

Banco de Dados II



O que é um banco de dados de documentos?

Um banco de dados de documentos (também conhecido como banco de dados orientado a documentos ou armazenamento de documentos) é um banco de dados que armazena informações em documentos.

```
{  
  "_id": "5cf0029caff5056591b0ce7d",  
  "firstname": "Jane",  
  "lastname": "Wu",  
  "address": {  
    "street": "1 Circle Rd",  
    "city": "Los Angeles",  
    "state": "CA",  
    "zip": "90404"  
  },  
  "hobbies": ["surfing", "coding"]  
}
```

Em vez de armazenar dados em linhas e colunas fixas, os bancos de dados de documentos usam documentos flexíveis.

O que são documentos?

Os documentos armazenam dados em pares campo-valor. Os valores podem ser de vários tipos e estruturas, incluindo strings, números, datas, matrizes ou objetos. Os documentos podem ser armazenados em formatos como JSON, BSON e XML.

```
{  
    name: "sue",  
    age: 26,  
    status: "A",  
    groups: [ "news", "sports" ]  
}
```



The diagram illustrates a JSON object structure. It consists of a left brace '{' followed by four key-value pairs. Each pair is preceded by a black arrow pointing to the right, labeled 'field: value'. The keys are 'name', 'age', 'status', and 'groups'. The values are '\"sue\"', '26', '\"A\"', and '[\"news\", \"sports\"]' respectively.

BSON Types

Type	Number	Alias	Notes
Double	1	"double"	
String	2	"string"	
Object	3	"object"	
Array	4	"array"	
Binary data	5	"binData"	
Undefined	6	"undefined"	Deprecated.
ObjectId	7	"objectId"	
Boolean	8	"bool"	
Date	9	"date"	
Null	10	"null"	
Regular Expression	11	"regex"	
DBPointer	12	"dbPointer"	Deprecated.
JavaScript	13	"javascript"	
Symbol	14	"symbol"	Deprecated.
JavaScript code with scope	15	"javascriptWithScope"	Deprecated in MongoDB 4.4.
32-bit integer	16	"int"	
Timestamp	17	"timestamp"	
64-bit integer	18	"long"	
Decimal128	19	"decimal"	
Min key	-1	"minKey"	
Max key	127	"maxKey"	

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

BSON (Binary JSON)

BSON é uma representação binária do JSON, utilizada pelo MongoDB para armazenar seus documentos. É utilizado devido a:

- Rápida escaneabilidade (fast scanability), ou seja, torna possível varrer um documento procurando um valor rapidamente.
- Novos tipos de dados (Date, ObjectId, Binary Data)

Todo documento tem um campo chamado `_id` obrigatoriamente.

Por padrão, é um campo do tipo `ObjectId`, mas pode ser definido como qualquer outro tipo de dado.

<https://bsonspec.org/>

Características do MongoDB

O MongoDB foi projetado para ser [escalável](#).

Por isso, algumas funcionalidades NÃO foram incorporadas ao seu sistema, como:

- Junções (JOINS) entre coleções
- Transações ACID [Elmasri and Navathe 2014]

Características do MongoDB

E como viver em um mundo sem junções e transações?

- Aninhando documentos (embedded documents)
- Criando links artificiais (textual, não é uma chave estrangeira)
- Lembrando que as operações de escritas são atômicas no nível de um único documento, mesmo quando uma operação modificar vários documentos.

Problemas?

O que pode acontecer com os nossos dados?

- A consistência pode ser eventual
- Pode haver duplicação e inconsistência entre campos usados como links de documentos

Em compensação, ganhamos desempenho e **escalabilidade**.

A **aplicação** torna-se responsável pela integridade dos **dados duplicados**

MongoDB x SQL

MongoDB	SQL
Documentos	Linhas
Campos/chaves/atributos	Colunas
Coleções	Tabelas

Operações básicas

CRUD	SQL	MongoDB
Create	Insert	Insert
Read	Select	Find
Update	Update	Update
Delete	Delete	Remove

Não existe uma linguagem separada para descrever as operações de CRUD no MongoDB.

As operações existem como métodos/funções dentro da API.

<https://www.mongodb.com/docs/manual/crud/>

(a) Create

```
db.users.insertOne( {  
  name: "sue",  
  age: 26,  
  status: "pending"  
})
```

(b) Read

```
db.users.find(  
  { age: { $gt: 18 } },  
  { name: 1, address: 1 }  
).limit(5)
```

collection
query criteria
projection
cursor modifier

(c) Update

```
db.users.updateMany(  
  { age: { $lt: 18 } },  
  { $set: { status: "reject" } })
```

collection
update filter
update action

(d) Delete

```
db.users.deleteMany(  
  { status: "reject" })
```

collection
delete filter

Criando uma base de dados no Mongodb Compass

use <“basededados”> // no ambiente de shell



Case-sensitive

db.createCollection(“tabela”) //Para a base existir é necessário uma collection

The screenshot shows the MongoDB Compass interface. In the top navigation bar, 'Databases' is selected. Below it, there's a 'Create database' button with a plus sign and the number '2'. The main area displays a list of databases: 'admin', 'alura_serie', 'aula_nosql' (which has a 'tabela' collection), 'biblioteca', 'config', 'local', and 'showbank'. The 'aula_nosql' database is expanded to show its collections. At the bottom, there's a 'MONGOSH' section with some command-line output.

Create Database

Database Name: aula_nosql

Collection Name: aluno

Time-Series

Time-series collections efficiently store sequences of measurements over a period of time. [Learn More](#)

Additional preferences (e.g. Custom collation, Capped, Clustered collections)

Create Database 3

The screenshot shows the MongoDB Compass interface with the 'Collections' tab selected. It lists the 'aula_nosql' database, which contains the 'tabela' collection. The 'tabela' collection has a storage size of 4.10 kB, 0 documents, and an average document size of 0 B. Below this, there's a 'Create Collection' dialog. The 'Collection Name' field is filled with 'tabela'. There are also sections for 'Time-Series' (with a link to learn more) and 'Additional preferences'.

Create Collection

Collection Name: tabela

Time-Series

Time-series collections efficiently store sequences of measurements over a period of time. [Learn More](#)

Additional preferences (e.g. Custom collation, Capped, Clustered collections)

Create Collection 3

Para saber mais: restrições de criação - Coleção

Assim como para criar um banco de dados, também existem restrições para se criar uma coleção aqui no MongoDB, que são:

- Os nomes das coleções devem começar com um sublinhado ou um caractere de letra.
- Não podem:
 - ✓ Conter o \$.
 - ✓ Ser uma string vazia (por exemplo "",).
 - ✓ Conter o caractere nulo.
 - ✓ Começar com o system.prefixo. (Reservado para uso interno).

Excluindo uma base de dados no Mongodb Compass

- db.collection.drop()
- db.dropDatabase()

```
>_MONGOSH
>
>
> db.tabela.drop()
< true
> db.dropDatabase()
< { ok: 1, dropped: 'aula_nosql' }
aula_nosql>
```

Inserir documentos

```
db.users.insertOne(  
  {  
    name: "sue",  
    age: 26,  
    status: "pending"  
  })
```

db.collection.insertOne()

```
db.aluno.insertOne({  
  "CPF": "12345647798",  
  "nome": "Manoela Castro",  
  "endereco": "Rua 10 numero 40 bairro 3",  
  "telefone": "3322-3326"  
});
```

db.collection.insertMany()

```
db.aluno.insertMany([  
  {"CPF": "12345647798",  
   "nome": "Carlos Pereira",  
   "endereco": "Rua 17 numero 42 bairro 3",  
   "telefone": "3322-3328"},  
  {"CPF": "5656565655",  
   "nome": "Maria Ribas",  
   "endereco": "Rua 11 numero 41 bairro 5",  
   "telefone": "3322-3328"}])
```

Inserir documentos

The screenshot illustrates the process of inserting documents into the 'aluno' collection within the 'biblioteca' database using the MongoDB Compass interface.

- Collection Selection:** The 'aluno' collection is selected in the left sidebar under the 'biblioteca' database. A blue box labeled '1' highlights this selection.
- Insert Data:** The 'ADD DATA' button is clicked, opening the 'Insert Document' dialog. A blue box labeled '2' highlights this step.
- Data Input:** The JSON data to be inserted is entered into the dialog. A blue box labeled '3' highlights the input area.

```
1 {  
2   "CPF": "12345647798",  
3   "nome": "Manoela Castro",  
4   "endereco": "Rua 10 numero 40 bairro 3",  
5   "telefone": "3322-3326"  
6 }  
7
```
- Execution and Confirmation:** The 'Insert' button is clicked, and the results are displayed in the MONGOSH shell at the bottom. A blue box labeled '4' highlights the confirmation message. The message shows two new document IDs inserted:

```
acknowledged: true,  
insertedIds: {  
  '0': ObjectId("6539b52ea051eb361b5d4802"),  
  '1': ObjectId("6539b52ea051eb361b5d4803")  
}
```

Inserir documentos

The screenshot shows the MongoDB Compass application interface. The left sidebar lists databases and collections, with 'biblioteca.aluno' selected. The main area displays the 'Documents' tab for the 'aluno' collection. A modal window titled 'Insert Document' is open, prompting the user to add data to the 'biblioteca.aluno' collection. The modal contains two JSON documents:

```
1  _id: ObjectId('6539b86ad3bcc82b5ae4c1a5')          ObjectId
2  CPF: "8965565656/"           String
3  nome: "Cleiton/"             String
4  endereco: "rua 3 num 43 bairro/"      String
5  telefone: "33652556/"            String

{
  "_id": "$oid",
  "CPF": "8965565656/",
  "nome": "Cleiton/",
  "endereco": "rua 3 num 43 bairro/",
  "telefone": "33652556/"
}

{
  "_id": "$oid",
  "CPF": "8965565656/",
  "nome": "Manoela Castro",
  "endereco": "Rua 10 numero 40 bairro 3",
  "telefone": "3322-3326"
}
```

At the bottom right of the modal are 'Cancel' and 'Insert' buttons.

Restrição em coleções

Assim, como para criar um banco de dados e coleções, também existem restrições ao se criar um documento:

- O nome do campo `_id` é reservado para uso como chave primária. Seu valor deve ser único na coleção, é imutável e pode ser de qualquer tipo que não seja um array.
- Os nomes dos campos não podem conter o caractere `NULL`.
- Os documentos BSON, também possuem restrições de tamanho:
- O tamanho máximo de um documento BSON é 16 megabytes.

Consultando os documentos

localhost:27017

Documents
base.empregador...

My Queries

Databases

Search

admin

alura_serie

series

base

empregadores

biblioteca

aluno

config

local

showbank

Import completed.
700 documents imported.

base.empregadores

700 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' } or [Generate query](#)

Explain Reset Find Options

1 - 20 of 700

`_id: ObjectId('6539be5ad3bcc82b5ae4c4ca')`
rank: 1
company: "Samsung Electronics"
industries: "Conglomerate"
country_territory: "South Korea"
employees: 270372
publish_year: 2023

`_id: ObjectId('6539be5ad3bcc82b5ae4c4cb')`
rank: 2
company: "Microsoft"
industries: "IT, Internet, Software & Services"
country_territory: "United States"
employees: 221000
publish_year: 2023

Visualizando os documentos

Consultando os documentos

The screenshot shows the MongoDB Compass interface. At the top, there are two tabs: 'Documents' (selected) and 'alura_serie.series'. The left sidebar lists databases: 'admin', 'alura_serie' (selected), 'base', 'biblioteca', 'config', 'local', and 'showbank'. Under 'alura_serie', there is a 'series' folder. The main area displays the 'alura_serie.series' collection with 93 documents and 1 index. The 'Documents' tab is selected. A query builder is visible, showing a filter condition: {"Ano de lançamento": 2019}. The results section shows three documents:

- Série: "The Family Man"
IMDb Avaliação: 8.6
Classificação: "18+"
- Série: "Modern Love"
IMDb Avaliação: 8
Classificação: "16+"
- Série: "The Boys"
IMDb Avaliação: 8.7

The screenshot shows the Compass MongoDB query interface. At the top, there's a search bar with placeholder text "Type a query: { field: 'value' } or [Generate query](#)". Below the search bar are several buttons: "Filter" (with a dropdown icon), "Explain", "Reset", "Find" (which is green), and "Options". Under the search bar, there are sections for "Project", "Sort", "Collation", "MaxTimeMS", "Skip", and "Limit". The "Project" section contains the query "{ field: 0 }". The "Sort" section contains "{ field: -1 } or [['field', -1]]". The "Collation" section contains "{ locale: 'simple' }". The "MaxTimeMS" section is set to 60000. The "Skip" section is set to 0. The "Limit" section is set to 0.

- **FILTER**: utilizado para especificar qual será a condição que os documentos devem atender para serem retornados na busca.
- **PROJECT**: utilizado para especificar quais campos serão ou não retornados na consulta.
Ao Informar o nome do campo e informar 0, todos os campos, exceto os campos especificados no campo project, são retornados. Se o campo receber o valor de 1, ele será retornado na consulta. O campo _id é retornado por padrão, a menos que este seja especificado no campo project e definido como 0.
- **SORT**: especifica a ordem de classificação dos documentos retornados.
Para especificar a ordem crescente de um campo, defina o campo como 1.
Para especificar a ordem decrescente de um campo, defina o campo como -1.
- **MAX TIME MS**: define o limite de tempo cumulativo em milissegundos para processar as operações da barra de consulta. Se o limite de tempo for atingido antes da conclusão da operação, o Compass interrompe a operação.
- **COLLATION**: utilizado para especificar regras específicas do idioma para comparação de textos, como regras para letras maiúsculas ou minúsculas, acentos, entre outros.
- **SKIP**: especifica quantos documentos devem ser ignorados antes de retornar o conjunto de resultados.
- **LIMIT**: especifica o número máximo de documentos a serem retornados.

Consultando os documentos – Operadores de Comparação

Nome	Descrição
<code>\$eq</code>	Corresponde a valores iguais a um valor especificado.
<code>\$gt</code>	Corresponde a valores maiores que um valor especificado.
<code>\$gte</code>	Corresponde a valores maiores ou iguais a um valor especificado.
<code>\$in</code>	Corresponde a qualquer um dos valores especificados em uma matriz.
<code>\$lt</code>	Corresponde a valores menores que um valor especificado.
<code>\$lte</code>	Corresponde a valores menores ou iguais a um valor especificado.
<code>\$ne</code>	Corresponde a todos os valores que não são iguais a um valor especificado.
<code>\$nin</code>	Não corresponde a nenhum dos valores especificados em uma matriz.

The screenshot shows the MongoDB Compass interface with a query builder. The query is:

```
{"Ano de lançamento": {$in: [2019, 2020]}}
```

The results section displays three documents:

- Série: "Absentia"
Linguagem: "Inglês"
- Série: "American Gods"
Linguagem: "Inglês"
- Série: "Bates Motel"
Linguagem: "Inglês"

Below the results is an "EXPORT DATA" button.

<https://www.mongodb.com/docs/manual/reference/operator/query/>

Consultando os documentos – Operadores Lógico

Nome	Descrição	
\$and	Unir cláusulas de consulta com uma lógica AND retorna todos os documentos que correspondem às condições de ambas as cláusulas.	Filter ⓘ <code>[\$and:[{"Ano de lançamento":2018}, {"Classificação":"18+"}]</code> Project { field: 0 } Sort { field: -1 } or [['field', -1]] Collation { locale: 'simple' }
\$not	Inverte o efeito de uma expressão de consulta e retorna documentos que não <i>correspondem</i> à expressão de consulta.	
\$nor	Unir cláusulas de consulta com uma lógica NOR retorna todos os documentos que não correspondem a ambas as cláusulas.	Filter ⓘ <code>[\$or:[{"Ano de lançamento":2018}, {"Classificação":"18+"}]</code> Project { field: 0 } Sort { field: -1 } or [['field', -1]] Collation { locale: 'simple' }
\$or	Une cláusulas de consulta com uma lógica OR que retorna todos os documentos que correspondem às condições de qualquer uma das cláusulas.	

<https://www.mongodb.com/docs/manual/reference/operator/query/>

Consultando os documentos

- As operações de leitura recuperam documentos de uma coleção ; ou seja, consulte uma coleção de documentos. O MongoDB fornece os seguintes métodos para ler documentos de uma coleção:
- `db.collection.find()`
- Você pode especificar filtros de consulta ou critérios que identifiquem os documentos a serem retornados.

```
db.users.find(  
  { age: { $gt: 18 } },  
  { name: 1, address: 1 }  
).limit(5)
```

← collection
← query criteria
← projection
← cursor modifier

Consultando os documentos

```
/lembre-se de selecionar o database para realizar as consultas
//Consulta todos os documentos da coleção series
db.series.find()
//Consulta todos os registros que tem ano de lançamento 2018
db.series.find({"Ano de lançamento":2018})
//Consulta todos os registros e retorna a projeção (campos) série ano de lançamento e exclui do resultado
o id
db.series.find({}, {"Série":1, "Ano de lançamento":1, "_id":0})
//Consulta todos os registros que em ano de lançamento tem 2019 e 2020
db.series.find({"Ano de lançamento": {$in:[2019,2020]}})
//Consulta todos os registros limitando a resposta a apenas 5
db.series.find().limit(5)
```

<https://devopedia.org/mongodb-query-language>

Atualizando os documentos

ADD DATA ▾ EXPORT DATA ▾ 1 - 20 of 93 ⏪ ⏴ ⏵ ⏶ ⏷ ⏸ ⏹

series						
	_id ObjectId	Série String	Ano de lançamento Int32	Temporadas disponíveis Int32	Linguagem String	Genero Array
1	ObjectId('6539bc0cd3bcc82b5ae...')	Mirzapur	String	1	"Hindi"	[] 1 elements
Document modified.						
2	ObjectId('6539bc0cd3bcc82b5ae...')	"The Family Man"	2019	1	"Hindi"	[] 1 elements
3	ObjectId('6539bc0cd3bcc82b5ae...')	"Modern Love"	2019	1	"Inglês"	[] 3 elements
4	ObjectId('6539bc0cd3bcc82b5ae...')	"Comicstaan"	2018	2	"Hindi"	[] 3 elements

Atualizando os documentos

As operações de atualização modificam documentos existentes em uma coleção . MongoDB fornece os seguintes métodos para atualizar documentos de uma coleção:

- `db.collection.updateOne()` Novo na versão 3.2
- `db.collection.updateMany()` Novo na versão 3.2
- `db.collection.replaceOne()` Novo na versão 3.2

No MongoDB, as operações de atualização têm como alvo uma única coleção. Todas as operações de gravação no MongoDB são atômicas no nível de um único documento.

```
db.users.updateMany(  
  { age: { $lt: 18 } },  
  { $set: { status: "reject" } } )
```



The diagram illustrates the components of the MongoDB `updateMany` command with the following labels and arrows:

- `collection`: Points to the `users` part of `db.users`.
- `update filter`: Points to the `{ age: { $lt: 18 } }` filter.
- `update action`: Points to the `{ $set: { status: "reject" } }` action.

Atualizando os documentos

- Alura_Series> db.series.find({"Série": "Grimm"})
- db.series.updateOne({"Série": "Grimm"}, {\$set: {"Temporadas disponíveis": 6}})
//Atualizando o documento que corresponde a série Grimm adicionado um campo que não existia
- db.series.updateOne({"Série": "Grimm"}, {\$set: {"Classificação": "16+"}})
- db.series.find({"Série": {\$in: ["Four More Shots Please", "Fleabag"]}})
- db.series.updateMany({"Série": {\$in: ["Four More Shots Please", "Fleabag"]}}, {\$set: {"Classificação": "18+"}})

Excluindo documentos

As operações de exclusão removem documentos de uma coleção. O MongoDB fornece os seguintes métodos para excluir documentos de uma coleção:

- `db.collection.deleteOne()` Novo na versão 3.2
- `db.collection.deleteMany()` Novo na versão 3.2
- No MongoDB, as operações de exclusão têm como alvo uma única coleção . Todas as operações de gravação no MongoDB são atômicas no nível de um único documento.

```
db.users.deleteMany(  
  { status: "reject" } )
```

The diagram illustrates the components of the MongoDB `deleteMany` command. It shows the command structure: `db.collection.deleteMany(filter)`. Annotations with arrows point to specific parts: one arrow points from the label "collection" to the collection name "users"; another arrow points from the label "delete filter" to the filter condition `{ status: "reject" }`.

Excluindo documentos

- Alura_Series> `db.series.find({"Série": "The Boys"})`
- `db.series.deleteOne({"Série": "The Boys"})`
- `db.series.deleteMany({"Temporadas disponíveis": 1})`
- `db.series.deleteMany({})`