Data Modelling in MongoDB

MongoDB has a flexible schema model which means the following:

- Documents in a single collection aren't required to have the same set of fields. [1]
- A field's data type can differ within documents in a collection. [1]

Documents in a collection usually have a similar collection of data. [1]

The flexible data model allows us to store data as our application requires it. It avoids joins. By avoiding joins, we ensure that application performance is increased. [1]

We must ensure that our data model has a logical structure and achieves optimal performance, we need to plan our schema before deploying it on a production database. To do this, we should take the following steps:

- Identify application workload [1]
- Map relationships between objects in our collections [1]
- Apply design patterns [1]

Identify Application Workload

When identifying the application workload, we need to identify the operations that the application runs most frequently. Knowing this detail will help us create effective indexes and minimise the calls to the DB. [2]

We need to consider the queries that occur / will occur based on the present and future functionalities of the application [2].

To identify the application workload, we need to do 3 steps:

1) Identify the data that our application needs. This is done based on the application's users and the i**nformation they need**; **business domain** and **frequently run queries**. [2]
2) The following workload table should be created with the following data in it: [2]

Action	Query Type	Info	Frequency	Priority
The action that a user takes to trigger the query. [2]	The type of query (read or write). [2]	The document fields that are either written or returned by the query. [2]	How frequently your application runs the query. [2] Queries that are run frequently benefit from indexes and should be optimized to avoid lookup operations. [2]	How critical the query is to your application. [2]

Map Schema Relations in the Data

Here's an overview whether to embed data or to use references. It depends on the relative importance of the following goals. This is the first step. [3]

Improve queries on related data
If your application frequently queries one entity to return data about another entity, embed the data to avoid the need for frequent **$lookup** operations. [3]

Improve data returned from different entities
If your application returns data from related entities together, embed the data in a single collection. [3]

Improve Update Performance
If your application frequently updates related data, consider storing the data in its own collection and using a reference to access it. When you use a reference, you reduce your application's write workload by only needing to update the data in a single place. [3]

The 2nd step is to map relationships between the data fields. Do the following:

Identify Related Data in the Application
Identify the data that your application queries and how entities relate to each other. Consider the operations you identified from your application's workload in the first step of the schema design process. Note the information these operations write and return, and what information overlaps between multiple operations. [3]

Create a Schema Map for the Related Data
We need to create an Entity-Relationship Diagram which shows the relationship between the data fields and the type of relationship between those fields. [3]

Choose Whether to Embed Data or Use References
The decision to embed data or use references depends on your application's common queries. Review the queries you identified in the Identify Application Workload step and use the guidelines mentioned earlier on this page to design your schema to support frequent and critical queries. Configure your databases, collections, and application logic to match the approach you choose. [3]

REFERENCES
[1] Data Modeling - MongoDB Manual v8.0
[2] Identify Application Workload - MongoDB Manual v8.0
[3] Map Schema Relationships - MongoDB Manual v8.0