SQL – NOSQL

1. INSTALAÇÃO E CONFIGURAÇÃO

MongoDB é um sistema de gestão de bases de dados orientada a documentos que são guardados num formato semelhante ao JSON. O MongoDB insere-se nas bases de dados do tipo NoSQL e permite guardar os dados num formato muito natural.

Nesta aula iremos utilizar uma aplicação open-source chamada Robo 3T (aka Robomongo). Esta aplicação disponibiliza uma interface para gerir e interagir com bases de dados MongoDB. Para além disso, disponibiliza também uma *shell* onde podem ser executadas instruções MongoDB. Vamos começar por fazer download do Robo 3T, disponível para Linux, Mac e Windows.

De seguida iniciamos o Robomongo e selecionamos ***File/Connect...*** para abrir a janela ***MongoDB Connections***. Selecionamos ***Create*** de forma a criar uma ligação ao servidor remoto MongoDB.

Em ***Connection Settings*** introduzimos o endereço do servidor:

Address - bdad.fe.up.pt

Port - 27017

e os parâmetros de autenticação em ***Authentication***:

Database - twitter

User Name - bdad

Password - 2017

Marcar o campo “Manually specify visible databases”

Databases - twitter

De seguida, clicar em ***Test***. Em caso de sucesso podemos guardar a ligação.

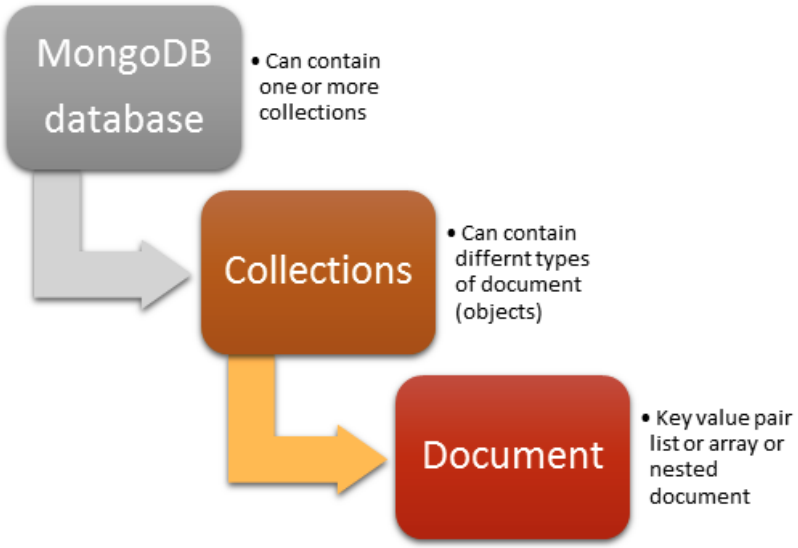
Na janela ***MongoDB Connections***, selecionar a ligação que acabamos de criar e clicar em ***Connect***.

Vamos então selecionar o Explorer dentro do menu ***View***. Deverá aparecer a base de dados twitter no ***Explorer*** no lado esquerdo da janela principal. Clicar em twitter com o botão direito do rato e selecionar ***Open Shell***.

A partir deste momento vamos inserir instruções via linha de comando. Podemos executar as instruções clicando em play na barra superior da janela ou então através dos atalhos ctrl+enter (Windows) e cmd+enter (Mac OS).

2. TUTORIAL

O MongoDB está organizado em 3 camadas conforme visível na imagem seguinte. A primeira camada é a **base de dados.** Esta é composta por **coleções** que por sua vez são compostas por **documentos**.

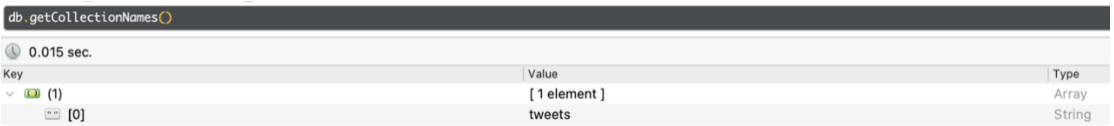


Em MongoDB as queries começam sempre pelo identificador da base de dados em questão, representado pela variável **db.** Esta variável tem vários métodos que podem inclusive ser concatenados entre si.

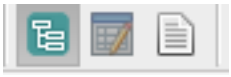
Utilizando esta lógica, vamos começar por obter o nome das coleções existentes na base de dados utilizando o método *getCollectionNames*.

Escreva a seguinte instrução na linha de comandos e execute (play ou ctrl+enter ou cmd+enter):

Por omissão, no Robo 3T, os resultados são exibidos em modo de árvore:



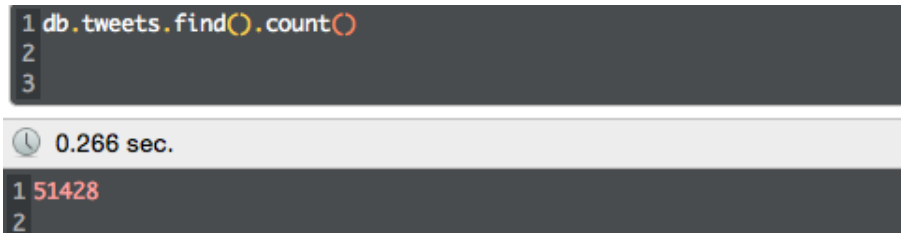
Os modos de exibição de resultados podem ser selecionados no canto superior da linha de comandos (árvore, tabela e texto).



De forma a facilitar a leitura devemos selecionar sempre o modo de leitura de texto (ícone mais à direita):

Estamos interessados na coleção **tweets**, desta forma sempre que criarmos instruções teremos que selecionar essa coleção escrevendo ***db.tweets***. De forma a retornar um cursor para todos os tweets na coleção utilizamos a função *find()*.

Se o objetivo for obter o número total de documentos (tweets) na coleção basta concatenar o método ***count*** ao método ***find*:**



Vamos então ver como é estruturado um tweet:

```
1 db.tweets.findOne()
```

0.029 sec.


```
/* 1 */
{
    "_id" : ObjectId("5567669ae9e57318a849ea0c"),
    "favorited" : false,
    "entities" : {
        "user_mentions" : [],
        "hashtags" : [],
        "urls" : []
    },
    "contributors" : null,
    "truncated" : false,
    "text" : "eu preciso de terminar de fazer a minha tabela, está muito
    "created_at" : "Thu Sep 02 18:11:23 +0000 2010",
    "retweeted" : false,
    "coordinates" : null,
    "source" : "web",
    "in_reply_to_status_id" : null,
    "in_reply_to_screen_name" : null,
    "user" : {
        "follow_request_sent" : null,
        "profile_use_background_image" : true,
        "id" : 5.35078e+07,
        "verified" : false,
        "profile_sidebar_fill_color" : "768575",
        "profile_text_color" : "25b8c2",
        "followers_count" : 102,
        "protected" : false,
        "location" : "",
        "profile_background_color" : "081114",
        "listed_count" : 0,
        "utc_offset" : -10800,
        "statuses_count" : 3504,
        "description" : "só os loucos sabem (:",
        "friends_count" : 73,
        "profile_link_color" : "eb55b6",
        "profile_image_url" : "http://a2.twimg.com/profile_images/103641
        "notifications" : null,
        "show_all_inline_media" : false,
        "geo_enabled" : false,
```

```
        "profile_background_image_url" : "http://a1.twimg.com/profile_ba
        "name" : "Beatriz Helena Cunha",
        "lang" : "en",
        "profile_background_tile" : true,
        "favourites_count" : 1,
        "screen_name" : "Bia_cunha1",
        "url" : "http://http://www.orkut.com.br/Main#Profile?uid=1433295
        "created_at" : "Fri Jul 03 21:44:05 +0000 2009",
        "contributors_enabled" : false,
        "time_zone" : "Brasilia",
        "profile_sidebar_border_color" : "1c9dbd",
        "following" : null
    },
    "place" : null,
    "retweet_count" : null,
    "geo" : null,
    "id" : NumberLong(22819396900),
    "in_reply_to_user_id" : null
}
```


Como podemos verificar, um tweet possui uma estrutura relativamente complexa. A lista de atributos da raiz do documento é composta pelos seguintes atributos:

favorited; retweet_count; in_reply_to_user_id; contributors; truncated; text; created_at; retweeted; coordinates; entities; in_reply_to_status_id; in_reply_to_screen_name; source; place; _id; geo; id; user.

Podemos efetuar pesquisas condicionadas pelos valores dos atributos de cada documento sejam strings, inteiros ou arrays.

Para definir igualdade (=) utilizamos a estrutura chave-valor dos documentos {atributo:valor}.

Vamos testar:

```
db.tweets.findOne({'source':'web'})
```

As projeções permitem-nos retornar apenas os atributos e respetivos valores que nos interessam.

Alterando a interrogação anterior podemos especificar através da projecção para que o resultado apenas retorne o campo ***'user.friends_count'***:

```
db.tweets.findOne({'source':'web'}, {'user.friends_count': 1})
```

0.017 sec.


```
/* 1 */
{
    "_id" : ObjectId("58da3e48fbbb830f71685164"),
    "user" : {
        "friends_count" : 73
    }
}
```

A interrogação anterior permite-nos selecionar atributos. Em alguns cenários, pode ser preferível indicar os atributos que queremos esconder. Experimente a instrução seguinte e compare os resultados:

O documento tweet possui alguns atributos com documentos 'nested', isto é, atributos cujo valor são outros documentos, como é o caso do atributo *entities*, *user* e *location*. Neste caso, sempre que quisermos pesquisar por valores de atributos dos documentos 'nested' utilizamos a notação '.' (ponto), como por exemplo: {'user.name':'John'}.

Vamos experimentar:

```
1 db.tweets.findOne({'user.name':'John'})
```

O MongoDB oferece-nos vários [operadores de comparação]:

`$gt`	Matches values that are greater than a specified value.
`$gte`	Matches values that are greater than or equal to a specified value.
`$lt`	Matches values that are less than a specified value.
`$lte`	Matches values that are less than or equal to a specified value.
`$ne`	Matches all values that are not equal to a specified value.
`$in`	Matches any of the values specified in an array.
`$nin`	Matches none of the values specified in an array.

A sintaxe do MongoDB obriga-nos a utilizar os operadores como valores de um atributo entre chavetas, como por exemplo {'idade': { '$gt': 18 } }

Para encontrar um tweet com hashtags (*entities.hashtags* é um array), temos de pesquisar por um tweet cujo array *hashtags* não seja vazio.

```
db.tweets.findOne( { 'entities.hashtags': { '$ne':[ ] } } )
```

0.154 sec.


```
/* 1 */
{
    "_id" : ObjectId("5567669ae9e57318a849ea0f"),
    "favorited" : false,
    "entities" : {
        "user_mentions" : [],
        "hashtags" : [
            {
                "indices" : [
                    17,
                    33
                ],
                "text" : "BoardwalkEmpire"
            }
        ],
        "urls" : []
    },
```


E se quisermos saber o número total de tweets com *hashtags*?

```
db.tweets.find( { 'entities.hashtags': {'$ne': [ ] } }).count()
```

0.078 sec.

```
6558
```

Agora que conhecemos a estrutura de uma *hashtag* podemos pesquisar por uma *hashtag* com valor específico:

```
db.tweets.findOne( { 'entities.hashtags.text': 'swagg' } )
```

O MongoDB tem funcionalidades de pesquisa de texto em atributos cujo valor é uma string através da função *$regex*. Vamos então encontrar um tweet que fale sobre futebol:

```
db.tweets.findOne({'text':{'$regex':'futebol'}})

```

0.035 sec.


```
/* 1 */
{
    "_id" : ObjectId("5567669ae9e57318a849ec5f"),
    "favorited" : false,
    "entities" : {
        "user_mentions" : [],
        "hashtags" : [],
        "urls" : []
    },
    "contributors" : null,
    "truncated" : false,
    "text" : "Time do Flamengo tá pior que meu futebol
    "created_at" : "Thu Sep 02 18:12:37 +0000 2010",
    "retweeted" : false,
    "coordinates" : null,
```

As funções de ordenação são simples e podemos selecionar qual o atributo pelo qual queremos ordenar os resultados utilizando a função *.sort({'atributo':1})* para ordenar por ordem crescente ou -1 para decrescente. Por questões de memória, devemos limitar o número de resultados aos quais iremos aplicar a ordenação.

Vamos então ver qual o utilizador com maior número de *followers* nos primeiros 100 *tweets*:

```
db.tweets.find().limit(100).sort({'user.followers_count':-1})
```

tweets 0.346 sec.


```
    "user" : {
        "follow_request_sent" : null,
        "profile_use_background_image" : true,
        "id" : 5.76887e+06,
        "verified" : true,
        "profile_sidebar_fill_color" : "efe6ce",
        "profile_text_color" : "996633",
        "followers_count" : 854199,
```


O operador *distinct* permite-nos obter todos os valores distintos de um dado atributo. Vamos então ver os nomes únicos de utilizadores que existem na collection tweets:

```
db.tweets.distinct('user.name')
```

0.996 sec.


```
/* 1 */
{
    "0" : "Beatriz Helena Cunha",
    "1" : "Medscape Pathology",
    "2" : "Travis Siebrass",
    "3" : "Emma Smith",
    "4" : "Lia afriani chaniago",
    "5" : "ピヨ山ピヨ彦",
    "6" : "kaitlinn spears",
    "7" : "ベリーさん",
    "8" : "Nitin",
    "9" : "marisa alfiani",
    "10" : "brittany =]",
    "11" : "Ariadna",
    "12" : "Catherine Mullane",
    "13" : "aundrea",
```


As funções de agregação servem para processar vários tipos de dados de uma coleção e calcular resultados de forma agregada. As operações permitidas assemelham-se às do SQL quando as funções de agregação são usadas com o operador GROUP BY. Por exemplo, o total de tweets por cada timezone ordenando o resultado por ordem decrescente pode ser obtido da seguinte forma:

BDAD bdad.fe.up.pt:27017 twitter

```
db.tweets.aggregate([
    {$group:{ _id: "$user.time_zone", totalByTimezone: {$sum: 1}}},
    {$sort: { totalByTimezone: -1 }}
])
```

Como podemos ver acima, a expressão de agregação utiliza um formato diferente para aceder aos campos dos documentos que servem de input à pipeline de agregação. No exemplo acima, o operador grupo especifica a campo alvo utilizando a chave ***_id*** com um valor no formato field path. Este formato começa pelo símbolo $ (dólar) e é seguido pelo nome do campo para valores simples ou então pelo nome do campo seguido de ponto (.) e dos campos subjacentes - utilizando a dot notation - para aceder a valores compostos.

A função de agregação permite encadear várias operações. Por exemplo, para se obter o total de tweets que contêm 'Br' nas timezones executa-se:

Outros operadores da função de agregação podem ser consultados em: https://docs.mongodb.com/manual/reference/operator/aggregation-pipeline.

3. LINKS ÚTEIS

Outros operadores para interrogações e projeções:
https://docs.mongodb.com/manual/reference/operator/query/

Fases pipelines de agregação:
https://docs.mongodb.com/manual/reference/operator/aggregation-pipeline/

4. EXERCÍCIOS

Escreva instruções que respondam às seguintes questões.

1. Quantos tweets têm como source o twitter versão 'web'?
2. Quantos tweets têm a hashtag 'javascript'?
3. Quantos tweets têm utilizadores com mais de 100 followers?
4. Quantos tweets têm utilizadores com timezone 'Lisbon'?
5. Encontre timezones distintas na coleção.
6. Encontre um tweet com menções a outros utilizadores.
7. Quantos tweets têm menções a outros utilizadores?
8. Quantos tweets têm 3 menções a outros utilizadores?
9. Quantos tweets têm 3 menções a outros utilizadores e 2 hashtags?
10. Quantos tweets falam do Cristiano Ronaldo?
11. Quais são as hashtags mais populares? (utilize a função aggregate())

12. Devolva 100 tweets aleatórios ordenados crescentemente pelo número de amigos dos utilizadores.
13. Qual é a timezone com mais de 100 tweets? (utilize a função aggregate())
14. Existe um utilizador com mais de 7 tweets na coleção. Qual o seu screen_name? (utilize a função aggregate())
15. Qual é o tweet com maior número de hashtags? (utilize a função aggregate())