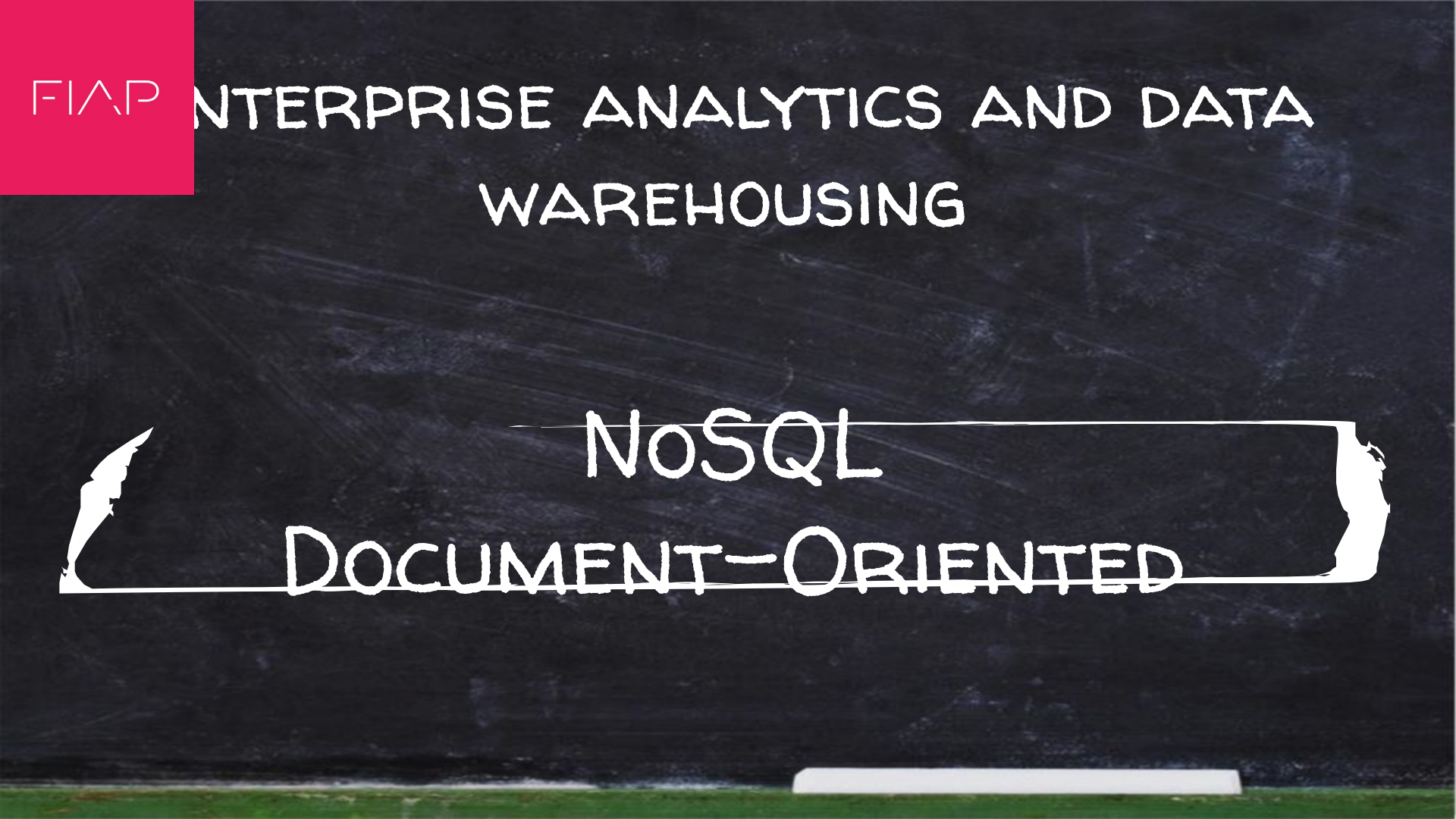
FIAP

Sistemas de informação

Enterprise analytics and data warehousing

FIAP


NTERPRISE ANALYTICS AND DATA
WAREHOUSING
NoSQL
DOCUMENT-ORIENTED

FIAP

Agenda

- ✘ Introdução
- ✘ MongoDB
- ✘ Hands On
- ✘ Atividade de Projeto III
- ✘ Referências

Introdução

Introdução

FIAP

Serialização

- Em computação, é comum termos a necessidade de representar entidades em temos de um conjunto de objetos em memória;

- Eventualmente é necessário enviar os dados desses objetos de um computador para outro ou até mesmo armazenar esses dados em um mecanismo de memória permanente;

- Para realizar a cópia de um objeto podemos utilizar a sua mesma representação em memória (tipo de dado previamente definido) ou então utilizar uma outra representação, para um formato texto por exemplo;

FIAP

Serialização

- Um objeto é construído utilizando uma classe como base. Portanto, um objeto do tipo pessoa deve conter as informações relacionadas aos atributos: Nome, Endereço, Telefone, Idade e Altura;

- Em boa parte das linguagens de programação, os atributos de um objeto ficam em memória a partir de em tipo (int, double, string) representados a partir de um formato em bytes;

- O uso do formato padrão nem sempre é interessante, quando se busca reconstruir objetos de diversos tipos, para outras aplicações rodando na mesma máquina ou em máquinas remotas. Em situações assim é conveniente armazenar objetos em um formato texto

Serialização

- Objetos do tipo pessoa podem ser representados como segue:

  João silva;rua projetada 13, 27;11964541212;35;1,78

  Jose Santos;rua projetada 14, 12;11966645552;42;1,67

  Maria Sá;rua projetada 15, 32;11944225544;24;1,91

- Veja que no formato acima, os campos são separados por “;” e não há como saber qual é o dado de cada atributo, senão pela ordem;

- Caso um novo atributo seja incluído entre telefone e idade podemos ter impacto nos dados armazenados anteriormente;

Serialização

- Uma outra opção é utilizar o formato texto XML:

```
<Pessoa>
    <Nome>João silva</Nome>
    <Endereco>rua projetada 13, 27</Endereco>
    <Telefone>11964541212</Telefone>
    <Idade>35</Idade>
    <Altura>1,78</Altura>
</Pessoa>
<Pessoa>
    <Nome>Jose Santos</Nome>
    <Endereco>rua projetada 14, 12</Endereco>
    <Telefone>11966645552</Telefone>
    <Idade>42</Idade>
    <Altura>1,67</Altura>
</Pessoa>
```

FIAP

Serialização

- A manipulação dos dados utilizando o formato XML permite muitos ganhos;
- É possível incluir um novo atributo sem que seja gerado um impacto nos dados previamente armazenados;
- É possível utilizar mecanismos prontos, denominados parsers, que manipulam o conteúdo de um XML produzido a partir de um objeto arbitrário;

Serialização

- De forma semelhante podemos utilizar o formato texto JSON:

```
{"Pessoas":[
    { "Nome": "João silva",
    "Endereço": [ "Rua
    projetada 13", 27 ],
    "Telefone":11964541212,
    "Idade": 35, "Altura":
    1.78 },
    { "Nome": "Jose Santos",
    "Endereço": [ "rua
    projetada 14", 12 ],
    "Telefone":11966645552,
    "Idade": 42, "Altura":
    1.67 },
    { "Nome": "Maria Sa",
    "Endereço": [ "Rua
    projetada 15", 32 ],
    "Telefone":11944225544,
    "Idade": 24, "Altura":
    1.91 },
]}
```

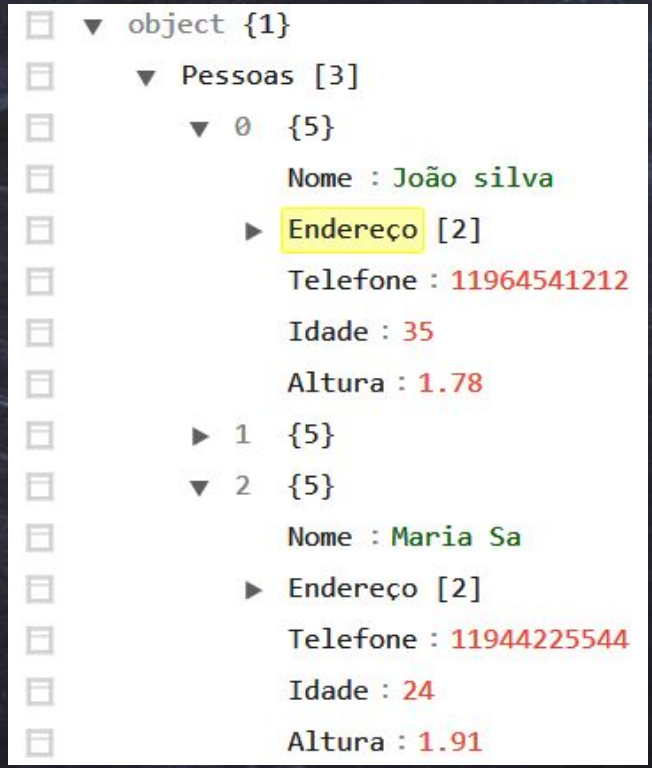
```
▼ object {1}
  ▼ Pessoas [3]
    ▼ 0 {5}
        Nome : João silva
      ▶ Endereço [2]
        Telefone : 11964541212
        Idade : 35
        Altura : 1.78
    ▶ 1 {5}
    ▼ 2 {5}
        Nome : Maria Sa
      ▶ Endereço [2]
        Telefone : 11944225544
        Idade : 24
        Altura : 1.91
```

Serialização

- Tanto para XML quanto para JSON existem ferramentas genéricas para manipulação dos dados:

```
−<Pessoas>
  −<Pessoa>
      <Nome>João silva</Nome>
      <Endereco>rua projetada 13, 27</Endereco>
      <Telefone>11964541212</Telefone>
      <Idade>35</Idade>
      <Altura>1,78</Altura>
    </Pessoa>
  −<Pessoa>
      <Nome>Jose Santos</Nome>
      <Endereco>rua projetada 14, 12</Endereco>
      <Telefone>11966645552</Telefone>
      <Idade>42</Idade>
      <Altura>1,67</Altura>
    </Pessoa>
  </Pessoas>
```

Firefox Application



https://jsoneditoronline.org/

Sintaxe JSON vs XML

JSON Example

```
{"employees":[
  { "firstName":"John", "lastName":"Doe" },
  { "firstName":"Anna", "lastName":"Smith" },
  { "firstName":"Peter", "lastName":"Jones" }
]}
```

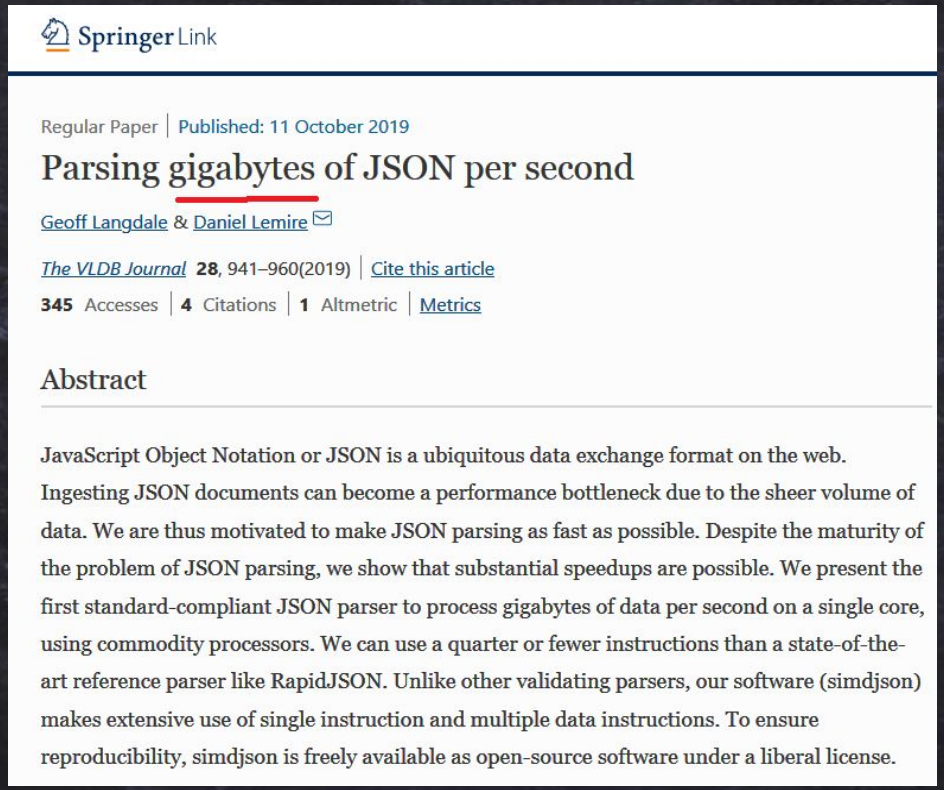
XML Example

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```

Fonte: www.w3schools.com/js/js_json_xml.asp

FIAP

JSON, ESTUDO DE DESEMPENHO...



Springer Link

Regular Paper | Published: 11 October 2019

Parsing gigabytes of JSON per second

Geoff Langdale & Daniel Lemire

The VLDB Journal **28**, 941–960(2019) | Cite this article

345 Accesses | **4** Citations | **1** Altmetric | Metrics

Abstract

JavaScript Object Notation or JSON is a ubiquitous data exchange format on the web. Ingesting JSON documents can become a performance bottleneck due to the sheer volume of data. We are thus motivated to make JSON parsing as fast as possible. Despite the maturity of the problem of JSON parsing, we show that substantial speedups are possible. We present the first standard-compliant JSON parser to process gigabytes of data per second on a single core, using commodity processors. We can use a quarter or fewer instructions than a state-of-the-art reference parser like RapidJSON. Unlike other validating parsers, our software (simdjson) makes extensive use of single instruction and multiple data instructions. To ensure reproducibility, simdjson is freely available as open-source software under a liberal license.

Fonte: https://link.springer.com/article/10.1007/s00778-019-00578-5

FIAP

Introdução NoSQL orientado a documento

- Um documento pode ser uma coleção de atributos e valores;
- Um atributo pode ser multi-valorado;
- Não possuem esquema/estrutura;
- Permitem o armazenamento de dados semi estruturados
- Alguns sistemas que suportam esse tipo de armazenamento são: MongoDb, CouchDB, RavenDb


FIAP
2.
Instalação MongoDB
Instalando MongoDB em uma máquina linux

FIAP

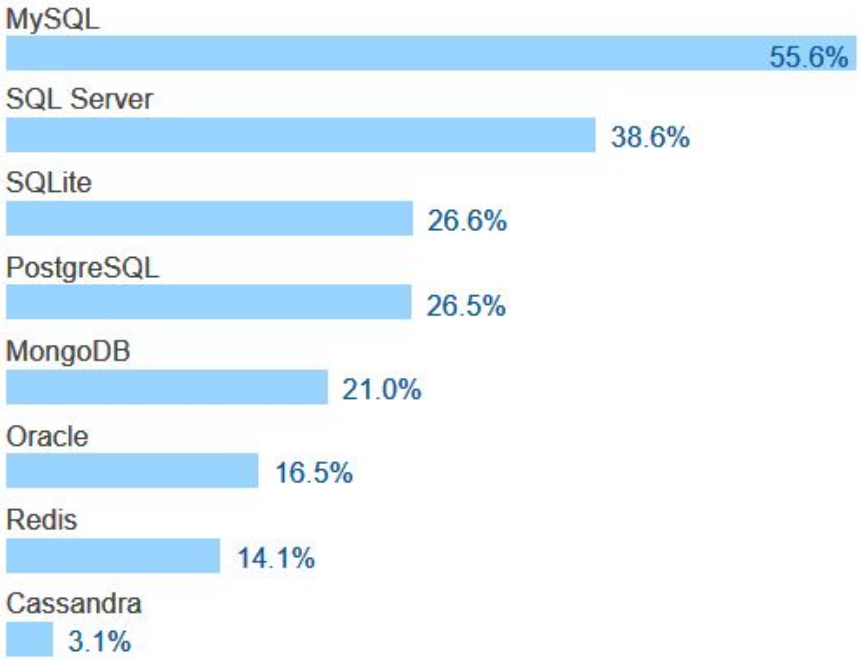
Introdução ao DB mongoDB

- O mondoDb é um banco de dados NoSQL, de código aberto, alto desempenho, escrito em linguagem de programação C++ pela empresa 10gen;
- O mongoDb permite o armazenamento de dados no formato de documentos, suportando a manipulação dos dados de diferentes formas;
- No mongoDB, os dados são armazenados em documentos BSON, um formato de arquivo semelhante ao formato JSON;
- O emprego de mongoDB é uma opção interessante em aplicações online, principalmente em situações de baixa latência e alta disponibilidade de conjuntos expressivo de dados;

Pesquisas stackoverflow para DBs

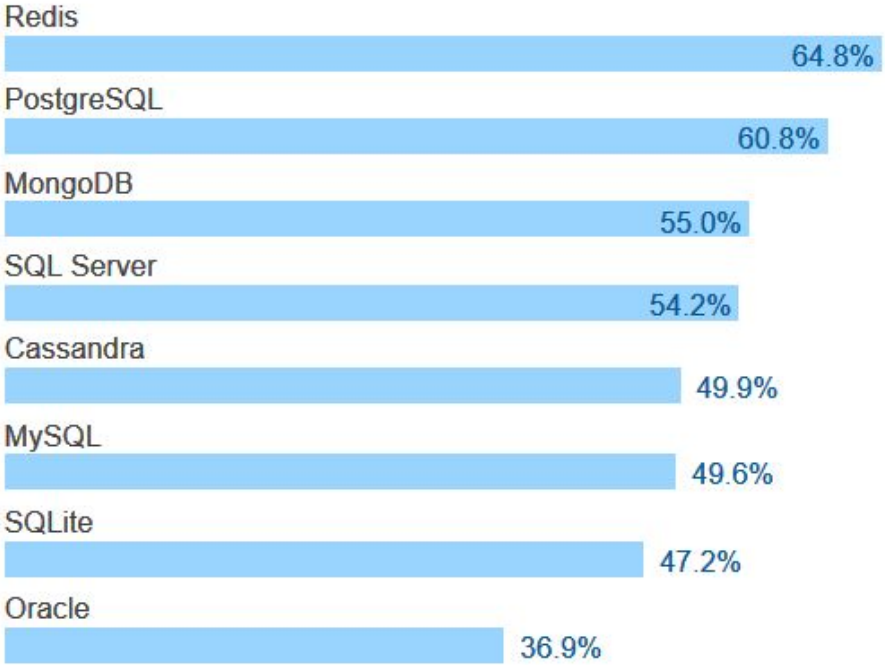
Databases

% of This Category



Most Loved, Dreaded, and Wanted Databases

Loved



Instalando o Serviço MongoDB

- Acesso ao servidor cloud (free):
  https://www.mongodb.com/cloud/atlas

- Atualize as informações de pacotes no linux:

```
$ sudo apt-get update
```

- Faça a instalação dos pacotes do DB Mongo:

```
$ sudo apt install -y mongodb
```

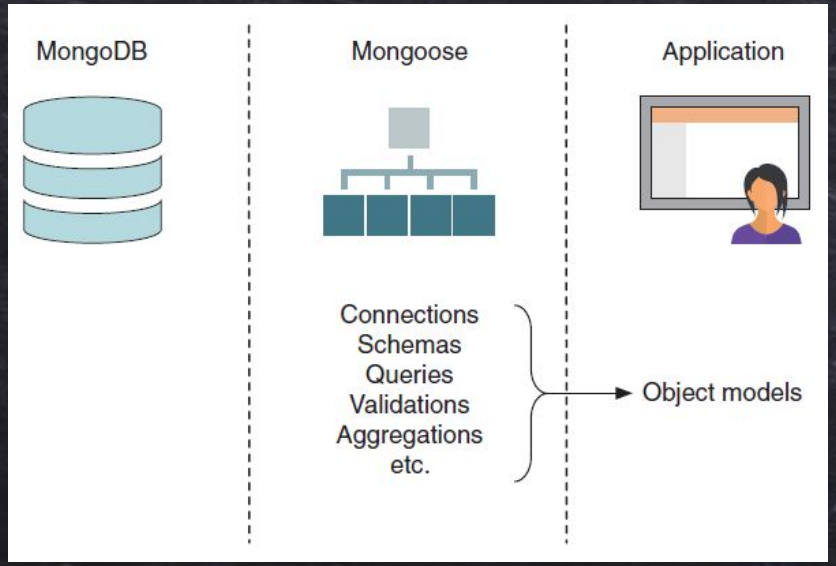
FIAP

MONGODB – PRINCIPAIS CARACTERÍSTICAS

- Um banco de dados (database) contem coleções (collections) que contem documentos (documents);
- Cada documento pode ter diferentes quantidades de campos (fields);
- O tamanho e o conteúdo de cada documento pode ser diferente;
- A estrutura do documento é fortemente ligado à estrutura de classes do sistema definido pelo time de desenvolvimento (os desenvolvedores entendem o modelo como um par chave-valor);
- A definição dos campos (fields) de cada linha de registro (aqui denominado documento) é feito em tempo de voo;
- O modelo de dados ganha expressiva flexibilidade, permitindo definir estruturas mais complexas de forma simples;

MongoDB – principais características

- A partir da versão 4.0, mongodb implementa um mecanismo de transação para múltiplos documentos que garante ACID.
- Outras características envolvem a definição do mongoose;



- Mongoose permite um balanceamento entre alguma validação para

MONGODB – PRINCIPAIS CARACTERÍSTICAS

- Banco de dados relacional, onde existe o conceito de linhas e colunas:

Table 1.1 An example of rows and columns in a relational database table

firstName	middleName	lastName	maidenName	nickname
Simon	David	Holmes		Si
Sally	June	Panayiotou		
Rebecca		Norman	Holmes	Bec

- Banco de dados mongoDB, uma linha corresponde a um documento:

Table 1.2 Each document in a document database defines and holds the data, in no particular order.

firstName: "Simon"	middleName: "David"	lastName: "Holmes"	nickname: "Si"
lastName: "Panayiotou"	middleName: "June"	firstName: "Sally"	
maidenName: "Holmes"	firstName: "Rebecca"	lastName: "Norman"	nickname: "Bec"

MONGODB – PRINCIPAIS CARACTERÍSTICAS

- Exemplo de um documento armazenado no mongodb:

```
{
   "firstName" : "Simon",
   "lastName" : "Holmes",
   _id : ObjectId("52279effc62ca8b0c1000007")
}
```

_id – campo com número único de 12 bytes hexadecimal:

4 bytes timestamp,

3 bytes machine id,

2 bytes process id,

3 bytes incrementer

Servidor MongoDB

- Atualize as informações de pacotes no linux:

  $ sudo apt-get update

- Faça a instalação dos pacotes do DB Mongo:

  $ sudo apt install –y mongodb

- Com o servidor ligado, acesse-o utilizando o cliente "mongo" na porta 27017:

  $ mongo

  >

- Obtenha informações do serviço:

  > db.serverCmdLineOpts()

- Para desconectar do DB execute o comando:

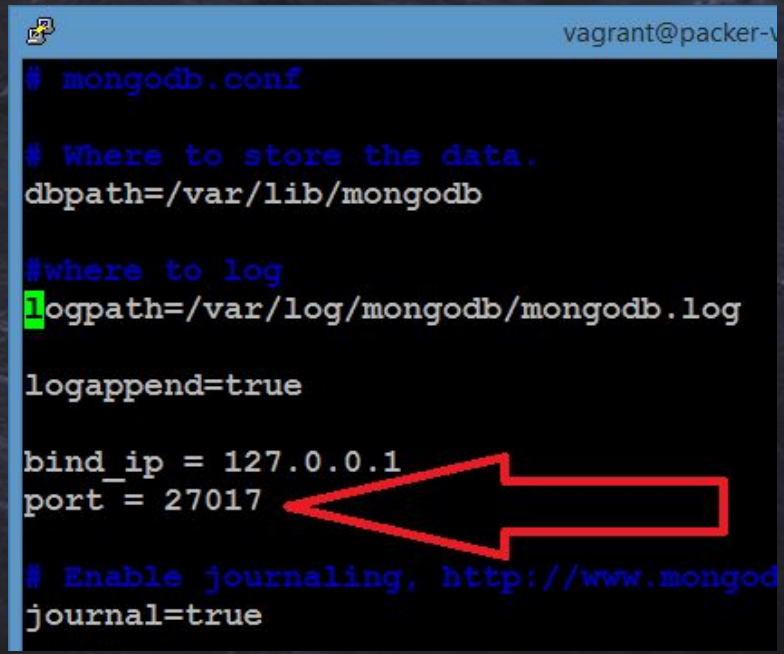
  $ exit

FIAP

Servidor MongoDB

- As configurações do servidor mongoDB ficam no arquivo mongo.config.
- Para acessar mongo.config:

  $ sudo vim /etc/mongodb.conf
- Altere a porta para 27017:



- Reinicialize o servidor mongoDb: $ sudo /etc/init.d/mongodb restart

MongoDb – Informações e definição de db

- Podemos desligar/ligar o servidor mongodb utilizando os comandos abaixo:

  $ sudo /etc/init.d/mongodb stop

  $ sudo /etc/init.d/mongodb start

  $ sudo /etc/init.d/mongodb restart

- Com o servidor ligado, acesse-o utilizando o cliente "mongo" na porta 27017:

  $ mongo –port 27017

- O comando "Use" utiliza um database existente ou cria um novo database caso ele não exista:

  > use Cliente

  switched to db Cliente

  > use Professor

  switched to db Professor

MongoDb – Informações e definição de db

- Criando collections:

  > db.createCollection("Colecao01")

- Exibir coleções:

  > show collections

- Exibir databases:

  > show databases

MongoDB – Incluir (CRUD)

✘ A inclusão de um novo registro é feita utilizando o comando insert, ex:

um documento por vez:

```
db.Cliente.insert([
    {
        ClienteId: 1,
        NomeCliente: 'Pedrao',
        Idade: 50
    }
])

db.Cliente.insert([
    {
        ClienteId: 2,
        NomeCliente: 'Tiao',
        Idade: 30
    }
])
```

um conjunto de documentos:

```
db.Cliente.insert([
    {
        ClienteId: 3,
        NomeCliente: 'Katia',
        Idade: 35
    },
    {
        ClienteId: 4,
        NomeCliente: 'Sebastiao',
        Idade: 72
    },
    {
        ClienteId: 5,
        NomeCliente: 'Pedro',
        Idade: 87
    },
    {
        ClienteId: 6,
        NomeCliente: 'Fabiana',
        Idade: 18
    },
])
```

MongoDB – Incluir (CRUD)

- Método find() é utilizado para consultas, e o método pretty() é utilizado para formatar os resultados de uma forma mais amigável, ex:

```
>db.Cliente.find()

>db.Cliente.find().pretty()
```

- Uso de AND e OR:

```
db.Cliente.find(
    {
        $and: [
            {"ClienteId":4}, {"NomeCliente": "Sebastiao"}
        ]
    }
).pretty()
```

```
db.Cliente.find(
    {
        $or: [
            {"ClienteId":4}, {"NomeCliente": "Pedro"}
        ]
    }
).pretty()
```

MongoDB – Incluir (CRUD)

- Lógica maior, menor, menor igual, etc:

Menor igual à 30:

```
db.Cliente.find(
    {
    "Idade":{$lte:30}
    }
).pretty()
```

Menor **que** 30:

```
db.Cliente.find(
    {
    "Idade":{$lt:30}
    }
).pretty()
```

Maior igual à 72:

```
db.Cliente.find(
    {
    "Idade":{$gte:72}
    }
).pretty()
```

Maior que 72:

```
db.Cliente.find(
    {
    "Idade":{$gt:72}
    }
).pretty()
```

Not equal to "Fabiana":

```
db.Cliente.find(
    {
    "NomeCliente":{$ne:"Fabiana"}
    }
).pretty()
```

MongoDB – Incluir (CRUD)

- Lógica AND e OR juntas:

(Idade > 30) AND (Nome == Katia OR Nome == Seabastiao OR Nome == Tiao):

```
db.Cliente.find({
    "Idade": {$gt:30},
    $or: [{"NomeCliente": "Katia"},
          {"NomeCliente": "Sebastiao"},
          {"NomeCliente": "Tiao"}]
}).pretty()
```

(Idade > 30) AND (Nome == Katia OR Nome == Seabastiao OR Nome == Tiao) AND (ClientId < 4):

```
db.Cliente.find({
    "Idade": {$gt:30},
    $or: [{"NomeCliente": "Katia"},
          {"NomeCliente": "Sebastiao"},
          {"NomeCliente": "Tiao"}],
    "ClienteId": {$lt:4},
}).pretty()
```

MongoDB – Incluir (CRUD)

- Atualização de dados:

```
db.Cliente.update(
    {'NomeCliente':'Katia'},
    {$set:
        {'NomeCliente':'Katia Rodrigues'}
    }
)
```

- O primeiro parâmetro pode ser utilizada lógica combinada;

MongoDB – Incluir (CRUD)

- Para remover dados deve ser utilizado o comando remove(), ex:

Remove apenas todos os elementos:

```
db.Cliente.remove()
```

Remove apenas os elementos condicionados:

```
db.Cliente.remove(
    {'NomeCliente':'Katia Rodrigues'}
)
```

Remove utilizando lógica AND:

```
db.Cliente.remove({
    "Idade": {$gt:30},
    "ClienteId": {$lt:4}
})
```

MongoDB – Index

- Para criar índices deve ser utilizado o método ensureInex({KEY:1}), ex:

Criando índice do campo ClienteID ascendente:

```
db.Cliente.ensureIndex(
    {"ClienteId":1}
)
```

Criando índice do campo ClienteID descendente:

```
db.Cliente.ensureIndex(
    {"NomeCliente":-1}
)
```

Consultando índices:

> db.system.indexes.find()

> db.Cliente.getIndexes()

Acessando MongoDB com Python

- Baixe o módulo para acessar o DB:

```
python -m pip install pymongo
```

- É necessário importar o módulo pymongo e MongoClient
- O driver disponibiliza classes para clientes, servidor, DataBase, etc...
- Abaixo um código de exemplo:

```python
import pymongo
from pymongo import MongoClient

client = MongoClient(host="localhost", port=27017)
db = client["python"]
emp = db.emp
result_one = emp.find_one()
result_all = emp.find()
print()
print("Imprimindo campos dos resultados encontrado:")
for r in result_all:
   print (r)
print()
print("Print result_one:")
print (result_one)
```

Acessando MongoDB com C#

- Baixe o driver para acessar o DB, a partir de:

  https://github.com/mongodb/mongo-csharp-driver/releases

- É necessário adicionar como referencia duas DLLs (MongoDB BOSON e MogoDbdriver) e utilizar os namespaces:

```
using MongoDB.Bson;
using MongoDB.Driver;
using MongoDB.Driver.Builders;
using MongoDB.Driver.GridFS;
using MongoDB.Driver.Linq;
```

- O driver disponibiliza classes para clientes, servidor, DataBase, etc...

```
MongoClient cliente = new MongoClient("mongodb://localhost");
MongoServer servidor = client.GetServer();
MongoDatabase dbmongo = servidor.GetDatabase("Teste");
```

- Existe um site com exemplos passo a passo (Acessos e CRUD):

  https://www.c-sharpcorner.com/UploadFile/87b416/getting-started-mongodbwithcsharp/

Referências

- SIMON HOLMES, CLIVE HARBER. Getting MEAN WITH MONGO, EXPRESS, ANGULAR, AND NODE, Manning Publications Co, Shelter Island, 2011.
- Gaurav Vaish. Getting Started with NoSQL, packt publiching, 2013.


OBRIGADO!