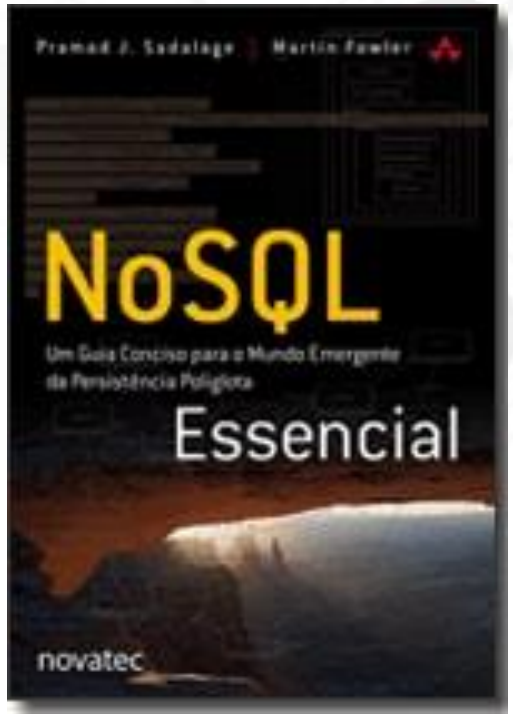
# Banco de Dados não Relacionais - NoSQL

PUC Minas Virtual

# Banco de dados de Documentos

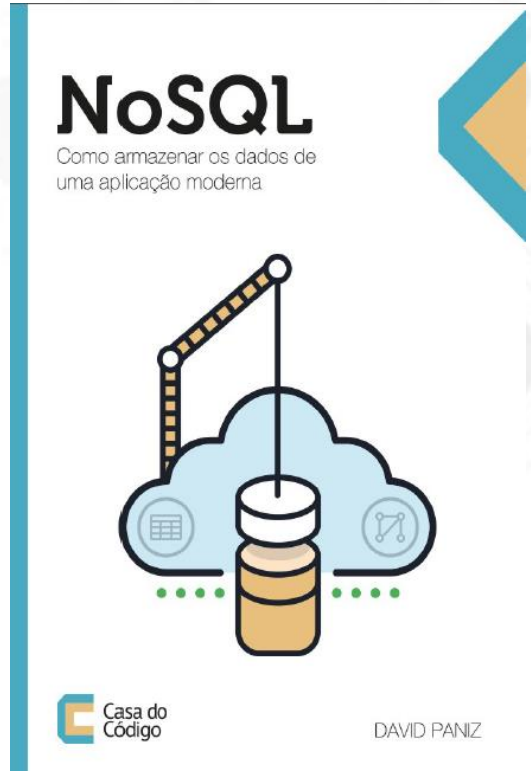
# Principais Referências

Pramod J.; Sadalage, Martin Fowler. **NoSQL Essencial: Um Guia Conciso para o Mundo Emergente da Persistência Poliglota**. Novatec Editora, 2013.



# Principais Referências

Paniz, David. NoSQL: **Como armazenar os dados de uma aplicação moderna**. Casa do Código, 2017.



# Principais Referências

Boaglio, Fernando. MongoDB: **Construa novas aplicações com novas tecnologias**. Casa do Código, 2017.

# Banco de dados de documento

O MongoDB é um projeto Open Source (Linux, Mac e Windows)

A linguagem utilizada para manipulação dos dados é JavaScript.



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Banco de dados de documento

Para instalação do MongoDB para Windows siga o tutorial (também disponível em outras plataformas):

https://docs.mongodb.com/v3.0/tutorial/install-mongodb-on-windows/

# Banco de dados de documento

Antes de iniciar, crie o diretório C:\data\db, usando o PowerShell: **md \data\db**

Para iniciar o MongoDB, execute no prompt de comando para iniciar o servidor:

C:\Program Files\MongoDB\Server\3.6\bin\mongod.exe

# Banco de dados de documento

E para conectar, abra uma nova janela no prompt e digite (abrir o cliente):

C:\Program Files\MongoDB\Server\3.6\bin\mongo.exe

# Criando um documento

No MongoDB precisamos criar uma instância para adicionar os documentos.

Com o MongoDB instalado o comando **`show dbs`** mostra quais são os banco já criados.

Comando para criar o primeiro banco: **`use nome_do_banco`** (se não existir, ele irá criar)



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Criando um documento

Após criar um novo banco de dados, precisamos criar uma nova coleção (equivalente a uma tabela no SGBDR).

Para criar uma nova coleção, basta adicionar um documento vazio utilizando o comando `insert`.



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Criando um documento

Iremos chamar a função `insert` a partir do nosso objeto base db e do nome da coleção, veja:

```
db.albuns.insert({})
```

Nome da coleção (sendo criada agora)

Corpo do documento (vazio)

Use o comando "`show collections`" para visualizar as coleções criadas

PUC Minas Virtual

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Criando um documento

Da mesma forma que utilizamos o `insert` para inserir um novo documento, usamos o comando `find` para retornar documentos na coleção.

Retorna o _id do documento, equivalente a chave primária de uma linha no SGBDR (mas é gerenciado internamente pelo mongoDB)

```
db.albuns.find({})

{“_id” : ObjectId (“5921f80828b4776a45fd6e64”)}
```



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Criando documentos

# Criando um documento

No MongoDB um documento é criado utilizando JavaScript. Portanto, o documento deve ser um objeto JavaScript, ou um JSON.

```
db.albuns.insert({"nome" : "The Dark Side Of The Moon ",
"data" : new Date(1973, 3, 29)})
```

Chave

Valores (strings, date, etc) – no tipo Date, o mês começa do zero.

Utilizamos { e } para criar um documento e definimos as chaves e seus respectivos valores

PUC Minas Virtual

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Criando um documento

## Criando alguns documentos:

```
db.albuns.insert({"nome" : "Master of Puppets", "dataLancamento" : new
Date(1986, 2, 3), "duracao" : 3286})
```

```
db.albuns.insert({"nome" : "...And Justice for All", "dataLancamento" :
new Date(1988, 7, 25), "duracao" : 3929})
```

```
db.albuns.insert({"nome" : "Among the Living", "produtor" : "Eddie
Kramer"})
```



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Criando um documento

## Criando alguns documentos:

```
db.albuns.insert({"nome" : "Nevermind", "artista" : "Nirvana",
"estudioGravacao" : ["Sound City Studios", "Smart Studios (Madison)"],
"dataLancamento" : new Date(1992, 0, 11)})
```

```
db.albuns.insert({"nome" : "Reign in Blood", "dataLancamento" : new
Date(1986, 9, 7), "artista" : "Larry Carroll", "duracao" : 1738})
```



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Criando um documento

## Criando alguns documentos:

```
db.albuns.insert({"nome" : "The Dark Side Of The Moon", "artista" :
"Pink Floyd", "dataLancamento" : new Date(1973, 3, 29)})
```

```
db.albuns.insert({"nome" : "Seventh Son of a Seventh Son", "artista" :
"Iron Maiden", "produtor" : "Martin Birch", "estudioGravacao" :
"Musicland Studios", "dataLancamento" : new Date(1988, 3, 11)})
```



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Buscando um documento

Após a inserção dos documentos no banco, vamos realizar consultas sobre estes documentos.

Vimos que podemos usar o comando "`find`" para encontrar um documento passando como argumento {} (criteria).



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Buscando um documento

Mas se desejar apenas listar todos os documentos inseridos, utilize: **`db.album.find()`** ou **`db.album.find().pretty()`** para exibição de forma estruturada.



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Buscando um documento

Para buscar documentos por campos específicos, podemos fazer da seguinte forma:

```
db.albuns.find({"nome" : "Seventh Son of a Seventh Son"})
```

Equivalente ao SGBDR:

```
SELECT *
FROM albuns
WHERE nome = "Seventh Son of a Seventh Son"
```



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Buscando um documento

Para buscar documentos por campos específicos, podemos fazer da seguinte forma (para exibição de forma estruturada):

```
db.albuns.find({"nome" : "Seventh Son of a Seventh Son"}).pretty()
```

```
> db.albuns.find({"nome" : "Seventh Son of a Seventh Son"}).pretty()
{
        "_id" : ObjectId("59244ae5c4375868538e4c4b"),
        "nome" : "Seventh Son of a Seventh Son",
        "artista" : "Iron Maiden",
        "produtor" : "Martin Birch",
        "estudioGravacao" : "Musicland Studios",
        "dataLancamento" : ISODate("1988-04-11T03:00:00Z")
}
```



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Buscando um documento

Este comando irá retornar uma lista de documentos que satisfazem a condição.

Se desejar retornar apenas um documento (o primeiro que satisfaz a condição ou `null` se nenhum documento for encontrado), utilize `findone()`



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Excluindo um documento

As mesma condições usada para filtrar um documento no MongoDB em uma consulta podem ser utilizadas para excluir um documento de uma coleção.

Ao invés de usar o comando `find`, utilizamos o comando `remove`.



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Excluindo um documento

Exemplo: vamos deletar o álbum "The Dark Side Of The Moon".

```
db.albuns.remove({"nome": "The Dark Side Of The Moon"})
```

Equivalente em SQL →

```
DELETE
FROM albuns
WHERE nome = " The Dark Side Of The Moon "
```

PUC Minas Virtual

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Excluindo um documento

Se desejar remover todos os documentos, utilize o comando remove sem nenhum critério:

```
db.albuns.remove({})
```



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Alterando um documento

Podemos utilizar o comando update para atualizar um documento na coleção:

```
db.albuns.update({"nome" : "Among the Living"}, {$set : {"duracao" : 3013}})
```

Agora busque pelo documento e verifique a alteração

```
db.albuns.find({"nome" : "Among the Living"})
```



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Consultas mais complexas

# Consultas mais complexas

Podemos realizar consultas mais complexas utilizando o MondoDB. Por exemplo, como encontrar um (ou mais) álbum(ns) cuja duração seja menor que um valor especificado na consulta.

Em um SGBDR, teríamos:



```
SELECT *
FROM albuns
WHERE duracao < 1800
```



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Consultas mais complexas

Há vários operadores de comparação:

| Operador | Descrição |
|---|---|
| $gt | maior que o valor específico na query. |
| $gte | maior ou igual ao valor específico na query. |
| $in | quaisquer valores que existem em um array específico em uma query |
| $lt | valores que são menores que o valor específico na query. |
| $lte | valores que são menores ou iguais que o valor específico na query |
| $ne | todos os valores que não são iguais ao valor específico na query. |
| $nin | valores que não existem em um array específico da query. |

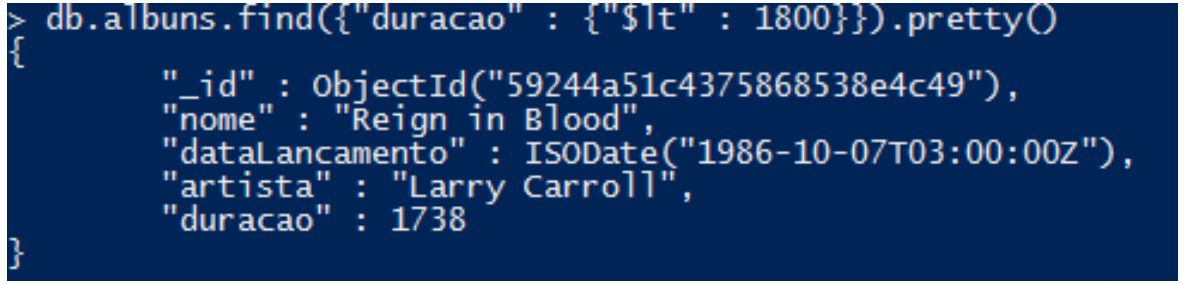


Tabela adaptada de : Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Consultas mais complexas

Por exemplo, como encontrar um (ou mais) álbum(ns) cuja duração seja menor que um valor especificado na consulta.

```
db.albuns.find({"duracao" : {"$lt" : 1800}}).pretty()
```






Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Consultas mais complexas

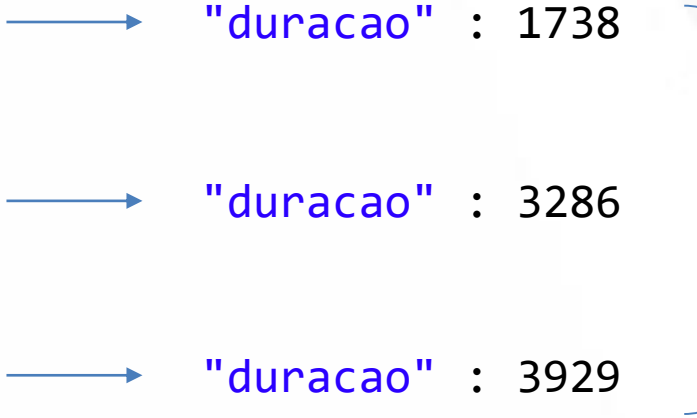
```
{"nome" : "Reign in Blood",
"dataLancamento" : new Date(1986, 9, 7),
"artista" : "Larry Carroll",
"duracao" : 1738}
```

```
{"nome" : "Master of Puppets",
"dataLancamento" : new Date(1986, 2, 3),
"duracao" : 3286}
```

```
{"nome" : "...And Justice for All",
"dataLancamento" : new Date(1988, 7, 25),
"duracao" : 3929}
```

```
{"nome" : "Among the Living",
"produtor" : "Eddie Kramer"}
```

Não possui duração




```
{"nome" : "Reign in Blood",
"dataLancamento" : new Date(1986, 9, 7),
"artista" : "Larry Carroll",
"duracao" : 1738}
```

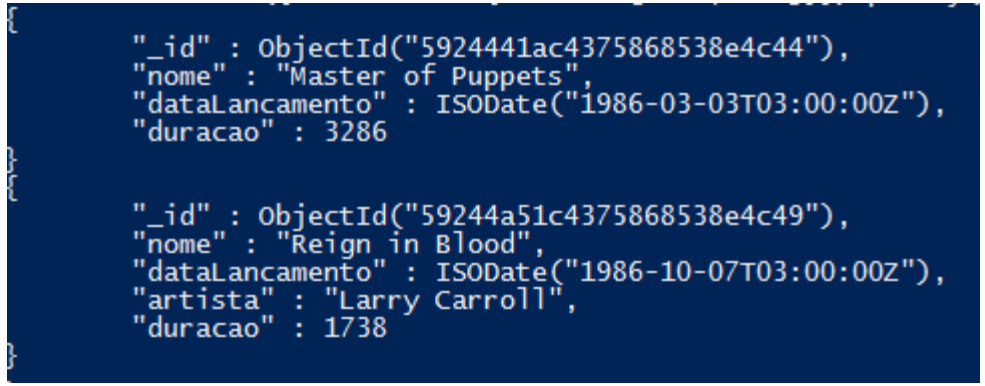


Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Consultas mais complexas

Outro exemplo, como encontrar um (ou mais) álbum(ns) cuja duração seja 1738 ou 3286.

```
db.albuns.find({"duracao" : {"$in" : [1738,3286]}}).pretty()
```




PUC Minas Virtual

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Consultas mais complexas

Exercício: retorne os álbuns que foram lançados antes de 1990.



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Consultas mais complexas

No MongoDB temos vários operadores lógicos:

| Operador | Descrição |
|---|---|
| $and | Retorna documentos com ambas as condições verdadeiras |
| $nor | Retorna documentos com ambas as condições falsas |
| $not | Inverte o resultado de uma condição |
| $or | Retorna documentos com um das condições verdadeiras. |



Tabela adaptada de : Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Consultas mais complexas

## Sintaxe da consulta com operadores lógicos:

```
{operador : [expressão 1, expressão 2, expressão n]}.
```



Critérios para a consulta



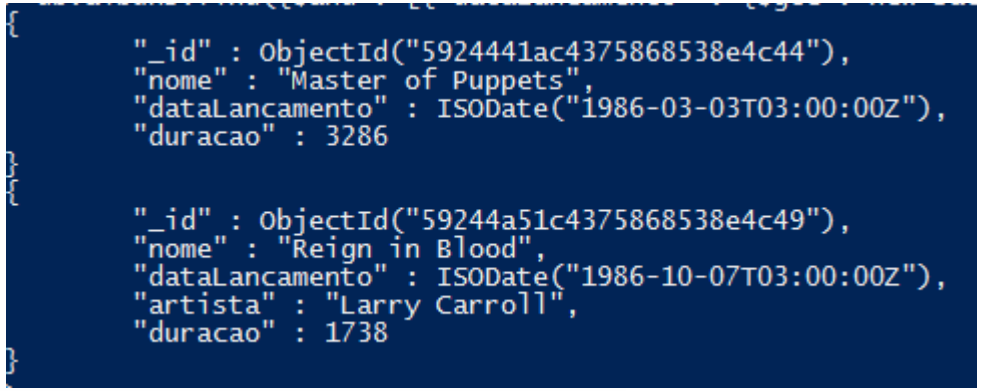
Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Consultas mais complexas

Exemplo: retornar todos os discos lançados em 1986.

```
db.albuns.find(
{$and : [{"dataLancamento" : {$gte : new Date(1986, 0, 1)}},
         {"dataLancamento" : {$lt : new Date(1987, 0, 1)}}]}
).pretty()
```

O valor deve ser maior na primeira cláusula e menor na segunda






Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Consultas mais complexas

Exemplo: retornar todos os discos lançados em 1986.

```
db.albuns.find(
{$and : [{"dataLancamento" : {$gte : new Date(1986, 0, 1)}},
         {"dataLancamento" : {$lt : new Date(1987, 0, 1)}}]}
).pretty()
```

Equivalente em SQL (SGBDR)

```
SELECT *
FROM albuns
WHERE dataLancamento >= '1986-01-01'
AND dataLancamento < '1987-01-01'
```



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Criando relacionamentos no MongoDB

# Inserindo relacionamento

No MongoDB é possível criar relacionamento entre suas coleções.

Já criamos nosso relação *Albuns* e agora vamos criar a relação *Artitas*.



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Inserindo relacionamento

Estamos criando a coleção artista e ao mesmo tempo criando diversos documentos.

```
db.artistas.insert([  {"nome" : "Metallica"},
                      {"nome" : "Megadeath"},
                      {"nome" : "Slayer"},
                      {"nome" : "Anthrax"},
                      {"nome" : "Iron Maiden"},
                      {"nome" : "Nirvana"},
                      {"nome" : "Pink Floyd"}])
```

Observe que cada documento só contem o campo nome

```
db.artistas.find().pretty()
```

Pesquise pelos artistas

PUC Minas Virtual

Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Inserindo relacionamento

## Resposta obtida:

```
> db.artistas.find().pretty()
{ "_id" : ObjectId("59247ab4c4375868538e4c4e"), "nome" : "Metallica" }
{ "_id" : ObjectId("59247ab4c4375868538e4c4f"), "nome" : "Megadeath" }
{ "_id" : ObjectId("59247ab4c4375868538e4c50"), "nome" : "Slayer" }
{ "_id" : ObjectId("59247ab4c4375868538e4c51"), "nome" : "Anthrax" }
{ "_id" : ObjectId("59247ab4c4375868538e4c52"), "nome" : "Iron Maiden" }
{ "_id" : ObjectId("59247ab4c4375868538e4c53"), "nome" : "Nirvana" }
{ "_id" : ObjectId("59247ab4c4375868538e4c54"), "nome" : "Pink Floyd" }
```



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Inserindo relacionamento

Agora precisamos criar o conceito de "chave estrangeira" dentro da coleção álbuns por meio do _id de cada artista (obtido anteriormente).

```
db.albuns.update( {"nome" : "Master of Puppets"}, {$set : {"artista_id" : ObjectId("5aeec74f9de69e42ae369bd5")}});
```

```
{ "_id" : ObjectId("5aeec74f9de69e42ae369bd5"), "nome" : "Metallica" }
```



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Inserindo relacionamento

```
db.albuns.update( {"nome" : "Among the Living"}, {$set : {"artista_id" : ObjectId("5aeec74f9de69e42ae369bd8")}} );
```

```
{ "_id" : ObjectId("5aeec74f9de69e42ae369bd8"), "nome" : "Anthrax" }
```

```
db.albuns.update( {"nome" : "Nevermind"}, {$set : {"artista_id" : ObjectId("5aeec74f9de69e42ae369bda")}} );
```

```
{ "_id" : ObjectId("5aeec74f9de69e42ae369bda"), "nome" : "Nirvana" }
```



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Inserindo relacionamento

```
db.albuns.update( {"nome" : "Reign in Blood"}, {$set : {"artista_id" : ObjectId("5aeec74f9de69e42ae369bd7")}} );
```

```
{ "_id" : ObjectId("5aeec74f9de69e42ae369bd7"), "nome" : "Slayer" }
```

```
db.albuns.update( {"nome" : "Seventh Son of a Seventh Son"}, {$set : {"artista_id" : ObjectId("5aeec74f9de69e42ae369bd9")}});
```

```
{ "_id" : ObjectId("5aeec74f9de69e42ae369bd9"), "nome" : "Iron Maiden" }
```



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Inserindo relacionamento

```
db.albuns.update( {"nome" : "...And Justice for All"}, {$set : {"artista_id" : ObjectId("5aeec74f9de69e42ae369bd5")}});
```

```
{ "_id" : ObjectId("5aeec74f9de69e42ae369bd5"), "nome" : "Metallica" }
```

```
db.albuns.update( {"nome" : "The Dark Side Of The Moon"}, {$set : {"artista_id" : ObjectId("5aeec74f9de69e42ae369bdb")}});
```

```
{ "_id" : ObjectId("5aeec74f9de69e42ae369bdb"), "nome" : "Pink Floyd" }
```



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Recuperando informações

A partir deste momento, podemos trabalhar buscando informações em ambas as coleções.

Para isso iremos utilizar a linguagem JavaScript para formular nossa consultas.



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Recuperando informações

```
var artista = db.artistas.findOne({"nome" : "Metallica"});
```

Criamos uma variável artista que recebe o objeto artista cujo nome é igual a “metálica”

```
var albuns = db.albuns.find({"artista_id" : artista._id})
```

Criamos uma variável álbuns. Esta variável armazena todos os álbuns do artista._id (armazenado anteriormente)

Observe que artista é um objeto que contém os campos “nome” e “_id”

Digite a variável “albuns” na tela e veja o resultado.



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Considerações sobre relacionamentos

- Apesar de mapearmos um relacionamento entre documentos no MongoDB, **as restrições de integridade referencial** (presentes no modelo relacional) **não** se aplicam a este banco.
- O controle sobre tal característica é de responsabilidade da aplicação.



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Documentos aninhados no MongoDB

# Documentos aninhados

Em muitas situações, pode ser mais interessante (por questões de performance por exemplo) aninhar um documento dentro do outro, como por exemplo:

```
{"nome" : "Master of Puppets",
"dataLancamento" : new Date(1986, 2, 3),
"duracao" : 3286
"artista" : {"nome" : "Metallica"}}
```

Ao invés de criar a coleção artistas, embutindo suas informações dentro do documento do álbum.



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Documentos aninhados

Incialmente pode parecer estranho ter que repetir a mesma informação do artista dentro de vários documentos.

Entretanto, isto ajudará na performance da consulta (a mesma ficará mais simples).



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Documentos aninhados

É possível realizar buscas pelos atributos dos subdocumentos:

```
db.albuns.find({"artista" : {"nome" : "Blind Guardian"}}).pretty();
```

Esta consulta retornar todos os álbuns do “Blind Guardian”, que é um atributo do subdocumento de album



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Documentos aninhados

Resolvemos um problema de performance duplicando informações dos artistas (que possuem apenas dois campos).

No caso de artistas possuírem muitos campos, uma solução parcial seria duplicar apenas o *nome* e o *_id* de cada artista e manter as demais informações em coleções separadas.



Referências: Paniz, David. NoSQL: Como armazenar os dados de uma aplicação moderna. Casa do Código, 2017.

# Dicas úteis

# Robo 3T

## Software para gerenciamento do MongoDB via interface gráfica

Robo 3T (antigo RoboMongo)
https://robomongo.org/download

Clicar em download e escolher "Download installer for Windows 64-bit"

# PyMongo

- Biblioteca para acesso e manipulação do MongoDB em Python

Após a instalação do Python 3.6.1, abra o prompt de comando e execute o comando: "python".
O console do Python será executado, e então execute o comando abaixo para instalação do Pymongo.

pip install pymongo

Referência:https://api.mongodb.com/python/2.7.2/installation.html

# PyMongo

```
import pymongo

client = pymongo.MongoClient("localhost", 27017)
db = client.aula

albuns = db.albuns.find()

file = open("C:\\Users\\albuns.txt", "a")

for item in albuns:
    nome = item["nome"]
    file.write(nome + '\n’)

file.close()
```

Importação da biblioteca do pymongo

Conexão com o banco de dados (deve-se iniciar o serviço antes na pasta de instalação do mongo)

Recupera todos os documentos

Cria arquivo em disco para salvar os dados recuperados

Recupera o campo “nome” de cada documento

# Quando usar e não usar

# Situações apropriadas para o uso

Registro de eventos: Diversos aplicativos tem necessidades de salvar logs de eventos. O banco de dados de documento pode atuar como um repositório central de eventos. Ideal principalmente quando há mudanças constantes no tipo de dados obtido pelo evento.



Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

# Situações apropriadas para o uso

Sistema de gerenciamento de conteúdo Web: por identificar o padrão JSON, é muito apropriado para aplicativos de publicações de websites, trabalhando com comentário de usuário, perfis e documentos visualizados.



Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

# Situações apropriadas para o uso

Comércio eletrônico: Muito último para armazenar informações de produtos que possuem diferentes características.

Análise de dados: fácil para armazenar visualizações de paginas e visitantes em tempo real.



Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

# Situações para *não* usar

Transações complexas: Operações atômicas em múltiplos documentos podem não ser ideias para este tipo de banco (controlar falhas na transferência entre duas contas bancárias, por exemplo).



Referências: Pramod J.; Sadalage, Martin Fowler. NoSQL Essencial. 2013

# Para saber mais

Consulte o site do desenvolvedor:

The MongoDB 3.0 Manual

https://docs.mongodb.com/v3.0/#getting-started

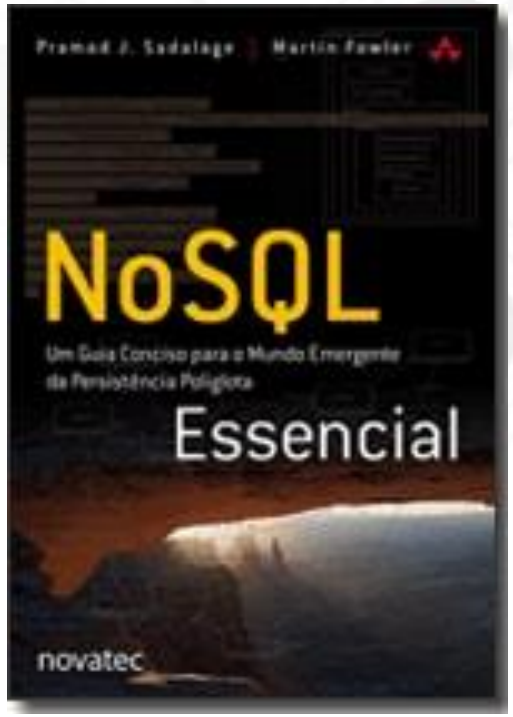

**Para saber mais: Consulte o site dos desenvolvedores**

**The MongoDB 3.0 Manual**

https://docs.mongodb.com/v3.0/#getting-started

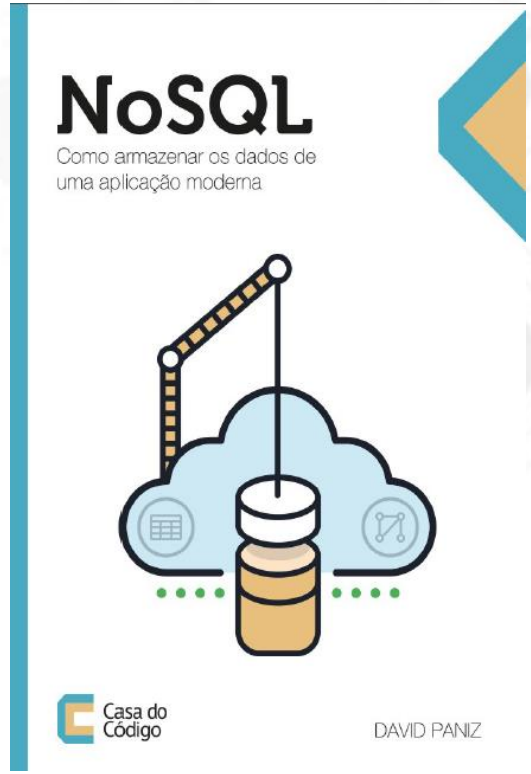
# Principais Referências

Pramod J.; Sadalage, Martin Fowler. **NoSQL Essencial: Um Guia Conciso para o Mundo Emergente da Persistência Poliglota**. Novatec Editora, 2013.



# Principais Referências

Paniz, David. NoSQL: **Como armazenar os dados de uma aplicação moderna**. Casa do Código, 2017.



# Principais Referências

Boaglio, Fernando. MongoDB: **Construa novas aplicações com novas tecnologias**. Casa do Código, 2017.

PUC Minas

Virtual