



mongoDB

**Guia rápido para iniciantes
por Diego Silveira Mota**

<http://www.mundointerativo.com>

Sobre este livro

Licença

Este e-book é licenciado sobre atribuição não comercial. **Você não precisa pagar por este livro.**

Sobre o autor

Sou Diego Silveira Mota, desenvolvedor desde 2000 com experiência em diversas tecnologias, tendo como especialidade o desenvolvimento web com PHP como linguagem principal de programação, bancos de dados PostgreSQL, MySQL e MongoDB, Javascript, CSS e HTML5.

Agradecimentos

Primeiramente a Deus, depois minha família e meus amigos. Agradeço também a você por estar lendo este e-book.

Prefácio

Este e-book é destinado para profissionais que possuem interesse em conhecer o banco de dados não-relacional MongoDB e desejam adquirir conhecimento de uma forma bastante prática e didática.

Neste livro irei abordar de maneira simples as principais ações com o MongoDB, como instalação no ambiente Linux e Windows, operações CRUD, modelos de dados, administração, segurança entre outros.

Para escrever este livro eu me baseei na própria documentação do MongoDB que pode ser encontrada atualmente em <https://docs.mongodb.org/master/core/indexes-introduction/>, no seu manual oficial que pode ser encontrado em <https://docs.mongodb.org/manual/MongoDB-manual-v3.0.pdf> e no e-book **The Little MongoDB Book** de **Karl Seguin** que pode ser baixado em <http://openmymind.net/mongodb.pdf>

Espero sinceramente que lhe seja útil e que goste do material aqui exposto, também conto com sua opinião sobre o livro e divulgação.

Diego S Mota

Introdução

Para falar de MongoDB temos que explicar NoSQL (Não somente Sql ou não relacional), que é um termo genérico para uma classe definida de bancos de dados não-relacionais.

Estes bancos de dados existem desde 1960 mas nesta época ainda não havia o termo NoSQL, que veio a surgir somente após uma onda de popularidade no início do século XXI desencadeada pela necessidade das grandes empresas web como Facebook, Google e Amazon.

As principais razões para o sucesso destes bancos de dados devem-se a simplicidade de design, escalabilidade horizontal de clusters de máquinas (que são ainda hoje um problema grande para bancos de dados relacionais, sejam pelo custo das máquinas, sejam pelo valor dos profissionais envolvidos) e por fim a alta disponibilidade oferecida por estes serviços.

Os bancos de dados NoSQL são cada vez mais utilizados em processamento de BigData e aplicações em tempo real. Neste último caso imagine um banco de dados que não tenha performance e seja utilizado por milhares de pessoas ao mesmo tempo em um jogo online, caso o banco de dados não tenha performance ninguém iria jogar pois teria atraso na informação e o jogo perderia o sentido.

Algumas características destes bancos de dados:

- Alta performance e desempenho
- Escalabilidade
- Suporte de replicação nativo
- Suporte a dados estruturados
- Mapreduce
- Baixo custo operacional
- Raízes Open Source
- Balanceamento de carga

Cases de sucesso

- Twitter
- Facebook
- Digg
- Amazon
- LinkedIn
- Google
- Yahoo
- The New York times
- Bit.ly

Instalação no Ubuntu

Visão geral

No exemplo aqui citado estou usando o Ubuntu 14.04.2 LTS num ambiente em cloud. O procedimento a seguir será para a versão especificada, porém caso você esteja utilizando outro ambiente acesse o link <https://docs.mongodb.org/master/installation/> e veja o tutorial correspondente.

Pacotes

MongoDB fornece os pacotes oficiais em seu próprio repositório. São eles:

- **mongodb-org** – este é o pacote principal e é automaticamente instalado com qualquer outro pacote desta lista.
- **mongodb-org-server** – este pacote contém o mongod e é associado aos “init-scripts”
- **mongodb-org-mongos** – Serviço de roteamento para configurações compartilhadas que processa consultas na camada de aplicação e localiza as informações no cluster de servidores. Na visão de aplicação o mongos se comporta como qualquer outra instancia de MongoDB.
- **mongodb-org-shell** – este pacote contém a shell do Mongo
- **mongodb-org-tools** – este pacote contém as ferramentas, como: [mongoimport](#), [bsondump](#), [mongodump](#), [mongoexport](#), [mongofiles](#), [mongooplog](#), [mongoperf](#), [mongorestore](#), [mongostat](#) e [mongotop](#).

Scripts de inicialização (Init scripts)

O pacote principal mongodb-org inclui vários “init scripts” incluindo o principal **/etc/init.d/mongod** este script é utilizado para parar, iniciar e reiniciar o processo do banco de dados.

Este pacote contém a configuração do MongoDB que é localizada no arquivo **/etc/mongod.conf** em conjunto com os scripts de inicialização.

Atenção

Você não deve instalar os pacotes acima, do repositório oficial simultaneamente com os pacotes do Mongo fornecidos pelo Ubuntu.

Depois de tanta história, se você é como eu já deve estar entediado e esperando por alguma ação. Vamos a instalação deste banco de dados no Ubuntu.

1. Importando chave pública utilizada no sistema gerenciador de pacotes

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv  
7F0CEB10
```

2. Importando chave publica utilizada no sistema gerenciador de pacotes

```
echo "deb http://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.0.list
```

3. Atualizando o banco de dados de pacotes local

```
sudo apt-get update
```

4. Instalando os pacotes do MongoDB

```
sudo apt-get install -y mongodb-org
```

Outros arquivos

O MongoDB armazena os arquivos de dados em **/var/lib/mongodb** e os arquivos de logs em **/var/log/mongodb** por padrão e utiliza a conta de usuário **mongodb**. Você pode alterar estes arquivos em **/etc/mongod.conf**

Comandos básicos

1. Iniciar

```
sudo service mongod start
```

Para verificar se o banco de dados iniciou normalmente verifique o conteúdo do arquivo de log localizado em: **/var/log/mongodb/mongod.log**

2. Parar

```
sudo service mongod stop
```

3. Reiniciar

```
sudo service mongod restart
```

Caso você ainda tenha dúvidas quanto a instalação ou **se quiser saber como desinstalar o MongoDB** no Ubuntu acesse o link

<https://docs.mongodb.org/master/tutorial/install-mongodb-on-ubuntu/>

Introdução ao MongoDB

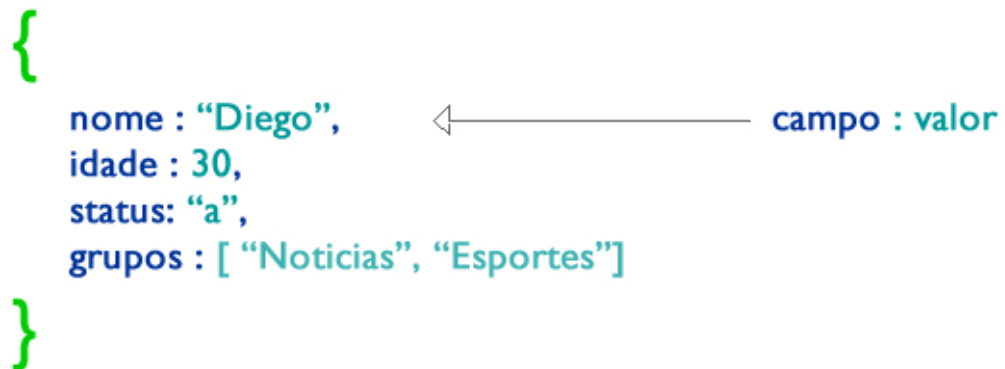
O MongoDB é um banco de dados de código aberto orientado a documentos que fornece uma grande performance, alta disponibilidade e automática e fácil escalabilidade com seu suporte nativo a replicação e balanceamento de carga.

Documentos

Cada registro no MongoDB é um documento e sua estrutura de dados é composta por um campo seu valor. Os documentos são similares a objetos JSON (Objetos javascript). Os valores de cada campo podem ser outros documentos, arrays e arrays de documentos.

Veja o exemplo abaixo:

```
{
  nome : "Diego",
  idade : 30,
  status: "a",
  grupos : [ "Noticias", "Esportes"]
}
```



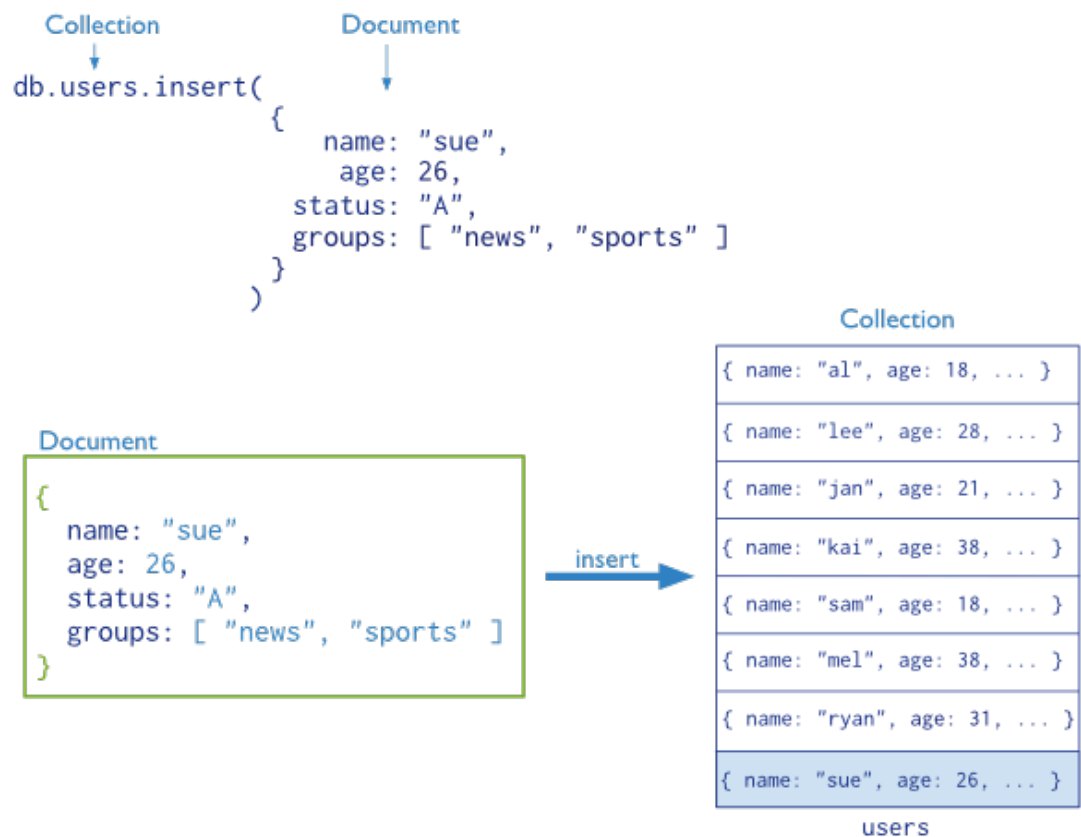
Vantagens de utilizar documentos:

- Documentos (ex. Objetos) correspondem a tipos nativos das principais linguagens de programação.
- Documentos incorporados e arrays reduzem a necessidade de grandes joins.
- Esquemas dinâmicos suportam polimorfismo.

Coleções

O MongoDB armazena todos os seus documentos em coleções. Uma coleção é um grupo de documentos similares que possuem um conjunto de índices em comum. Em uma analogia as coleções são como tabelas nos bancos de dados relacionais.

Imagem que demonstra como o banco funciona:



Comandos básicos

Acessando o mongo via terminal

Para acessar o mongo via Shell, entre no terminal e digite:

```
mongo
```

Help

O comando help é o mais útil desta ferramenta ;)

```
help()
```

Listar bancos de dados disponíveis

Para visualizar os bancos de dados existentes, use o comando:

```
show dbs
```

Criar ou acessar um banco de dados

Para criar um banco de dados novo ou selecionar um banco de dados utilize:

```
use NOME_DO_BANCO_DE_DADOS
```

Detalhe importante: por padrão o MongoDB está desabilitado para acesso externo.

Coleções / Tabelas

Visão geral

Aqui vamos ver os principais comandos relacionados a coleções

db.createCollection();

Esta função serve para criar tabelas.

```
db.createCollection(<NAME>, {  
  
    capped: <boolean>,  
    autoIndexId: <boolean>,  
    size: <number>,  
    max: <number>,  
    storageEngine: <document>  
  
})
```

Função dividida em dois parâmetros:

1 – name : tipo string – nome da criação que está sendo criada;

2 – opções : tipo documento (JSON)

contém os seguintes campos:

- **capped** – tipo booleano – serve para criar uma coleção que automaticamente irá sobrescrever as entradas mais antigas quando atingir o limite máximo.
- **autoIndexId** – tipo booleano – Por padrão vem TRUE e isso faz o auto-incremento no id gerado automaticamente, caso não queira você deverá informar false
- **size** – tipo numérico – parâmetro opcional que serve para especificar a quantidade de bytes de uma coleção do tipo capped
- **max** – tipo numérico – parâmetro opcional para informar o limite máximo de documentos alocados na coleção.

Para ver mais detalhes desta função acesse :

<https://docs.mongodb.org/v3.0/reference/method/db.createCollection/>

db.getCollectionNames();

Esta função serve para listar as tabelas existentes deste banco de dados

db.getCollectionInfos();

Esta função retorna as informações de coleções

db.collection.drop();

esta função apaga uma tabela do banco de dados, exemplo: **db.TABELA.drop();**

db.collection.count();

Retorna o número de registros de uma coleção, exemplo: **db.TABELA.count();**

db.collection.dataSize();

Retorna o tamanho em bytes de uma coleção, exemplo: **db.TABELA.dataSize();**

db.collection.find();

Lista os documentos de uma coleção, exemplo: **db.TABELA.find();**

db.collection.insert();

Comando utilizado para inserir documentos na coleção.

Exemplo:

db.TABELA.insert({ OBJETO JSON });

Atenção se a tabela não existir ainda e você utilizar este comando, a tabela será criada automaticamente com os parâmetros padrões.

db.collection.update();

Comando utilizado para atualizar os documentos da coleção.

Exemplo:

```
db.collection.update(  
    <query>,  
    <update>,  
    {  
        upsert: <boolean>,  
        multi: <boolean>,  
        writeConcern: <document>  
    }  
);
```

Parâmetros:

1 – query – tipo documento – é o critério para atualização. São os mesmos seletores utilizados no método find()

2 – update - tipo documento – Modificações a serem aplicadas

3 – upsert – tipo booleano – Se for setado como **true** irá criar um novo documento quando a consulta não retornar resultados. Se **false** a não irá inserir um novo documento quando não retornar resultados.

db.collection.remove();

Comando utilizado para remover um documento de uma coleção

Exemplo : **db.TABELA.remove (<query>, <justone>);**

MongoDB na prática

Agora que você já viu como criar banco de dados, criar uma coleção, e os principais métodos de coleções, iremos na prática aplicar este conhecimento.

Criando o banco de dados

Comando:

```
use livraria;
```

Feito isto, será criado nosso banco de dados.

Iremos criar as tabelas de exemplo:

Comando

```
db.createCollection("autor");  
db.createCollection("livro");
```

Vamos agora listar as coleções:

Comando

```
db.getCollectionNames();
```

Isto irá listar as duas tabelas criadas e uma do sistema, veja exemplo:

```
["autor", "livro", "system.indexes"];
```

Inserindo documentos na coleção:

```
db.autor.insert (
    {
        "_id" : "diego.silveira.mota",
        "nome" : "Diego Silveira Mota",
        "idade" : "30",
        "sexo" : "m"
    }
);

db.autor.insert (
    {
        "_id" : "marcelino.saraiva",
        "nome" : "Marcelino Saraiva Mota",
        "idade" : "54",
        "sexo" : "m"
    }
);

db.livro.insert ({
    "nome" : "E-book de MongoDB",
    "genero" : "Técnico",
    "autor" : "diego.silveira.mota"
});
```

Agora que você criou duas tabelas vamos a uma simples consulta

```
db.livro.find({ "autor" : "diego.silveira.mota"});
```

Este comando irá retornar a lista:

```
{ "_id" : ObjectId("564575d32565e881dc537f11"), "nome" : "E-book de MongoDB",  
  "genero" : "Técnico", "autor" : "diego.silveira.mota" }
```

Para visualizar melhor o retorno, adicione o método pretty(); ao final da consulta.

```
db.livro.find({ "autor" : "diego.silveira.mota"}).pretty();
```

Isto irá retornar a lista da seguinte forma:

```
{  
  "_id" : ObjectId("564575d32565e881dc537f11"),  
  "nome" : "E-book de MongoDB",  
  "genero" : "Técnico",  
  "autor" : "diego.silveira.mota"  
}
```

Realizando consultas a documentos

db.TABELA.find(<query>, <projection>);

Análise do método de pesquisa.

Parâmetros

- query – tipo documento – Opcional, especifica-se nele os critérios de consulta utilizando operadores de consulta. Se não informar parâmetro ou se este parâmetro for vazio irá retornar todos os registros da coleção.
- projection – documento – Opcional. Especifica os campos de retorno usando os operadores de projeção. Para retornar todos os campos combinados na pesquisa omita este parâmetro.

Operadores

Para entender melhor as consultas temos que voltar para o básico e conhecer os principais operadores da QUERY.

Operadores de comparação

- **\$eq** – Combina valores que são iguais ao valor especificado.
- **\$gt** – Encontra valores maior do que o valor especificado.
- **\$gte** – Encontra valores maior ou igual ao valor especificado.
- **\$lt** – Encontra valores menores do que o valor especificado.
- **\$lte** – Encontra valores menores ou iguais ao valor especificado.
- **\$ne** – Encontra valores não iguais ao valor especificado.
- **\$in** – Encontra valores especificados em um determinado array.
- **\$nin** – Encontra valores diferentes do informado em um determinado array.

Operadores lógicos

- **\$or** – Encontra uma consulta ou outra
- **\$and** – Une outras consultas lógicas e retorna todos os documentos encontrados em ambas cláusulas.
- **\$not** – Inverte o efeito da consulta e retorna os documentos que não satisfazem o critério informado.
- **\$nor** – Não correspondem uma pesquisa nem outra

Operadores de elementos

- **\$exists** – Verifica se existe um campo ou não
- **\$type** – Verifica se determinado campo é de um tipo

Operadores de avaliação

- **\$mod** – Executa uma operação de módulo sobre o valor de um campo e seleciona documentos com um resultado especificado.
- **\$regex** – Seleciona documentos em que os valores correspondem a uma expressão regular especificada.
- **\$text** – Executa uma pesquisa por texto.
- **\$where** – Combina documentos que satisfaçam uma expressão JavaScript.

Operadores de array

- **\$all** – corresponder a todos os valores em um array
- **\$elemMatch** – Procura em um campo array específico do documento por elementos dentro desse array.
- **\$size** – este operador encontra qualquer array que contenha o número de campos especificado.

Para ver os operadores geo-espaciais e os demais acesse:

<https://docs.mongodb.org/manual/reference/operator/query/>

Quadro comparativo

SQL Select Statements	MongoDB find() Statements
select * from users	db.users.find();
select id, user_id, status from users	db.users.find({ }, { user_id : 1, status : 1 });
select user_id, status from users	db.users.find({ }, { user_id : 1, status: 1, _id: 0 });
select * from users where status = "A"	db.users.find({ status : "A" });
select * from users where status = "A" or age = 50	db.users.find({ \$or : [{status : "A" }, {age : "50" }] });
select * from users where age > 25	db.users.find({ age : { \$gt : 25 } })
select * from users where age < 25	db.users.find({ age : { \$lt : 25 } });
select * from users where age > 25 and age <= 50	db.users.find({ age : { \$gt : 25 , \$lte : 50 } });
select * from users where user_id like "%bc%"	db.users.find({ user_id : /bc/ });

Operador \$elemMatch

Para explicar melhor o funcionamento da pesquisa informando o parâmetro \$elemenMatch, exemplifico de forma prática a seguir:

Inserindo dados em uma nova tabela:

```
db.grade.insert({
  "name" : "amy",
  "age" : 21,
  "score" : [
    {
      "course" : "computer",
      "credit" : 2,
      "teacher" : "nola"
    },
    {
      "course" : "chinese",
      "credit" : 3,
      "teacher" : "mary"
    }
  ]
});
```

```
db.grade.insert({
  "name" : "jim",
  "age" : 20,
  "score" : [
    {
      "course" : "computer",
      "credit" : 2,
      "teacher" : "mary"
    },
    {
      "course" : "english",
      "credit" : 1,
      "teacher" : "join"
    }
  ]
});
```

Exemplo

Caso você queira pesquisar dentro do array "score" apenas o campo "course" que contenha a palavra "english", seria da seguinte forma:

```
db.grade.find({ "score" : { "$elemMatch" : { "course": "english" } } }).pretty();
```

Tudo sobre os métodos da pesquisa você deverá encontrar no link:

<https://docs.mongodb.org/manual/reference/method/db.collection.find/>

Outros métodos que podem ser usados associados ao find()

- db.TABELA.find().count()
- db.TABELA.find().distinct()
- db.TABELA.find().sort()
- db.TABELA.find().max()
- db.TABELA.find().min()
- db.TABELA.find().limit()
- db.TABELA.find().skip()

Mais sobre cursores em :

<https://docs.mongodb.org/manual/reference/method/#cursor>

Alterando documentos

db.collection.update(<query>, <update>, <options>)

Este é o principal método para alterar documentos no MongoDB

```
db.collection.update(  
  <query>,  
  <update>,  
  {  
    upsert: <boolean>,  
    multi: <boolean>,  
    writeConcern: <document>  
  }  
)
```

Então vamos aos exemplos práticos

SQL Update Statements	MongoDB update() Statements
update users SET status = "C" where age > 25	db.users.update({ age : { \$gt : 25 } }, { \$set : { status : "C" } }, { multi: true });
update users SET age = age + 3 where status = "A"	db.users.update({ status : "A" }, { \$inc : { age : 3 } }, { multi : true });

Links úteis:

<https://docs.mongodb.org/manual/reference/operator/update/>

<https://docs.mongodb.org/manual/reference/method/db.collection.update/#db.collection.update>

Removendo documentos

db.collection.delete(<query>, <justone>);

Este é o método principal para remover documentos de uma coleção.

Para grandes operações de eliminação, pode ser mais eficiente copiar os documentos que você deseja manter para uma nova coleção e, em seguida, usar o método drop() sobre a coleção original.

Exemplo:

```
db.inventory.remove( { type : "food" }, 1 )
```

SQL Delete Statements	MongoDB remove() Statements
delete from users WHERE status = "D"	db.users.remove({status : "D"});
delete from users	db.users.remove();

Considerações finais

Este é apenas um guia rápido para pequenas transações no MongoDB. Acredite ele faz bem mais que isto e tudo que você precisar saber, você encontrará na documentação, que é super organizada.

Novamente agradeço pela leitura do livro e aguardo sugestões.

Links interessantes

<https://docs.mongodb.org/v3.0/reference/>

<https://docs.mongodb.org/v3.0/reference/method/db.collection.find/>

<http://nomadev.com.br/>

<http://christiano.me/>

<http://agiletesters.com.br/topic/53/n%C3%A3o-seja-um-mongo-com-mongodb>

<http://www.programpt.com/blog/MQDMwADMwQT2.html>

<http://imasters.com.br/artigo/17308/mongodb/como-utilizar-selects-com-mongodb/>

<http://leandropiga.nothus.com.br/2013/09/capitulo-4-modelagem-de-dados.html>