

# **Aula 14**

## **Introdução ao MongoDB (Parte 1)**

27 de abril de 2018

Profa. Kelly Rosa Braghetto














Slides baseados no material confeccionado por  
Elaine Naomi Watanabe (elainew@ime.usp.br),  
mestre pelo DCC-IME-USP

# MongoDB

- Sistema gerenciador de banco de dados NoSQL
  - Modelo orientado a documentos
  - Esquema flexível (*schemaless*)
  - API de manipulação de dados em JavaScript
- Software livre, multiplataforma
- Criado em 2007
- Nome vem de “*huMONGOus*” – capaz de lidar com *enormes* volumes de dados
- <https://www.mongodb.com/>


















# Popularidade entre os SGBDs

342 systems in ranking, April 2018

Rank			DBMS	Database Model	Score		
Apr 2018	Mar 2018	Apr 2017			Apr 2018	Mar 2018	Apr 2017
1.	1.	1.	Oracle 	Relational DBMS	1289.79	+0.18	-112.21
2.	2.	2.	MySQL 	Relational DBMS	1226.40	-2.46	-138.22
3.	3.	3.	Microsoft SQL Server 	Relational DBMS	1095.51	-9.28	-109.26
4.	4.	4.	PostgreSQL 	Relational DBMS	395.47	-3.88	+33.69
5.	5.	5.	MongoDB 	Document store	341.41	+0.89	+15.98
6.	6.	6.	DB2 	Relational DBMS	188.95	+2.28	+2.29
7.	7.	7.	Microsoft Access	Relational DBMS	132.22	+0.27	+4.04
8.	 9.	 11.	Elasticsearch 	Search engine	131.36	+2.81	+25.69
9.	 8.	9.	Redis 	Key-value store	130.11	-1.12	+15.75
10.	10.	 8.	Cassandra 	Wide column store	119.09	-4.40	-7.10

# Popularidade entre os *Document Stores*

45 systems in ranking, April 2018

Rank			DBMS	Database Model	Score		
Apr 2018	Mar 2018	Apr 2017			Apr 2018	Mar 2018	Apr 2017
1.	1.	1.	MongoDB 	Document store	341.41	+0.89	+15.98
2.	2.	2.	Amazon DynamoDB 	Multi-model 	43.14	+0.69	+11.08
3.	3.	3.	Couchbase 	Document store	32.34	-0.56	+1.53
4.	4.	4.	CouchDB	Document store	19.85	-0.35	-2.52
5.	5.	 9.	Microsoft Azure Cosmos DB 	Multi-model 	17.19	+0.43	+12.91
6.	6.	 5.	MarkLogic	Multi-model 	10.63	-0.35	-0.57
7.	7.	 11.	Firebase Realtime Database	Document store	6.05	-0.54	+3.01
8.	8.	 6.	OrientDB 	Multi-model 	5.64	-0.82	+0.20
9.	9.	 7.	RethinkDB	Document store	4.30	-0.51	-0.83
10.	 11.	 15.	ArangoDB	Multi-model 	3.80	-0.32	+1.19

# Um BD no MongoDB

- Banco de dados = conjunto de coleções
- Coleção (*collection*) = conjunto de documentos
- Documento  $\approx$  objeto em JSON

# [Relembrar é viver]

## Tipos de dados do JSON

- **Null**
- **Boolean** – *true* ou *false*
- **String** (codificação Unicode)
- **Number** (pode ter sinal, e também pode ser decimal ou científico)
- **Object** – conjunto de itens do tipo chave-valor, onde cada chave é uma string única dentro do objeto
- **Array** – lista ordenada de dados, possivelmente de tipos diferentes, delimitados por colchetes e separados por vírgulas

# [Relembrar é viver]

## Exemplo de documento em JSON

```
{  
  "nome": "João", //string  
  "idade": 21, //number  
  "eleitor": true, //boolean  
  "escolaridade": null, //null  
  "hobbies": ["tênis", "xadrez"], //array  
  "endereço": {  
    "cidade": "São Paulo",  
    "estado": "SP"  
  } //object/document  
}
```

# BSON – Binary JSON

- É um formato binário de serialização de documentos JSON
- É usado no MongoDB para o armazenamento e passagem de dados
- Possui extensões que permitem a representação de tipos de dados não existentes em JSON:
  - **Date**
  - **Binary Data (binData)** – vetores de bytes
  - **ObjectId** – identificador de um registro no MongoDB
  - **32-bit integer (int)**
  - **64-bit integer (long)**
  - **Regular Expression (regex)**
  - **MaxKey, MinKey, Timestamp** – tipos usados internamente pelo MongoDB
  - ...

<https://docs.mongodb.com/manual/reference/bson-types/>



# Identificação de objetos

- Todo documento em uma coleção deve ter, obrigatoriamente, um campo chamado **\_id** que funciona como chave primária para o documento
  - O campo **\_id** pode ser de qualquer tipo de dado
  - Mas se na criação de um objeto o campo **\_id** não for definido, o MongoDB atribuirá um valor do tipo *ObjectId* para o campo

# Identificação de objetos

- *ObjectIds* são “valores pequenos, ordenados, (provavelmente!) únicos e fáceis de se gerar”
- Um *ObjectId* ocupa 12 bytes:
  - 4 bytes para representar o timestamp da criação do ID
  - 3 bytes para o identificador da máquina onde o ID foi criado
  - 2 bytes para o identificador do processo que criou o ID
  - 3 bytes para um contador, iniciado com um valor aleatório

# Para garantir escalabilidade, o MongoDB ...

- Não possui o conceito chaves estrangeiras
  - Mas é possível estabelecer ligações (relacionamentos) entre documentos por meio de atributos comuns
- Não possui suporte a operações de junção de coleções
- Não possui suporte a transações
  - Mas garante **atomicidade em nível de documento** nas operações de escrita
  - Já está anunciado no site do MongoDB que uma nova versão com suporte a transações multidocumentos estará disponível ainda em 2018

# Principais componentes do MongoDB

- **mongo** : cliente do MongoDB (Mongo Shell)
- **mongod** : servidor do MongoDB
- **mongod --replSet "nomeReplica"**: servidor de réplica do MongoDB
- **mongod --configsvr**: serviço de configuração do particionamento de dados (ele processa as consultas do cliente e determina a localização dos dados)
- **mongos**: serviço de roteamento de consultas do cliente para o serviço de configuração do cluster
- **Lista de drivers:** <http://docs.mongodb.org/ecosystem/drivers/>

# Replicação / Particionamento no MongoDB

**Métodos utilizados para a obtenção de escalabilidade horizontal:**

- ***Sharding*** (particionamento de dados): o conjunto de dados é dividido em diferentes partes ( shards ) e essas partes são distribuídas em servidores distintos
- ***ReplicaSet*** (conjunto de réplicas): é um grupo de servidores (mongod) que hospedam/armazenam o mesmo conjunto de dados

# Conexão com o MongoDB

Nas aulas sobre o MongoDB, usaremos duas opções de clientes de conexão:

- MongoDB Shell (linha de comando)
  - Já vem no pacote *core* do MongoDB
  - Ele já está instalado nas máquinas do CEC
- Nosqlclient (antigo MongoClient) – interface gráfica

# Cliente de conexão Nosqlclient

- Baixe-o de <https://nosqlclient.com/>
- Descompacte o arquivo e entre no diretório criado
- Execute o programa com o comando: `./nosqlclient`
- Entre no menu “Connect > Create New”
- Dados para a conexão com um MongoDB local:
  - Na aba “Connection”
    - **Connection name:** qualquer nome a sua escolha
    - **Hostname:** localhost
    - **Port:** 27017 → essa é a porta padrão do MongoDB
    - **DB Name:** local



Os exemplos/comandos que  
veremos a seguir introduzem o  
uso do MongoDB Shell



# A API do MongoDB é baseada em JavaScript, assim pode-se...

- Executar scripts js

```
for(i=0; i < 3; i++){  
    print ("hello, " + i);  
}
```

- Declarar variáveis

```
var test = "abc";  
print (test);  
test
```

- Manipulador objetos JSON

```
dict = {"a":1, "b":2};  
dict  
dict.a  
dict["a"]  
w = "a"  
dict[w]
```

# Acessando o *help* em diferentes níveis

```
//geral  
help;  
//nível do banco de dados  
db.help();  
//nível de uma coleção  
db.<nome_coleção>.help();  
//nível do comando find()  
db.<nome_coleção>.find().help();  
//definição de uma função  
db.<nome_coleção>.find;
```

<nome\_coleção> é o nome da coleção dentro do banco de dados selecionado com `use <nome_bd>`

# Alguns comandos úteis

`show dbs` – Lista todos os BDs do servidor mongod

`use <nome_bd>` – Conecta (ou cria, se ainda não existir) o BD especificado

`db` – obtém ponteiro para BD atualmente em uso

`show collections` – Lista todas as coleções de BD atualmente em uso

`quit()` – sai do shell

# Criação e uso de BDs

- Para usar um banco de dados já existente ou criar um novo banco de dados, basta fazer:

```
use <nome_do_BD>;
```

Se ainda não existir um BD com o nome indicado, esse BD será criado assim que uma coleção for criada dentro dele.

# Criação de coleções

- Pode-se incluir um documento em uma coleção; se a coleção não existe ainda, ela será automaticamente criada
  - O comando abaixo cria a coleção `minha_nova_colecao` com o documento `{"a":1}` dentro dela

```
db.minha_nova_colecao.insert({"a":1});
```

- Ou pode-se criar uma coleção vazia

```
db.createCollection("minha_nova_colecao");
```

# Organizando as coleções com *namespaces*

- Pode-se agrupar as coleções em *namespaces*
  - A notação com ponto é usada para designar a qual *namespace* pertence uma coleção
- Exemplos:

```
db.bcc.alunos.insert({"nusp":12345, "nome":"John Lennon"});  
db.bcc.disciplinas.insert({"codigo":"MAC0439"});
```

ou

```
db.createCollection("bcc.alunos");  
db.createCollection("bcc.disciplinas");
```

# Criando uma coleção para testes

- Baixe o script "criacao\_colecao\_bios.js" disponível no Paca
- No mongo shell, execute:

```
load("criacao_colecao_bios.js");
```

Isso criará no BD em uso uma coleção chamada **bios**, com dados da biografia de personalidades da Computação (incluindo suas contribuições para a área e os prêmios que receberam)

# Exemplo de documento da coleção “bios”

```
{  "_id" : 6,
   "name" : {
     "first" : "Guido",
     "last" : "van Rossum" },
   "birth" : ISODate("1956-01-31T05:00:00Z"),
   "contribs" : [ "Python" ],
   "awards" : [
     {
       "award" : "Award for the Advancement of Free Software",
       "year" : 2001,
       "by" : "Free Software Foundation"
     },
     {
       "award" : "NLUUG Award",
       "year" : 2003,
       "by" : "NLUUG"
     }
   ]
}
```



# Busca de documentos

No MongoDB, toda busca se dá no escopo de uma única coleção.

- Para listar todos os documentos da coleção bios:

```
db.bios.find();
```

- Para listar as personalidades da coleção que possuem primeiro nome “James” e sobrenome “Gosling”:

```
db.bios.find({"name":{"first": "James", "last":"Gosling"}});
```

ou

```
db.bios.find({"name.first": "James", "name.last":"Gosling"});
```

# Busca de documentos

- Atenção: Na base ***bios*** usada nesta aula,

```
db.bios.find({"name":{"first": "James"}});
```

**não** gera o mesmo resultado que

```
db.bios.find({"name.first": "James"});
```

# Busca de documentos

- Formato mais geral da busca:

```
db.bios.find(  
  {"birth": {$gte: ISODate("1930-01-01T00:00:00Z")}},  
  {"name":1, "contribs":1 }  
) .limit(5);
```

**Coleção  
consultada**

**Condição de  
seleção dos  
documentos**

**Modificador  
da resposta**

**Atributos da  
resposta (projeção)**

# Busca de documentos

- Busca com condição **OU** – Exemplo

Seleciona os nomes e as contribuições das personalidades que ganharam um prêmio Turing ou que morreram antes dos anos 2000.

```
db.bios.find(  
  { $or:[{"awards.award":"Turing Award"},  
          {"death":{"$lt: ISODate("2000-01-01T00:00:00Z")}}]  
  },  
  {"name":1, "contribs":1 }  
);
```

# Busca de documentos

- Busca com condição **E** e **OU** - Exemplo

Seleciona os nomes e as contribuições das personalidades que nasceram depois dos anos 30 e que (ganharam um prêmio Turing ou que morreram antes dos anos 2000).

```
db.bios.find(  
  {  
    "birth": {$gte: ISODate("1930-01-01T00:00:00Z")},  
    $or:[{"awards.award":"Turing Award"},  
          {"death": {$lt: ISODate("2000-01-01T00:00:00Z")}}  
  ],  
  {"name":1, "contribs":1 }  
);
```

# Busca de documentos

- Busca com condições **E** - Exemplo

Seleciona os nomes e as contribuições das personalidades que nasceram depois dos anos 30 e antes dos anos 80 e que ganharam um prêmio Turing.

```
db.bios.find(  
  {  "birth": {$gte: ISODate("1930-01-01T00:00:00Z"),  
             $lt: ISODate("1980-01-01T00:00:00Z")},  
    "awards.award": "Turing Award"  
  },  
  {"name":1, "contribs":1 }  
);
```

# Busca de documentos

- A consulta abaixo **não** gera o mesmo resultado que a do slide anterior porque quando aparece mais de uma condição para um mesmo atributo, somente a última delas vai ser considerada (no caso, `birth < 1980-01-01` )

```
db.bios.find(  
  {  "birth": {$gte: ISODate("1930-01-01T00:00:00Z")},  
    "birth": {$lt: ISODate("1980-01-01T00:00:00Z")},  
    "awards.award": "Turing Award"  
  },  
  {"name":1, "contribs":1 }  
);
```

# Operadores lógicos para condições de busca

Operadores lógicos são usados para unir cláusulas na condição de busca

- **\$or**
- **\$and**
- **\$not**
- **\$nor** – devolve os documentos para os quais todas as cláusulas unidas pelo NOR são avaliadas como falsas

<https://docs.mongodb.com/manual/reference/operator/query/#query-selectors>



# Operadores para condições de busca

- **\$gt** – *greater than* (>)
- **\$gte** – *greater than or equal* (>=)
- **\$lt** – *less than* (<)
- **\$lte** – *less than or equal* (<=)
- **\$ne** – *not equal* (<>)
- **\$in** – verifica a pertinência num conjunto de valores
- **\$nin** – verifica a não pertinência num conjunto de valores

# Operadores para condições de busca - Exemplo

- Encontre as disciplinas que possuem código MAC426 ou MAC439

```
db.disciplinas.find(  
    {"codigo":{"$in":["MAC0439","MAC0426"]}});
```

# Operadores para condições de busca

- **\$exists** – verifica a existência de atributos
- **\$elemMatch** – verifica se existe pelo menos um elemento no vetor que satisfaz a condição de comparação
- **\$size** – compara tamanho de vetor
- **\$regex** – “casa” com expressão regular
- ...

# Operadores para condições de busca - Exemplo

- Encontre os nomes das personalidades que tiveram 3 contribuições para a Computação:

```
db.bios.find({"contribs":{"$size":3}},  
             {"name":1, "contribs":1, "_id":0});
```

# Operadores para condições de busca - Exemplo

- Encontre os nomes das personalidades que criaram a linguagem de programação *Simula*:

```
db.bios.find({"contribs":  
             {$elemMatch: {$eq: "Simula"}}},  
             {"name": 1, "contribs": 1, "_id": 0});
```

# Operadores para condições de busca - Exemplo

- Encontre as disciplinas que contêm “MAC” no código:

```
db.bcc.disciplinas.find(  
    {"codigo":{"regex":"mac",$options:'i'}});
```

ou

```
db.bcc.disciplinas.find({"codigo":/mac/i});
```

Obs.: a opção 'i' é para ignorar a diferença entre maiúsculas e minúsculas.

# Operadores para condições de busca - Exemplo

- Encontre as disciplinas que contêm “MAC” como **prefixo** no código:

```
db.bcc.disciplinas.find(  
    {"codigo":{"regex":"^mac","options":"i"}});
```

ou

```
db.bcc.disciplinas.find({"codigo":"/^mac/i});
```

# Operadores para condições de busca - Exemplo

- Encontre as disciplinas que contêm “MAC” como **sufixo** no código:

```
db.bcc.disciplinas.find(  
    {"codigo":{"$regex":"mac$","options":"i"}});
```

ou

```
db.bcc.disciplinas.find({"codigo":/mac$/i});
```

- Mais sobre o uso de regex de JavaScript pode ser visto em:
  - [http://www.w3schools.com/jsref/jsref\\_obj\\_regexp.asp](http://www.w3schools.com/jsref/jsref_obj_regexp.asp)



# Modificadores de resultados de consulta - Exemplos

- Mostra a sexta disciplina (uma depois de 5) da lista de disciplinas ordenada decendentemente por código:

```
db.bcc.disciplinas.find().  
    sort({"codigo":-1}).limit(1).skip(5);
```

# Remoção de coleções e BDs

- Para apagar a coleção <nome\_coleção>:

```
db.<nome_colecao>.drop();
```

- Para apagar o banco de dados atualmente em uso:

```
db.dropDatabase();
```

# Referências Bibliográficas

- Documentação do MongoDB
  - <https://docs.mongodb.com/>
- Tutoriais oficiais:
  - <https://docs.mongodb.com/manual/>
  - <https://docs.mongodb.com/getting-started/shell/>
- Tabela de mapeamento de SQL para MongoDB
  - [http://s3.amazonaws.com/info-mongodb-com/sql\\_to\\_mongo.pdf](http://s3.amazonaws.com/info-mongodb-com/sql_to_mongo.pdf)
- Livros:
  - “MongoDB: The Definitive Guide, 2nd Edition – Powerful and Scalable Data Storage“, de Kristina Chodorow, Editora: O'Reilly Media
  - “MongoDB: Construa novas aplicações com novas tecnologias“, de Fernando Boaglio, Editora: Casa do Código