Banco de Dados – MongoDB

Prof. Washington Almeida, MSc, ISF 27002
@profwashington.almeida - Instagram

INTRODUÇÃO

- O **MongoDB** é uma nova ideia de banco de dados trazendo conceitos de Banco de Dados Orientado a Documentos.

- Será explorado os conceitos presentes no **MongoDB** como suas vantagens, desvantagens, instalação e manipulação.

Banco de Dados Orientado a Documentos

- Os **Banco de Dados Orientados a Documentos** utilizam o conceito de dados e documentos autocontidos e auto descritivos.

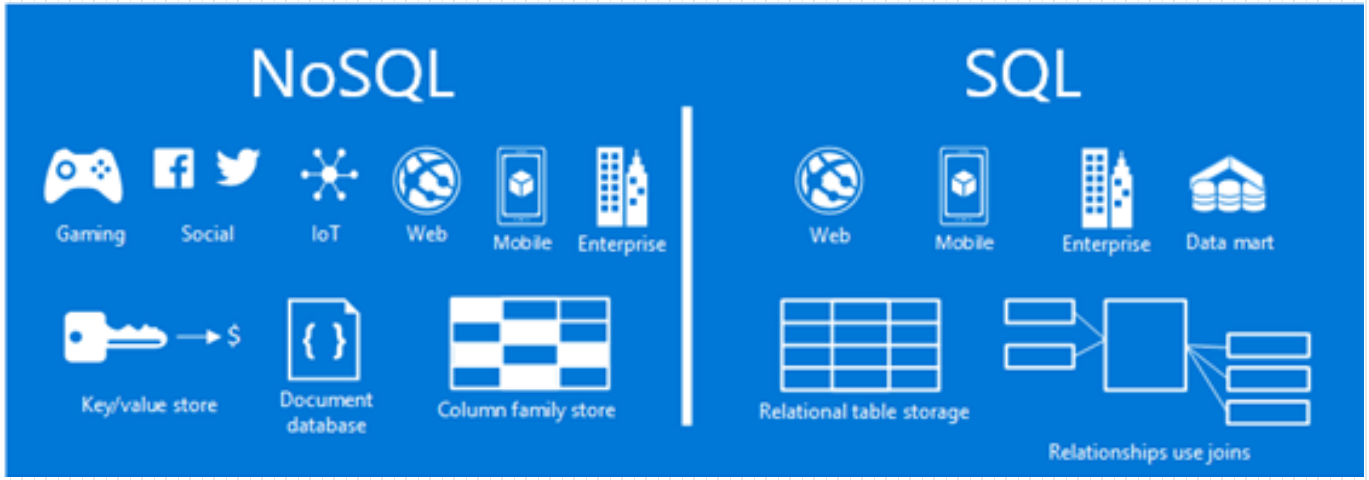
- Isso implica que o documento em si já define como ele deve ser apresentado e qual o significado dos dados armazenados na sua estrutura.

- **Banco de Dados Orientados a Documentos** tem como característica conter todas as informações importantes em um único documento, ser livre de esquemas, possuir **identificadores únicos universais** (UUID).

Banco de Dados Orientado a Documentos

- Também possibilitam a consulta de documentos através de métodos avançados de agrupamento e filtragem (**MapReduce**) e também permitir redundância e inconsistência.

- Esses banco de dados também são chamados de Bancos **NoSQL** (Not Only SQL).



Banco de Dados Orientado a Documentos

- Esse termo **NoSQL** é devido à ausência do SQL, mas esse tipo de Banco de Dados não se resume em apenas não usar SQL.
- Por isso o termo não é o mais correto para esse novo tipo de Banco de Dados.
- O termo já foi aceito por que já se popularizou na comunidade.
- Alguns chegaram a defender o termo **NoREL** (Not Relational), mas não chegou ser aceito como o **NoSQL**.

Banco de Dados Orientado a Documentos

- Esse topo de Banco de Dados não traz consigo as ideias do **modelo relacional** e nem a **linguagem SQL**.
- Uma diferença fundamental entre os dois modelos surge quando precisamos criar relacionamentos nos bancos de dados relacionais
- Diferente dos bancos orientados a documentos que não fornecem relacionamentos entre documentos, o que mantém seu design sem esquemas.

Banco de Dados Orientado a Documentos

- Ao invés de armazenar dados relacionados em uma área de armazenamento separado, os **bancos de dados de documento** integram esses dados ao próprio documento.

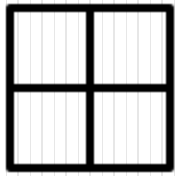
- Contudo, ainda assim é possível armazenar dados relacionados de forma separada, isso pode ser feito usando **uma coleção separada**.

- Porém, é preciso tomar cuidado com isso visto que os ganhos de desempenhos podem ser perdidos.

Características do MongoDB

- O **MongoDB** é um banco de dados orientado a documentos, diferente dos bancos de dados tradicionais que seguem o modelo relacional.
- Dessa forma, já temos uma primeira diferença entre os dois modelos:

- O banco orientado a documentos lida com documentos e não com registros como no modelo relacional, onde tudo é representado usando uma **abordagem bidimensional** (tabelas representadas através de duas dimensões: linhas e colunas).



Características do MongoDB

- Este Banco de Dados tem como característica ser código-fonte aberto licenciado pela **GNU AGPL** (**Affero General Public License**) versão 3.0.

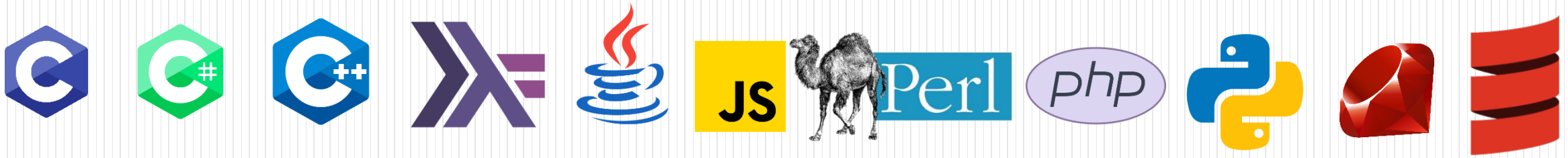
Características do MongoDB

- Possui alta performance, não possuir esquemas, ser escrito em **C++**, multiplataforma e ser formado por um conjunto de **aplicativos JSON**.

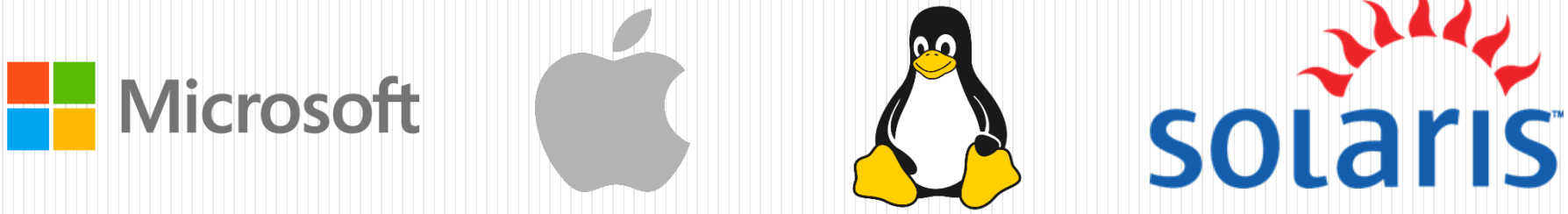
- Apesar do projeto **MongoDB** ter iniciado em 2007 o Banco de Dados somente foi concluído em 2009 lançando assim a primeira versão do **MongoDB**.

Características do MongoDB

- Diversas linguagens e plataforma já possuem drivers para o MongoDB, entre elas destacam-se:
- C, C#, C++, Haskell, Java, JavaScript, Perl, PHP, Python, Ruby e Scala.



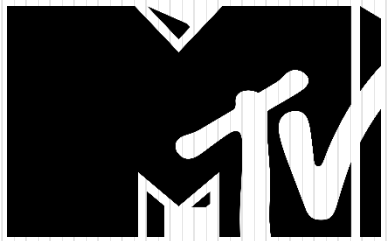
- Além disso, o MongoDB possui binários para diversas plataformas como Windows, Mac OS X, Linux e Solaris.



Características do MongoDB

- Entre as empresas que já utilizam o MongoDB destacam-se:

- Globo.com
- SourceForge
- FourSquare
- MailBox (serviço de e-mail do Dropbox)
- LinkedIn
- SAP
- MTV
- Pearson Education
- e muitos outros.

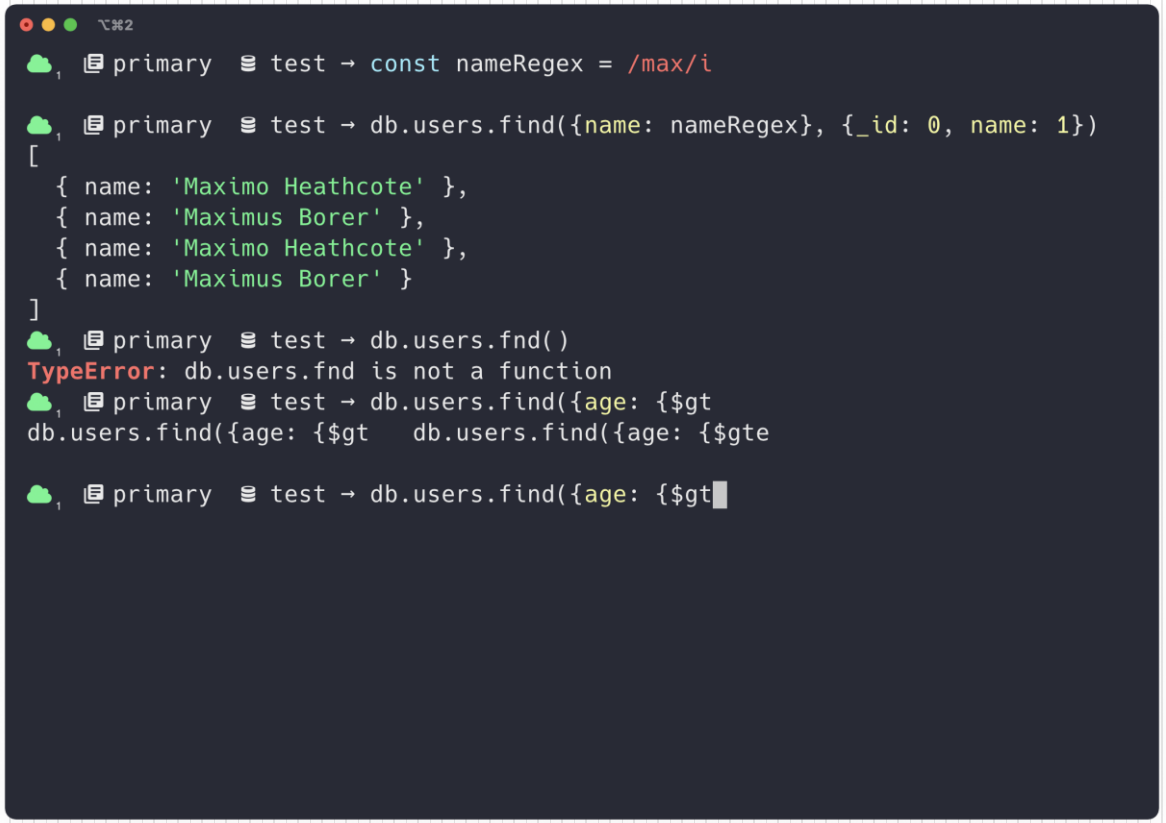


Instalando o MongoDB

- Para instalar o **MongoDB** devemos primeiramente baixa-lo, escolhendo uma versão de sistema operacional.
- Agora já podemos descompactar o **zip** para alguma pasta.
- Após isso devemos ir até a pasta onde descompactamos o **MongoDB** e ir na pasta "**bin**".
- Por fim executamos o "mongod.exe".

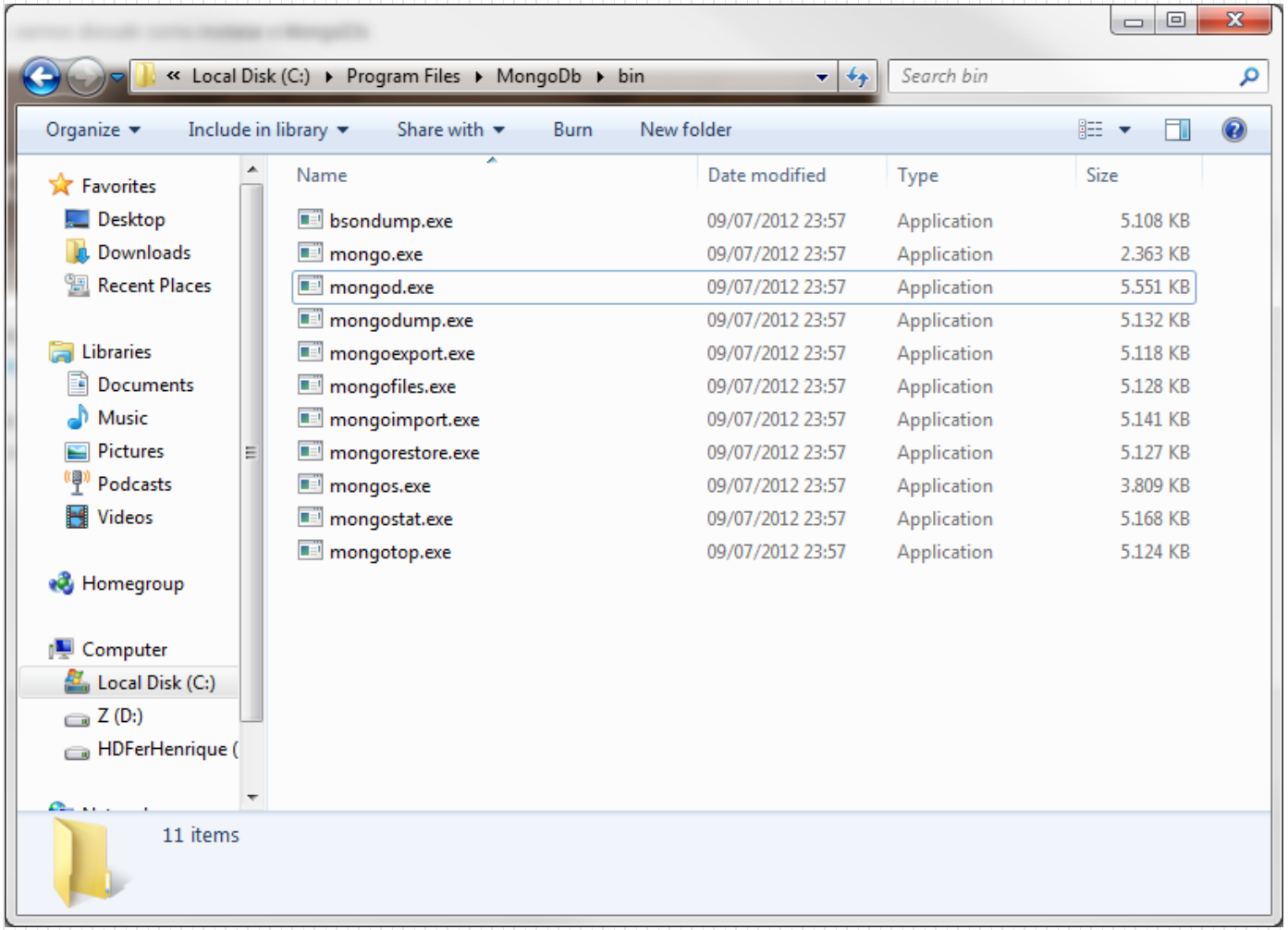
Instalando o MongoDB

- Agora já podemos executar o cliente de **Shell** do MongoDB.



Instalando o MongoDB

- O aplicativo de Shell do MongoDB está incluído junto com a distribuição sendo localizado na pasta “bin”.
- No Windows, está na forma do aplicativo **mongo.exe**.



Instalando o MongoDB

- Através deste Shell podemos criar:

- Bancos de dados 
- Documentos 
- Coleções 

- Se em qualquer momento for preciso obter ajuda, basta dar o comando "**help**" na linha de comando do Shell do Mongo.

Instalando o MongoDB

- Por padrão, o Shell do Mongo se conecta ao banco de dados "**test**".
- Para mudar para outro banco de dados, usamos o comando "**use nomeDoBanco**".
- Se o banco de dados não existir o MongoDB o criará assim que forem incluídos dados nele.
- O Shell deve apresentar a mensagem: **switched to db meuMongoDB**.

Instalando o MongoDB

- Para criar um documento e armazená-lo em uma nova coleção chamada "**colors**" podemos usar o comando abaixo no Shell:

```
1 | > db.colors.save({name:"red",value:"FF0000"});
```

Instalando o MongoDB

- Para verificar se o documento foi armazenado no banco de dados executamos o comando abaixo no Shell:

```
1 | > db.colors.find();
```

- Visto que o Shell do MongoDB usa JavaScript, é possível escrever construções regulares em **JavaScript** ao interagir com os bancos de dados.

Instalando o MongoDB

- A Listagem 1 cria uma coleção de documentos de caracteres, cada um contendo a representação da cadeia de caractere e seu código **ASCII** associado.

```
1 > var chars = "abcdefghijklmnopqrstuvwxyz"
2   > for(var i =0; i<chars.length; i++) {
3   ... var char = chars.substr(i, 1);
4   ... var doc = {char:char, code: char.charCodeAt(0)};
5   ... db.alfabeto.save(doc);
6   ... }
```

Listagem 1. **Coleção de documentos**

Instalando o MongoDB

```
> var chars = "abcdefghijklmnopqrstuvwxyz"
  > for(var i =0; i<chars.length; i++) {
  ... var char = chars.substr(i, 1);
  ... var doc = {char:char, code: char.charCodeAt(0)};
  ... db.alfabeto.save(doc);
  ... }
```

- Este loop criará 26 documentos, um para cada letra minúscula do alfabeto, cada documento contendo o caractere em si e seu código de caractere ASCII.

Utilizando MongoDB

- O **MongoDB** utiliza a sintaxe JSON que retém os dados usando pares de chave/valor. Segue na Listagem 2 um exemplo criação de um esquema:

Listagem 2. **Retendo dados**

```
db.usuarios.insert( {
                        nome: "Higor Medeiros",
                        cidade: "Porto Alegre",
                        estado: "Rio Grande do Sul"
                    }
   )
```

- O equivalente em SQL seria como o código da Listagem 3.

Utilizando MongoDB

Listagem 3. **Retendo dados no SQL**

```
1  CREATE TABLE USUARIOS (
2
3  id
4  MEDIUMINT NOT NULL AUTOINCREMENTS,
5  nome
6  Varchar(30),
7  cidade
8  Varchar(60),
9  estado
10 Varchar(60),
11 PRIMARY
12 KEY (id))
```

- Depois ainda teríamos que criar um **INSERT** para adicionar as informações referentes ao usuário.

Utilizando MongoDB

- No primeiro exemplo acima temos um documento criado no MongoDB que possui três atributos: **nome, cidade e estado**.

- Como podemos observar não temos nenhuma definição de tipos dos dados, tamanho máximo para cada atributo, regras de validação ou restrições.

- Para armazenar outros dados no banco de dados executamos o código abaixo:

```
1 | db.meudb.save(MeusDados)
```

Utilizando MongoDB

- Sendo que o documento **MeusDados** poderia ter o conteúdo da Listagem 4:

Listagem 4. **Conteúdo de MeusDados**

```
MeusDados = {
            nome: "Higor Medeiros",
            cidade: "Porto Alegre",
            estado: "Rio Grande do Sul"
}
```

Utilizando MongoDB

- No exemplo acima estamos armazenando o documento "**MeusDados**" no banco "**meudb**".
- Também podemos criar estruturas diferentes. Segue na Listagem 5 outra forma de armazenar os dados acima:

Listagem 5. **Conteúdo de MeusDados em outra armazenagem**

```
1  MeusDados = {
2              nome: "Higor Medeiros",
3              endereco: {cidade: "Porto Alegre", estado: "Rio Grande do Sul"}
4    }
```

Utilizando MongoDB

- Também podemos fazer uma estrutura ainda mais complexa, como a da Listagem 6:

Listagem 6. **MeusDados sob outra armazenagem**

```
MeusDados = {
            nome: "Higor Medeiros",
            endereco: {cidade: "Porto Alegre", estado: {nome:"Rio Grande do Sul", sigla:"RS", pais:"Brasil"}}
}
```

Utilizando MongoDB

- Podemos notar que não há regras de validação muito rígidas, assim podemos armazenar qualquer tipo de documento no banco de dados.

- Quando precisarmos adicionar um novo atributo no banco de dados podemos incluir este atributo apenas no documento onde ele é necessário.

- Sendo diferente do modelo relacional em que uma coluna aplica-se a todos os registros.

Utilizando MongoDB

- Obviamente que isso deve ser realizado com cuidado para que a base de dados não fique totalmente **desorganizada** e **com diversos atributos perdidos**.

- O interessante é que por meio de um update o MongoDB permite que possamos acrescentar dados.

- Isso acontece porque não existe um esquema dentro das coleções, ou seja, as **collections** podem ser dinâmicas.

Utilizando MongoDB

- O MongoDB também permite a exclusão dos dados que é feito através do comando abaixo:

```
db.usuarios.remove( { estado: "Rio Grande do Sul" } )
```

Utilizando MongoDB

- Utilizando um **Banco de Dados Relacional** seria o equivale ao código abaixo:

```sql
DELETE FROM usuarios WHERE estado = "Rio Grande do Sul"
```

Utilizando MongoDB

- Outro comando interessante para deleção muito utilizado é simplesmente colocarmos o código da Listagem 7:

Listagem 7. **Comando Delete**

```
"DELETE FROM usuarios"
#Utilizando o MongoDB faríamos semelhante conforme abaixo:
db.usuarios.remove()
```

Utilizando MongoDB

- Para atualização de dados com SQL temos o código da Listagem 8:

Listagem 8. **Comando Update**

```
1 | UPDATE usuarios SET estado = "Rio de Janeiro" WHERE cidade = "Rio de Janeiro"
```

Utilizando MongoDB

- No MongoDB o equivalente seria conforme o código da Listagem 9.

Listagem 9. **Comando Update no Mongo**

```
1 | db.usuarios.update( { cidade: { $eq:"Rio de Janeiro" } },
2 |                                 $set:   { estado: "Rio de Janeiro"} },
3 |                                 { multi: true }
4 |   )
```

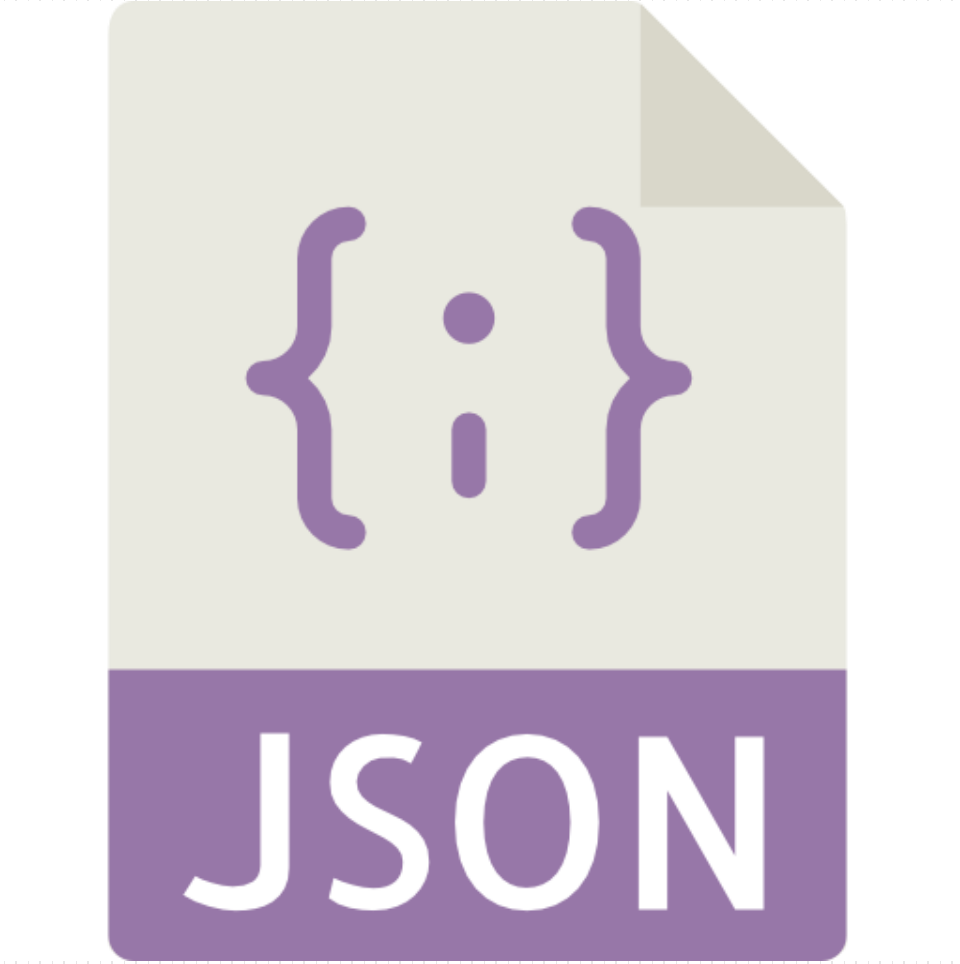
Utilizando MongoDB

- Outro comando muito utilizado é para fazer consultas.

- O MongoDB possui uma poderosa linguagem de consulta baseada em documentos, que torna a transição de um banco de dados relacional para o MongoDB bastante fácil, pois as consultas são convertidas com bastante facilidade.

JSON

- O formato JSON (JavaScript Object Notation) é um formato aberto usado como alternativa ao XML para a transferência de dados estruturados entre um servidor de Web e uma aplicação Web.
- Sua lógica de organização tem semelhanças com o XML, mas possui notação diferente.



Utilizando MongoDB

- Segue na Listagem 10 uma consulta realizada com SQL:

Listagem 10. **Comando no SQL**

```
1 | SELECT * FROM usuarios WHERE estado = "Rio de Janeiro"
```

Utilizando MongoDB

- Utilizando o MongoDB teríamos o equivalente a Listagem 11:

Listagem 11. **Consulta no Mongo**

```
db.usuarios.find(
                    { estado: { $eq: "Rio de Janeiro" } }
)
```

Utilizando MongoDB

- Também poderíamos ter comparações numéricas, conforme exemplo abaixo:

```
1 | SELECT * FROM usuarios WHERE idade > 25 AND idade <= 50
```

Utilizando MongoDB

- Utilizando o MongoDB teríamos o equivalente conforme abaixo:

```
db.usuarios.find(
                        { idade: { $gt: 25, $lte: 50 } }
  )
```

Utilizando MongoDB

- Também poderíamos ter o operador "**like**", que é muito utilizado como mostra a consulta a seguir:

```
1 | SELECT * FROM usuarios WHERE nome like "Higor%"
```

Utilizando MongoDB

- Utilizando o MongoDB teríamos o equivalente conforme demonstrado abaixo:

```
1 | db.usuarios.find(
2 |                 { nome: /^Higor/ }
3 |   )
```

Utilizando MongoDB

- Outra operação muito comum é a ordenação ascendente e descendente em SQL conforme exemplificado abaixo:

```
1 | SELECT * FROM usuarios WHERE cidade = "Porto Alegre" ORDER BY nome ASC
```

Utilizando MongoDB

- Utilizando o MongoDB teríamos o equivalente conforme abaixo:

```
1 | db.usuarios.find({cidade : "Porto Alegre"}).sort({nome:1})
```

Utilizando MongoDB

- Para ordenar de forma descendente teríamos o SQL:

```
1 | SELECT * FROM usuarios WHERE cidade = "Porto Alegre" ORDER BY nome desc
```

Utilizando MongoDB

- Utilizando o MongoDB teríamos o equivalente conforme abaixo:

```
1 | db.usuarios.find({cidade : "Porto Alegre"}).sort({nome:-1})
```

Utilizando MongoDB

- Outro comando muito utilizado nas consultas SQL é o **join**. Abaixo temos uma consulta SQL usando um join e procurando dentro desta tabela, segue abaixo o código:

```
1 | SELECT * FROM usuarios u INNER JOIN eventos e ON e.id = u.id
2 |              WHERE e.publicado is not null
3 |              GROUP BY u.email
```

Utilizando MongoDB

- Utilizando o MongoDB teríamos o equivalente conforme abaixo:

```
db.usuarios.find(('eventos.publicado': ($ne: null)))
```

Ano: 2021 Banca: CESGRANRIO Órgão: Banco do Brasil Prova: CESGRANRIO - 2021 - Banco do Brasil - Agente de Tecnologia

Um administrador de um banco de dados construído por meio do MongoDB inseriu dados em uma coleção (collection) de dados da seguinte forma:

db.fornecedores.insert({ codigo: “thx1138“, nome: “Roupas Syfy ltda“, pais: “Arabia Saudita“ })

Posteriormente, esse administrador construiu uma consulta que retornou apenas o nome, sem repetição, de todos os países que fazem parte dessa coleção (collection).

O comando utilizado para tal consulta foi:

Alternativas

a) db.fornecedores.find("pais")
b) db.fornecedores.find().pretty({"pais":1})
c) db.fornecedores.find().sort({"pais":1})
d) db.fornecedores.distinct({"pais":0})
e) db.fornecedores.distinct("pais")

Ano: 2021 Banca: CESGRANRIO Órgão: Banco do Brasil Prova: CESGRANRIO - 2021 - Banco do Brasil - Agente de Tecnologia

Um administrador de um banco de dados construído por meio do MongoDB inseriu dados em uma coleção (collection) de dados da seguinte forma:

db.fornecedores.insert({ codigo: “thx1138“, nome: “Roupas Syfy ltda“, pais: “Arabia Saudita“ })

Posteriormente, esse administrador construiu uma consulta que retornou apenas o nome, sem repetição, de todos os países que fazem parte dessa coleção (collection).

O comando utilizado para tal consulta foi:

Alternativas

a) db.fornecedores.find("pais")
b) db.fornecedores.find().pretty({"pais":1})
c) db.fornecedores.find().sort({"pais":1})
d) db.fornecedores.distinct({"pais":0})
e) db.fornecedores.distinct("pais")

Ano: 2020 Banca: INSTITUTO AOCP Órgão: MJSP Prova: INSTITUTO AOCP - 2020 - MJSP - Cientista de Dados - Big Data

Um cientista de dados necessita apresentar a quantidade de documentos em uma coleção do MongoDB. Sabendo disso, assinale a alternativa que apresenta corretamente o operador que o cientista precisa utilizar.

Alternativas

a) db.collection.agregate ()

b) db.collection.count ()

c) db.collection.total ()

d) db.collection.find ()

e) db.collection.sum ()

Ano: 2020 Banca: INSTITUTO AOCP Órgão: MJSP Prova: INSTITUTO AOCP - 2020 - MJSP - Cientista de Dados - Big Data

Um cientista de dados necessita apresentar a quantidade de documentos em uma coleção do MongoDB. Sabendo disso, assinale a alternativa que apresenta corretamente o operador que o cientista precisa utilizar.

Alternativas

a) db.collection.agregate ()

b) db.collection.count ()

c) db.collection.total ()

d) db.collection.find ()

e) db.collection.sum ()

Ano: 2021 Banca: FGV Órgão: TJ-RO Prova: FGV - 2021 - TJ-RO - Analista Judiciário - Analista de Sistema - Desenvolvimento de Sistema

Com referência ao MongoDB, nas consultas usando o método find é possível exibir o resultado formatado por meio do método:

Alternativas

a) format;
b) organize;
c) pretty;
d) structure;
e) tidy.

Ano: 2021 Banca: FGV Órgão: TJ-RO Prova: FGV - 2021 - TJ-RO - Analista Judiciário - Analista de Sistema - Desenvolvimento de Sistema

Com referência ao MongoDB, nas consultas usando o método find é possível exibir o resultado formatado por meio do método:

Alternativas

a) format;

b) organize;

c) pretty;

d) structure;

e) tidy.

Utilizando MongoDB

- Fazendo uma analogia com o Banco de Dados Relacional tem-se que:
- o documento JSON seria o registro

- a collection seria a tabela

- os índices são os mesmos

- e o embedding e o linking seria o join.

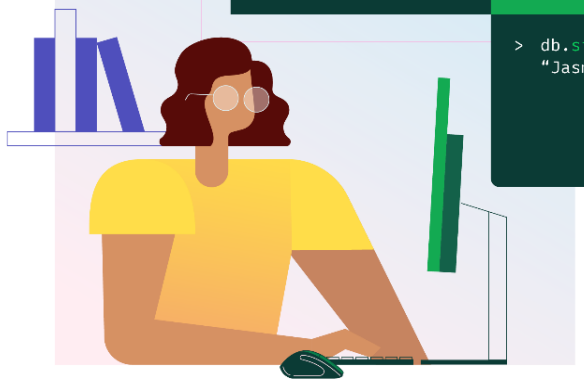
Utilizando MongoDB

MongoDB

```
{
"_id": 1,
"student_name": "Jasmin Scott",
  "school": {
     "school_id": 226,
     "name": "Tech Secondary",
     "address": "100 Broadway St",
     "city": "New York",
     "state": "NY",
     "zipcode": "10001"
  },
"marks": [98, 93, 95, 88, 100],
}
```

mongo

```
> db.students.find({"student_name":
  "Jasmin Scott"})
```



SQL

students

id	name	school_id
1	Jasmin Scott	226
...	...	...

marks

id	student_id	mark
10	1	98
...	...	...

school_details

id	name	address	city	state	zipcode
226	Tech Secondary	100 Broadway St	New York	NY	10001
...	...	...	...	...	...

Results

name	mark	school_name	city
Jasmin Scott	98	Tech Secondary	New York
...	...	...	...

sql

```
SELECT s.name, m.mark, d.name as "school name",
d.city
FROM students s
INNER JOIN marks m ON s.id = m.student_id
INNER JOIN school_details d ON s.school_id = d.id
WHERE s.name = "Jasmin Scott";
```

Ano: 2021 Banca: CESPE / CEBRASPE Órgão: SERPRO Prova: CESPE / CEBRASPE - 2021 - SERPRO - Analista - Especialização: Ciência de Dados

A respeito de conceitos de NoSQL orientado a grafos, orientado a documentos e orientado a colunas, julgue o item a seguir.

Uma coleção e um documento, no MongoDB, são equivalentes à tabela e à linha, no Modelo Relacional de Dados.

Alternativas

Certo

Errado

Ano: 2021 Banca: CESPE / CEBRASPE Órgão: SERPRO Prova: CESPE / CEBRASPE - 2021 - SERPRO - Analista - Especialização: Ciência de Dados

A respeito de conceitos de NoSQL orientado a grafos, orientado a documentos e orientado a colunas, julgue o item a seguir.

Uma coleção e um documento, no MongoDB, são equivalentes à tabela e à linha, no Modelo Relacional de Dados.

Alternativas

Certo

Errado

Ano: 2020 Banca: CPCON Órgão: Câmara de Cerro Corá - RN Prova: CPCON - 2020 - Câmara de Cerro Corá - RN - Agente Administrativo

Sobre Banco de Dados, assinale a alternativa CORRETA.

Alternativas

a) Os sistemas NoSQL, quando comparados a bancos de dados relacionais, oferecem mais escalabilidade, melhor performance em consultas e flexibilidade para manipular dados que podem apresentar mudanças em seu formato.

b) Um Banco de Dados é uma coleção de programas que permite aos usuários definir, construir e manipular uma base de dados para o uso em diversas aplicações.

c) Um modelo lógico consiste em uma descrição do Banco de Dados de forma independente de implementação em um SGBD (Sistema de Gerenciamento de Banco de Dados).

d) Um modelo conceitual de um Banco de Dados descreve como os dados serão armazenados no banco e também seus relacionamentos. Esse modelo adota alguma tecnologia, pode ser: relacional, orientado a objetos, orientado a colunas, entre outros.

e) São exemplos de SGBD: Oracle, SQLServer, PostgreSQL, MySQL, Microsoft Edge e MongoDB.

Ano: 2020 Banca: CPCON Órgão: Câmara de Cerro Corá - RN Prova: CPCON - 2020 - Câmara de Cerro Corá - RN - Agente Administrativo

Sobre Banco de Dados, assinale a alternativa CORRETA.

Alternativas

a) **Os sistemas NoSQL, quando comparados a bancos de dados relacionais, oferecem mais escalabilidade, melhor performance em consultas e flexibilidade para manipular dados que podem apresentar mudanças em seu formato.**

b) Um Banco de Dados é uma coleção de programas que permite aos usuários definir, construir e manipular uma base de dados para o uso em diversas aplicações.

c) Um modelo lógico consiste em uma descrição do Banco de Dados de forma independente de implementação em um SGBD (Sistema de Gerenciamento de Banco de Dados).

d) Um modelo conceitual de um Banco de Dados descreve como os dados serão armazenados no banco e também seus relacionamentos. Esse modelo adota alguma tecnologia, pode ser: relacional, orientado a objetos, orientado a colunas, entre outros.

e) São exemplos de SGBD: Oracle, SQLServer, PostgreSQL, MySQL, Microsoft Edge e MongoDB.

Utilizando MongoDB

- As vantagens em utilizar o **MongoDB** começam a aparecer quando podemos representar objetos do mundo real da forma que eles são, e caso novos atributos surjam podemos aplica-los somente onde é necessário e não em todos os casos como ocorria no **modelo relacional**.

Utilizando MongoDB

- Enquanto no modelo relacional somos obrigados a pensar em evitar o tempo todo a **redundância de dados**, motivo pelo qual existem os relacionamentos.

- No **MongoDB** temos uma situação diferente em que não há relacionamentos e a duplicação por vezes é até mesmo incentivada como nos casos observados anteriormente onde os dados são armazenados do jeito que quisermos.

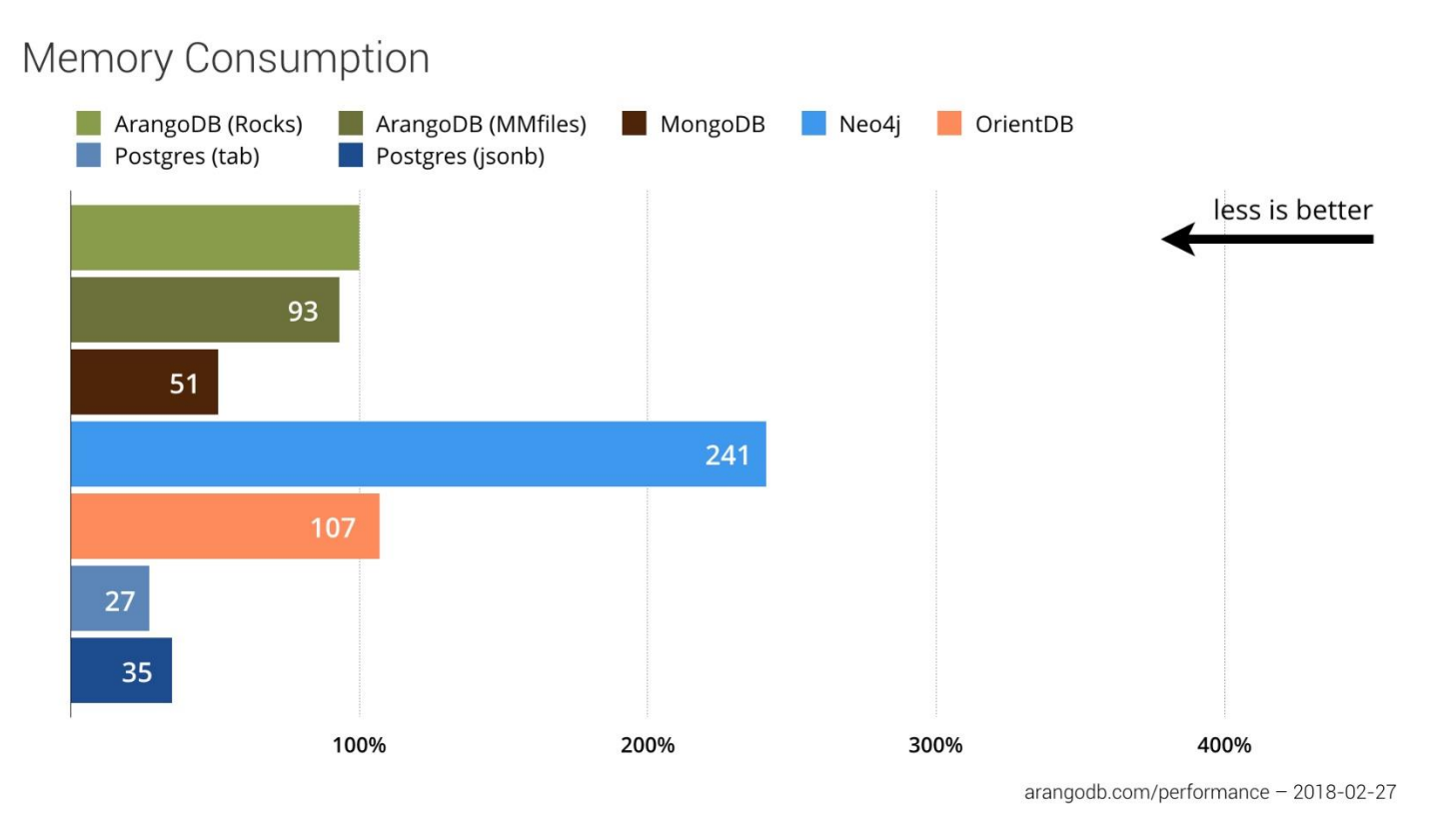
Utilizando MongoDB

- A ideia do MongoDB é que tenhamos **documentos autocontidos** obtendo todas as informações que necessitamos sem que seja necessário realizarmos vários **joins**.

- Dessa forma fazemos apenas uma consulta e o retorno será o documento inteiro com todas as informações resultando num ganho significativo de **performance**.

Vantagens

- Utilizando MongoDB temos uma melhor performance, visto que uma única consulta retorna tudo o que precisamos saber sobre o documento.



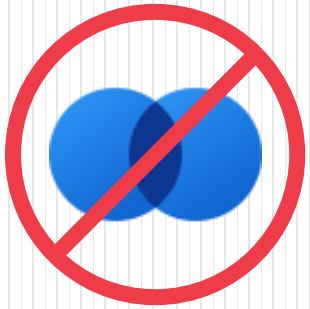
Vantagens

- Os banco de dados NoSQL apresentam vantagens sobre os outros quando precisamos de:

- Escalabilidade
- Flexibilidade
- Manipulação de quantidade massiva de dados
- Bom desempenho
- Facilidade para consultas.

Vantagens

- O MongoDB possui consultas bastantes simples de serem realizadas, visto que **não existem transações** e **joins**.



- As consultas são mais fáceis de escrever e mais fáceis de ajustar.

Vantagens

- O escalonamento horizontal com **Sharding** é muito bem implementado no MongoDB.

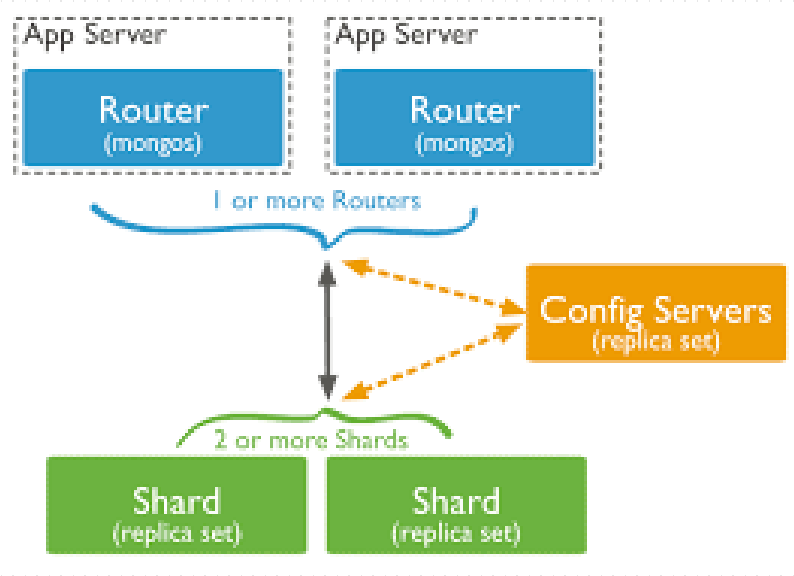
- **Sharding** é utilizado quando temos muitos dados e estamos no limite do disco, dessa forma dividimos esses dados entre várias máquinas.

- Dessa maneira temos mais rendimento e maior capacidade de armazenamento em disco.

Vantagens

- Portanto, quanto mais Shards maior será o armazenamento e o desempenho.

- O MongoDB disponibiliza essa opção.



Vantagens

- Para mais detalhes sobre **Sharding** podemos consultar a documentação oficial na página oficial do MongoDB.

https://www.mongodb.com/docs/

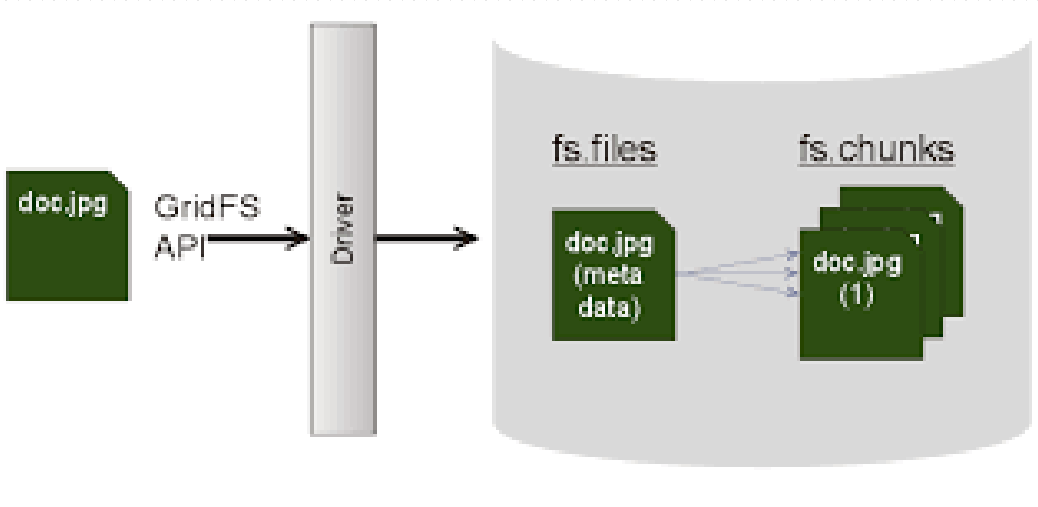
- Banco de dados relacionais muito utilizados como o MySQL não suportam esse tipo de solução por padrão, para isso teríamos que manipular os dados em uma camada acima da base de dados, sendo muito mais trabalhoso.

Vantagens

- Como a linguagem de **consulta SQL** é fácil de ser convertida para **MongoDB** temos uma maior facilidade para migração de organizações que atualmente usam bancos de dados relacionais.

Vantagens

- Por fim, o MongoDB também possui a funcionalidade chamada **GridFS** que é responsável por armazenar arquivos de grandes dimensões.



Vantagens

- Isso significa que vídeos, arquivos, etc, podem ser armazenados diretamente no banco de dados.

- Diferente do modelo relacional que tinhamos que armazenar tudo no **Filesystem** e disponibilizar apenas uma referência para esses arquivos no banco de dados.

- O tempo de resposta desses arquivos armazenados no MongoDB é comparado ao tempo de resposta dos arquivos em Filesystems.

Desvantagens

- Uma desvantagem é quando queremos alterar todos os registros relacionados a uma unidade semântica, nesse caso é preciso tratar um a um.

Desvantagens

- Uma pergunta a ser respondida ainda é a dúvida que todos os usuários possuem em relação ao suporte do MongoDB que todas as grandes empresas e projetos de alto valor necessitam.

- **Até que ponto o MongoDB oferece esse suporte necessário?**

- Outro ponto é em relação a disponibilidade do serviço.

Desvantagens

- Essas e outras perguntas serão respondidas ao longo do tempo.
- O fato é que muitas empresas de grande porte tem utilizado em projetos de complexidade média.
- Uma escolha natural devido ao pequeno tempo que essa nova tecnologia está no mercado e os resultados têm sido muito satisfatórios.

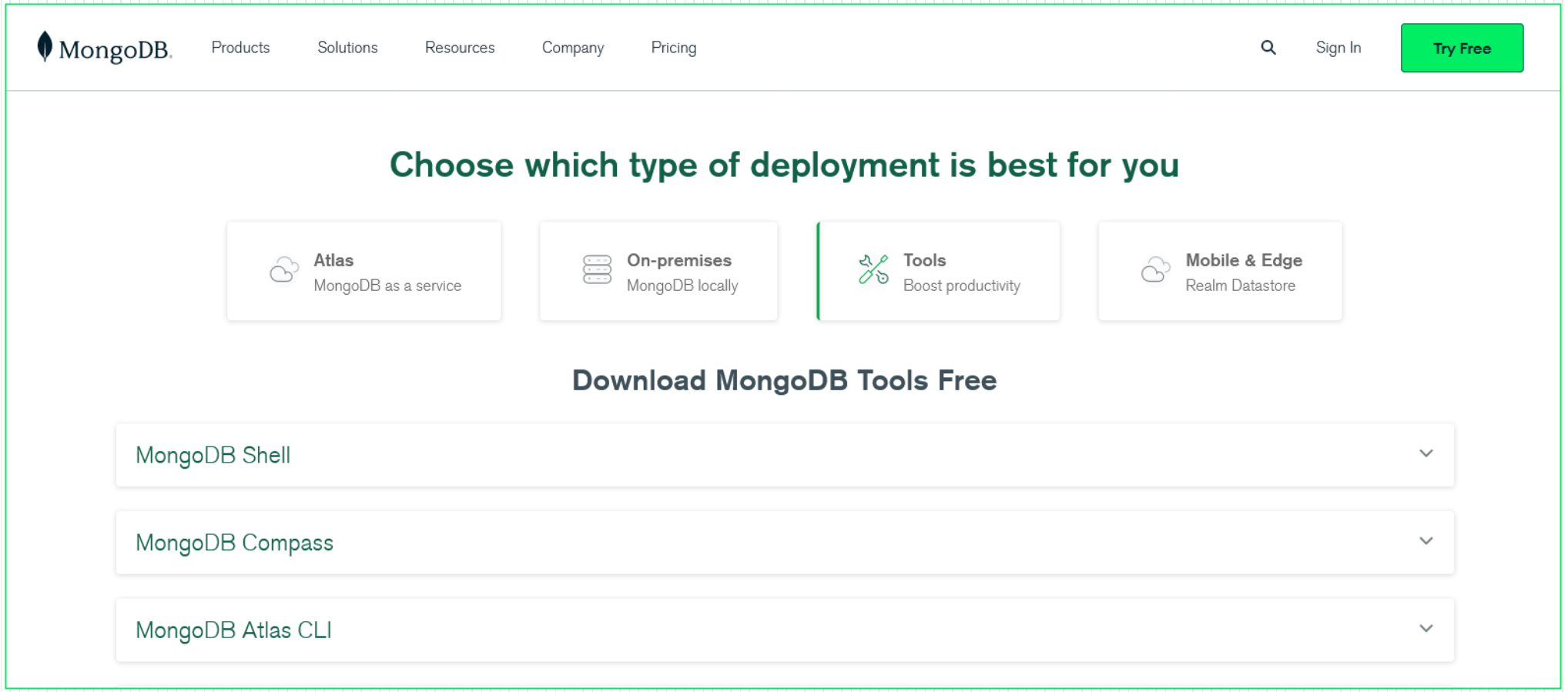
Criando uma Aplicação com MongoDB

- Para exercitar os conhecimentos adquiridos com MongoDB vamos criar uma aplicação de exemplo para manipular um banco de dados MongoDB.

- Primeiramente devemos baixar o MongoDB em ***www.mongodb.org/downloads*** escolhendo a versão para o nosso sistema operacional.

Criando uma Aplicação com MongoDB

www.mongodb.org/downloads



Criando uma Aplicação com MongoDB

- Após isso, basta descompactar numa pasta ou executar o instalador (se for ".msi").

- Feito isso devemos criar uma pasta "data" e dentro da pasta "data" uma pasta "db" no **C:/** do Windows.

- Nessa pasta o MongoDB colocará os bancos de dados criados.

Criando uma Aplicação com MongoDB

- Agora já podemos executar o aplicativo “**mongod.exe**” que está dentro da pasta “**bin**” do MongoDB.
- Esse aplicativo executa os bancos de dados do MongoDB na **porta 27017**.
- Para gerenciar as tabelas podemos utilizar o Shell do MongoDB executando o aplicativo “mongo.exe”.

Criando uma Aplicação com MongoDB

- Também devemos baixar o arquivo ”.jar” para que o nosso projeto Java consiga se comunicar com o Banco de dados.

- Para isso baixe o arquivo e coloque na raiz do projeto que será criado abaixo:

https://github.com/mongodb/mongo-java-driver

- Agora devemos ir até o Eclipse e criar um novo projeto Java, conforme mostram as **Figuras 1** a **3**.

Criando uma Aplicação com MongoDB

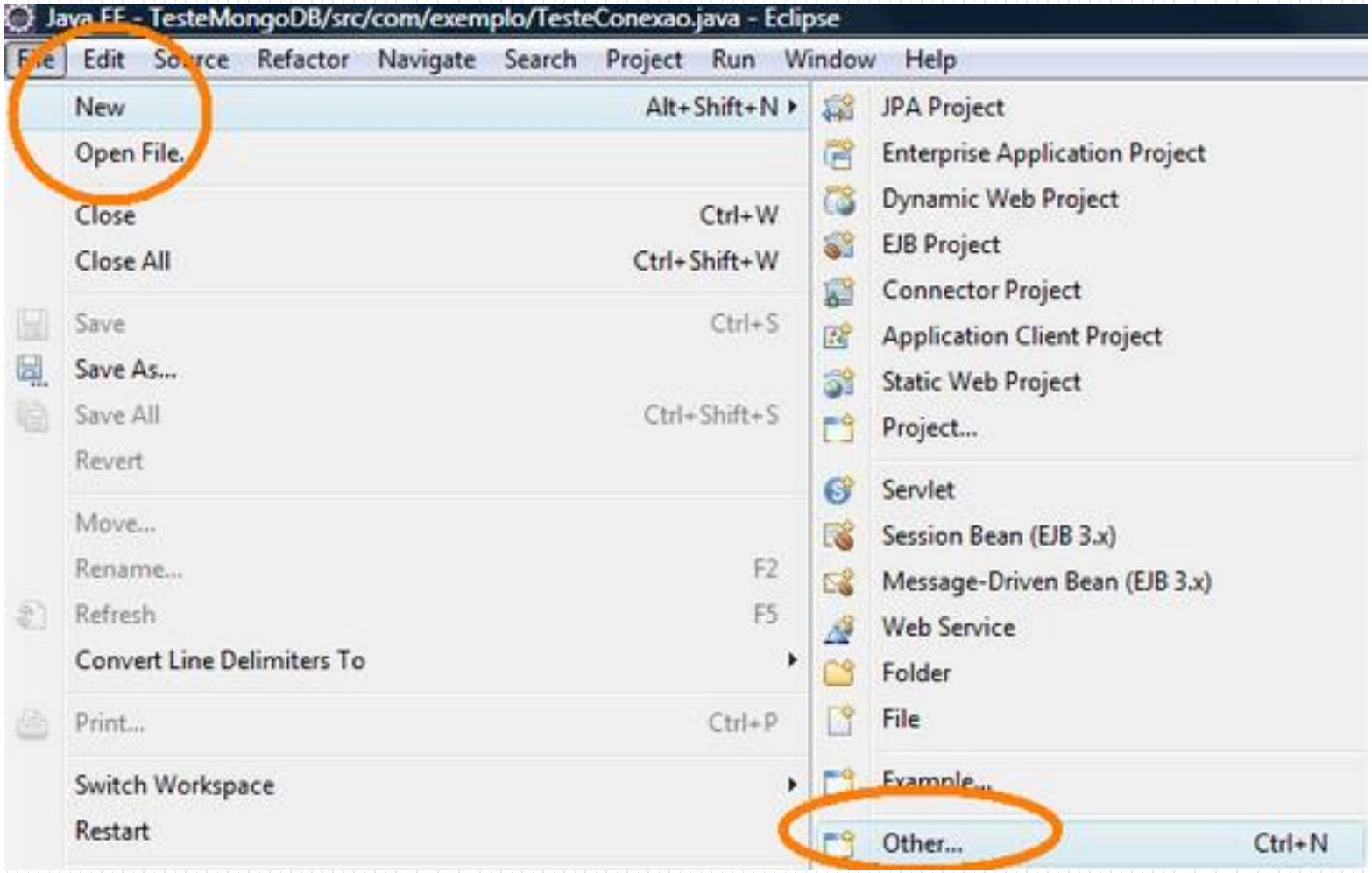


Figura 1. Selecionando *Other* para selecionar o tipo de projeto

Criando uma Aplicação com MongoDB

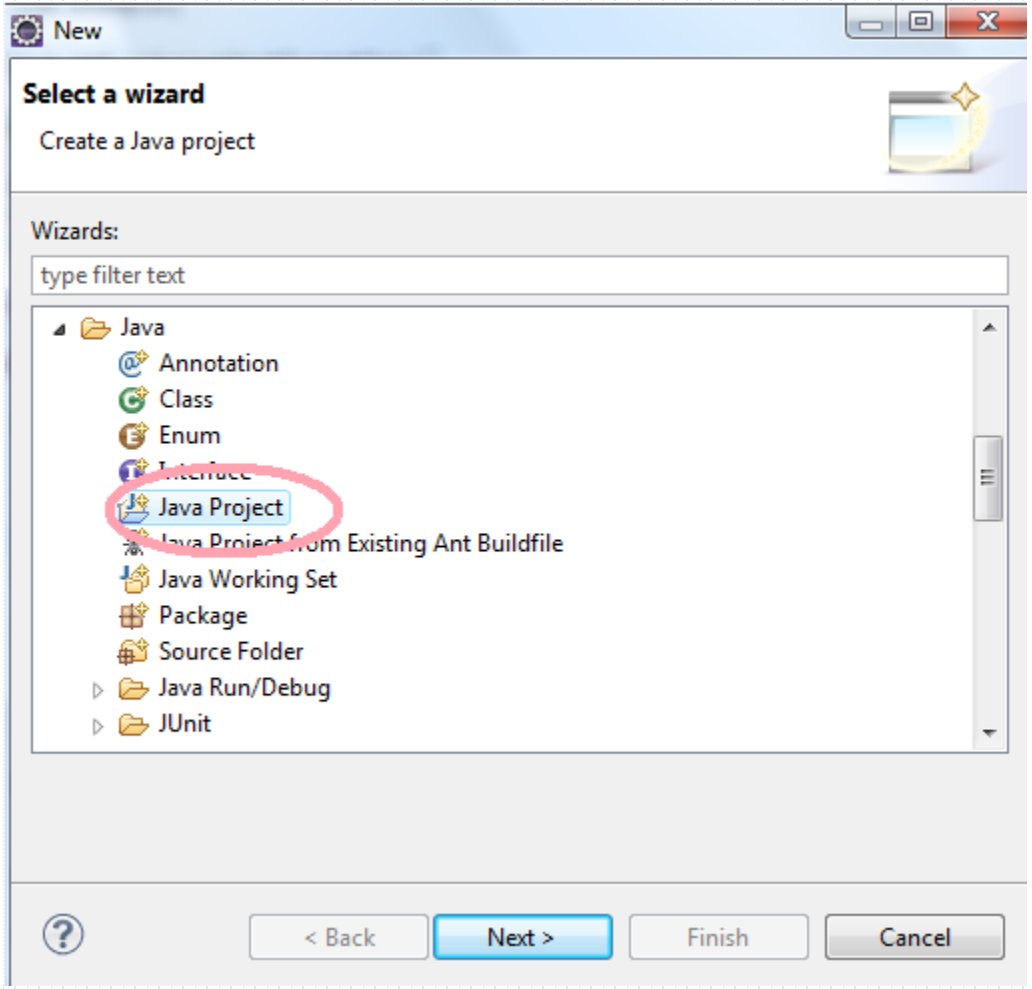


Figura 2. Criando um projeto Java

Criando uma Aplicação com MongoDB

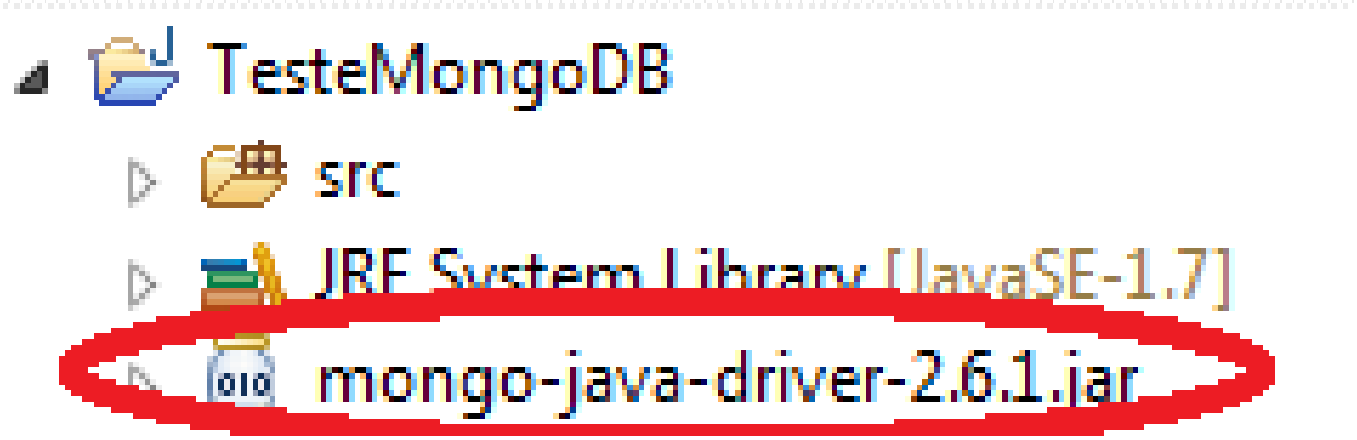


Figura 3. Colocando o jar na raiz do projeto

Criando uma Aplicação com MongoDB

- Agora que já temos o nosso projeto Java criado vamos fazer um exemplo criando um pacote chamado "**com.exemplo**" e uma classe **ExemploMongoDB.java** dentro do pacote criado.

- A nossa classe deverá ficar como a da Listagem 12.

Criando uma Aplicação com MongoDB

Listagem 12. **Exemplo de conexão e manipulação de dados com MongoDB 1/3**

```
1  package com.exemplo;
2
3    import java.net.UnknownHostException;
4    import java.util.List;
5
6
7    import com.mongodb.BasicDBObject;
8    import com.mongodb.DB;
9    import com.mongodb.DBCollection;
10   import com.mongodb.DBCursor;
11   import com.mongodb.Mongo;
12   import com.mongodb.MongoException;
13
```

Criando uma Aplicação com MongoDB

Listagem 12. **Exemplo de conexão e manipulação de dados com MongoDB 2/3**

```
15  public class ExemploMongoDB {
16
17      public static void main(String args[]) throws UnknownHostException, MongoException {
18
19          Mongo mongo = new Mongo("localhost", 27017);
20
21          DB db = mongo.getDB("testemongodb");
22
23          BasicDBObject doc = new BasicDBObject("username", "higormed").
24                  append("nome",   "Higor Medeiros").
25                  append("cidade", "Porto Alegre");
26
27          DBCollection collec = db.getCollection("dados");
28
29          collec.insert(doc);
30
```

Criando uma Aplicação com MongoDB

Listagem 12. **Exemplo de conexão e manipulação de dados com MongoDB 3/3**

```
31              //lista as coleções
32              DBCursor cursor = collec.find();
33          int i=1;
34          while (cursor.hasNext()) {
35              System.out.println("Documento Inserido: "+i);
36              System.out.println(cursor.next());
37              i++;
38          }
39
40          System.out.println("Banco de dados armazenados:");
41                List<String> dbs = mongo.getDatabaseNames();
42                for(String dbStr : dbs){
43                        System.out.println(dbStr);
44                }
45
46                System.out.println("fim execucao");
47
48      }
49
50  }
```

Criando uma Aplicação com MongoDB

- O código acima cria um novo banco de dados caso ainda não exista um com o nome “**testemongodb**”
- Cria um novo documento com o nome “dados” e com os valores definidos
- Lista os documentos inseridos
- E por fim lista todos os bancos de dados criados até o momento.

Criando uma Aplicação com MongoDB

- Com isso já podemos criar um banco de dados, documentos, valores, listagens, e muitos mais com o MongoDB.

- Como exercício tente atualizar os dados, listá-los e por fim excluir os dados e listá-los novamente para verificar a remoção.

Ano: 2018 Banca: FCC Órgão: MPE-PE Prova: FCC - 2018 - MPE-PE - Analista Ministerial - Informática

No gerenciador de bancos de dados MongoDB, os comandos para exibir a lista de bancos de dados presentes no servidor e para gerar um backup são, respectivamente,

Alternativas

a) show dbs e mongodump.
b) list dbs e mongodrop.
c) copy dbs e mongouse.
d) use dbs e mongoname.
e) serve dbs e mongofind.

Ano: 2018 Banca: FCC Órgão: MPE-PE Prova: FCC - 2018 - MPE-PE - Analista Ministerial - Informática

No gerenciador de bancos de dados MongoDB, os comandos para exibir a lista de bancos de dados presentes no servidor e para gerar um backup são, respectivamente,

Alternativas

a) **show dbs e mongodump.**

b) list dbs e mongodrop.

c) copy dbs e mongouse.

d) use dbs e mongoname.

e) serve dbs e mongofind.

Ano: 2020 Banca: FUNDATEC Órgão: CIGA-SC Prova: FUNDATEC - 2020 - CIGA-SC - Programador

Em um banco de dados MongoDB vazio, um usuário abre o MongoDB Shell e executa os comandos abaixo, na ordem em que são mostrados:

1. use ciga
2. db.cidades.insertOne({ id: 1, descricao:"Fortaleza" })
3. db.cidades.insertOne({ id: 1, descricao:"Fortaleza", uf:"CE" })
4. db.cidades.find({})

É correto afirmar que:

Alternativas

a) Ocorre erro no comando 1.
b) Ocorre erro no comando 2, pois é necessário criar a coleção com “create collection (id: integer, uf: string)” antes.
c) Ocorre erro no comando 3, pois a coleção não tem o atributo “uf” definido para o documento.
d) Apenas um (1) documento será mostrado após o último comando.
e) Dois (2) documentos serão mostrados após o último comando.

Ano: 2020 Banca: FUNDATEC Órgão: CIGA-SC Prova: FUNDATEC - 2020 - CIGA-SC - Programador

Em um banco de dados MongoDB vazio, um usuário abre o MongoDB Shell e executa os comandos abaixo, na ordem em que são mostrados:

1. use ciga
2. db.cidades.insertOne({ id: 1, descricao:"Fortaleza" })
3. db.cidades.insertOne({ id: 1, descricao:"Fortaleza", uf:"CE" })
4. db.cidades.find({})

É correto afirmar que:

Alternativas

a) Ocorre erro no comando 1.

b) Ocorre erro no comando 2, pois é necessário criar a coleção com “create collection (id: integer, uf: string)” antes.

c) Ocorre erro no comando 3, pois a coleção não tem o atributo “uf” definido para o documento.

d) Apenas um (1) documento será mostrado após o último comando.

e) Dois (2) documentos serão mostrados após o último comando.

Ano: 2020 Banca: INSTITUTO AOCP Órgão: MJSP Prova: INSTITUTO AOCP - 2020 - MJSP - Cientista de Dados - Big Data

Um cientista de dados deve criar índices como uma forma de aprimorar o desempenho do banco de dados MongoDB em nível de coleção. Assim, a sintaxe correta de criação de índice do MongoDB que esse cientista de dados deve executar é

Alternativas

a) db.collection.createIndex ({<fieldname>:(1|-1)});

b) db.collection.createIndex ({<collectionname});

c) db.collection.newIndex ({<collectionname>:(1|-1)});

d) db.collection.createIndex ({<dbname>:(1|-1)});

e) db.collection.newIndex({<dbname});

Ano: 2020 Banca: INSTITUTO AOCP Órgão: MJSP Prova: INSTITUTO AOCP - 2020 - MJSP - Cientista de Dados - Big Data

Um cientista de dados deve criar índices como uma forma de aprimorar o desempenho do banco de dados MongoDB em nível de coleção. Assim, a sintaxe correta de criação de índice do MongoDB que esse cientista de dados deve executar é

Alternativas

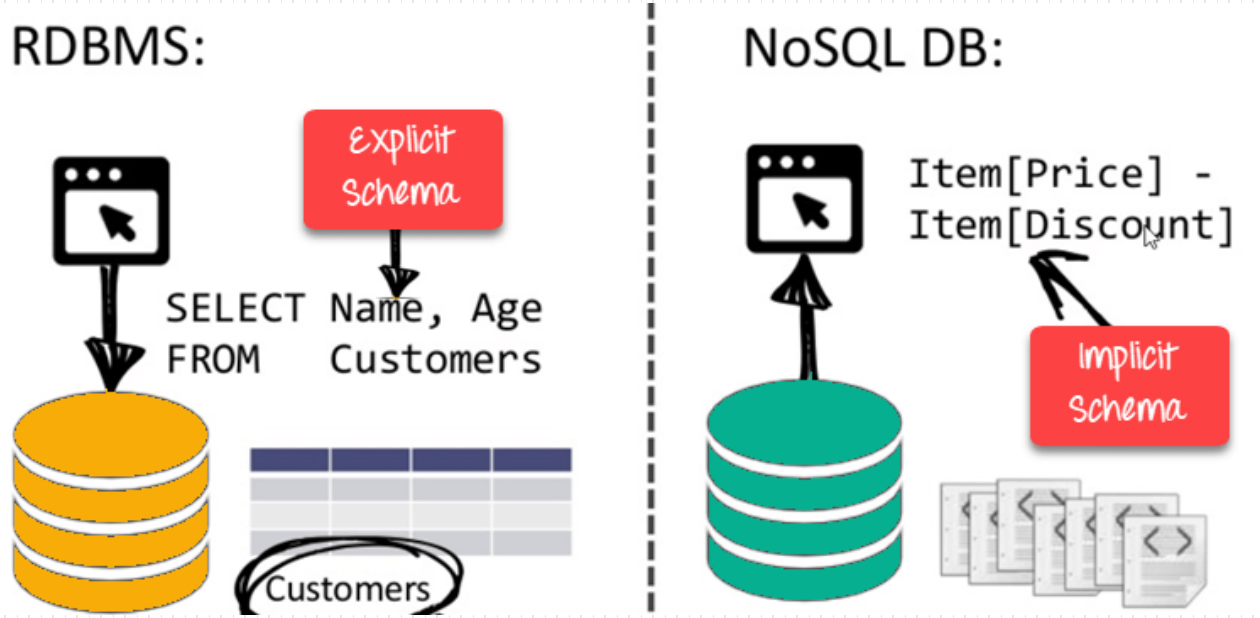
a) **db.collection.createIndex ({<fieldname>:(1|-1)});**
b) db.collection.createIndex ({<collectionname});
c) db.collection.newIndex ({<collectionname>:(1|-1)});
d) db.collection.createIndex ({<dbname>:(1|-1)});
e) db.collection.newIndex({<dbname});

NoSQL

- **NoSQL** ("não SQL" ou "não relacional", posteriormente estendido para ***Not Only SQL*** - Não Somente SQL) é um termo genérico que representa os bancos de dados não relacionais.
- É um movimento que promove soluções de armazenamento de dados não relacionais.
- Ele é composto por diversas ferramentas que buscam resolver problemas como tratamento de grandes volumes de dados, execução de consultas com baixa latência e modelos flexíveis de armazenamento de dados, como documentos XML ou JSON.

NoSQL

- As tecnologias NoSQL não têm como objetivo substituir os bancos de dados relacionais, mas apenas propor algumas soluções que em determinados cenários são mais adequadas.
- Desta forma é possível trabalhar com tecnologias NoSQL e banco de dados relacionais dentro de uma mesma aplicação.



Onde utilizar ?

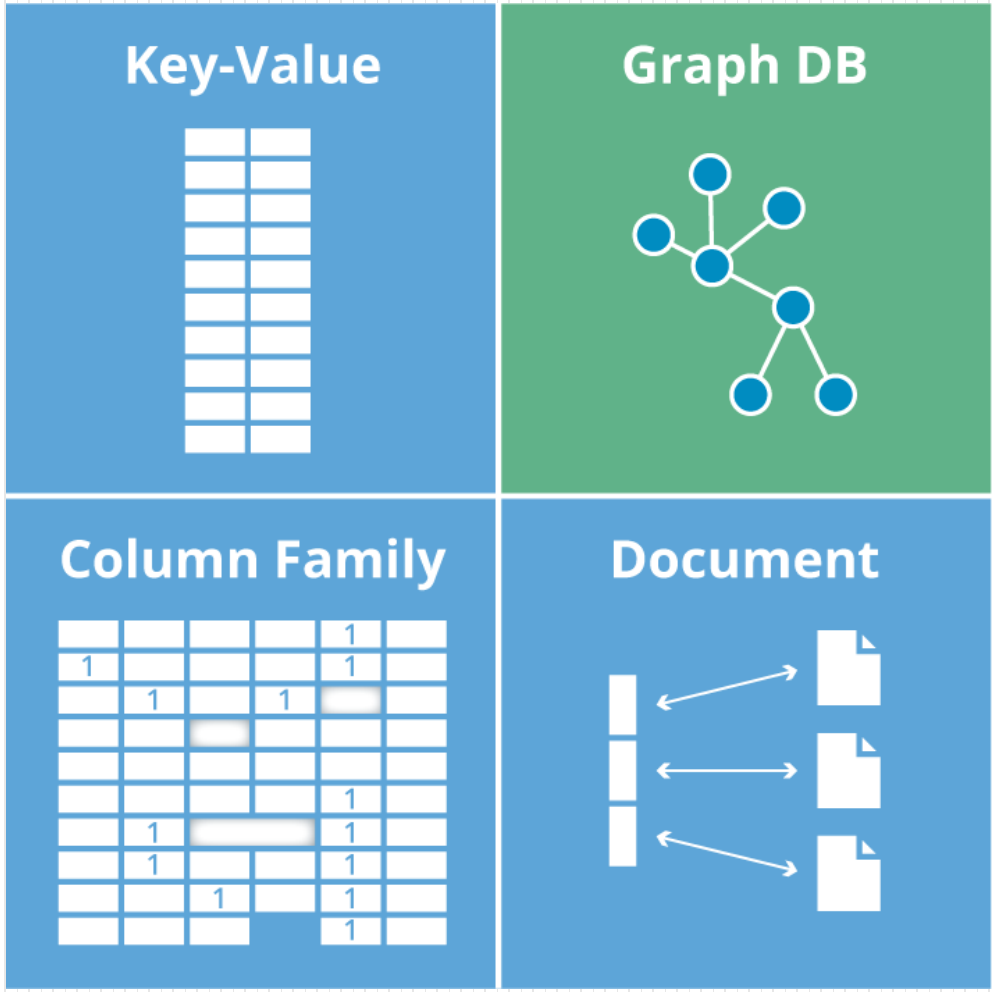
- Em cenários onde sistemas de banco de dados tradicionais não são suficientes ou adequados às necessidades específicas, tais como:
  - baixa latência, grandes volumes de dados, escalabilidade ou estruturas em que as conexões entre os dados são tão importantes quanto o próprio dado.
- Todas as tecnologias abordadas apresentam situações de uso particulares, que podem ser úteis na substituição dos tradicionais bancos de dados relacionais.

Bancos

- Redis adota o modelo **chave-valor**, utiliza a memória para alocação de dados e é ideal para utilização de Cache.
- MongoDB implementa o modelo baseado em **documentos**, tem foco no tratamento de grandes volumes de dados e é ideal para grande parte das aplicações web.
- Neo4j, uma das ferramentas NoSQL mais maduras, tem o modelo baseado em **grafos** e seus principais casos de uso estão relacionados a motores de recomendação, análise de rotas geográficas e redes sociais.
- Cassandra, que é uma implementação *open source* do modelo de dados do BigTable com a arquitetura distribuída do Dynamo.

Tipos

- **Chave-Valor**
  - Armazena dados como um conjunto de pares de chave-valor em que uma chave funciona como um identificador exclusivo.
  - A chave e os valores podem ser qualquer coisa, desde objetos simples até objetos compostos complexos.
  - São altamente particionáveis e permitem escalabilidade horizontal que outros tipos de bancos de dados não conseguem alcançar.
- **XML**
  - subclasse de bancos de dados orientados a documentos que são otimizados para trabalhar com documentos XML.
- **Grafos**
  - são similares, mas adicionam outra camada, o relacionamento, que permite que eles liguem documentos para acesso rápido.



Tipos

- **Coluna**
  - Enquanto um BD relacional é otimizado para armazenar linhas de dados, geralmente para aplicativos transacionais, um banco de dados colunar é otimizado para recuperação rápida de colunas de dados, normalmente em aplicativos analíticos.
  - O armazenamento orientado a colunas para tabelas do banco de dados é um fator importante para a performance de consulta analítica, pois ele reduz expressivamente os requisitos gerais de E/S de disco e diminui a quantidade de dados que você precisa carregar do disco.

Tipos

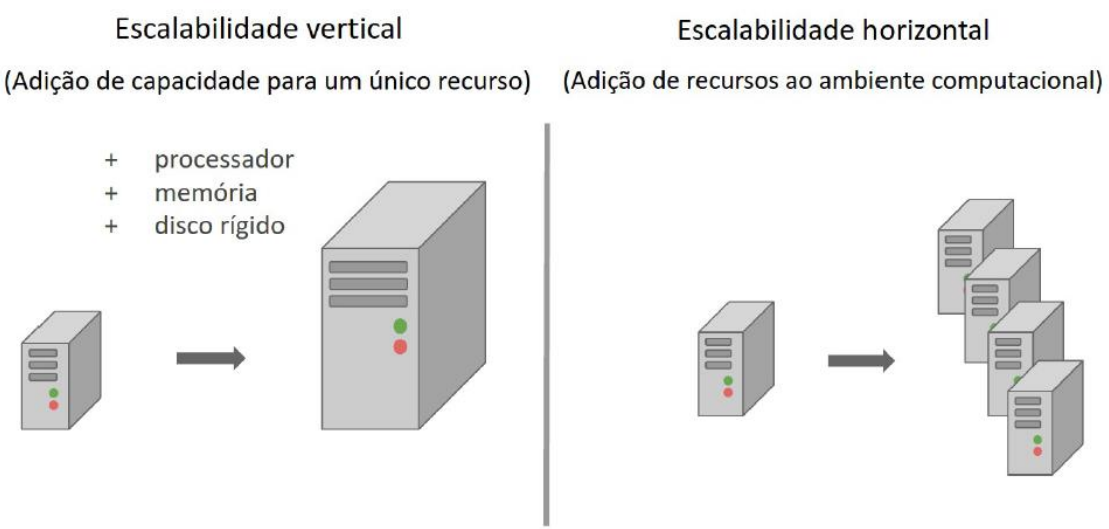
- **Documento (document store)**
  - Projetado para armazenar, recuperar e gerenciar informações orientadas a documentos, também conhecidas como dados semiestruturados.
  - São uma subclasse do armazenamento chave-valor. A diferença está na maneira que os dados são processados.
  - No armazenamento chave-valor, os dados são considerados como sendo inerentemente opacos ao banco de dados, enquanto que um sistema orientado a documentos depende da estrutura interna no documento afim de extrair metadados que o mecanismo do banco de dados utiliza para otimização adicional.
  - Apesar da diferença ser geralmente discutível, devido às ferramentas nos sistemas, conceitualmente o armazenamento por documentos é projetado para oferecer uma experiência mais rica com técnicas de programação modernas.

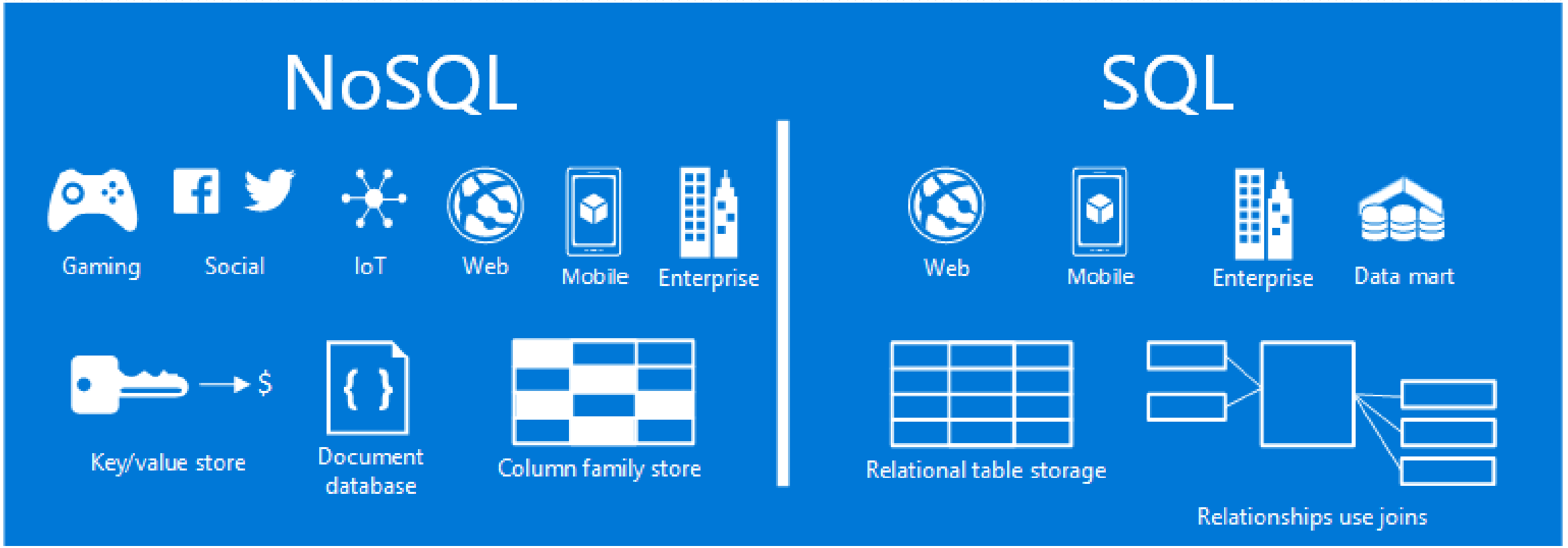
Tipos

- **Memória**
  - Aplicativos de jogos e tecnologia de publicidade têm como exemplos de uso placares de líderes, armazenamentos de sessões e análises em tempo real que exigem tempos de resposta em microssegundos e podem ter grandes picos de tráfego a qualquer momento.
- **Pesquisa**
  - Construído especificamente para fornecer visualizações e análises quase em tempo real de dados gerados por máquina ao indexar, agregar e pesquisar registros e métricas semiestruturadas. Também é um poderoso mecanismo de pesquisa de alta performance para casos de uso de pesquisa de texto completo.

Escalabilidade

- Vertical (scale up):
  - significa adicionar recursos em um único nó do sistema (mais memória ou um disco rígido mais rápido).
- Horizontal (scale out):
  - significa adicionar mais nós ao sistema, tais como um novo computador com uma aplicação para clusterizar o software.




NoSQL
Gaming
Social
IoT
Web
Mobile
Enterprise
Key/value store
$
Document database
{ }
Column family store
SQL
Web
Mobile
Enterprise
Data mart
Relational table storage
Relationships use joins

Exemplos

Document Database	Graph Databases
 Couchbase MarkLogic mongoDB	 InfiniteGraph The Distributed Graph Database
Wide Column Stores	**Key-Value Databases**
   AEROSPIKE 	  HYPERTABLE INC   Amazon SimpleDB

Propriedades BASE

- **BA (Basically Available)**: disponibilidade é prioridade.
- **S (Soft State)**: não precisa ser consistente o tempo todo.
- **E (Eventually Consistent)**: consistente em momento indeterminado.

ACID	BASE
Consistência Forte	Fraca Consistência
Isolamento	Foco em Disponibilidade
Concentra-se em “commit”	Melhor esforço
Transações aninhadas	Respostas aproximadas
Disponibilidade	Mais simples e mais rápido
Conservador (pessimista)	Agressivo (otimista)
Evolução difícil	Evolução mais fácil

Referências

- Artigo base do Curso – https://www.devmedia.com.br/introducao-ao-mongodb/30792
- Site do MongoDB - https://www.mongodb.com/pt-br
- Documentação do MongoDB - https://www.mongodb.com/docs/
- Instalação - https://www.mongodb.com/docs/manual/


STUDY HARD