Modelling Data Relationships in MongoDB

There are 3 types of relationships in the database are:

- One-to-One (1:1)
- One-to-Many (1:N)
- Many-to-Many (M:N)

Let us take 2 entities:
Headquarters (street, city, state, zip, country);
Publisher (publisherId, name, founded, description, socials);

An example of a 1:1 relationship is that between Headquarters and Publisher. In this type of relationship, a Publisher can have 1 Headquarters and vice-versa.

An example of a 1:N relationship is that between a book and reviews. A book can have many reviews but all those reviews can only belong to 1 book.

An example of an M:N relationship is that between a book and authors. A book can have many authors and an author could also have written many books.

Embedding vs. Referencing
Referencing is the process of linking 2 documents together using the DocumentID of 1 document in the other.

Embedding is the process of directly putting a key-value pair in a document.

We should follow the golden rule of schema design: **Data that is accessed together should be stored together**.

We ask the following questions as to whether to embed or reference. We apply them to the M:N relationship between books & authors.

Simplicity: Would keeping the pieces of information (book & author) together lead to a simpler data model and simpler code? -> Yes -> Embed (+1)

Go Together: Do the pieces of info (book & author) have “has a” or “contains a” relationship? -> Yes -> Embed (+1)

Atomicity: Does the application query the pieces of information (books and authors) together? -> Yes -> Embed (+1)

Update Complexity: Are the entities (books & authors) updated together? -> No -> Reference(+1). Eg - Change an author’s biography w/o changing anything about a book that they wrote.

Archival: Do the entities (books & authors) need to be archived together? -> No -> Reference(+1). Eg - (The author has many books on our app. We wouldn't want to archive the author in case only 1 of their books was deleted.)

Cardinality: Is there a high cardinality (current or growing) on the child side (authors) of the relationship? (This question is asking us to think of the possibility of an unbounded array. However, an unbounded array is not possible here since the number of authors for a book are limited.) -> No -> Embed (+10

Data Duplication: Would data duplication be too complicated to manage and undesired? -> No -> Embed (+1)

Document Size: Would the combined size of the pieces of information (1 book & its author(s)) take up too much memory or transfer too much bandwidth to the application. (Since one book only has few authors, so) -> No -> Embed (+1)

Document Growth: Would the embedded piece grow without bound? (Limited growth over time) -> No -> Embed (+1)

Write-Heavy Workload: Are the pieces of info written at different times in a write-heavy workload? (When adding or updating books, both the book & author info will be written together. It is expected to occur 10 times / hour.) -> No -> Embed (+1)

Individuality: For the child side of the relationship (author), can the pieces exist by themselves without a parent? (In our app, an author can't exist without a book) -> No -> Embed (+1)

Embed(9), Reference (2) -> Thus we should embed the authors in the book document.

We have to consider the priority of each guideline in relation to application requirements.

Modelling One-to-One Relationships

This type of relationship is usually implemented by embedding. In this section, we will use 2 entities **Publisher** (publisherId, name, founded, description, headquarters, socials) and **headquarters** (street, city, state, country, zip). Here, **Publisher** is the parent and **headquarters** is the child.

There are 2 options for embedding:

1) Co-locate the fields of the **headquarters** entity in the **Publisher** document. This is the equivalent of a row in a relational model.
2) Group related fields together in a **headquarters** sub-document inside the **Publisher** document. This is useful if the **Headquarters** will have different addresses in different countries.

There are 3 options for referencing:

1) Add a reference to the child in the parent document. This is useful if our app primarily reads **Publisher** documents.
2) Add a reference to the parent in the child document. This is useful if our app primarily reads **headquarters** documents.
3) Add references to the parent in the child document and vice-versa.This is called bi-directional referencing.

After tabulating the 11 guidelines for this relationship, we find that the simplest solution would be to embed **headquarters** as a sub-document in the **Publisher** document.

Modelling 1:N Relationships
Here, we will use the Books - Reviews relationship.

There are 2 ways for embedding:

1) The most common way to embed 1:N relationships is to embed the N side as an array of sub-documents in the document of the 1 side. In our case, create an array of review sub-documents for the “reviews” key in the **Books** document.

2) The other way is to embed the N side as a sub-document on the one-side. Here, each review is stored as a sub-document in the “reviews” sub-document in the **Books** document.
3) The advantage of embedding is that it doesn’t create any duplicate data.

There are 3 ways of referencing:

1) Include reference(s) to the child in the parent document. In our case, references to the Reviews in the **Book** document.
2) Include reference(s) to the parent in the child document. In our case, references to the book in each review document.
3) Have bidirectional references.

In such a kind of relationship, embedding the N side as an array of sub-documents in the document of the 1 side is usually preferred. However, in our case, since a book can have unlimited (potentially, billions) of reviews, we choose to **reference the id of the book document in each review document**.

Modelling M:N Relationships
Here, we will use the relationship between **Books** and **Authors**.

There are 3 ways to embed Authors in Books:

- We can store Authors an array of sub-documents in each Book document.
- We can store Authors as a sub-document with key-value pairs, in which each author is a sub-document in the “authors” sub-document in the Books document.

There are 2 ways to reference:

- We can have an array of references to authors in the Books document. This is feasible since a Book can have limited Authors. This will lead to a little data duplication since the author ids have to be present in each Book document. (An author's id will be there in the document of each Book that he wrote). A little data duplication is OK.
- We can have an array of references to Books in the Authors document. This second approach will lead to large arrays since an author can write many books and is thus infeasible. This will lead to a lot of data duplication, since the Book ids will be there in the document of each author that wrote the same books and especially for an author who wrote many books.
- Bi-directional referencing, however, is more expensive.

The solution (according to the 11 guidelines / rules) is to embed the Authors as an array of sub-documents in the Book document.