Collections in MongoDB

This lesson delves further into the concept of documents and collections; it also goes over the differences between MongoDB and Relational Databases.

WE'LL COVER THE FOLLOWING

In the previous lesson, we discussed documents briefly. Now let's get into the details.

What are Collections?

Documents are stored inside of *collections*.

> **Collections** are groups of somehow related documents, but these documents don't need to have the same structure.

Here lies one of the biggest benefits of MongoDB: developers don't need to know the schema of the database beforehand but can modify the schema, dynamically, during development. This is especially great in systems where we can't get the schema quite right in the beginning, or there are plenty of edge cases to cover.

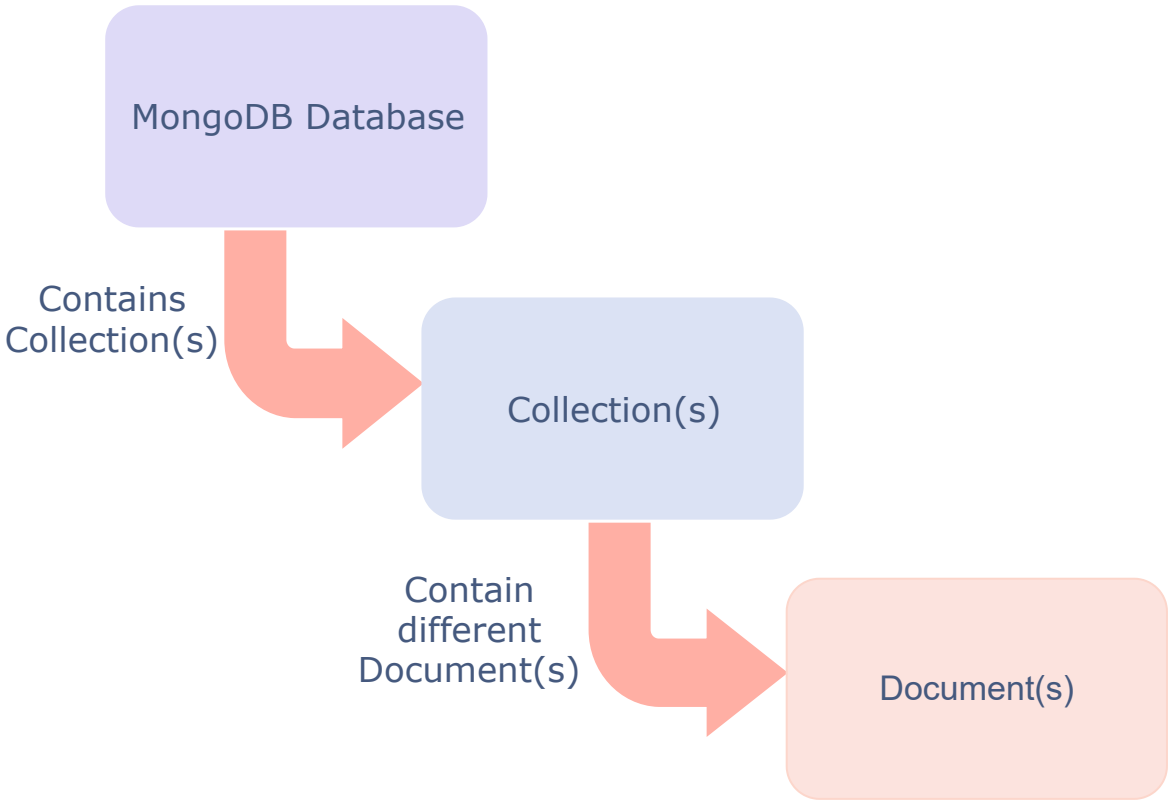
Also, this way, the entire problem with impedance mismatch is avoided (i.e., elimination of the object-relational mapping layer).

What does this look like?

Well, let's say that the previous document is stored in the collection called `users`; we could add another document into that collection which would

users, we could add another document into that collection which would

contain fields that the previous document didn't have, or we could add a document that may not have the fields that the previous document had.



Example

As an example, we could add the next document into the collection:

Document1 | Document2

```
//previous document
{
    "_id" : ObjectId("58e28d41b1ad7d0c5cd27549"),
    "name" : "Nikola Zivkovic",
    "blog" : "rubikscode.net",
    "numberOfArticles" : 10,
    "Adress" : [
        "street" : "some street",
        "city" : "Novi Sad",
        "country" : "Serbia"
        ],
    "company" : "Vega IT Sourcing",
    "expertise" : [".NET", "JavaScript", "NoSQL", "Node.js"]
}
```

Adding documents to the collection

As you can see these documents are similar, but not the same. The new document doesn't contain the numberOfArticles field, but it does contain an

document doesn't contain the `numberOfArticles` field, but it does contain an additional `location` field which the previously added document didn't have.

Collection groups then, give you the ability to add indexes to these documents. Indexes are one of the concepts that MongoDB inherited from relational databases.

Differences Between MongoDB & Relational Databases

It's important to emphasize some of the differences between MongoDB and Relational databases.

Firstly, MongoDB doesn't have foreign keys; but, it has a feature that looks quite like that – **References**.

Any object can have a reference to some other object, using its `id`, but this is not automatically updated, and it's up to the application to keep track of these connections.

This is done in this way because once a *foreign key* is introduced in a relational database, it can be hard to unwind the database from it.

Thanks to the document data model, and due to the fact that all the necessary information for one "record" is stored inside one document, *joins* are not provided in MongoDB. However, a similar mechanism called **Lookup** is available.

Among other differences, it should be mentioned that there is *no* equivalent of multiple-table transactions in MongoDB.

That is quite a lot of theory-- so let's see how it looks in practice.