Nested Documents

The document storage model in NoSQL databases is well-suited to use nested documents. For example, instead of having two collections—`posts` and `users`—we can have a single collection (`users`), with each item of that collection having `posts`.

The decision of whether to use separate collections or nested documents is more of an architectural question, and its answer depends on usage. For example, if posts are used only in the context of users (their authors)—say, on the users' profile pages—then it's best to use nested documents. However, if the blog features multiple users' posts that need to be queried independently of their user context, then separate collections fit better.

To implement nested documents, we can use the type `Schema.Types.Mixed` in Mongoose schemas (`Schema`, e.g., `bookSchema` or `postSchema`) or we can create a new schema for the nested document. An example of the former approach is as follows:

```
const userSchema = new mongoose.Schema({
  name: String,
  posts: [mongoose.Schema.Types.Mixed]
})
//attach methods, hooks, etc.
let User = mongoose.model('User', userSchema)
```

However, the latter approach of using a distinct new subschema is more flexible and powerful:

```
var postSchema = new mongoose.Schema({
  title: String,
  text: String
})
//attach methods, hooks, etc., to post schema
var userSchema = new mongoose.Schema({
  name: String,
  posts: [postSchema]
})
//attach methods, hooks, etc., to user schema
var User = mongoose.model('User', userSchema)
```

To create a new user document or to save a post to an existing user when working with a nested posts document, treat the `posts` property as an array and just use the `push` method from the JavaScript/Node.js API, or use the MongoDB `$push` operand (http://docs.mongodb.org/manual/reference/operator/update/push/). For example, we can add a post (`newPost`) to a user object, which is found by a matching ID (`_id` is `userId`):

```
User.update(
  {_id: userId},
  {$push: {posts: newPost}},
  function(error, results) {
    //handle error and check results
  })
```