**MongoDB- Guia Rápido**

**MongoDB Overview**

MongoDB é um banco de dados multi-plataforma orientado ao documento que fornece alta performance, alta disponibilidade e fácil escalabilidade. MongoDB funciona no conceito de coleções de documentos.

**Banco de Dados**

Banco de dados é um recipiente físico para coleções. Cada banco de dados tem o seu próprio conjunto de arquivos no sistema de arquivos. Um único servidor MongoDB normalmente tem vários bancos de dados.

**Coleções (Collection)**

Collection é um grupo de documentos MongoDB. É o equivalente de uma tabela de RDBMS. Uma coleção existe dentro de um único banco de dados. Coleções não impõem um esquema. Documentos dentro de uma coleção pode ter diferentes campos. Normalmente, todos os documentos em uma coleção são propositalmente  semelhantes ou afins.

**Documento**

Um documento é um conjunto de pares de valores-chave. Documentos têm esquema dinâmico. Esquema dinâmico significa que os documentos na mesma coleção não precisam ter o mesmo conjunto de campos ou estrutura e campos comuns em documentos de uma coleção podem conter diferentes tipos de dados.

Exemplo:

O exemplo abaixo mostra a estrutura do documento de um blog onde uma simples vírgula separa o par valor da chave.

{

\_id:ObjectId(7df78ad8902c)

title:'MongoDB - Guia Rapido',

description:'MongoDB - Guia Rapido',by:'MongoDBWise',

url:'http://www.mongodbwise.worpress.com',

tags:['mongodb','database','NoSQL'],

likes:100,

comments:[{

user:'user1',

message:'My first comment',

dateCreated:newDate(2014,5,21,2,15),

like:0},{

user:'user2',

message:'My second comments',

dateCreated:newDate(2014,5,21,7,45),

like:5}]}

**Instalar MongoDB no Windows**

Para instalar o MongoDB no Windows, primeiro faça o download da última versão do MongoDB no site <http://www.mongodb.org/downloads>

Agora extraia o arquivo baixado para uma pasta de sua escolha. Certifique-se que o nome da pasta extraída é [versão] mongodb-win32-i386 ou x86\_64 mongodb-win32-[versão]. Aqui [versão] é a versão do MongoDB baixada.

Agora abra uma janela com Prompt de comando e execute o seguinte comando:

C:\>move mongodb-win64-\* mongodb

1 dir(s) moved.

C:\>

No caso de ter extraído o mondodb em local diferente, em seguida, ir para esse caminho usando o comando cd FOLDER/DIR e executar o processo acima indicado.

O MongoDB requer uma pasta de dados para armazenar seus arquivos. O local padrão para o diretório de dados MongoDB é c: \data\db. Então, você precisa criar essa pasta usando o Prompt de Comando. Execute a seguinte sequência de comandos:

C:\>md data

C:\md data\db

Se você tem instalar o MongoDB em local diferente então você precisa especificar qualquer caminho alternativo para \data\db, definindo o caminho ***dbpath*** no mongod.exe. Para a mesma questão execute os seguintes comandos

No prompt de comando vá para o diretório ***bin*** localizado na pasta de instalação do MongoDB. Suponha que a pasta de instalação seja  **D:\set up\mongodb**

C:\Users\XYZ>d:

D:\>cd "set up"

D:\set up>cd mongodb

D:\set up\mongodb>cd bin

D:\set up\mongodb\bin>mongod.exe --dbpath "d:\set up\mongodb\data"

Isto irá mostrar à espera de conexões mensagem na saída do console indicando que o processo mongod.exe está sendo executado com sucesso.

Agora, para executar o mongodb você precisa abrir outro prompt de comando e executar o seguinte comando

D:\set up\mongodb\bin>mongo.exe

MongoDB shell version: 2.6.1

connecting to: test

>db.test.save( { a: 1 } )

>db.test.find()

{ "\_id" : ObjectId(5879b0f65a56a454), "a" : 1 }

>

Isso vai mostrar que o MongoDB foi instalado e executado com sucesso. Da próxima vez quando você executa mongodb você precisa emitir apenas comandos:

D:\set up\mongodb\bin>mongod.exe --dbpath "d:\set up\mongodb\data"

D:\set up\mongodb\bin>mongo.exe

**Create Database**

Para criar um banco de dados basta utilizar o comando “use database\_name”, o comando irá criar um novo banco de dados se caso ele não existir, de outra forma ele irá retornar o banco de dados existente.

**Sintaxe:**

A sintaxe básica para a utilização do USE\_DATABASE é a seguinte:

**use DATABASE\_NAME**

**Exemplo**

Se você quer criar um banco de dados com o nome <mydb>, então a sintaxe seria a seguinte:

>use mydb

switched to db mydb

Para verificar o banco de dados selecionado atualmente usar o comando **db**

>db

mydb

Se você quiser verificar a sua lista de bancos de dados, em seguida, use o comando **show dbs**.

>show dbs

local 0.78125GB

test 0.23012GB

Você deve ter notado que seu banco de dados criado (mydb) não está presente na lista. Para exibir dados é necessário inserir pelo menos um documento para ele.

>db.mydb.insert({"name":"mongodbwise"})

>show dbs

local0.78125GB

mydb 0.23012GB

test 0.23012GB

***Nota : No banco de dados “test” é o padrão do MongoDB, portanto se você não criar qualquer banco de dados, as coleções serão armazenadas no banco de dados de teste.***

**Drop Database**

Para dropar um banco de dados no MongoDB utilizamos o comando **db.dropDatabase()**.

A sintaxe básica do comando **dropDatabase ()**  é a seguinte:

db.dropDatabase()

Isso excluirá o banco de dados selecionado. Se você não tiver selecionado qualquer banco de dados, então ele vai apagar dados do banco de dados default ‘test’

**Exemplo**

Se você deseja excluir o novo banco de dados **<mydb>**, então o comando **dropDatabase()** seria o seguinte:

>use mydb

switched to db mydb

>db.dropDatabase()>{"dropped":"mydb","ok":1}>

**Create Collection**

Para esta tarefa, utilizaremos o comando db.createCollection(name, options), onde “name” é o nome da Collection e “options” é utilizada para especificar as configurações da Colletion.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| Name | String | Nome da coleção a ser criada |
| Options | Document | (Opcional) Especifica opções sobre tamanho de memoria e indices |

Opções parâmetro é opcional, por isso você precisa especificar apenas o nome da coleção.

**Sintaxe**

A sintaxe básica do método **createCollection ()** é como se segue:

>use test

switched to db test

>db.createCollection("mycollection"){"ok":1}>

Você pode verificar a coleção criada usando o comando show collections

>show collections

mycollection

system.indexes

Também podemos utilizar o comando show tables que irá retornar o mesmo resultado

>show tables

mycollection

system.indexes

**Lista de Opções**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| capped | Boolean | (Opcional) Se “true”, habilita uma “capped collection”. Capped collection é uma coleção com tamanho fixo que sobrescreve automaticamente as atualizações antigas sempre que atinge o tamanho máximo. **Se você especificar “true”, será necessário especificar também o parametro “size”.** |
| autoIndexID | Boolean | (Opcional) Se true, o indice é criado automaticamente no campo \_id. O Valor Default é “false”. |
| size | number | (Opcional) Especifica o tamanho máximo em bytes para uma capped collection. Se **capped is true, então será necessário especificar este campo também.** |
| max | number | (Opcional) Especifica o numero máximo de documentos permitidos em uma the capped collection. |

Ao inserir o documento, o MongoDB primeiro verifica o tamanho do campo na Capped Collection, em seguida, ele verifica o campo max.

**Sintaxe**

>db.createCollection("mycol",{ capped :true, autoIndexID :true, size :6142800, max :10000}){"ok":1}>

No mongoDB você não precisa criar a coleção, pois o MongoDB cria a coleção automaticamente quando você insere algum documento.

>db.mongodbwise.insert({"name":"mongodbwise"})

>show collections

mycol

mycollection

system.indexes

mongodbwise

>

**Drop Collection**

No MongoDB utilizamos o comando **db.collection.drop()**  para descartar (dropar) uma coleção (Collection)

**Sintaxe**

A sintaxe básica do comando **drop()** é a seguinte:

db.COLLECTION\_NAME.drop()

**Exemplo**

Abaixo veremos um exemplo onde vamos dropar a coleção com o nome  **myCollection**

>use mydb

switched to db mydb

>db.mycollection.drop()true>

**Inserir Documentos (Insert Document)**

Para inserir dados em uma coleção MongoDB, você precisa usar o comando **insert()** do MongoDB.

**Sintaxe**

A sintaxe básica do comando**insert()** é a seguinte:

>db.COLLECTION\_NAME.insert(document)

**Exemplo**

>db.mycol.insert({

\_id:ObjectId(7df78ad8902c),

title:'MongoDB Guia Rapido',

description:'MongoDB - Guia Rapido',by:'MongodbWise',

url:'http://www.mongodbwise.wordpress.com',

tags:['mongodb','database','NoSQL'],

likes:100})

Aqui é a nossa coleção mycol. Se a coleção não existir no banco de dados, então o MongoDB irá criar esta coleção e  em seguida, inserir documento para ele.

No documento inserido, se não especificar o parâmetro **\_id**, então MongoDB atribui um ObjectId exclusivo para esse documento.

O **\_id** é de 12 bytes é um número hexadecimal exclusivo para cada documento em uma coleção. 12 bytes serão divididos da seguinte forma:

\_id:ObjectId(4 bytes timestamp,3 bytes machine id,2 bytes process id,3 bytes incrementer)

Para inserir vários documentos em única query, você pode passar uma matriz de documentos no comando **insert()**.

>db.post.insert([{

title:'MongoDB - Guia Rapido',

description:'MongoDB - Guia Rapido',by:'MongoDBWise',

url:'http://www.mongodbwise.wordpress.com',

tags:['mongodb','database','NoSQL'],

likes:100},{

title:'NoSQL Database',

description:'NoSQL database doesn't have tables',

by: 'tutorials point',

url: 'http://www.mongodbwise.wordpress.com',

tags:['mongodb','database','NoSQL'],

likes:20,

comments:[{

user:'user1',

message:'My first comment',

dateCreated:newDate(2013,11,10,2,35),

like:0}]}])

**Consulta de Documentos**

Para consultar os dados de Collection MongoDB, você precisa usar o comando **find()** do MongoDB.

**Sintaxe**

A sintaxe básica do comando **find()** é como se segue:

>db.COLLECTION\_NAME.find()

O comando**find ()** irá exibir todos os documentos de uma forma não estruturada. Para exibir os resultados de uma forma estruturada, você pode usar o comando **pretty()**.

Sintaxe

>db.mycol.find().pretty()

Exemplo

>db.mycol.find().pretty()

{"\_id":ObjectId(7df78ad8902c),

"title":"MongoDB - Guia Rapido",

"description":"MongoDB - Guia Rapido","by":"MongoDBWise",

"url":"http://www.mongodbwise.wordpress.com",

"tags":["mongodb","database","NoSQL"],

"likes":"100"}>

Além do comando find(), existe também o comando findOne() que irá retornar tudo em um único documento

**Cláusula Where equivalentes entre RDBMS e MongoDB**

Para consultar o documento com base em alguma condição, você pode usar seguintes operações

|  |  |  |  |
| --- | --- | --- | --- |
| **Operação** | **Syntaxe** | **Examplo** | **RDBMS Equivalente** |
| Equality | {<key>:<value>} | db.mycol.find({“by”:”tutorials point”}).pretty() | where by = ‘tutorials point’ |
| Less Than | {<key>:{$lt:<value>}} | db.mycol.find({“likes”:{$lt:50}}).pretty() | where likes < 50 |
| Less Than Equals | {<key>:{$lte:<value>}} | db.mycol.find({“likes”:{$lte:50}}).pretty() | where likes <= 50 |
| Greater Than | {<key>:{$gt:<value>}} | db.mycol.find({“likes”:{$gt:50}}).pretty() | where likes > 50 |
| Greater Than Equals | {<key>:{$gte:<value>}} | db.mycol.find({“likes”:{$gte:50}}).pretty() | where likes >= 50 |
| Not Equals | {<key>:{$ne:<value>}} | db.mycol.find({“likes”:{$ne:50}}).pretty() | where likes != 50 |

**“AND” no MongoDB**

No comando find () se você passar várias chaves, separando-os por ‘,’ então MongoDB irá tratar como condição AND. A sintaxe básica é mostrada abaixo:

>db.mycol.find({key1:value1, key2:value2}).pretty()

**Exemplo**

>db.mycol.find()

{"\_id":ObjectId(7df78ad8902c),

"title":"MongoDB - Guia Rapido",

"description":"MongoDB - Guia Rapido","by":"MongoDBWise",

"url":"http://www.mongodbwise.wordpress.com",

"tags":["mongodb","database","NoSQL"],

"likes":"100"}>

No exemplo acima, onde  a clausula ‘ where by=’MongoDBWise’ AND title=’MongoDB – Guia Rapido’ ‘.Você pode passar qualquer número de pares de chaves de valor na cláusula find.

**“OR” no MongoDB**

**Sintaxe**

Para consultar os documentos com base na condição OR, você precisa usar $or como palavra-chave. A sintaxe básica de OR é mostrado abaixo:

>db.mycol.find({

$or:[{key1: value1},{key2:value2}

]}

).pretty()

Abaixo dado exemplo mostrará todos os documentos escritos por ‘mongoDBWise”ou cujo título é’ MongoDB – Guia Rapido ‘

>db.mycol.find({$or:[{"by":"tutorials point"},{"title":"MongoDB Overview"}]}).pretty()

{"\_id":ObjectId(7df78ad8902c),

"title":"MongoDB - Guia Rapido",

"description":"MongoDB - Guia Rapido",

"by":"MongoDBWise",

"url":"http://www.mongodbwise.wordpress.com",

"tags":["mongodb","database","NoSQL"],

"likes":"100"}>

**Utilizando “AND” e “OR” juntos**

O exemplo a seguir mostra documentos que possuem “likes” acima de 100 e cujo o titulo é “MongoDB – Guia Rapido” escrito por “MongoDBWise”. Seria o equivalente a clausula “where likes>10 AND (by = “MongoDBWise” OR title=”MongoDB – Guia Rapido”.

>db.mycol.find("likesgt:10}, $or: [{"by": "MongoDBWise"}, {"title": "MongoDB - Guia Rapido"}] }).pretty()

{

"\_id": ObjectId(7df78ad8902c),

"title": "MongoDB - Guia Rapido",

"description": "MongoDB - Guia Rapido",

"by": "MongoDBWise",

"url": "http://www.mongodbwise.wordpress.com",

"tags": ["mongodb", "database", "NoSQL"],

"likes": "100"

}

>

**Atualização de Documentos (Update Document)**

As atualizações de documentos em uma coleção no MongoDB são feitas através dos comandos update() e save(). O comando update() atualiza os valores enquanto o comando save() substitui os documentos existentes com os novos documentos passados através do comando save().

**MongoDB Update()**

**Sinataxe**

A sintaxe básica do comando update() é como segue abaixo:

>db.COLLECTION\_NAME.update(SELECTIOIN\_CRITERIA, UPDATED\_DATA)

Vamos considerar que a Collection Mycol possui os seguintes dados:

{"\_id":ObjectId(5983548781331adf45ec5),"title":"MongoDB - Guia Rapido"}

{"\_id":ObjectId(5983548781331adf45ec6),"title":"NoSQL Overview"}

{"\_id":ObjectId(5983548781331adf45ec7),"title":"MongoDB - Overview"}

Seguindo o exemplo, vamos definir o novo título ‘New MongoDB – Guia Rapido’ dos documentos cujo título é “MongoDB – Overview ‘

>db.mycol.update({'title':'MongoDB{$set:{'title':'New MongoDB - Guia Rapido'}})

>db.mycol.find()

{ "\_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB - Guia Rapido"}

{ "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}

{ "\_id" : ObjectId(5983548781331adf45ec7), "title":"New MongoDB - Guia Rapido"}

>

Por default o mongodb irá atualizar somente um único documento, para atualizações múltiplas, é necessário definir o parâmetro, “multi” para true.

db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB - Guia Rapido'}},{multi:true})

**MongoDB Save() Method**

O método save() substitui o documento existente com o novo documento passado pelo comando save()

A Sintaxe básica para o comando save() é como passada a seguir:

db.COLLECTION\_NAME.save({\_id:ObjectId(),NEW\_DATA})

Seguindo o exemplo vamos substituir o documento com a\_id ‘5983548781331adf45ec7’

>db.mycol.save(

{

"\_id" : ObjectId(5983548781331adf45ec7), "title":"New MongoDB - Guia Rapido", "by":"MongoDBWise"

}

)

>db.mycol.find()

{ "\_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB - Guia Rapido", "by":"MongoDBWise"}

{ "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}

{ "\_id" : ObjectId(5983548781331adf45ec7), "title":"New MongoDB - Guia Rapido"}

>

**Excluir Documentos (Delete Document)**

O método remove() de MongoDB  é usado para remover documentos da coleção. O método remove() aceita dois parâmetros. Um deles é critério de exclusão e segundo é o Flag Justone

1. Critérios de exclusão: (Opcional) Critérios de exclusão de acordo com documentos serão removidos.
2. Justone: (opcional) Se definido como “true” ou 1, em seguida, retira apenas um documento.

Abaixo segue a sitaxe básica para o comando remove():

>db.COLLECTION\_NAME.remove(DELLETION\_CRITTERIA)

Considerando que a coleção “mycol”  tem os seguintes dados:

{ "\_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB - Guia Rapido"}

{ "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}

{ "\_id" : ObjectId(5983548781331adf45ec7), "title":"New MongoDB - Guia Rapido"}

Seguindo o exemplo vamos remover todos os documentos cujo título é ‘MongoDB – Guia Rapido”

>db.mycol.remove({'title':'MongoDB - Guia Rapido'})

>db.mycol.find()

{"\_id":ObjectId(5983548781331adf45ec6),"title":"NoSQL Overview"}

{"\_id":ObjectId(5983548781331adf45ec7),"title":"New MongoDB - Guia Rapido"}>

**Removendo apenas um documento**

Se houver vários registros e você quer apagar apenas o primeiro registro, em seguida, defina o parâmetro Justone no comando remove()

>db.COLLECTION\_NAME.remove(DELETION\_CRITERIA,1)

Se você não especificar os critérios de exclusão, então o mongodb vai excluir todos documentos  da coleção. Isto seria o equivalente do comando de SQL truncate.

>db.mycol.remove()

>db.mycol.find()

>

**MongoDB Projection**

O significado de uma projeção mongodb é que você está selecionando apenas os dados necessários ao invés de selecionar todo os dados de um documento. Se um documento tem 5 campos e você precisa mostrar apenas 3, e em seguida selecionar apenas 3 campos a partir deles.

O comando **find()** do MongoDB, como explicado anteriormente, aceita um segundo parâmetro opcional que é a lista de campos que você deseja recuperar. Em MongoDB quando você executar o método **find()**,  ele exibe todos os campos de um documento. Para limitar isso, você precisa configurar como vai querer a lista de campos, definindo com valor 1 ou 0. 1 é usado para mostrar a arquivado enquanto 0 é usado para ocultar o campo.

A sintaxe básica do método **find()** com projeção é a seguinte:

>db.COLLECTION\_NAME.find({},{KEY:1})

Considerando que a coleção **“mycol”**  tem os seguintes dados:

{ "\_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB - Guia Rapido"}

{ "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}

{ "\_id" : ObjectId(5983548781331adf45ec7), "title":"New MongoDB - Guia Rapido"}

Seguindo o exemplo, irá exibir o título do documento ao consultar o documento.

>db.mycol.find({},{"title":1,\_id:0})

{"title":"MongoDB - Guia Rapido"}

{"title":"NoSQL Overview"}

{"title":"New MongoDB - Guia Rapido"}>

Por favor, note campo \_id sempre é exibido ao executar o comando **find()**, se você não quer que este campo seja listado, então você precisará configurá-lo como 0.

**Limite de Documentos**

**MongoDB Limit() Method**

Para limitar os registros no MongoDB, você precisa usar o método **limit()**. O método **limit()** aceita um argumento de tipo de número, que é o número de documentos a serem exibidos.

A sintaxe básica do método **limit()** é como segue:

>db.COLLECTION\_NAME.find().limit(NUMBER)

Considerando que a coleção **mycol** possui os seguintes dados

{"\_id":ObjectId(5983548781331adf45ec5),"title":"MongoDB - Guia Rapido"}

{"\_id":ObjectId(5983548781331adf45ec6),"title":"NoSQL Overview"}

{"\_id":ObjectId(5983548781331adf45ec7),"title":"New MongoDB - Guia Rapido"}

O exemplo a seguir irá mostrar apenas 2 documentos ao consultarmos o documento.

>db.mycol.find({},{"title":1,\_id:0}).limit(2)

{"title":"MongoDB - Guia Rapido"}{"title":"NoSQL Overview"}

>

Se você não especificar o argumento número no método limit(), em seguida, ele irá mostrar todos os documentos da coleção.

**MongoDB Skip() Method**

Além de método **limit()** não é nada mais um método **Skip()**, que também aceita argumento de tipo de número e usado para pular número de documentos.

A sintaxe básica do método skip() é como segue abaixo:

>db.COLLECTION\_NAME.find().limit(NUMBER).skip(NUMBER)

No exemplo a seguir deverá ser exibido apenas o segundo documento.

>db.mycol.find({},{"title":1,\_id:0}).limit(1).skip(1){"title":"NoSQL Overview"}>

**Nota: O valor default no método skip() é 0**

**Classificando documentos**

Para classificar documentos no MongoDB, você precisa usar o método **sort()**. O método **sort()** aceita uma lista de documentos que contém campos, juntamente com a sua ordem de classificação. Para especificar a ordem utiliza-se 1 e -1 como ​​classificação. 1 é usado para a ordem crescente enquanto -1 é usado para a ordem descendente.

A sintaxe básica do comando sort() é como segue abaixo:

>db.COLLECTION\_NAME.find().sort({KEY:1})

Considerando que a coleção mycol possui os seguintes dados

{"\_id":ObjectId(5983548781331adf45ec5),"title":"MongoDB - Guia Rapido"}

{"\_id":ObjectId(5983548781331adf45ec6),"title":"NoSQL Overview"}

{"\_id":ObjectId(5983548781331adf45ec7),"title":"New MongoDB - Guia Rapido"}

O exemplo a seguir irá exibir os documentos ordenados por título em ordem decrescente.

>db.mycol.find({},{"title":1,\_id:0}).sort({"title":-1})

{"title":"NoSQL Overview"}

{"title":"New MongoDB - Guia Rapido"}

{"title":"MongoDB - Guia Rapido"}

>

Observe se você não especificar a preferência de classificação, então o método **sort()** irá exibir os documentos em ordem crescente.

**Indexação no MongoDB**

Índices servem para apoiar a resolução eficiente de consultas. Sem índices, o MongoDB deverá digitalizar todos os documentos de uma coleção para depois selecionar os documentos que correspondem a instrução de consulta. Essa verificação é altamente ineficiente e exigem muito do mongod para processar um grande volume de dados.

Índices são estruturas de dados especiais, que armazenam uma pequena parte do conjunto de dados em um formato mais prático. O índice armazena o valor de um campo específico ou um conjunto de campos, ordenados pelo valor do campo, tal como especificado no índice.

Para criar um índice que você precisa para usar o método **ensureIndex()** do MongoDB.

**Sintaxe:**

>db.COLLECTION\_NAME.ensureIndex({KEY:1})

Aqui chave é o nome do registro em qual você deseja criar o índice e 1 é para ordem crescente. Para criar o índice em ordem decrescente você precisa usar -1.

**Exemplo**

>db.mycol.ensureIndex({"title":1})>

No método **ensureIndex()** você pode passar vários campos, para criar um índice em vários campos.

>db.mycol.ensureIndex({"title":1,"description":-1})>

O método **ensureIndex()** também aceita uma lista de opções (que são opcionais), cuja lista é a seguinte:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| background | Boolean | Constrói o índice em segundo plano para que a construção de um índice não bloqueie outras atividades de banco de dados. Especifique true para construir em segundo plano. O valor padrão é **false**. |
| unique | Boolean | Cria um índice exclusivo para que a coleção aceite a inserção de documentos em que a chave de índice ou as chaves que correspondam a um valor existente no índice. Especifique **true** para criar um índice exclusivo. O valor padrão é **false**. |
| name | string | O nome do índice. Se não for especificado, o MongoDB gera um nome de índice concatenando os nomes dos campos indexados e a ordem de classificação. |
| dropDups | Boolean | Cria um índice exclusivo em um campo que pode ter duplicatas. Índices MongoDB  registram apenas a primeira ocorrência de uma chave e removem de todos os documentos da coleção que contêm ocorrências subsequentes desta chave. Especifique **true** para criar um índice único. O valor padrão é **false**. |
| sparse | Boolean | Se for **“true”**, o índice só faz referência a documentos com o campo especificado. Estes índices usam menos espaço, mas se comportam de forma diferente em algumas situações (particularmente os tipos). O valor padrão é **false**. |
| expireAfterSeconds | integer | Especifica um valor, em segundos, como um TTL para controlar quanto tempo MongoDB retém documentos nesta coleção. |
| v | index version | O número da versão do índice. A versão índice padrão depende da versão do mongod em execução durante a criação do índice. |
| weights | document | O peso é um número que varia de 1 a 99.999 e denota a importância do campo em relação aos outros campos indexados em termos de pontuação. |
| default\_language | string | Para um índice de texto, a linguagem que determina a lista de palavras de parada e as regras para a **stemmer** e **tokenizador**. O **isenglish** é valor padrão. |
| language\_override | string | Para um índice de texto, especifique o nome do campo no documento que contém, a linguagem para substituir o idioma padrão. O valor padrão é a linguagem. |

**MongoDB Aggregation**

Operações de agregações (Aggregation) processa todos os registros de dados e retorna o resultado calculado. Grupo de operações de agregação de valores a partir de vários documentos juntos podem executar uma variedade de operações nos dados agrupados para retornar um único resultado. Os comandos SQL “count (\*)” com  “grupo by”  seria equivalentes aos grupos de agregação no MongoDB. Para a agregação em mongodb você deve usar o método agregate().

Sintaxe

>db.COLLECTION\_NAME.aggregate(AGGREGATE\_OPERATION)

Na coleção você tem os seguintes dados:

{

\_id:ObjectId(7df78ad8902c)

title:'MongoDB - Guia Rapido',

description:'MongoDB - Guia Rapido',

by\_user:'mongodbwise',

url:'http://www.mongodbwise.wordpress.com',

tags:['mongodb','database','NoSQL'],

likes:100

},

{

\_id:ObjectId(7df78ad8902d)

title:'NoSQL Overview',

description:'No sql database is very fast',

by\_user:'mongodbwise',

url:'http://www.mongodbwise.wordpress.com',

tags:['mongodb','database','NoSQL'],

likes:10

},

{

\_id:ObjectId(7df78ad8902e)

title:'MongoDB Java',

description:'New MongoDB - Guia Rapido',

by\_user:'mongodbbrazil',

url:'http://www.mongodbwise.wordpress.com',

tags:['java','database','NoSQL'],

likes:750

},

Agora, a partir da coleção acima, se você quiser exibir uma lista que quantos tutoriais são escritos por cada usuário, você vai usar o método aggregate()  como mostrado abaixo:

> db.mycol.aggregate([{$group :{\_id :"$by\_user", num\_tutorial :{$sum :1}}}])

{"result":[

{"\_id":"mongodbwise","num\_tutorial":2

},

{"\_id":"mongodbbrazil","num\_tutorial":1

}

],

"ok":1

}

>

O SQL Query equivalente para o caso de uso acima seria ***“select by\_user, count(\*) from mycol group by by\_user”***

No exemplo acima, temos  documentos agrupados pelo campo **by\_user** e em cada ocorrência de **by\_user,** o valor anterior da soma é incrementado. Abaixo uma lista de expressões de agregação disponíveis.

|  |  |  |
| --- | --- | --- |
| **Expression** | **Description** | **Example** |
| $sum | Soma-se o valor definido a partir de todos os documentos da coleção. | db.mycol.aggregate([{$group : {\_id : “$by\_user”, num\_tutorial : {$sum : “$likes”}}}]) |
| $avg | Calcula a média de todos os valores dados a partir de todos os documentos da coleção. | db.mycol.aggregate([{$group : {\_id : “$by\_user”, num\_tutorial : {$avg : “$likes”}}}]) |
| $min | Obtém o mínimo dos valores correspondentes de todos os documentos da coleção. | db.mycol.aggregate([{$group : {\_id : “$by\_user”, num\_tutorial : {$min : “$likes”}}}]) |
| $max | Obtém o máximo dos valores correspondentes de todos os documentos da coleção. | db.mycol.aggregate([{$group : {\_id : “$by\_user”, num\_tutorial : {$max : “$likes”}}}]) |
| $push | Insere o valor de uma matriz no documento resultante. | db.mycol.aggregate([{$group : {\_id : “$by\_user”, url : {$push: “$url”}}}]) |
| $addToSet | Insere o valor de uma matriz no documento resultante, mas não cria duplicatas. | db.mycol.aggregate([{$group : {\_id : “$by\_user”, url : {$addToSet : “$url”}}}]) |
| $first | Obtém o primeiro documento dos documentos de origem de acordo com o agrupamento. Normalmente, isso só faz sentido em conjunto com alguns anteriormente aplicada “$sort”-stage. | db.mycol.aggregate([{$group : {\_id : “$by\_user”, first\_url : {$first : “$url”}}}]) |
| $last | Obtém o último documento dos documentos de origem de acordo com o agrupamento. Normalmente, isso só faz sentido em conjunto com alguns anteriormente aplicada “$sort”-stage. | db.mycol.aggregate([{$group : {\_id : “$by\_user”, last\_url : {$last : “$url”}}}]) |

**MongoDB Replication**

Replicação é o processo de sincronização de dados entre vários servidores. A replicação oferece redundância e aumenta a disponibilidade de dados com várias cópias dos dados em diferentes servidores de banco de dados. A  replicação de um banco de dados além de proteger da perda de um único servidor, também permite que você recupere devido a uma falha de hardware ou interrupções de serviço. Com as cópias adicionais dos dados, você pode dedicar uma para recuperação de desastres, relatórios, ou backup.

**Por que replicação?**

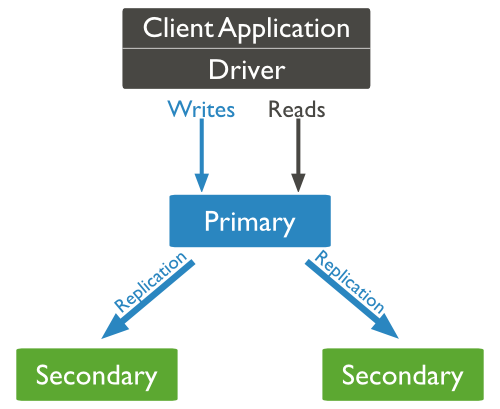
* Para manter seus dados seguros
* Alta disponibilidade de dados (24×7)
* Recuperação de Desastres
* Falta de tempo de inatividade para manutenção (como backups, reconstrução de índices, compactação)
* Escala  de Leitura(cópias extras para ler)
* Conjunto de réplicas transparentes para o aplicativo

**Como a replicação trabalha no MongoDB**

MongoDB obtêm sua  replicação através da utilização do conjunto de réplicas. Um conjunto de réplicas é um grupo de instances mongod que hospedam o mesmo conjunto de dados. Em uma réplica, um dos nós é o nó principal, que recebe todas as operações de escrita. Todos as outras instances, secundários, aplicam operações do primário de modo que eles tenham o mesmo conjunto de dados. Um conjunto de réplicas (Replica Set) pode ter apenas um nó principal.

1. Conjunto de réplicas é um grupo de dois ou mais nós (geralmente são necessárias 3 nós no mínimo).
2. Em um conjunto de réplicas, um nó é o nó principal e nós restantes são secundários.
3. Todos os dados replicam do primário para o nó secundário.
4. No momento do failover automático ou manutenção, a eleição estabelece qual será o primário e um novo nó principal é eleito.
5. Após a recuperação do nó que falhou, ele novamente irá se juntar ao conjunto de réplicas e trabalhará como um nó secundário. .

Um diagrama típico de replicação mongodb é mostrado abaixo no qual a aplicação do cliente sempre interage com o nó principal, e em seguida, o nó principal repete os dados para os nós secundários.

[](https://mongodbwise.files.wordpress.com/2014/05/replication.png)

**Características do conjunto de réplicas**

* Um Cluster de N nodess
* Qualquer nó pode ser o primário
* Todas as operações de gravação são efetuadas no nó primário
* Failover automático
* Recuperação Automática
* Consenso de eleição primária

**Configurar um conjunto de réplicas**

Neste tutorial vamos converter instância mongod autônoma em um conjunto de réplicas. Para converter em um conjunto de réplicas, siga os passos abaixo indicados:

* Shutdown do servidor MongoDB
* Agora inicie o servidor MongoDB, especificando a opção **– replSet**. Sintaxe básica do **– replSet** é dada abaixo:

mongod --port "PORT"--dbpath "YOUR\_DB\_DATA\_PATH"--replSet "REPLICA\_SET\_INSTANCE\_NAME"

**Exemplo**

mongod --port 27017--dbpath "D:\set up\mongodb\data"--replSet rs0

Ele vai iniciar uma instância mongod com o nome RS0, na porta 27017

* Agora inicie o prompt de comando e se conecte a essa instância mongod.
* No console do Mongo execute o comando **rs.initiate()**  para iniciar o novo conjunto de replicas (replica set)
* Para verificar algum problema no Conjunto de Replicas (Replica Set), execute o comando **rs.conf()**.
* Para verificar o status do Conjunto de Replicas (Replica Set), execute o comando **rs.status()**.

**Geração de Backups no MongoDB**

**Dump MongoDB Data**

Para criar um backup de banco de dados no MongoDB você deve usar o comando **mongodump**. Este comando irá copiar todos os dados de seu servidor no diretório de dump. Há muitas opções disponíveis através do qual você pode limitar a quantidade de dados ou criar backup do seu servidor remoto.

**Sintaxe**

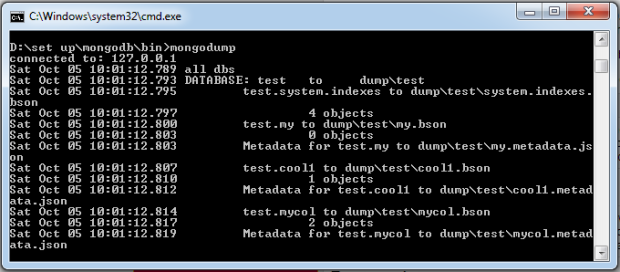
>mongodump

**Exemplo**

Inicie o seu servidor mongod. Supondo-se que o servidor mongod está sendo executado no localhost e na porta 27017. Agora, abra um prompt de comando e vá para o diretório bin da sua instância MongoDB e digite o comando ***mongodump***

>mongodump

O comando vai se conectar ao servidor rodando em 127.0.0.1 e na porta 27017 e em seguida envia todos os dados do servidor para o diretório /bin/dump/. Saída do comando é mostrado abaixo:

[](https://mongodbwise.files.wordpress.com/2014/05/mongodump.png)

Abaixo, uma lista de opções disponíveis que podem ser utilizados com o comando mongodump.

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Description** | **Example** |
| mongodump –host HOST\_NAME –port PORT\_NUMBER | Este comando irá fazer obackup todos os bancos de dados especificados na Instance mongod. | mongodump –host tutorialspoint.com –port 27017 |
| mongodump –dbpath DB\_PATH –out BACKUP\_DIRECTORY |  | mongodump –dbpath /data/db/ –out /data/backup/ |
| mongodump –collection COLLECTION –db DB\_NAME | Este comando irá fazer o backup somente da collection do banco de dados, especificada. | mongodump –collection mycol –db test |

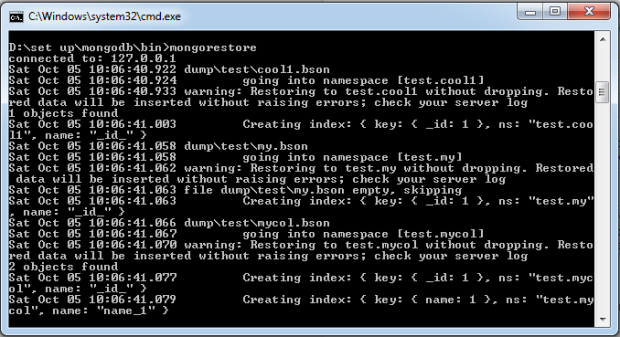
**Restaurando Dados (Restore Data)**

O comando para restaurar dados de backup do MongoDB é o ***mongorestore***. Este comando irá restaurar todos os dados do diretório de backup.

**Sintaxe**

>mongorestore

Saída do comando é mostrado abaixo:

[](https://mongodbwise.files.wordpress.com/2014/05/mongorestore.png)