

**INSTITUTO  
FEDERAL**  
Santa Catarina

# Banco de Dados 2



## Bancos de Dados não relacionais

1. Introdução
2. MongoDB
3. Operações DML
4. Operações DQL
5. Schema Design
6. Otimização
7. Agregação



1/4

# Introdução



# Introdução

- Os bancos de dados não relacionais foram criados a partir de uma necessidade de manipulação de grandes volumes de dados
- O termo “NoSQL” não possui uma definição clara, mas geralmente está vinculado a tecnologias de bancos de dados que utilizam dados sem schema.



# Por que Bds NoSQL é interessante?

- Mais simples de programar: Menos trabalho gasto no mapeamento dos dados no programa;
- Pode oferecer mais desempenho e escalabilidade sendo capaz de processar grandes quantidades de dados em *clusters* em vez de uma única máquina;
- Em vez de você evoluir em direção à integração do banco de dados, o objetivo passa a ser encapsular o banco de dados na aplicação e a integração de serviços.
- Portabilidade entre diferentes Bds não relacionais não é tão simples como as dos Bds relacionais, graças ao SQL.

# Tipos de NoSQL

Estrutura	Descrição	Ex. de softwares
<b>chave-valor</b>	É o BD NoSQL mais simples, sendo basicamente uma tabela hash, onde há uma chave primária e seu respectivo valor.	Redis, MemCached, Hamster-DB, Project Voldemort
<b>Documentos</b>	Armazena os dados em formatos como XML e JSON, localizados através de identificadores.	MongoDB, CouchDB, Lotus Notes, OrientDB
<b>Família de Colunas</b>	Família de colunas são grupo de dados relacionados. Armazenam dados com chaves mapeadas para valores. Voltados para bigdata.	Cassandra, HBase, Amazon SimpleDB, HyperTable
<b>Grafos</b>	Grafos são vários nós relacionados em uma estrutura complexa. As informações podem estar conectadas de forma uni ou bidirecional.	Neo4J, InfiniteGraph, OrientDB

# É o fim dos bancos de dados relacionais?

- Bancos de dados NoSQL foram criados para ser um complemento aos bancos de dados SQL, e não para substituí-los.
- NoSQL na verdade quer dizer Not Only SQL.
- Carlo Strozzi, criador do conceito, defende que o nome dado não reflete muito bem a intenção desse modelo.
- Para ele, ele deveria se chamar NoREL, ou seja, não-relacional.





2/4

mongoDB®



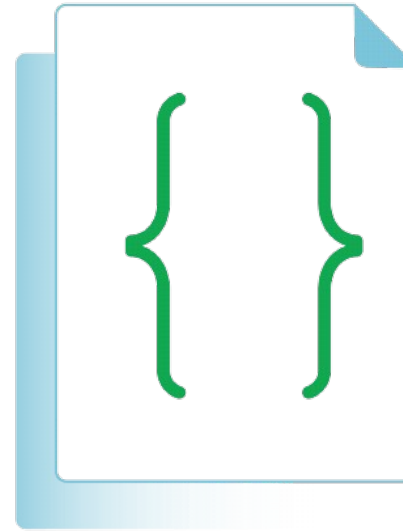


- MongoDB é um software livre de banco de dados NoSQL orientado a documentos de código aberto.
- Assim como no MySQL, existe a versão “*Enterprise*” paga e outros serviços igualmente pagos.
- A versão gratuita é a “*Community*”
- É um banco de dados NoSQL



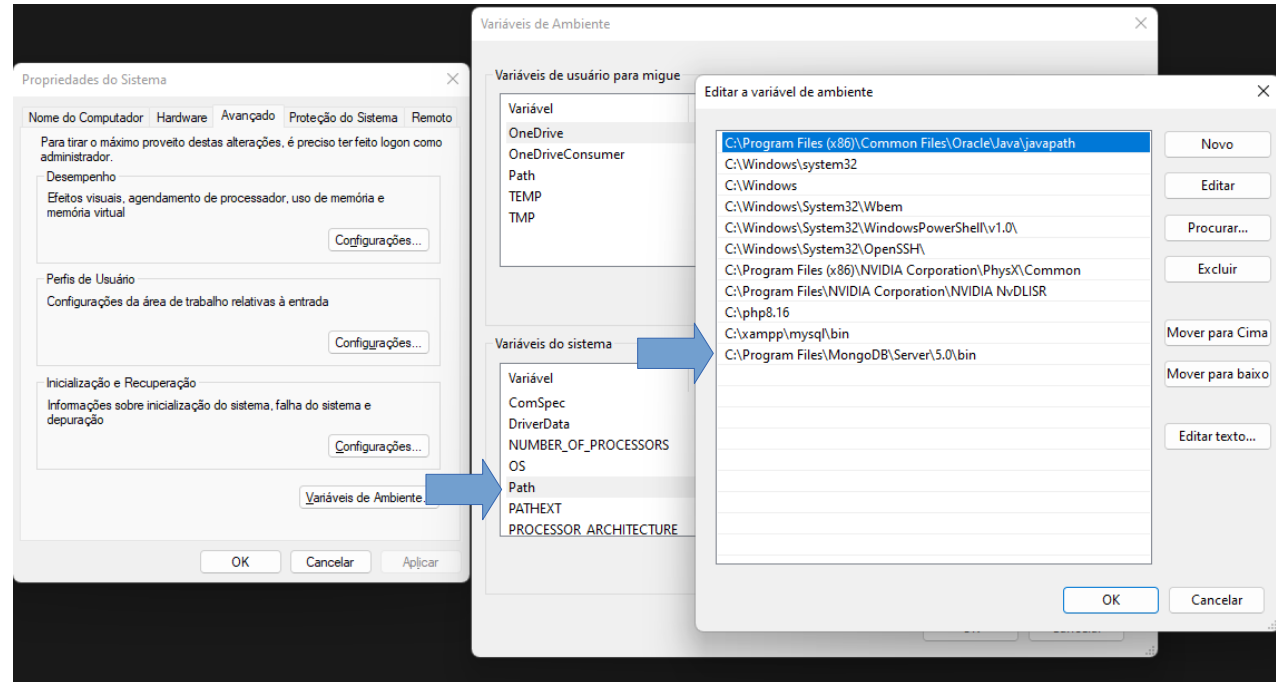
```
1  {  
2    _id: "5cf0029caff5056591b0ce7d",  
3    firstname: 'Jane',  
4    lastname: 'Wu',  
5    address: {  
6      street: '1 Circle Rd',  
7      city: 'Los Angeles',  
8      state: 'CA',  
9      zip: '90404'  
10   }  
11  }
```

- O MongoDB não trabalha com schemas definidos
- Ele utiliza documentos semelhantes a (JavaScript Object Notation) como esquemas
- Existe também o BSON (Binary JSON), sendo uma representação binário do formato que permite mais tipos de dados além dos que já fazem parte da especificação JSON.



```
{  
  "model": "Volvo C70",  
  "year": 2007,  
  "bodyStyle": [  
    "coupe",  
    "convertible"  
  ],  
  "engine": {  
    "model": "D5",  
    "power": "178hp"  
  }  
}
```

- Para instalar, baixe o server versão community no site <https://mongodb.org>
- Instalação clássica next, next.
- Instale opcionalmente o MongoDB Compass.
- No windows, adicione nas variáveis de ambiente o caminho da instalação.



- No terminal de comando, digite o comando “**mongo**” para se conectar.
- Para exibir os bancos de dados atuais:

**> show dbs**

- Para criar um banco, utilize “**use banco**”. Caso não exista, o mongoDB irá criá-lo:

**> use escola**

```
Windows PowerShell
Implicit session: session { "id" : UUID("77f3f3ce-9342-4377-9435-442b96791af5") }
MongoDB server version: 5.0.9
=====
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility. The "mongo" shell has been deprecated and will be removed in an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
=====
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
https://docs.mongodb.com/
Questions? Try the MongoDB Developer Community Forums
https://community.mongodb.com

---
The server generated these startup warnings when booting:
  2022-06-25T21:18:12.335-03:00: Access control is not enabled for the database. Read and write operations that change data (e.g. insert, update, save, delete, drop) will potentially modify data.

---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible via the MongoDB Cloud console.
```

- Coleções são como tabelas que podem ter informações relacionadas.
- No mongoDB, uma coleção é um grupo de documentos (linhas) que podem estruturas diferentes, incluindo outros documentos. Em um banco de dados relacional, seria o equivalente a uma junção.
- Uma coleção no mongoDB funciona como uma tabela não normalizada.

**> show collections**

```
alunos = [  
  {  
    "nome": "João",  
    "nascimento": "2000-05-25",  
    "telefones": [  
      "4899999999",  
      "4898888888"  
    ]  
  },  
  {  
    "nome": "Maria",  
    "nascimento": "2002-01-15",  
  }  
]
```

- Para apagar um banco de dados, após selecioná-lo

```
db.dropDatabase()
```

- Para apagar uma coleção

```
db.alunos.drop()
```



3/4

# Operações DML

Inserir, atualizar e excluir



# Operações de inserir, atualizar e excluir

- Criando registros (cria o schema automaticamente)

```
db.alunos.insert( ... )
```

```
db.alunos.insert( { "nome":"Pedro", "nascimento":"2098-01-22" } )
```

- Se não informar um `_id`, ele irá gerar automaticamente

```
db.alunos.insert( { "_id":1, "nome":"Ana", "nascimento":"2002-01-15" } )
```



# Operações de inserir, atualizar e excluir

- Você pode inserir várias informações ao mesmo tempo

```
db.alunos.insert(  
  [  
    { "nome":"João", "nascimento":"2005-01-01" },  
    { "nome":"Alessandra", "nascimento":"2003-03-12" }  
  ]  
)
```

# Operações de inserir, atualizar e excluir

- Para exibir informações

```
db.alunos.find()
```

- Para exibir o JSON formatado

```
db.alunos.find().pretty()
```

```
Windows PowerShell
> db.alunos.find().pretty()
{
  "_id" : ObjectId("62b8bf083c68855f38be8030"),
  "nome" : "Ana",
  "nascimento" : "2002-01-15"
}
{ "_id" : 1, "nome" : "Paula", "nascimento" : "2002-01-15" }
{
  "_id" : ObjectId("62b8cf273c68855f38be8032"),
  "nome" : "Pedro",
  "nascimento" : "2098-01-22"
}
{
  "_id" : ObjectId("62b8cfe13c68855f38be8033"),
  "nome" : "João",
  "nascimento" : "2005-01-01"
}
{
  "_id" : ObjectId("62b8cfe13c68855f38be8034"),
  "nome" : "Alessandra",
  "nascimento" : "2003-03-12"
}
> |
```

# Operações de inserir, atualizar e excluir

- Para remover documentos (precisa de uma *query*)

```
db.alunos.remove( {"nome":"João"} )
```

- Em versões antigas, não passar a *query* apagava todos documentos.

# Operações de inserir, atualizar e excluir

- Para editar documentos, você precisa passar dois parâmetros

```
db.alunos.update({"nome":"João"}, {"nome":"João", "nascimento":"2003-01-01"})
```

- O primeiro parâmetro é a query de consulta
- O segundo é o documento inteiro com as informações atualizadas

- Para atualizar parcialmente

```
db.alunos.update({"nome":"João"}, {$set:{"nascimento":"2001-01-01"}})
```



## Operações de inserir, atualizar e excluir

- Para remover uma parte do documento sem alterá-lo por inteiro, utilizamos o comando abaixo

```
db.alunos.update({"nome":"João"}, {$unset:{"nascimento":"2001-01-01"}})
```



- Para adicionar um novo atributo ao documento, basta utilizar o \$set normalmente

```
db.alunos.update({"nome":"João"}, {$set:{"idade":25}})
```



# Operações de inserir, atualizar e excluir

- Modificador para incrementar inteiro

```
db.alunos.update({"nome":"João"}, {$inc:{"idade":1}})
```



```
Windows PowerShell
> db.alunos.find().pretty()
{
  "_id" : ObjectId("62b8d74c3c68855f38be803c"),
  "nome" : "Alessandra",
  "nascimento" : "2003-03-12"
}
{
  "_id" : ObjectId("62b8eca33c68855f38be803e"),
  "nome" : "João",
  "idade" : 25
}
> |
```



```
Windows PowerShell
> db.alunos.find().pretty()
{
  "_id" : ObjectId("62b8d74c3c68855f38be803c"),
  "nome" : "Alessandra",
  "nascimento" : "2003-03-12"
}
{
  "_id" : ObjectId("62b8eca33c68855f38be803e"),
  "nome" : "João",
  "idade" : 26
}
> |
```

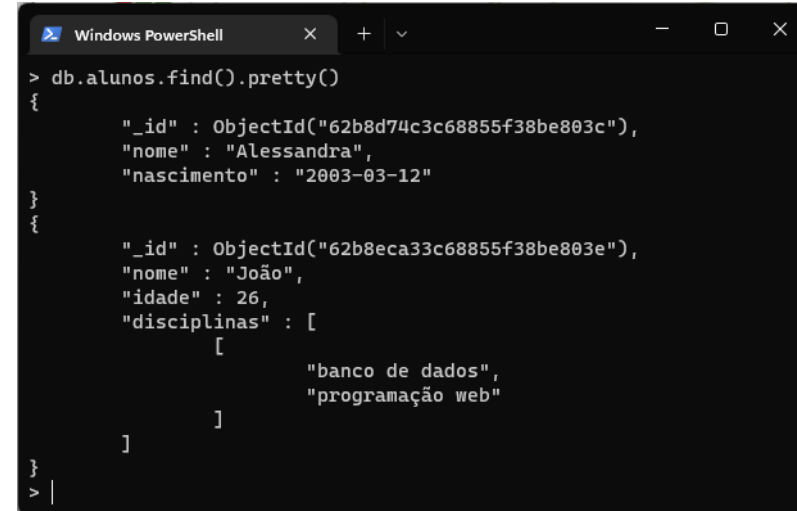
# Operações de inserir, atualizar e excluir

- Modificador para adicionar um atributo como array

```
db.alunos.update(  
  {"nome":"João"},  
  {$push:{"disciplinas":"banco de dados"}}  
)
```



```
db.alunos.update(  
  {"nome":"João"},  
  {$push:{"disciplinas":"programação web"}}  
)
```

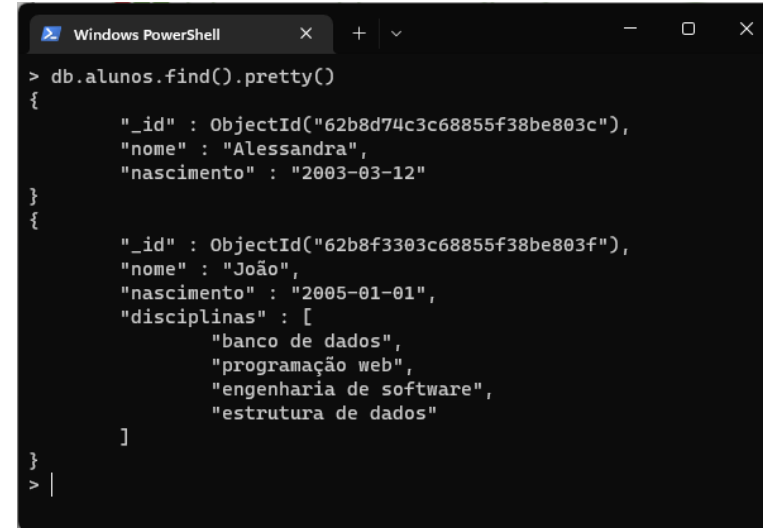


```
Windows PowerShell  
> db.alunos.find().pretty()  
{  
  "_id" : ObjectId("62b8d74c3c68855f38be803c"),  
  "nome" : "Alessandra",  
  "nascimento" : "2003-03-12"  
}  
{  
  "_id" : ObjectId("62b8eca33c68855f38be803e"),  
  "nome" : "João",  
  "idade" : 26,  
  "disciplinas" : [  
    "banco de dados",  
    "programação web"  
  ]  
}
```

# Operações de inserir, atualizar e excluir

- Modificador para adicionar vários valores a um atributo do tipo array combinamos o operador **\$push** com o **\$each**

```
db.alunos.update(  
  { "nome": "João" },  
  {  
    $push: {  
      "disciplinas": {  
        $each: ["engenharia de software", "estrutura de dados"]  
      }  
    }  
  }  
)
```



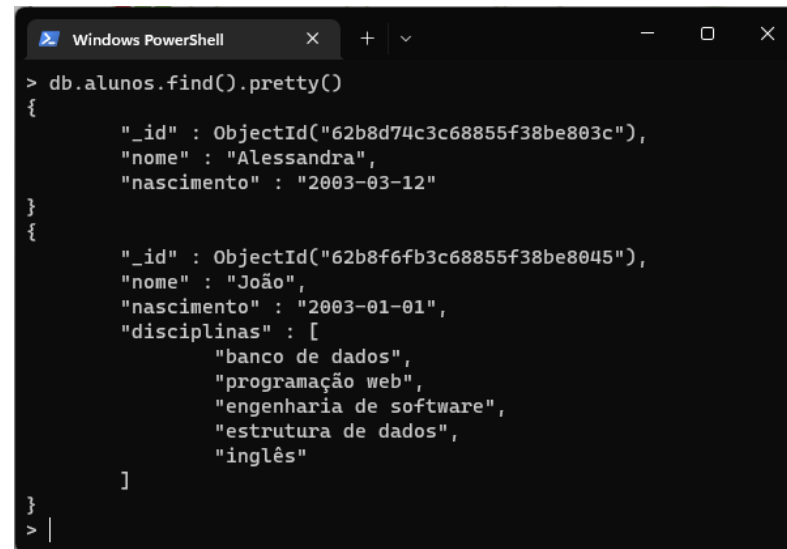
```
Windows PowerShell  
> db.alunos.find().pretty()  
{  
  "_id" : ObjectId("62b8d74c3c68855f38be803c"),  
  "nome" : "Alessandra",  
  "nascimento" : "2003-03-12"  
}  
{  
  "_id" : ObjectId("62b8f3303c68855f38be803f"),  
  "nome" : "João",  
  "nascimento" : "2005-01-01",  
  "disciplinas" : [  
    "banco de dados",  
    "programação web",  
    "engenharia de software",  
    "estrutura de dados"  
  ]  
}
```



# Operações de inserir, atualizar e excluir

- Modificador para adicionar vários valores a um atributo do tipo array sem repetições combinamos o operador **\$addToSet** com o **\$each**

```
db.alunos.update(  
  { "nome": "João" },  
  {  
    $addToSet: {  
      "disciplinas": {  
        $each: ["banco de dados", "inglês"]  
      }  
    }  
  }  
)
```




```
Windows PowerShell  
> db.alunos.find().pretty()  
  
{"_id" : ObjectId("62b8d74c3c68855f38be803c"),  
  "nome" : "Alessandra",  
  "nascimento" : "2003-03-12"  
}  
  
{"_id" : ObjectId("62b8f6fb3c68855f38be8045"),  
  "nome" : "João",  
  "nascimento" : "2003-01-01",  
  "disciplinas" : [  
    "banco de dados",  
    "programação web",  
    "engenharia de software",  
    "estrutura de dados",  
    "inglês"  
  ]  
}
```

# Operações de inserir, atualizar e excluir

- Modificador para adicionar vários valores a um atributo do tipo array sem repetições combinamos o operador **\$addToSet** com o **\$each**

```
db.alunos.update(  
  { "nome": "João" },  
  {  
    $addToSet: {  
      "disciplinas": {  
        $each: ["banco de dados", "inglês"]  
      }  
    }  
  })
```



```
Windows PowerShell  
> db.alunos.find().pretty()  
  
  {  
    "_id" : ObjectId("62b8d74c3c68855f38be803c"),  
    "nome" : "Alessandra",  
    "nascimento" : "2003-03-12"  
  },  
  {  
    "_id" : ObjectId("62b8f6fb3c68855f38be8045"),  
    "nome" : "João",  
    "nascimento" : "2003-01-01",  
    "disciplinas" : [  
      "banco de dados",  
      "programação web",  
      "engenharia de software",  
      "estrutura de dados",  
      "inglês"  
    ]  
  }  
]
```

# Operações de inserir, atualizar e excluir

- para remover o primeiro item da lista de um atributo array

```
db.alunos.update({ "nome": "João" }, { $pop: { "disciplinas": -1 } })
```



- para remover o último item da lista de um atributo array

```
db.alunos.update({ "nome": "João" }, { $pop: { "disciplinas": 1 } })
```

# Operações de inserir, atualizar e excluir

- Para remover por critério um item da lista de um atributo array, utilizamos o modificador **\$pull**

```
db.alunos.update(  
  { "nome": "João" },  
  { $pull: { "disciplinas": "engenharia de software" } }  
)
```



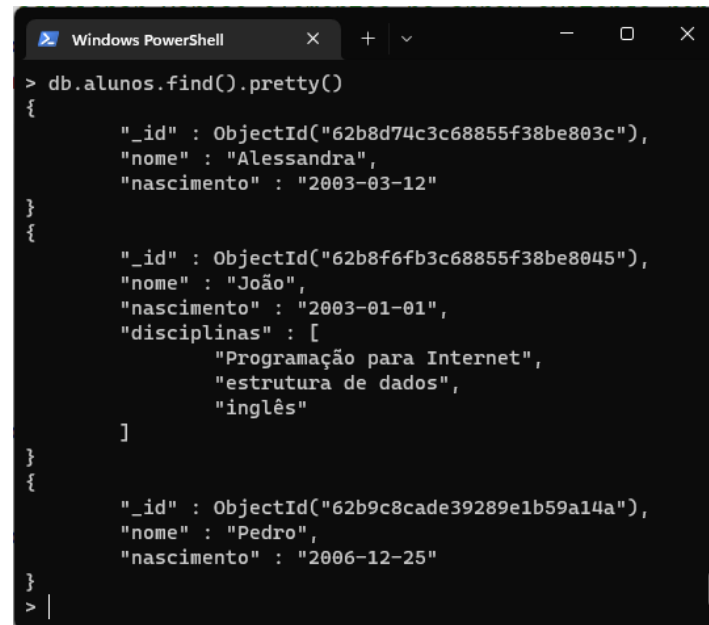
```
Windows PowerShell  
> db.alunos.find().pretty()  
  
{"_id" : ObjectId("62b8d74c3c68855f38be803c"),  
  "nome" : "Alessandra",  
  "nascimento" : "2003-03-12"  
}  
  
{"_id" : ObjectId("62b8f6fb3c68855f38be8045"),  
  "nome" : "João",  
  "nascimento" : "2003-01-01",  
  "disciplinas" : [  
    "programação web",  
    "estrutura de dados",  
    "inglês"  
  ]  
}
```

# Operações de inserir, atualizar e excluir



```
db.alunos.update( {"nome":"Pedro"}, {$set:{"nascimento":"2006-12-25"}}, true)
```

- Se você quer editar uma informação mas caso ela não exista seja incluída, basta adicionar um terceiro parâmetro **true** no update.



```
Windows PowerShell
> db.alunos.find().pretty()
{
  "_id" : ObjectId("62b8d74c3c68855f38be803c"),
  "nome" : "Alessandra",
  "nascimento" : "2003-03-12"
}

{
  "_id" : ObjectId("62b8f6fb3c68855f38be8045"),
  "nome" : "João",
  "nascimento" : "2003-01-01",
  "disciplinas" : [
    "Programação para Internet",
    "estrutura de dados",
    "inglês"
  ]
}

{
  "_id" : ObjectId("62b9c8cade39289e1b59a14a"),
  "nome" : "Pedro",
  "nascimento" : "2006-12-25"
}
```

4/4

# Operações DQL

Listagem e busca



# Operações de listagem e busca

- Para listar com um parâmetro o documento, utilizamos a sintaxe abaixo:

```
db.alunos.find( {nome:"Alessandra"} )
```

- ou com mais de um parâmetro de busca

```
db.alunos.find( {nome:"Alessandra", nascimento:"2003-03-12"} )
```

- Ou utilizando expressões regulares

```
db.alunos.find( {nascimento:/2003/} )
```

# Operações de listagem e busca

- Para buscar limitando a quantidade de documentos retornados:

```
db.alunos.find().limit(2).pretty()
```

- Para buscar apenas o primeiro resultado

```
db.alunos.findOne()
```

- Para ordenar os resultados (1 crescente, -1 decrescente)

```
db.alunos.find().sort( {nome: 1} )
```

```
db.alunos.find().sort( {nome: -1} )
```



# Operações de listagem e busca

- Para fazer uma paginação, podemos combinar skip com limit

```
db.alunos.find().skip(3).limit(3).pretty()
```

- Para buscar apenas partes (chaves) de um documento, utilizamos um segundo parâmetro

```
db.alunos.find( { }, { nome: 1 } )
```

# Operadores de comparação

\$lt	<
\$lte	<=
\$gt	>
\$gte	>=
\$eq	=
\$ne	!=

- Exemplos

```
db.alunos.find({ idade: {$gt:18, $lt:60} })  
db.alunos.find({ nome: {$ne:'Pedro'} })
```

# Operadores lógicos

\$or

\$and

\$not

\$nor

- Exemplo

```
db.alunos.find({$or: [ {nome: "Pedro"}, {nome: "Maria"} ] })
```