

Aula 5: Modelo de dados NOSQL orientado a documentos - parte I

Apresentação

Na Aula 4, concluímos o estudo do modelo de banco de dados NOSQL chave-valor, onde verificamos comandos NOSQL para a definição da estrutura de tabelas chave-valor com JSON e com JSONB, e escrevemos comandos NOSQL para realizar a manipulação de dados. Estudaremos agora, na Aula 5, as características do banco de dados orientado a documentos, analisaremos os formatos de documentos permitidos para uso nesse modelo, e estudaremos ainda os comandos NOSQL para a definição da estrutura do banco de dados.

Bom estudo!

Objetivo

- Reconhecer as características do banco de dados orientado a documentos;
- Analisar os formatos de arquivos permitidos para uso nesse modelo;
- Escrever comandos NOSQL para a definição da estrutura do banco de dados.

Introdução

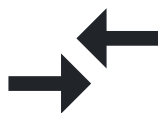
Nesta Aula 5, iniciaremos o estudo do modelo do banco de dados NOSQL no modelo orientado a documentos (SADALAGE; FOWLER, 2013).

Destacam-se como vantagens desse tipo de banco de dados o suporte a agrupamentos de conjuntos de pares do tipo chave-valor.

O modelo chave-valor foi visto nas Aulas 3 e 4, e apresenta registros com atributos formados por pares de chaves-valor. Porém, esse modelo tem como característica principal o armazenamento de documentos, diferente daquele que se restringe a valores. Neste, cada registro no banco de dados representa um documento e, os documentos devem ser organizados em coleções de documentos que apresentem características semelhantes entre si.

Modelo chave-valor

Nesse modelo, a base de dados é de uma forma simplificada um conjunto de pares do tipo chave-valor.



Modelo orientado a documentos

Nesse modelo, a base de dados pode ser vista como um repositório de documentos, e esses documentos podem ser representados com estruturas que apresentam grandes diferenças em seus formatos, porém são usadas no mesmo banco de dados.

Assim como o chave-valor, o modelo orientado a documentos não exige um esquema (*schema*) pré-definido.

Apesar do termo “documentos”, o banco pode armazenar dados comuns como registros de *logs* de operações de sistema (que poderiam também ser armazenados em uma tabela no modelo relacional). Entretanto, a vantagem é que os *logs* podem ser armazenados em diferentes estruturas, o que seria muito útil em uma situação de uso como uma coleção (que é uma tabela no relacional) recebendo dados de *logs* de diversas fontes ou de outros sistemas de *logs*.

A aula está organizada da seguinte forma: primeiramente vamos conhecer as características dos bancos de dados orientados a documentos, depois verificar os tipos de formatos de arquivos usados no modelo orientado a documentos e, no final da aula, vamos conhecer o sistema gerenciador de banco de dados (SGBD) [MongoDB](#) para usá-lo na escrita de comandos para a definição das estrutura do banco de dados.

Características do banco de dados orientado a documentos

O banco de dados orientado a documentos é considerado por usuários como uma evolução do modelo do tipo chave-valor. De uma forma análoga ao modelo relacional (ELMASRI; NAVATHE, 2018), os dados no modelo orientado a documentos são armazenados como coleções de documentos. Assim, temos:

No modelo relacional usamos a estrutura baseada em tabelas e registros.

No modelo orientado a documentos, uma coleção pode ser formada por um ou vários documentos.

Os documentos representam as unidades de armazenamento de dados, e esses são representados nos SGBDs como linhas ou registros de uma tabela. Em termos de composição, um documento pode ser um item simples ou composto.


Exemplo

Como exemplo, podemos considerar documentos representando listas, documentos simples ou documentos embutidos em outros.

Segue uma tabela comparativa entre os modelos relacional e orientado a documentos, sendo que o SGBD representante deste último é o MongoDB.

Tabela 1 - Comparação entre termos usados no modelo relacional de banco de dados e no SGBD MongoDB

Modelo relacional	Modelo orientado a documentos
Banco de dados (<i>database</i>)	Banco de dados (<i>database</i>)
Tabela	Coleção
Linha, registro ou tupla	Documento

 **Atenção!** Para visualização completa da tabela utilize a rolagem horizontal

Formatos de arquivos permitidos no modelo orientado a documentos

Quanto aos tipos de formatos de arquivos que podem ser usados para o armazenamento de um documento, esses podem ser codificados de formas diferentes, como nos formatos:

- XML (eXtensible Markup Language);
- JSON (JavaScript Object Notation);
- JSONB (usado no PostgreSQL);
- BSON (Binary JSON);
- YAML ([YetAnother](#) Markup Language);
- Com textos simples.

Comentário

O formato JSON é o mais usado atualmente. Como exemplos desses bancos de dados podemos citar o MongoDB, [RavenDB](#), [CouchDB](#) e PostgreSQL com o formato JSON e JSONB.

Todos os SGBDs listados anteriormente armazenam dados em JSON. Verificaremos a seguir exemplos de documentos nos formatos XML e JSON:

Em XML:

```
<?xml version="1.0"?>
<!DOCTYPE acervo>
<acervo>
  <DATA_RECORD>
    <idtitulo>1</idtitulo>
    <acervo_detalhes>{"peso" : 200,
      "autor" : [ "Pramod J. Sadalage", "Martin Fowler" ] ,
      "titulo" : "NoSQL Essencial" ,
      "editora" : "Pearson" ,
      "idlivro" : [ 100, 101, 102, 103 ] ,
      "isbn_id" : "978-85-7522-338-3", "paginas": 220}
    </acervo_detalhes>
  </DATA_RECORD>
</acervo>
```

<https://pt.stackoverflow.com/questions/80881/o-que-%C3%A9-e-pra-que-serve-yaml>

Em JSON:

```
{
  "peso": 200,
  "autor": [
    "Pramod J. Sadalage",
    "Martin Fowler"
  ],
  "titulo": "NoSQL Essencial",
  "editora": "Pearson",
  "idlivro": [
    100,
    101,
    102,
    103
  ],
  "isbn_id": "978-85-7522-338-3",
  "paginas": 220
}
```

Comandos NOSQL para a definição da estrutura do banco de

Agora chegou o momento de praticar as operações básicas do tipo CRUD (Cadastrar, Consultar, Atualizar e Excluir – do Inglês Create, Read, Update, Delete) em um SGBD orientado a documentos. Para isso, vamos usar o MongoDB. Poderíamos usar vários outros, conforme os listados em diversos sites que apresentam rankings de uso de SGBDs em diversas categorias, como o caso do [DB-Engines Ranking](#), que apresenta o MongoDB em 5º lugar entre todos os modelos de bancos de dados.

O MongoDB, lançado em 2009, é um dos muitos SGBDs orientados a documentos, e com certeza um dos mais usados. Entre suas vantagens está sua velocidade na manipulação de grandes volumes de dados, na casa dos petabytes, tendo em vista que foi projetado para o ambiente distribuído.

É disponibilizado livremente, além disso tem código aberto e é multiplataforma. Você pode trabalhar com esse banco de dados de três formas:

————— 1 —————

Por meio de sua interface gráfica conhecida como [Compass](#).

————— 2 —————

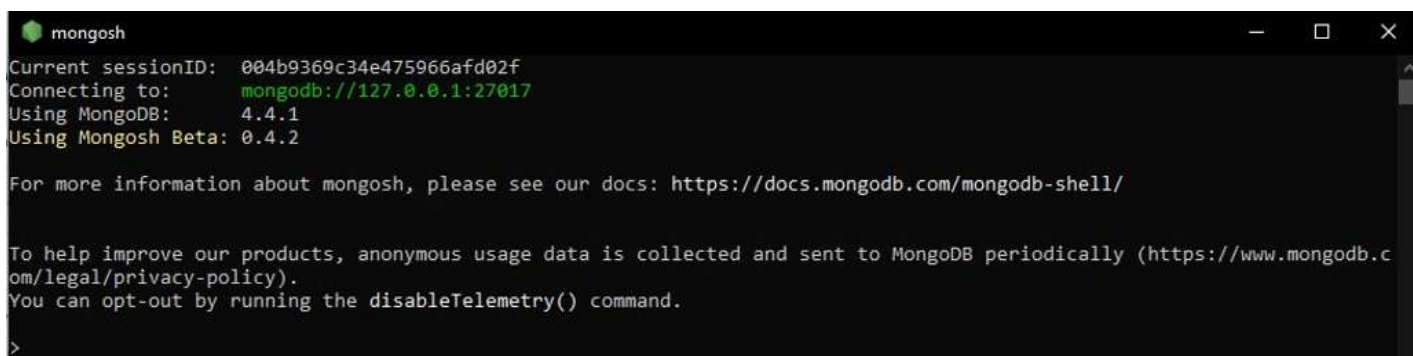
Por meio de linha de comando pelo [MongoDB Shell](#), também conhecido como mongosh (que iremos usar durante nessa aula).

————— 3 —————

Há ainda uma forma de estudo inicial online por meio de exemplos que podem ser acessados no site: <https://docs.mongodb.com/manual/tutorial/getting-started/>.

Ou seja, o SGBD disponibiliza diversas formas de uso aos seus usuários de acordo com a preferência de cada um.

Figura 1 - Interface do MongoDB Shell



```
mongosh
Current sessionID: 004b9369c34e475966afd02f
Connecting to:    mongodb://127.0.0.1:27017
Using MongoDB:    4.4.1
Using Mongosh Beta: 0.4.2

For more information about mongosh, please see our docs: https://docs.mongodb.com/mongodb-shell/

To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

>
```

📷 Fonte: MongoDB.

Antes de selecionar a forma com a qual prefere usar o SGBD, há a necessidade de baixar e instalar o servidor de bancos de dados do [MongoDB](#). Vamos, então, iniciar o estudo de códigos para a criação de coleções e de documentos no MongoDB. A primeira atividade será

criar o banco de dados.

Diferentemente de outros SGBDs, no MongoDB não existe o comando para a criação de um banco de dados, tal como o comando *create database* em SQL. O MongoDB vem instalado com os *databases* admin e local já criados. Eles podem ser verificados pelo comando *show* a seguir:

```
show dbs
```

Para criar um banco de dados no MongoDB, basta indicar o nome do banco como se ele já existisse. Para isso, digite apenas o comando *use* e o nome do database:

```
use aula
```

Se executarmos o comando para exibir os *databases* existentes, o banco aula não aparecerá, mas está criado. Ele vai aparecer apenas quando for feita a inclusão de dados.

Na sequência, criaremos uma coleção, destacando que uma coleção é composta por vários documentos. No comando a seguir, criaremos a coleção pedidos com os seguintes parâmetros:

———— 1 ————

Nome

Nesse exemplo, é igual a pedidos.

———— 2 ————

***Capped* (opcional)**

Indica que a coleção a ser criada será limitada e com um tamanho especificado.

———— 3 ————

***Size* (opcional)**

Indica o tamanho da coleção em bytes.

———— 4 ————

***Max* (opcional)**

Indica o total de documentos permitidos para a coleção.

Vejamos, então, o exemplo para a coleção pedidos:

```
db.createCollection("pedidos", {capped:true, size:1310720, max:500} )
```

Porém, a coleção pode ser criada sem os opcionais, apenas desta forma:

```
db.createCollection("pedidos" )
```

Para ver se a coleção foi criada, execute o comando a seguir:

```
show collections
```


A operação de copiar e colar no MongoDB Shell é concluída apenas ao clicar com o botão direito do mouse. Outra observação é que, para limpar a tela com os comandos anteriores, basta clicar a combinação de teclas CTRL + L.

Após criados o banco de dados e a coleção, vamos incluir um documento nessa coleção (o que seria semelhante a incluir um registro em uma tabela em um banco de dados relacional). Em nosso exemplo, trata-se do documento de um pedido de um cliente (semelhante ao exemplo apresentado na aula anterior no modelo chave-valor).

```
db.pedidos.insert (
  { cliente : 'Nei',
    produto : 'Produto_1' ,
    autor : 'curso NOSQL' ,
    qtd : 8 ,
    peso: 1,
    unidade_medida: 'kg' } )
```

O banco cria um `id_field` automaticamente para ser usado como identificador único.

O comando `db.collection.find()` apresentado a seguir serve para listar todos os documentos de uma coleção. Segue o código para a consulta para a coleção `pedidos`:

db.pedidos.find()

Atenção! Aqui existe uma videoaula, acesso pelo conteúdo online

A Figura 2 apresenta o resultado das operações anteriores:

Figura 2 - Resultado das operações de inclusão dos documentos anteriores na coleção

```
> db.pedidos.find()
[
  {
    _id: ObjectId("5f7f890f1ef8d02630563a38"),
    cliente: 'Nei',
    produto: 'Produto_1',
    autor: 'curso NOSQL',
    qtd: 8,
    peso: 1,
    unidade_medida: 'kg'
  },
  {
    _id: ObjectId("5f7f89161ef8d02630563a39"),
    cliente: 'Rui',
    produto: 'Produto_2',
    autor: 'curso NOSQL',
    qtd: 9,
    peso: 10,
    unidade_medida: 'kg'
  },
  {
    _id: ObjectId("5f7f89161ef8d02630563a3a"),
    cliente: 'Lia',
    produto: 'Produto_3',
    autor: 'curso NOSQL',
    qtd: 15,
    peso: 200,
    unidade_medida: 'kg'
  }
]
```

 Fonte: MongoDB.

Para inserir mais de um documento da coleção, usamos o comando a seguir:

```
db.pedidos.insert([
  { cliente: 'Rui',
    produto: 'Produto_2',
    autor: 'curso NOSQL',
    qtd: 9,
    peso: 10,
    unidade_medida: 'kg'},
  { cliente: 'Lia',
    produto: 'Produto_3',
    autor: 'curso NOSQL',
    qtd: 15,
```

```
peso: 200,  
  unidade_medida: 'kg'}  
])
```

Ainda no conjunto de comandos do CRUD, caso seja necessário excluir todos os registros de uma coleção, pode-se executar o comando a seguir, onde o nome da coleção é informado no comando:

```
db.pedidos.remove({})
```

Para excluir uma coleção completa no banco de dados (estrutura e dados), deve-se executar os comandos a seguir, depois de indicar o banco de dados que deseja excluir:

```
use aula  
db.dropDatabase()
```

A seguir serão apresentadas consultas simples de seleção na coleção em uso nos exemplos. A primeira consulta tem por objetivo listar os dados do produto de nome Produto_3:

```
db.pedidos.find( { produto : 'Produto_3' } )
```

Agora, vamos listar os dados dos produtos com quantidade igual a 9 ou a 15:

```
db.pedidos.find(  
  { qtd : { $in: [9, 15] } } )
```

Para listar os documentos com peso maior do que 10 kg, basta executar o comando a seguir, onde \$gt significa *greater then* (maior do que):

Atenção! Aqui existe uma videoaula, acesso pelo conteúdo online

```
db.pedidos.find( { peso: { $gt: 10 } } )
```

Como os comparadores são diferentes dos usados no modelo relacional, a seguir é apresentada uma Tabela com a lista completa de operadores de comparação de consultas usados no MongoDB e a forma correspondente em SQL ao lado:

Tabela 2 - Comparação entre operadores de comparação no MongoDB e em SQL relacional de banco de dados e no SGBD MongoDB

Operador

Expressão no MongoDB

Expressão em SQL

Igual	db.pedidos.find({peso:{\$eq: 10 }})	where peso = 10
Menor do que	db.pedidos.find({peso:{\$lt: 10}})	where peso < 10
Menor ou igual a	db.pedidos.find({peso:{\$lte: 10}})	where peso <= 10
Maior do que	db.pedidos.find({peso:{\$gt: 10}})	where peso > 10
Maior ou igual a	db.pedidos.find({peso:{\$gte: 10}})	where peso >= 10
Diferente	db.pedidos.find({peso:{\$ne :10}})	where peso != 10

 **Atenção!** Para visualização completa da tabela utilize a rolagem horizontal

Para atualizar o valor de um documento, podemos usar o comando apresentado no exemplo a seguir. No caso, alteraremos o valor da chave qtd para 25 onde o nome do produto é igual a Produto_3:

```
try {
  db.pedidos.updateOne(
    { "produto" : "Produto_3" },
    { $set: { "qtd" : 25 } }
  );
} catch (e) {
  print(e);
}
```

Para atualizar muitos documentos em uma só operação, devemos usar o comando updateMany(). Vejamos o exemplo para alterar o nome do autor do curso para todos os documentos. A Figura 3 apresenta o resultado da aplicação do comando:

```
try {
  db.pedidos.updateMany(
    { autor: "curso NOSQL" },
    { $set: { "autor " : "curso de BD NOSQL" } }
  );
} catch (e) {
  print(e);
}
```

Figura 3 - Listagem dos documentos da coleção após a operação de atualização

```
● mongosh
> try {
...   db.pedidos.updateMany(
.....     { autor: "curso NOSQL" },
.....     { $set: { "autor" : "curso de BD NOSQL" } }
.....   );
... } catch (e) {
...   print(e);
... }

> db.pedidos.find()
[
  {
    _id: ObjectId("5f7f890f1ef8d02630563a38"),
    cliente: 'Neli',
    produto: 'Produto_1',
    autor: 'curso de BD NOSQL',
    qtd: 8,
    peso: 1,
    unidade_medida: 'kg'
  },
  {
    _id: ObjectId("5f7f89161ef8d02630563a39"),
    cliente: 'Rui',
    produto: 'Produto_2',
    autor: 'curso de BD NOSQL',
    qtd: 9,
    peso: 10,
    unidade_medida: 'kg'
  },
  {
    _id: ObjectId("5f7f89161ef8d02630563a3a"),
    cliente: 'Lia',
    produto: 'Produto_3',
    autor: 'curso NOSQL',
    qtd: 25,
    peso: 200,
    unidade_medida: 'kg'
  }
]
```

 Fonte: MongoDB.

Para verificar os valores de uma chave sem repetição, basta usar a opção `distinct`, assim como na SQL:

```
db.pedidos.distinct( "autor" )
```

Para contar a quantidade de documentos de uma coleção, pode-se usar este comando:

```
db.pedidos.countDocuments({})
```

Quando for necessário excluir um documento de uma coleção, pode-se usar o comando `deleteOne()`. No exemplo a seguir, o documento cujo identificador foi passado na consulta, será excluído da coleção:

```
try {  
  db. Pedidos.deleteOne( { "_id" : ObjectId("5f7f89161ef8d02630563a3a") } );  
} catch (e) {  
  print(e);}
```

Atenção! Aqui existe uma videoaula, acesso pelo conteúdo online

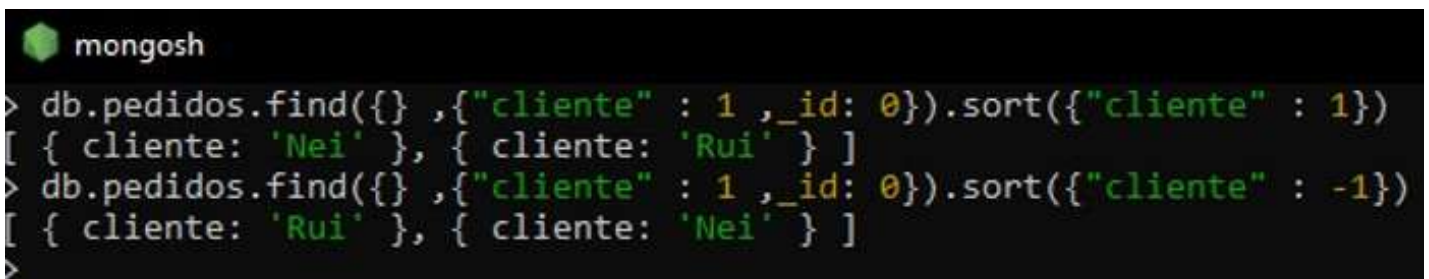
Pode-se apresentar os documentos de uma coleção de forma indentada, assim como foi feito na Aula 4 em relação aos registros com o tipo de dado JSONB. Vejamos como fica o comando para a coleção `pedidos`.

```
Db.pedidos.find({}).pretty()
```

É possível consultar os dados e apresentá-los de forma ordenada. Para isso, utiliza-se o método `sort()` para efetuar a classificação de documentos de uma coleção no MongoDB. Para a determinação da ordem, usa-se os valores 1 ou -1 para a ordem crescente e decrescente, respectivamente. Nos exemplos a seguir, a classificação aplicada aos documentos da coleção `pedidos` é efetuada na chave `cliente` de modo ascendente e logo depois de modo decrescente. Os resultados são apresentados da Figura 4.

```
Db.pedidos.find({},{ "cliente" : 1 ,_id : 0}).sort({ "cliente" : 1})  
db.pedidos.find({},{ "cliente" : 1 ,_id : 0}).sort({ "cliente" : -1})
```

Figura 4 – Listagem dos documentos ordenados usando a chave cliente



```
mongosh  
> db.pedidos.find({},{ "cliente" : 1 ,_id : 0}).sort({ "cliente" : 1})  
[ { cliente: 'Nei' }, { cliente: 'Rui' } ]  
> db.pedidos.find({},{ "cliente" : 1 ,_id : 0}).sort({ "cliente" : -1})  
[ { cliente: 'Rui' }, { cliente: 'Nei' } ]
```

Além dos documentos usados nos exemplos, pode-se ainda definir documentos embutidos, como no caso a seguir, para os dados de um cliente:

```
--criação da coleção
db.createCollection("clientes", {capped:true, size:1310720, max:500})

--uso da coleção e inclusão de um documento
use clientes
db.clientes.insert(
{
"nome" : "Luís",
"endereço" :
{
"rua" : "Rua A",
"cidade" : "Rio de Janeiro",
"state" : "RJ"
}
})
--consulta ao document cadastrado
db.clientes.find()
```

Conclusão

Nesta Aula 5, tivemos as primeiras noções do conteúdo de banco de dados orientado a documentos. Usamos o SGBD MongoDB para nosso estudo. Na aula 6, estudaremos o projeto de banco de dados orientado por meio de agregações. Utilizaremos ainda outro SGBD, o mesmo PostgreSQL que usamos na Aula 4, agora com o JSONB, para demonstrar como implementar os “agregados” para uso com coleções de documentos.

Atividade

1. Assinale a afirmativa errada quando a relação entre a nomenclatura usada na definição de uma estrutura de um banco de dados orientado a documentos e um banco de dados relacional.

- a) Banco de dados → banco de dados.
- b) Linha → documento.
- c) Tabela → coleção.
- d) Registro → para chave-valor.
- e) Tupla → documento.

2. Assinale a afirmativa que indica um formato de arquivo que não é usado para o armazenamento de dados em um banco de dados orientado a documentos:

- a) BSON.
 - b) XLS.
 - c) XML.
 - d) JSON.
 - e) JSONB.
-

3. De acordo com os comandos para a criação da tabela avaliações e da inclusão dos documentos apresentados a seguir, indique o comando que pode ser usado para exibir os documentos de forma indentada:

```
use aula
db.createCollection ("avaliacoes", { capped : true, size : 1410740, max : 500 } )
use avaliacoes
db
show dbs

db.avaliacoes.insert( [
{
  titulo : 'MongoDB' ,
  descricao : 'MongoDB é um banco de dados orientado a documentos' ,
  endereco : 'https://www.mongodb.com/' ,
  tags : [ 'banco de dados NOSQL', 'Mongodb', 'NOSQL' ] ,
  curtidas : 100
} ,
{
  titulo : 'CouchDB',
  descricao : 'CouchDB é um banco de dados orientado a documentos' ,
  endereco : 'https://couchdb.apache.org/' ,

  tags : [ 'CouchDB', 'banco de dados NOSQL', 'NOSQL' ] ,
  curtidas : 20
} ,
{
  titulo : 'Couchbase' ,
  descricao : 'Couchbase é um banco de dados orientado a documentos e diferente do CouchDB' ,
  endereco : 'http://couchbase.com/' ,
  tags : [ 'banco de dados NOSQL', 'Couchbase', 'NOSQL' ] ,
  curtidas : 10
}
] )
```

- a) SELECT jsonb_pretty(detalhes) FROM avaliacoes.
 - b) SELECT db. avaliacoes.find({}).pretty().
 - c) SELECT db.avaliacoes FROM pretty().
 - d) SELECT * FROM db. avaliacoes.find({}).
 - e) db. avaliacoes.find({}).pretty().
-

Notas

Referências

Próxima aula

Na próxima aula, daremos continuidade aos estudos vistos aqui, abordando os seguintes conteúdos:

- Modelos de dados agregados;
- Estudos de casos de modelos de dados agregados de documentos;
- Comandos NOSQL para a manipulação de dados em modelos de dados agregados de documentos.

Explore mais

Para aprofundar seus conhecimentos, fica a sugestão de leitura do livro *Sistema de banco de dados* de Elmasri, R.; Navathe, S.B.; *Sistemas de Banco de Dados* no Cap. 24.3: Sistemas NOSQL baseados em documentos e MongoDB.

De modo a entender com mais detalhes do modelo orientado a documentos, faça a leitura do Capítulo 9 do livro de Pramod J. Sadalage e Martim Fowler, *NoSQL Essencial - Um Guia Conciso para o Mundo Emergente da Persistência Poliglota*.