**Databases, documents and collections no MongoDB**

[](https://mongodbwise.files.wordpress.com/2014/05/mongodb.gif)

**Neste tutorial, vamos mostrar os conceitos e fatos importantes sobre**

**Databases, Documents e collections no MongoDB.**

**DATABASES**

Em um servidor MongoDB é possível executar vários bancos de dados. O Banco de dados padrão do MongoDB é chamado ‘db’, que é armazenado dentro da pasta de dados.

O MongoDB pode criar bancos de dados em tempo real, portanto não é necessário criar um banco de dados antes de começar a trabalhar com ele.

Execute no prompt do OS, o comando “mongo” para ter acesso ao console:

[root@Mongo01 ~]# mongo

MongoDB shell version: 2.6.1

connecting to: test

>

Automaticamente o MongoDB conecta ao Database “test”

Através do comando ***“show dbs”***no console onde você pode obter uma lista de todos os bancos de dados existentes.

> show dbs

admin (empty)

cadastro 0.078GB

local 0.078GB

test (empty)

>

Ao executar o comando “db”, você obtem o banco de dados no qual está:

> db

test

>

Para se conectar a um banco de dados específico, execute o comando ***“use”***

Neste exemplo vamos utilizar o banco de dados “cadastro” :

> use cadastro

switched to db cadastro

Nomes de banco de dados podem ser quase qualquer caractere no intervalo ASCII. Mas eles não podem conter uma cadeia vazia, um ponto (ou seja, “.”) Ou “” uma vez que isto é reservado ao “sistema” e não pode ser utilizado como um nome de banco de dados,porém podem contar “$”

**DOCUMENTOS**

O Dcumento é a unidade de armazenamento de dados em um banco de dados MongoDB.

O Mongodb utiliza documentos no formato JSON (JavaScript Object Notation, que é um formato leve, completamente explorável e muito utilizado para trocar dados entre diversas aplicações) para armazenar dados.

Um exemplo simples de um documento JSON é como se segue:

{Site: “mongodbwise.wordpress.com”}

Muitas vezes, o termo **“object”** é utilizado para referir um documento.

Documentos são análogos aos registros de um RDBMS. As operações Inserir, atualizar e excluir podem ser executadas em uma coleção (Collection). A tabela a seguir vai ajudar você a entender o conceito mais facilmente:

|  |  |
| --- | --- |
| **RDBMS** | **MongoDB** |
| Table | Collection |
| Column | Key |
| Value | Value |
| Records / Rows | Document / Object |

A tabela a seguir mostra os vários tipos de dados que podem ser utilizados no MongoDB.

|  |  |
| --- | --- |
| **Data Types** | **Description** |
| string | Pode ser uma cadeia vazia ou uma combinação de caracteres. |
| integer | Digitos. |
| boolean | Os valores lógicos Verdadeiro ou Falso. |
| double | Um tipo de número de ponto flutuante. |
| null | Not zero, not empty. |
| array | Lista de valores. |
| object | Uma entidade que pode ser utilizado na programação. Pode ser um valor, variável, função ou estrutura de dados. |
| timestamp | Um valor de 64 bits referindo-se a um momento único e em uma única instância “mongod”. O primeiro de 32 bits deste valor refere-se ao segundo desde o UTC 1 de janeiro de 1970. E últimos 32 bits referem-se ao incremento ordinal para operações dentro de um determinado momento. |
| Internationalized Strings | UTF-8 para strings. |
| Object IDs | Cada objeto ou documento MongoDB deve ter um ID do objeto que é único. Este é uma BSON (Binary JavaScript Object Notation, que é a interpretação binária de JSON) objeto id, um valor binário de 12 bytes que tem uma oportunidade muito rara de ser duplicado. Este ID consiste em uma timestamp de 4 bytes (segundos depois),  3 bytes para o ID de máquina, um ID de processo de 2 bytes e um contador de 3 bytes. |

**Coleções (Collections)**

Uma coleção pode armazenar vários documentos. A coleção funciona como uma tabela de um RDBMS.

Uma coleção pode armazenar documentos que não são os mesmos em estrutura. Isso é possível porque o MongoDB é um banco de dados sem esquema. Em um banco de dados relacional como MySQL, Oracle, etc.. um esquema define a organização/estrutura de dados no banco de dados. O MongoDB não exige tal conjunto de fórmula que define a estrutura de dados. Assim, é perfeitamente possível armazenar documentos de diferentes estruturas em uma coleção. Na prática, não é necessário definir uma coluna e é o tipo de dados ao contrário de RDBMS, trabalhando com MongoDB.

No código a seguir, mostra-se que dois documentos MongoDB, pertence à mesma coleção, o armazenamento de dados de diferentes estruturas.

{"tutorial" : "NoSQL"} {"topic\_id" : 7}

A Coleção é criada quando o primeiro documento é inserido.

**Nomes de coleção válidos**

Todos os nomes das Coleções devem começar com letras ou um sublinhado e pode conter números assim como também não é permitido a utilização do carácter “$”, pois é reservado para o sistema.

Um nome de coleção não deve exceder 128 caracteres. É indicado mantê-lo dentro de 80/90 caracteres.

Utilizando um “.” (ponto) notação, as coleções podem ser organizadas em grupos nomeados. Por exemplo, tutorials.php e tutorials.javascript ambos pertencem a tutoriais. Esse mecanismo é chamado de namespace collection que servem apenas para o usuário pois para os bancos de dados não tem importência alguma.

Como  usá-lo por meio de linha de comando:

>db.tutorials.php.findOne()

**Capped Collections**

Imagine que você deseja registrar as atividades que acontecem com a aplicação. você deseja armazenar dados na mesma ordem em que está sendo inserido. Para isso o MongoDB oferece as Capped Collections para fazê-lo.

Capped Collections são conjuntos circulares, de tamanho fixo que mantêm documentos bem ordenadas, mesmo sem o uso de um índice. Isto significa que as Capped Collections podem receber escritas em velocidades muito alta e leituras seqüenciais.

Estas coleções são particularmente úteis para manter os arquivos de log, mas não estão limitadas a esse fim. Use Capped Collections quando apropriado.

Atualmente, o tamanho máximo para uma Capped Collection 1e9 (ou seja, 1X109) para máquinas de 32 bits. Para máquinas de 64 bits, não há nenhum limite teóricamente falando. Na prática, ele pode ser estendido de acordo com sua autorização de recursos do sistema.

**Utilize várias Coleções (Collections) em vez de uma apenas**

Esta forma dispensa a exigência de se criar índices desde que você não está armazenando dados de repetição em cada objeto. Se aplicada em uma situação adequada, pode melhorar o desempenho.

As informações sobre um banco de dados são armazenadas em certas coleções. Elas estão agrupados como :

dbname.system. \*

A tabela a seguir mostra as coleções e o que elas armazenam

|  |  |
| --- | --- |
| **Collections with namespace** | **Description** |
| dbname.system.namespaces | list of all namespaces |
| dbname.system.indexes | list of all indexes |
| dbname.system.profile | stores database profiling information |
| dbname.system.users | list of users who may access the database |
| dbname.local.sources | stores replica slave configuration data and state |
| dbname.local.sources | stores replica slave configuration data and state |

Existem mais duas opções para armazenar metadados:

arquivos database.ns armazenam metadados e namespace/index adicional, se houver.

A informação sobre a estrutura de um objecto armazenado é armazenado dentro do próprio objeto.

**Como criar Capped Collections**

db.createCollection(nameOfCollection, {capped: Boolean, autoIndexId: Boolean, size: Number, max : Number})

Aqui o que os parâmetros especificam:

* nameOfCollection: Nome da capped collection
* “capped” option : Definir como “true” no no caso de criação de uma capped collection
* “size” option: especificar o limite de tamanho para a coleção em bytes (obrigatório no caso de capped collection)
* “max” option: especificar o limite para o número máximo de documentos permitidos na coleção (opção de tamanho é dada preferência sobre opção max)
* “autoIndexId” option: Definir como “false” no caso de capped collections para desabilitar a criação automática do index.

Como exemplo vamos criar uma Capped Collection chamada “demo”:

[root@Mongo01 ~]# mongo

MongoDB shell version: 2.6.1

connecting to: test

> db.createCollection("demo", { capped : true, size : 500, max : 3 } )

{ "ok" : 1 }

Aqui nós colocamos limite de tamanho de 500 bytes e número máximo de documentos a 3. Agora vamos inserir alguns documentos nesta coleção.

> db.demo.insert({'name':'PortoriteResult({ "nInserted" : 1 })

> db.demo.insert({'name':'Florianopolisult({ "nInserted" : 1 })

> db.demo.insert({'name':'Rio})

WriteResult({ "nInserted" : 1 })

>

> db.demo.find().pretty()

{ "\_id" : ObjectId("53742a383752473d26901b3b"), "name" : "Porto Alegre" }

{ "\_id" : ObjectId("53742a463752473d26901b3c"), "name" : "Florianopolis" }

{ "\_id" : ObjectId("53742a4f3752473d26901b3d"), "name" : "Rio de Janeiro" }

>

Agora, observe que o limite máximo de tamanho foi alcançado e se tentarmos adicionar mais documentos, os documentos antigos serão sobre escritos pelos novos.

> db.demo.insert({'name':'Brasiliault({ "nInserted" : 1 })

> db.demo.find().pretty()

{ "\_id" : ObjectId("53742a463752473d26901b3c"), "name" : "Florianopolis" }

{ "\_id" : ObjectId("53742a4f3752473d26901b3d"), "name" : "Rio de Janeiro" }

{ "\_id" : ObjectId("53742b3a3752473d26901b3e"), "name" : "Brasilia" }

**Nota:** O Comando “db.collection.is Capped()”  é usado para verificar se uma coleção é uma Capped collection  ou não.

Veja:

> db.demo.isCapped()

true

>

**Vantagens de Capped Collections:**

1. As Consultas não precisam de um índice para devolver os documentos em ordem de inserção, pelo fato de proporcionar um maior rendimento de inserção.
2. Capped collections apenas permitim atualizações que se encaixam o tamanho do documento original, garantindo assim que o documento não mude a sua localização no disco.
3. Útil para arquivos de Log

**Desvantagens das Capped Collections:**

1. Se a operação de atualização aumentar o tamanho original da Capped Collections, a operação de atualização pode falhar.
2. Não podemos excluir documentos de uma Capped Collections. Para remover todos os registros de uma Capped Collections, use o comando {emptycapped: nameOfCollection}
3. Não podemos utilizar o Shard em uma Capped Collection

**emptycapped**

O comando emptycapped remove todos os documentos de uma Capped Collection. Use a seguinte sintaxe:

{ emptycapped: "demo" }

**ATENÇÃO**

*Este comando gera um bloqueio de gravação no banco de dados afetado e irá bloquear outras operações até que ela seja concluída.*

**NOTA**   
*O  emptycapped é um comando interno que não está habilitado por padrão. O emptycapped deve ser ativado usando – setParameter enableTestCommands = 1 na linha de comando mongod. O emptycapped não pode ser ativado durante o tempo de execução.*

Assim, podemos fazer uma escolha se deseja usar Capped Collections de acordo com o nosso caso de uso.