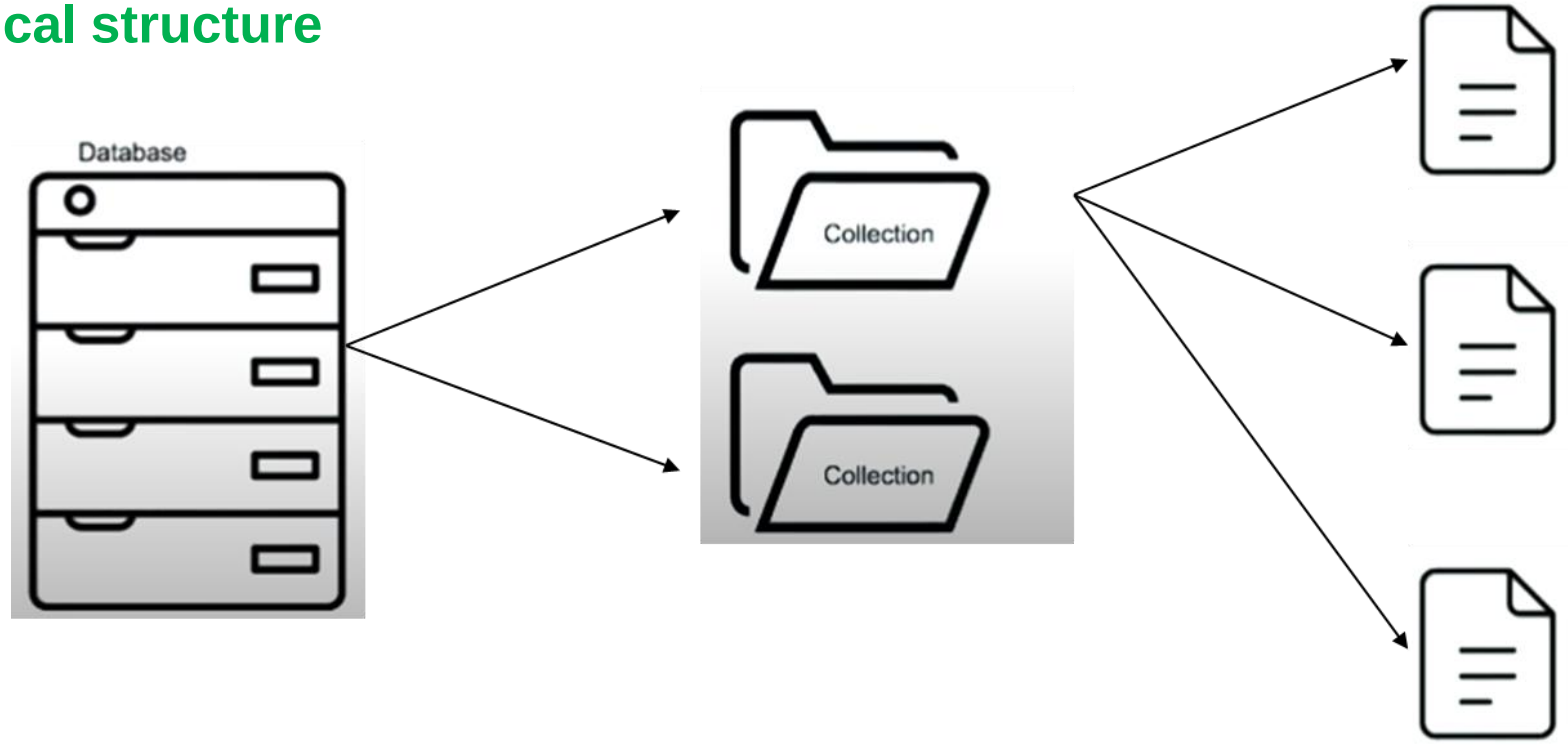


Document Model in MongoDB

- MongoDB Data Hierarchy
- Documents
- Document have schema

MongoDB Data Hierarchy

Hierarchical structure



- MongoDB stored in a hierarchical structure where the database are at the top level where each MongoDB deployment can have many databases.
- Then there are one or more collections in the database.
- And finally, there are documents which are kept at the collection level.

Document Model in MongoDB

- **Document Model** is a Document-Oriented Database, which means that data is stored as documents, and Documents are grouped in collections**.**

- The documents in a single collection don't necessarily need to have exactly the same set of fields. This is what we call a “flexible schema”

- Documents in MongoDB are stored in the BSON format, which is a binary-encoded JSON format.

Document Structure

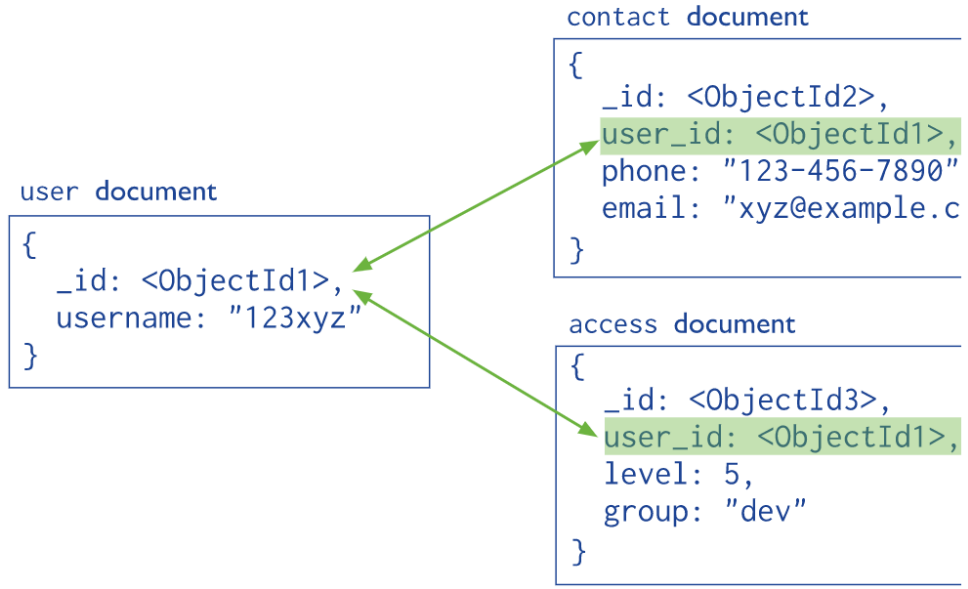
- The key decision in designing data models for MongoDB applications revolves around the **structure of documents** and how the application represents relationships between data.

- MongoDB allows related data to be embedded within a single document.

Document Structure

Embedded Data: MongoDB documents make it possible to embed document structures in a field or array within a document. These denormalized data models allow applications to retrieve and manipulate related data in a single database operation.

References: store the relationships between data by including links or references from one document to another. Applications can resolve these references to access the related data. Broadly, these are normalized data models.



Documents

- In MongoDB, data is stored as BSON documents that are composed of field **value pairs**, where BSON is a binary representation of JSON documents.

- Since BSON is not human-readable, we will stick with JSON in our examples throughout this course.

- If you're coming from SQL, a document is like a row in the table that has been joined with other relevant rows from other tables.

- You have a field and its assigned value, just like each column in a row has a value.

Documents

Flexible structure

Documents have Schema

```
{
   "firstName" : "John",
   "lastName" : "Doe",
   "age" : 25,
   "phone" :
      [8888888888,7777777777],
   "address" :
      {"street" : "One",
       "building" : 1,
       "city" : "Capitol",
       "state" : "Moosylvania",
       "country" : "USA"}
   "education" :
      [{"College" : "Hogwarts",
       "Degree" : "Auror"},
      {"College" : "Durmstrang",
       "Degree" : "Professor"}]
}
```

```
"firstName" -> String,
"lastName" -> String,
"age" -> int,
"phone" -> array,
"address" -> object
"education" -> array (of objects)
```

Documents

Flexible structure

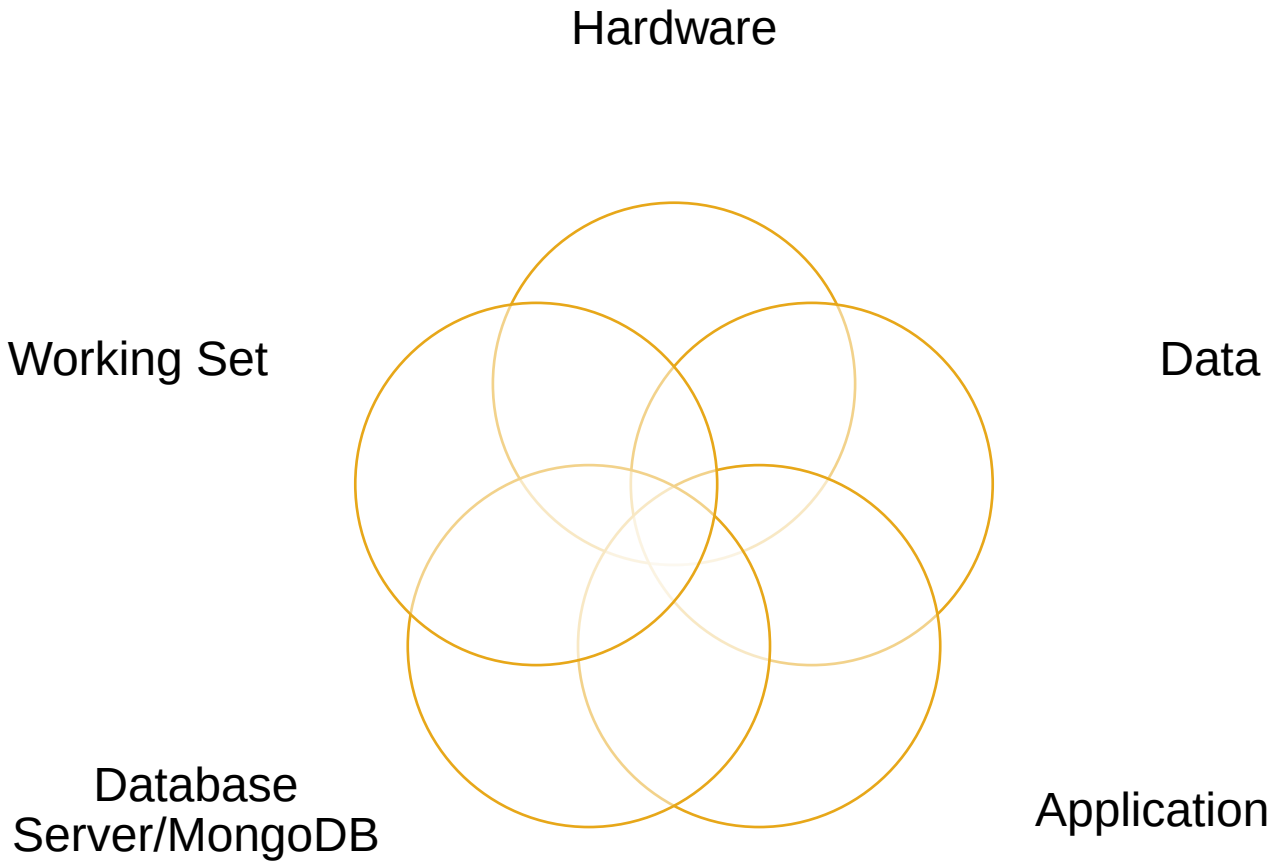
```
{
   "firstName" : "John",
   "lastName" : "Doe",
   "phone" : 8888888888
   "address" :
      {"street" : "One",
       "building" : 1,
       "city" : "Capitol",
       "state" : "Moosylvania",
      }
}
```

```
{
   "firstName" : "John",
   "lastName" : "Doe",
   "age" : 25,
   "phone" :
      [8888888888,7777777777],
   "address" :
      {"street" : "One",
       "building" : 1,
       "city" : "Capitol",
       "state" : "Moosylvania",
       "country" : "USA"}
   "education":
      [{"College" : "Hogwarts",
       "Degree" : "Auror"},
      {"College" : "Durmstrang",
       "Degree" : "Professor"}]
}
```

Recap

- MongoDB store data as Documents
- Document fields can be values, embedded documents, or arrays of values and documents
- MongoDB is a Flexible Schema Database

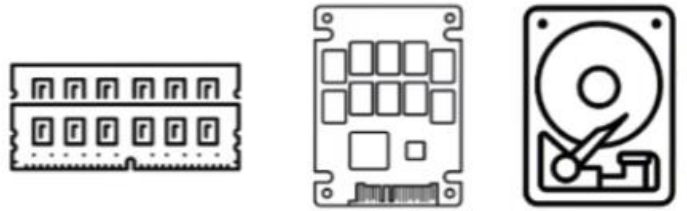
Constraints in Data Modeling



Constraints with Computer Applications

Let's start with the hardware

- Hardware
  - RAM
  - Solid State Drive/Hard Disk Drive



Constraints with Computer Applications

Restriction is related to data:

- Hardware
  - RAM
  - Solid State Drive/Hard Disk Drive
- Data
  - Size
  - Security, Sovereignty

Constraints with Computer Applications

Restriction is related to data:

- The data retention policies and data sovereignty regulations that may impact the way you define data model.

- Giving access to some pieces of data will influence how you group that data in your documents and collection.

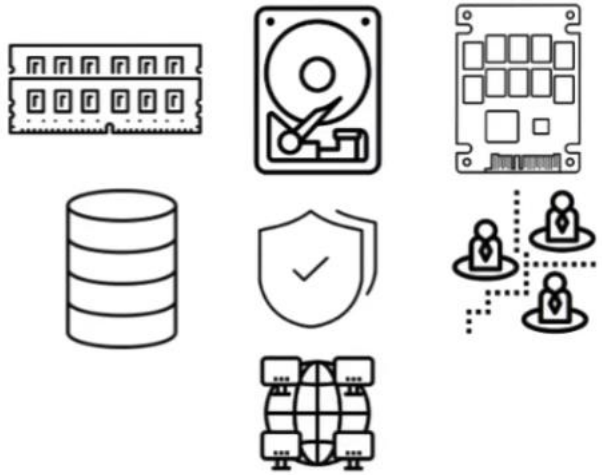
- To cover these concerns, MongoDB offers a set of access controls and security checks

Constraints with Computer Applications

The physical limits on the network's speed:

- Modeling for applications that are globally distributed and accessed by clients in different locations may require a bit more thinking about how we can make data accessible without compromising service quality.

- Hardware
  - RAM
  - Solid State Drive/Hard Disk Drive
- Data
  - Size
  - Security, Sovereignty
- Application
  - network latency

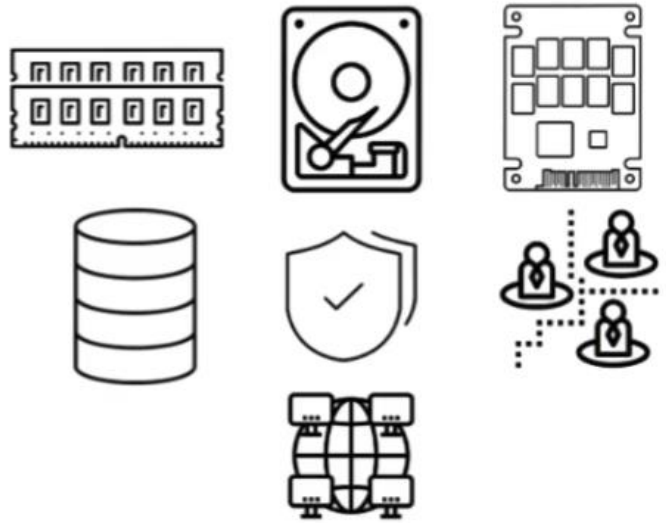


Constraints with Computer Applications

The limits of the Database Server:

- In MongoDB, a document can't be larger than 16 MB.
- And with the current WiredTiger engine, reading information from a document requires the full document to be loaded in RAM.
- Both of those constraints may drive your design to split frequently accessed data from the rest of the data less frequently accessed.

- Hardware
  - RAM
  - Solid State Drive/Hard Disk Drive
- Data
  - Size
  - Security, Sovereignty
- Application
  - network latency
- Database Server/MongoDB
  - max size of document (16 MB), atomicity of updates

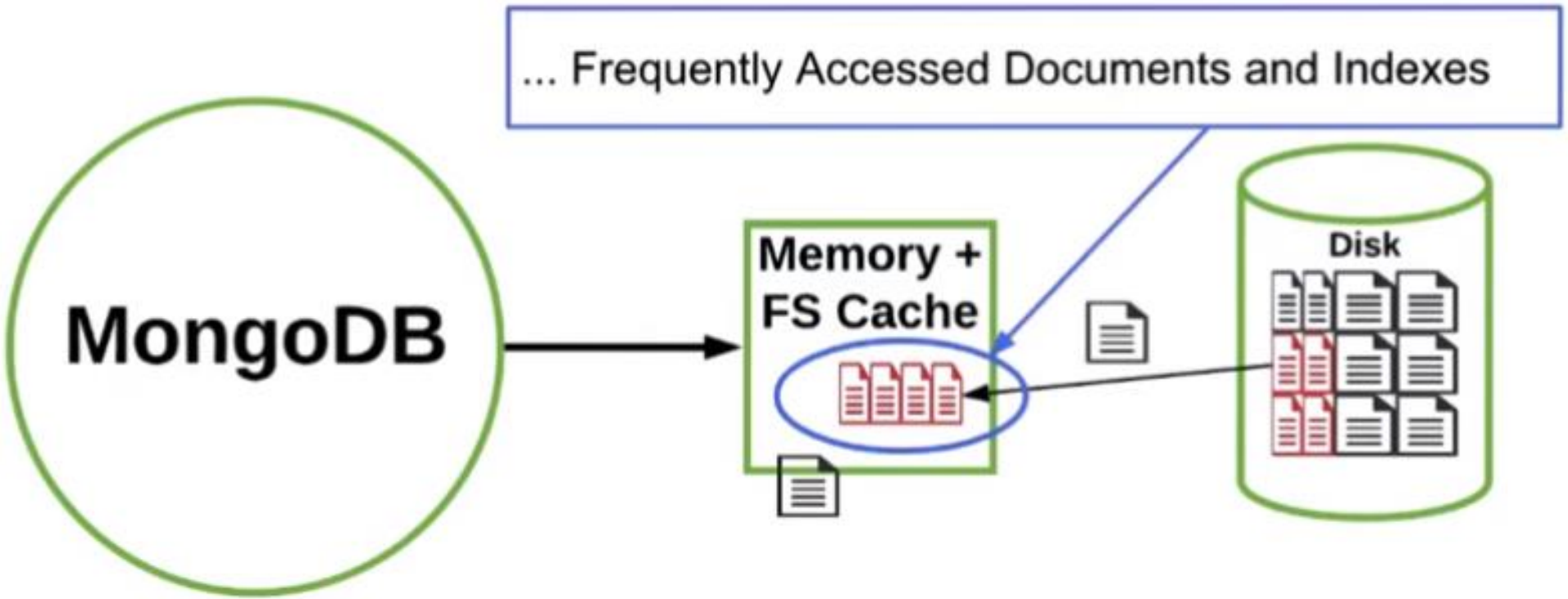


Constraints with Computer Applications

- **Summary**:
  - Keep the frequently used documents and indexes in memory
  - Keep historical data, data you don't use very often in hard disk drives
  - It is important to identify those exact constraints and their impact to create a better model. As your software and the technological landscape change, your model should be re-evaluated and updated accordingly.

Working Set

"the total body of data that the application uses in the course of normal operations"



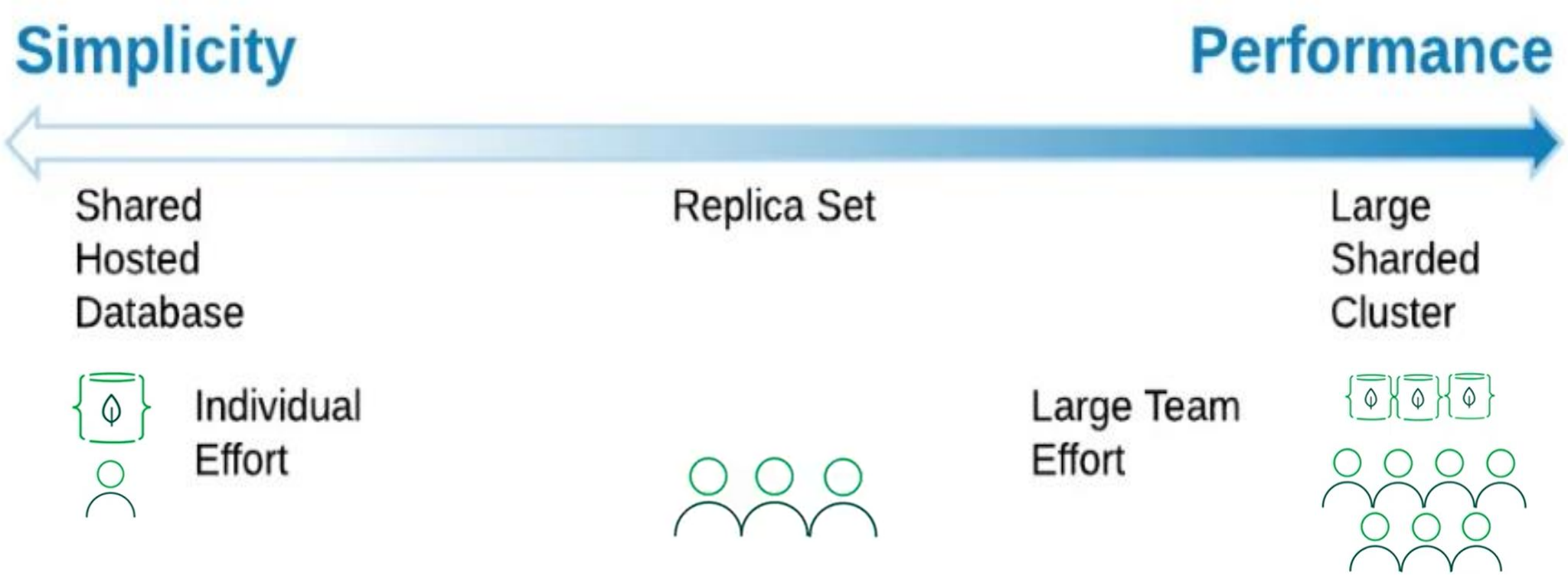
Working Set

- **Working set** represents the **total body of data** that the application uses in the course of normal operations… For best performance, the majority of your active set should fit in RAM.
- As you run queries through MongoDB, **working set** consists of **any data the server requires** to fulfill a query. As this happens, **server cache** will behave dynamically, according to what the **working set** needs.

Recap

- The nature of your dataset and hardware define the need to model your data
- It is important to identify those exact constraints and their impact to create a better model
- As your software and the technological landscape change, your model should be re-evaluated and updates accordingly

Main Trade Off in Mode



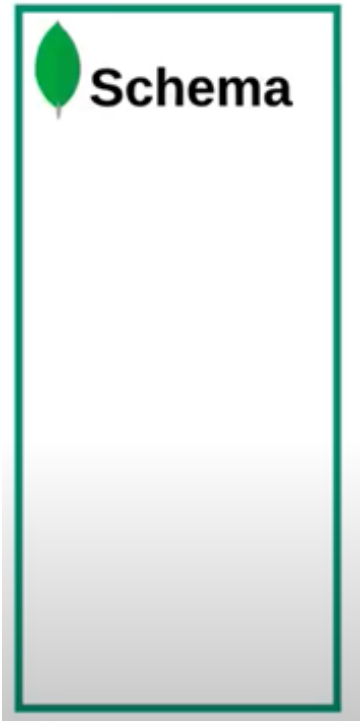
Main Trade Off in Modeling

- In different phases of methodology, It's pretty the **main trade off** when your model for something, not just for database, but for a lot of things, you often have to choose between **simplicity** and **performance**.
  - **You would choose simplicity**, if you have a very **small project** and you have an effort that is done by **one or two people**
  - **You would choose performance** if you have a **large system**, with a **big team**, you probably want to spend a little bit more time up front to design your system and performance will probably be a lot more important because it's going to have a bigger impact on larger systems

The Data Modeling Methodology

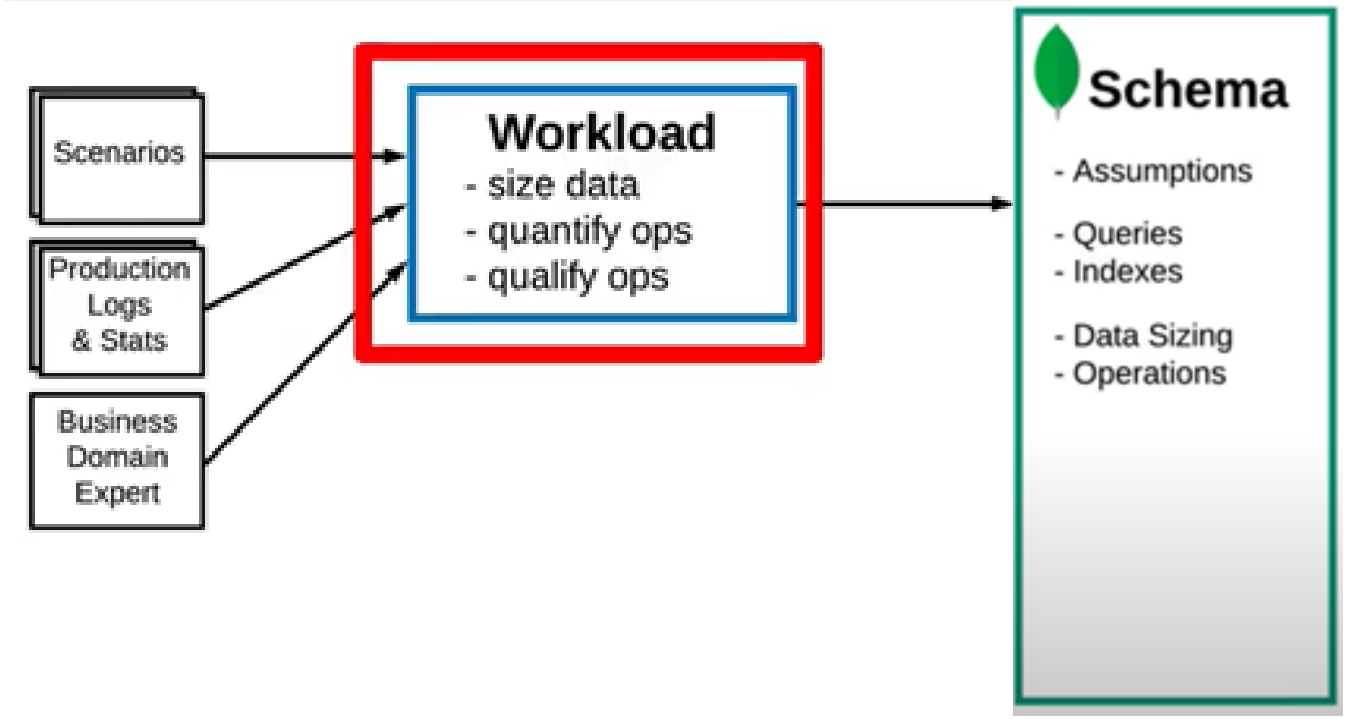
- **Methodology**
  1. Describe the workload
  2. Identify and Model the Relationships
  3. Apply Patterns
- **Flexible Methodology**
- **Recap**

Describe the Workload



Describe the Workload

- Feeding some of these inputs into **first phase**, so the first phase is describing the **Workload**, that in order to be able to model correctly.

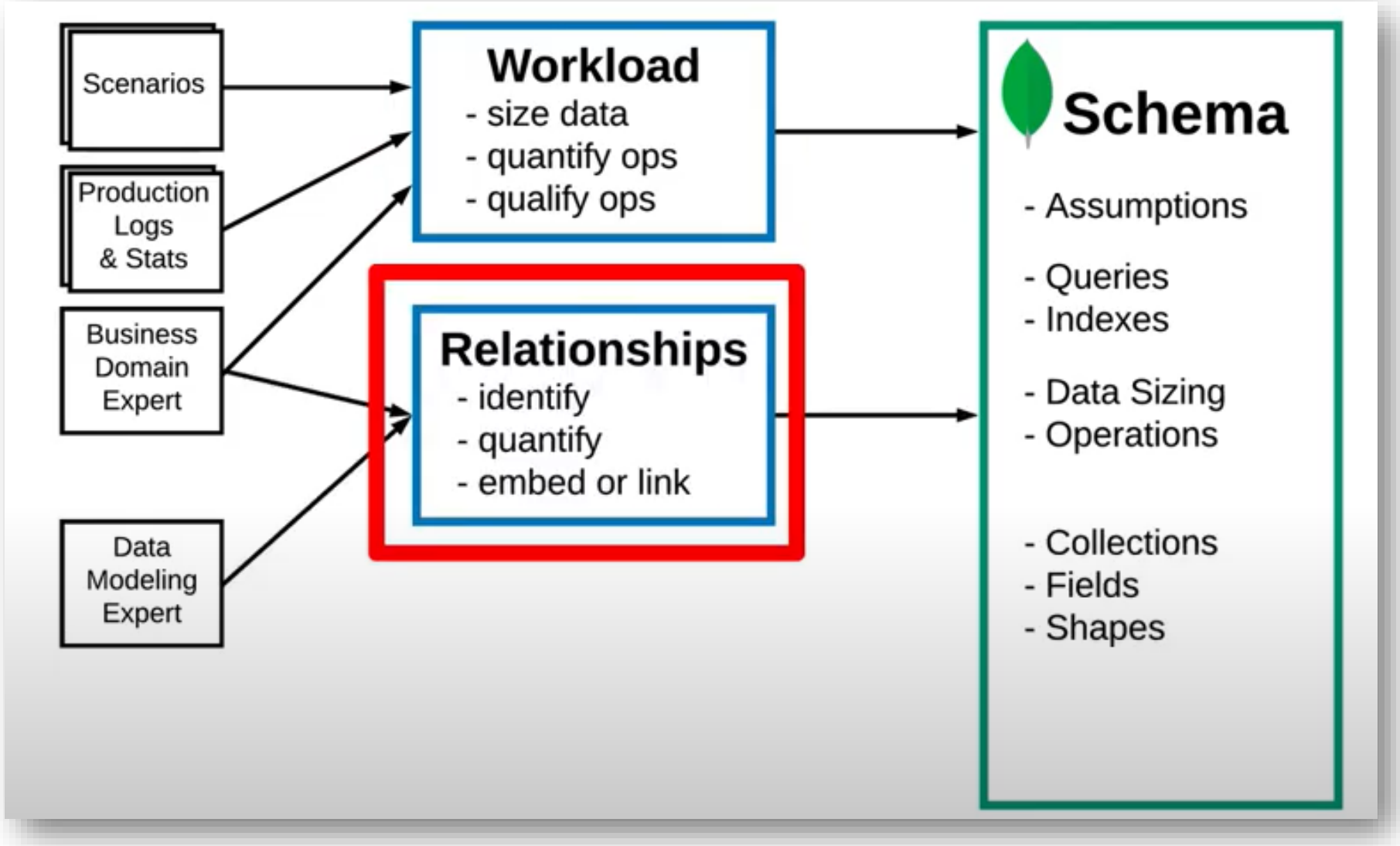


Describe the Workload

- **Example**: System will have different inputs order to design
  - **The scenarios** needs to support
  - **Production logs** if need migrating this system from a relational database to mongodb or a **prototype** so that generated logs can use
  - In both cases, logs, stats, et cetera, give you additional information about the **current state of the system**
  - Finally, we needs to assemble this information together in the **schema**

So the first phase is to look at the documents that you have in your input and create some artifacts out of them.

Identify and Model the Relationships

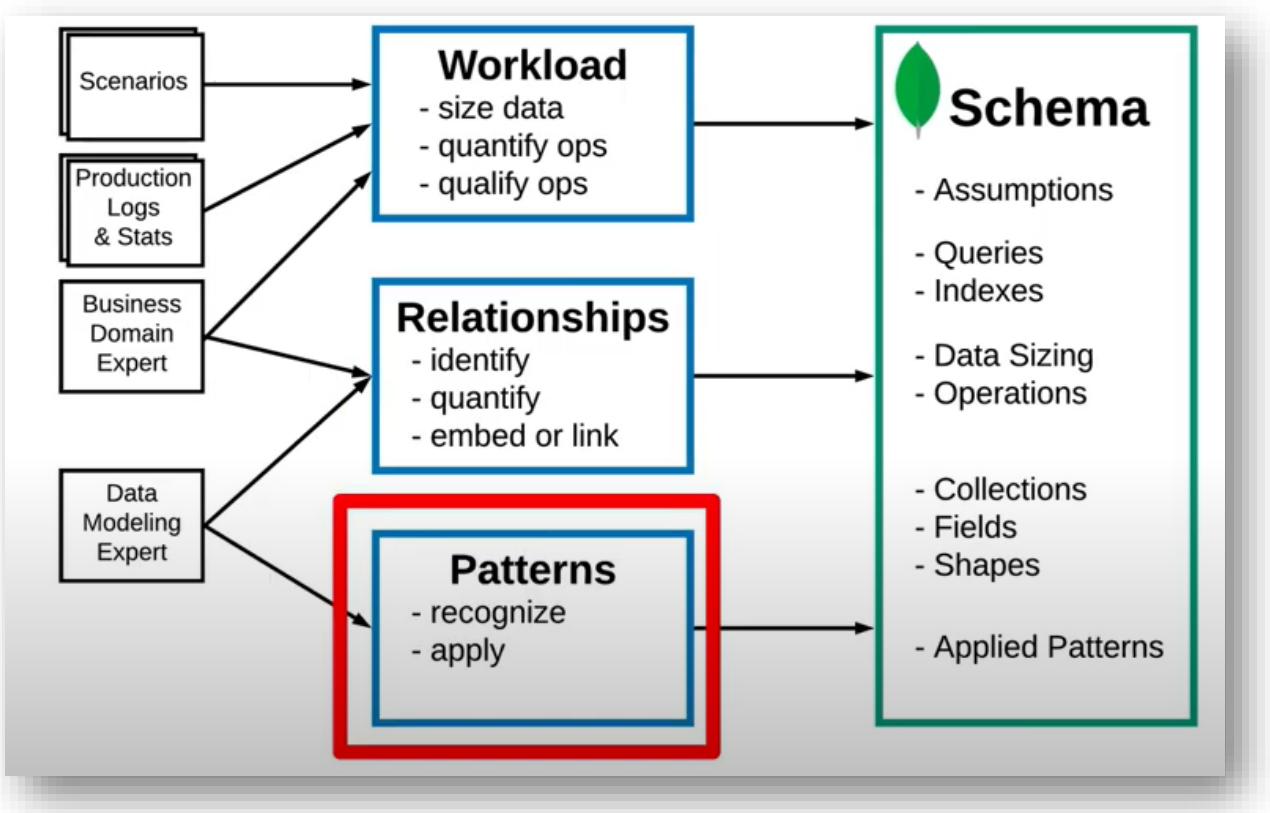


Identify and Model the Relationships

- **In second phase**, we start with a piece of information that were identified. Each piece has a **relationship** with another one.
- The ones that have a **one-to-one relationship** tend to be grouped together in the same table or **collection**.
- **At the end of this process**, you will have a **list of entities** with **their fields**, some of them grouped together inside the common **collection**.

Apply Patterns

- **The last phase is to apply schema design patterns**. This is where you will get to make your model more performant or more clear by applying some transformations.



Flexible Methodology

Goal	*Simplicity*	*Simplicity and Performance*	*Performance*
1. Describe the **Workload**	Most frequent Operation	Most Operations Quantify Ops	All Operations Quantify Ops Qualify Ops
2. Identify and Model the **Relationships**	Mostly embedding	Embedding and linking	Embedding and linking
3. Apply **Patterns**	Pattern A	Pattern A Pattern B	Pattern A Pattern B Pattern C

Recap

1. **Describe the workload**
   - Data size, important reads and writes
2. **Identify and Model the Relationships**
   - Identify them, link or embed the related entities
3. **Apply Patterns**
   - Apply the ones for need optimizations

Remain flexible by only doing the steps needed

Homeworks - Group Exercises

- Choose an information system that can use MongoDB to store data
- Describe the user's requirements for that system