# **MongoDB: Data Model Design**

Take five minutes to read the following:

* <https://docs.mongodb.com/manual/core/data-model-design>

Be prepared to answer the following questions:

* What is a denormalized data model?
* What is a normalized data model?
* What are the trade-offs between embedding documents vs referencing documents?

When building an app, we should consider how to design our database. Often depending on how we use the data in our app, we design the data model in such a way that supports our app's needs - performance being one of them.

"For instance, different data models can allow for more efficient queries, increase the throughput of insert and update operations .... When developing a data model, analyze all of your application’s read and write operations" - MongoDB Docs

In MongoDB there are two ways to relate data in our documents: embedding or referencing documents. In this lesson we will discuss the difference between the two. By the end of this lesson you should be informed enough to make decisions on how to design a MongoDB database according to your app's needs.

## **Denormalized Data Model**

In MongoDB you can do this:

user document

{

\_id: ObjectId("3e399709171f6188450e43d2"),

name: "Joe Schmoe",

email: "j.schmoe@gmail.com",

address: {

street: "123 Fake Street",

city: "Faketon",

state: "MA",

zip: "12345"

}

}

This looks like an object literal in JavaScript right? Well it is. In MongoDB it is common to embed related data inside a document. This structure (or schema) of embedding related data is a feature in MongoDB. In database design we call this type of schema "denormalized".

A HUGE (performance) benefit of having related data embedded in a document is that we can have all our data in one collection - that means we can create, read, update, and delete data all in one request!

"The embedded data model combines all related data in a single document instead of normalizing across multiple documents and collections. This data model facilitates atomic operations." - MongoDB Docs

## **Normalized Data Model**

With normalized data models we describe related data by referencing the related document:

publisher document

{

\_id: ObjectId("3e399709171f6188450e43d2"),

name: "Penguin Books",

location: "375 Hudson St, New York, NY 10014",

url: "https://penguin.com"

}

book document

{

title: "A New Earth",

author: "Eckhart Tolle",

published\_date: "2005",

publisher\_id: ObjectId("3e399709171f6188450e43d2")

}

Notice how we create the relationship between publisher and book via a publisher\_id field.

* in the book document we are creating a reference to a publisher document

According to MongoDB docs the reason we would want to use a normalized data model would be:

* when embedding would result in duplication of data but would not provide sufficient read performance advantages to outweigh the implications of the duplication. (We will look at an example of this in a later lesson)
* to represent more complex many-to-many relationships
* to model large hierarchical data sets

Don't worry about the last two bullet points. A lot of this knowledge will come with maturity of working with MongoDB - the more you work with it, the more you will begin to understand how the relationships work, and what relationship best suits your app's needs.