

# Técnico em desenvolvimento de sistemas

## Bancos de dados não relacionais: NoSQL

Você já deve ter aprendido nos outros materiais sobre os bancos de dados relacionais, nos quais você cria uma estrutura organizada com tabelas, colunas e campos com propriedades predefinidas, com grande parte das informações interligadas entre si. Um campo que contenha uma ID de outra tabela pode ser uma forma simples de relacionar ambos os dados em um contexto. Seu software atende aos aspectos definidos em sua base de dados, tudo muito organizado! O que você acha, porém, de explorar uma diferente forma de armazenar dados? Para este novo aprendizado sobre NoSQL, será necessário que você coloque um pouco de lado alguns conceitos para conseguir absorver novas ideias e uma metodologia de registro de dados bastante diferente do que foi visto neste momento.

### SQL versus NoSQL

O **SQL** (*structured query language*, ou “**linguagem de consulta estruturada**”, **em português**) é um meio de desenvolver uma base de dados relacional estruturadamente, definindo tabelas, regras e especificações aos seus dados, gerando forte consistência na manipulação de dados. Sua escalabilidade é normalmente voltada para uma grande estrutura, na qual, conforme o aumento do uso de banco de dados acontece, os servidores utilizados se tornam maiores e mais desenvolvidos.

Em contrapartida, o **NoSQL** (*not only SQL*, ou “**não relacional**”) apresenta diferentes formas de organização, não contendo esquema de tabelas. Suas formações são mais flexíveis, pois sua estrutura é menos rigorosa. Embora haja uma

menor consistência de informações e restrições, a compensação ocorre por meio da otimização de interação com grandes escalas de dados, tornando os processos mais rápidos. Sua escalabilidade ocorre horizontalmente, ou seja, de acordo com o crescimento no uso dos dados, existe a possibilidade de distribuir as informações por meio de diversos servidores distribuídos.

Enquanto a estrutura das bases de dados SQL são baseadas em tabelas, as bases de dados contêm formas diferentes de organizar as informações.

Veja a seguir algumas das formas de estruturas de tabelas não relacionais:

## Colunar

Sua ideia é representada em forma de tabela, na qual há diversas famílias de colunas. Dentro dessas famílias, encontram-se as colunas com as respectivas propriedades.

## Grafos

Seguindo a ideia básica de um grafo – nós ou pontos ligados a outros nós por meio de vértices, formando uma série de conexões direcionadas ou não –, as informações são representadas por esses nós e os relacionamentos pelos vértices. Essa estrutura de armazenamento de grafos também contém ligações que realizam o intermédio das associações entre os grafos e suas respectivas propriedades, relacionando-as.

## Chave-valor

Trata-se do armazenamento de dados vinculado a uma chave. Todo processo de manipulação de dados (inserção, leitura, edição ou exclusão) será realizado por meio do uso de sua chave específica, de seu campo em específico. Desta forma, será difícil ocorrer algum conflito de informações no banco de dados.

## Documento

Essa estrutura é baseada na ideia de uma coleção de documentos, na qual o documento corresponde a um objeto que contém um código único, junto a um conjunto de informações. Estas informações podem ser de vários tipos, como **int**, **string**, e podem ser até mesmo outros documentos ou listas. Embora possa ser visto como similar ao modelo chave-valor, o documento se diferencia por meio dos identificadores únicos de coleção. Além disso, o armazenamento de dados em **JSON** (*JavaScript Object Notation*) auxilia no suporte para vários tipos de dados.

Vale destacar também que ambas as tecnologias são eficientes para áreas de atuação específicas, sendo assim, não é preciso optar sempre por somente uma das opções. É importante avaliar seu projeto para adaptar conforme a necessidade, obtendo assim um melhor desempenho. Para sistemas em que as informações que deverão ser registradas são conhecidas e as informações podem ser melhor predefinidas (como sistemas de estoque, contabilidade, gerenciamentos de uma

forma geral), estruturas de código que utilizam o MySQL tendem a trazer ótimos resultados. Já para negócios com crescimento progressivo, no qual a base de dados pode não ficar extremamente clara e pode passar por mudanças, seu sistema pode se encaixar melhor com o uso de NoSQL.

Reflita sobre as seguintes questões: quais tipos de bancos de dados podem auxiliá-lo no seu projeto? De que forma isso será uma influência positiva? O contexto do seu sistema apresenta um cenário de constante atualização e acesso de informações? As informações que serão inseridas devem conter uma esquematização mais rígida ou flexível? Serão feitas alterações frequentes na base de dados?

Por meio de perguntas como essas, você conseguirá observar melhor qual opção poderá atender à sua necessidade.

## MongoDB

Para explorar um pouco mais esta tecnologia, será utilizado o banco de dados MongoDB.

O “**MongoDB** é um banco de dados de documentos com a escalabilidade e flexibilidade que você deseja junto com a consulta e indexação que você precisa” (MONGODB, 2021).

O site oficial do MongoDB apresenta alguns aspectos interessantes sobre seu funcionamento, como, por exemplo, o armazenamento de dados que ocorre por meio de documentos do tipo **JSON** flexíveis, ou seja, os campos utilizados podem variar de documento para documento, permitindo que a estrutura dos dados possa ser alterada com maior facilidade ao longo do tempo. Além disso, os processos de consultas *ad hoc* (consultas desenvolvidas conforme a necessidade), indexação (classificação de registros) e agregação em tempo real oferecem formas robustas de acesso e análise de dados. Sites como Google, eBay e Pearson são exemplos de grandes empresas que utilizam o NoSQL especificamente com o uso do MongoDB.

JSON é uma representação textual simplificada de um objeto de dados. O formato permite que se informe o nome de uma ou de várias propriedades desse objeto e o valor correspondente e também que sejam representadas listas ou objetos aninhados. Veja o exemplo a seguir para a representação dos dados de uma pessoa:

```
{  
  "nome": "Maria da Silva",  
  "idade": 35,  
  "endereco": {  
    "rua": "Arlindo Pereira",  
    "numero": 10,  
    "cidade": "Garças",  
  },  
  "hobbies": ["musica", "jogos", "viagens"]  
}
```

Nesse exemplo, há um objeto (da pessoa), um objeto aninhado (em “endereco”) e uma lista (em “hobbies”). JSON tem sido muito usado nas linguagens de programação modernas para transportar dados, especialmente pela web.

Agora que você conhece um pouco mais sobre esse banco de dados, que tal já começar a usá-lo?

## Processo de instalação: passos iniciais

Para possibilitar a experiência de inserção das linhas de comando e também permitir a visualização dos documentos por meio de uma interface gráfica, faça o *download* do MongoDB Compass para utilizar nos exemplos encontrados ao longo deste material. Para iniciar o *download*, basta buscar por MongoDB no seu navegador e acessar o *site* oficial (figura 1). Na guia de produtos, selecione a versão **Community** do sistema (figura 2). Nesta página, é possível escolher o sistema operacional e a versão que deseja fazer o *download* (figura 3).

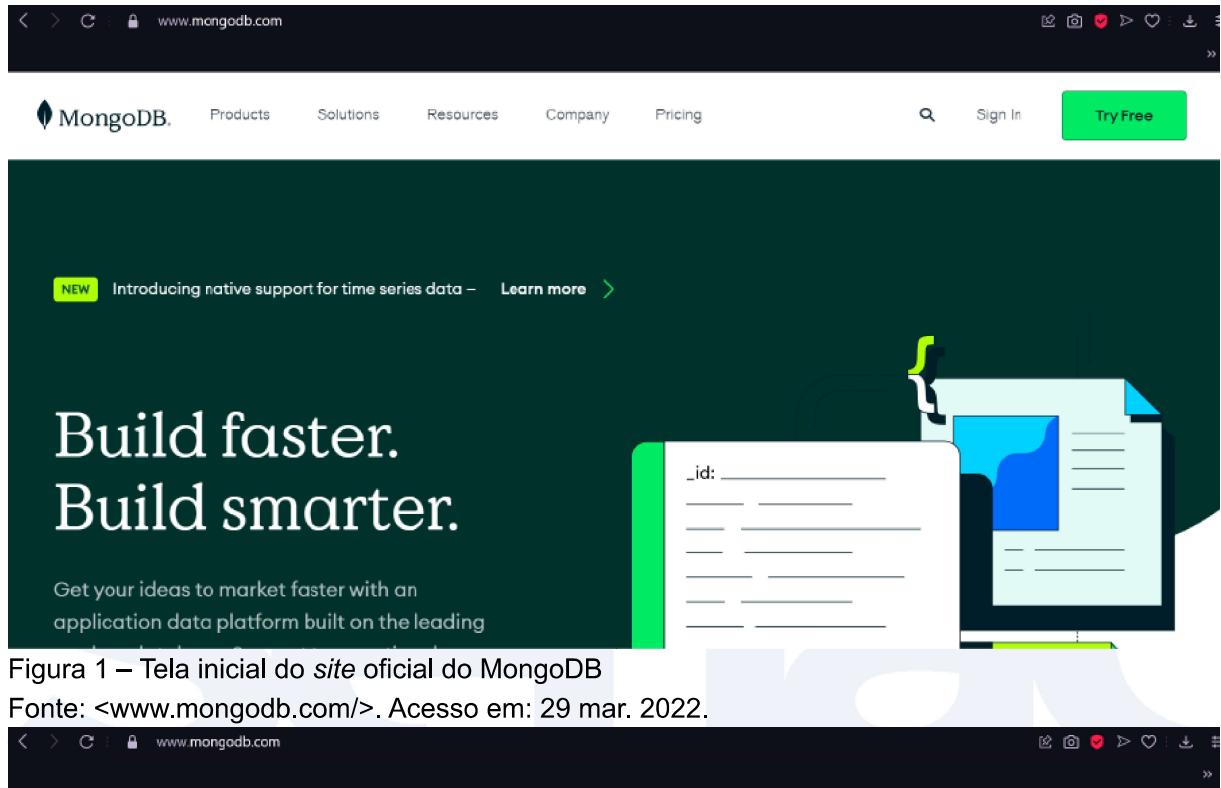


Figura 1 – Tela inicial do site oficial do MongoDB  
Fonte: <[www.mongodb.com/](http://www.mongodb.com/)>. Acesso em: 29 mar. 2022.

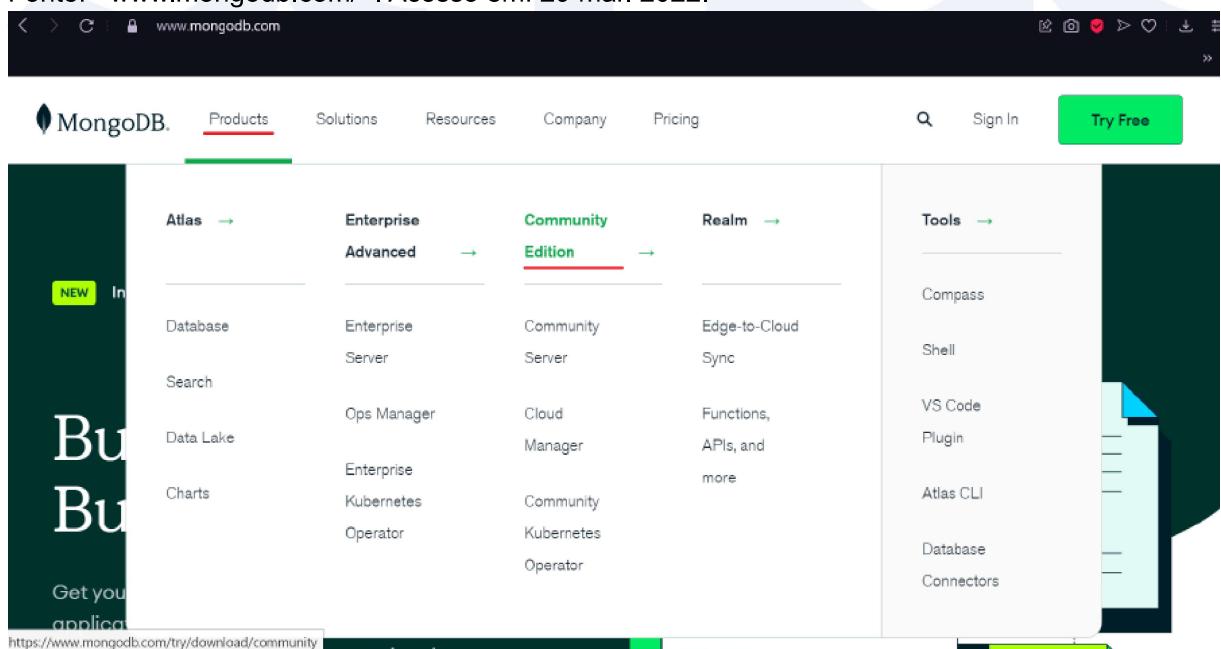


Figura 2 – Indicação de download do MongoDB  
Fonte: <[www.mongodb.com/](http://www.mongodb.com/)>. Acesso em: 29 mar. 2022.

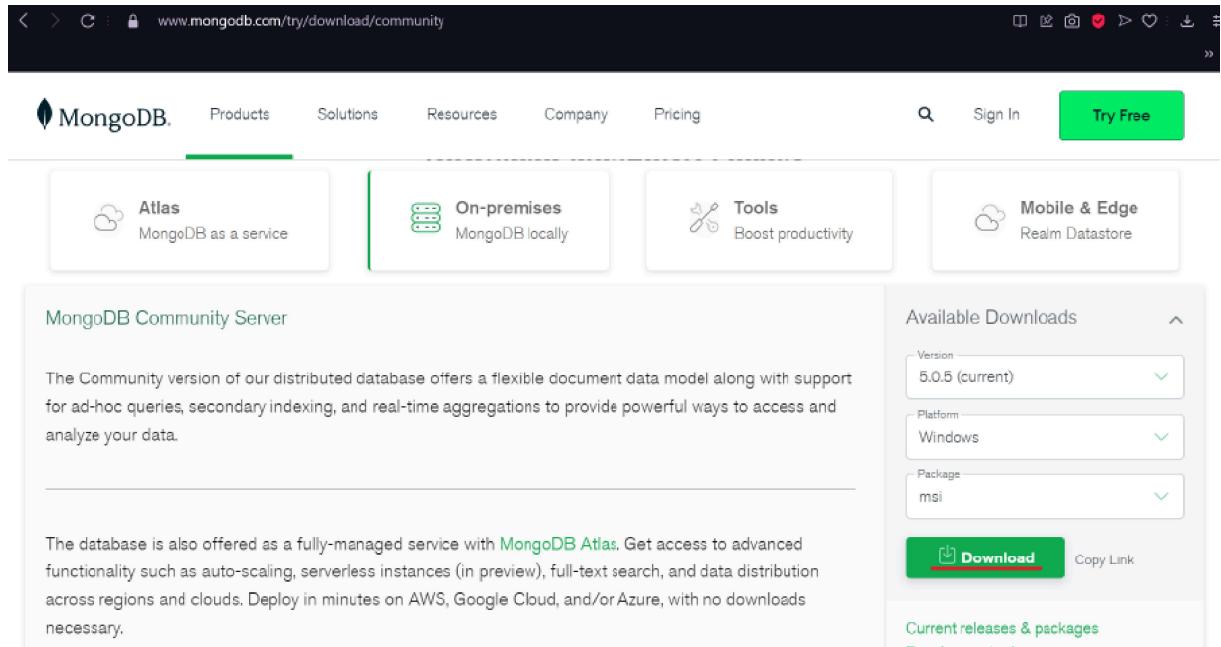


Figura 3 – Tela de download do MongoDB

Fonte: <[www.mongodb.com/](http://www.mongodb.com/)>. Acesso em: 29 mar. 2022.

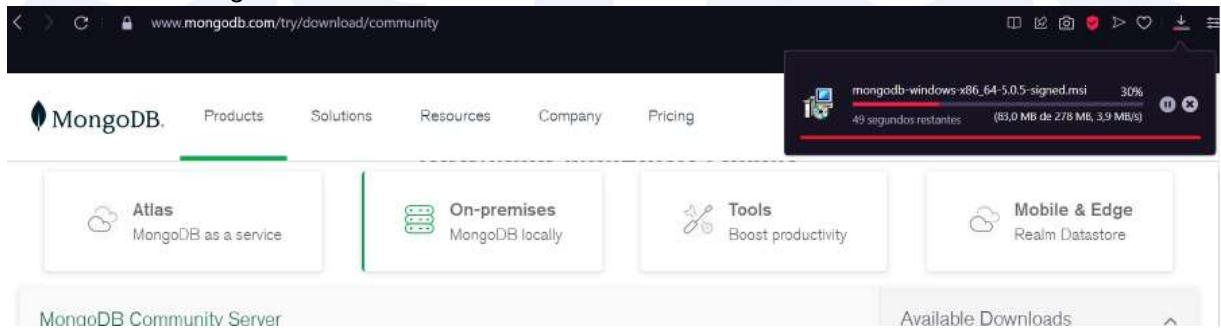
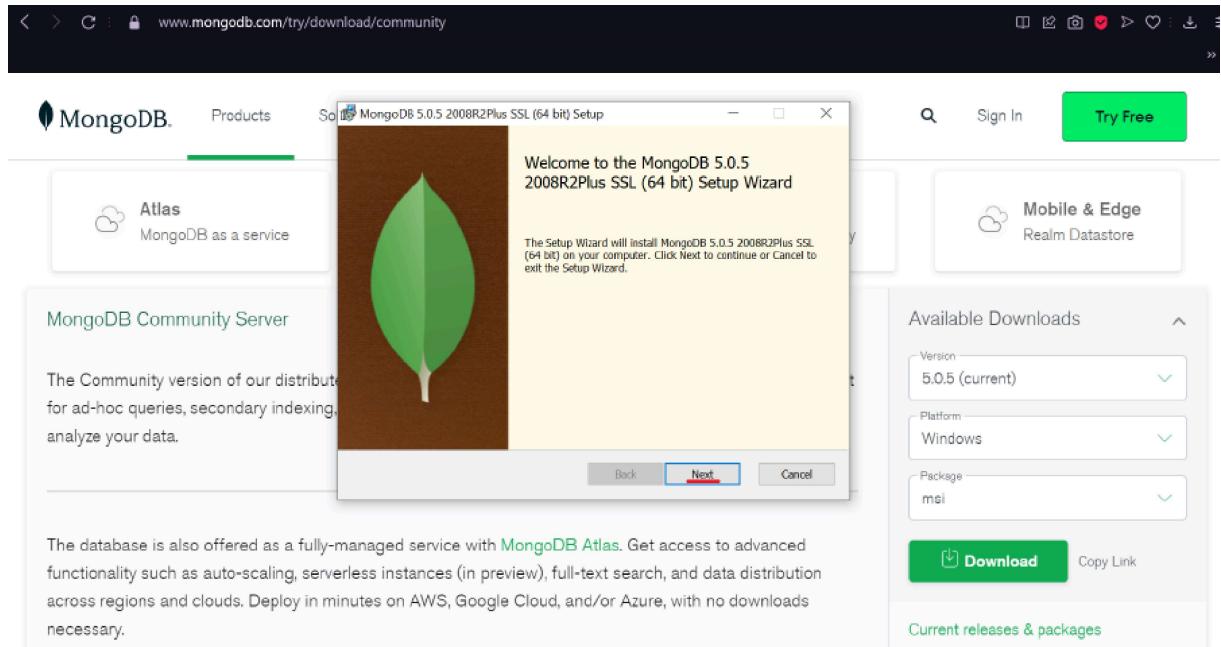


Figura 4 – Indicação do surgimento do início do download do arquivo pelo navegador

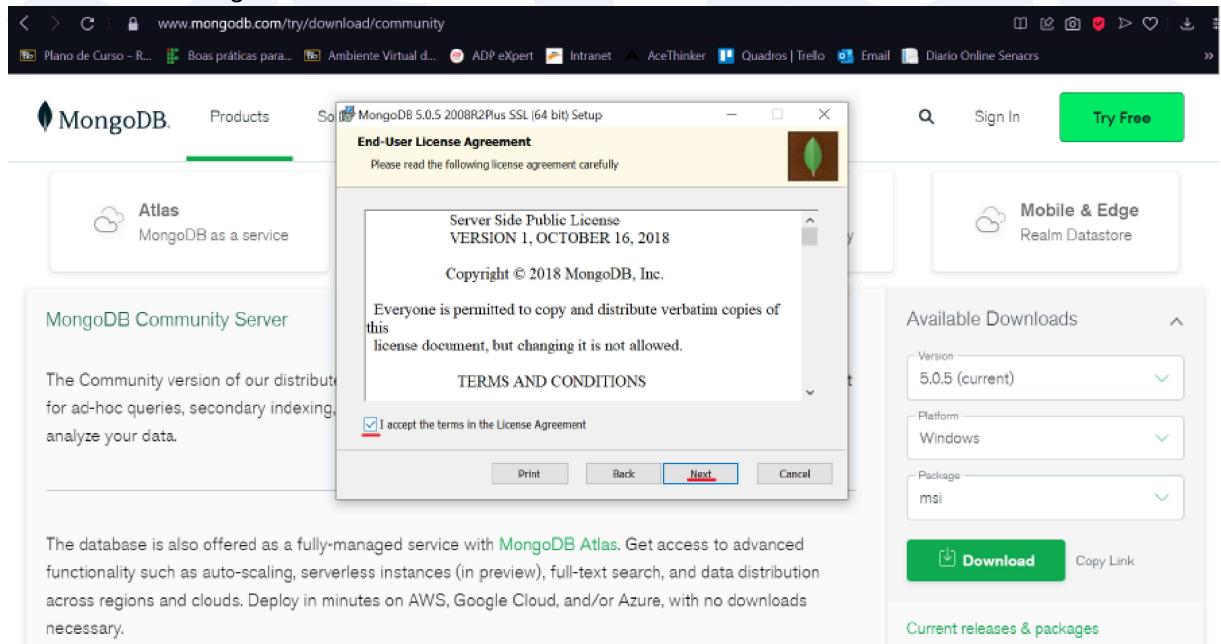
Fonte: <[www.mongodb.com/](http://www.mongodb.com/)>. Acesso em: 29 mar. 2022.

A instalação do sistema após o *download* é bem simples! Basta dar dois cliques no arquivo que você baixou para iniciar a instalação. Você pode manter as configurações previamente apresentadas, devendo ler e confirmar os termos de uso e verificar se a instalação do MongoDB Compass está selecionada. No restante da instalação, clique em **Next** até chegar à tela de conclusão de instalação do software.



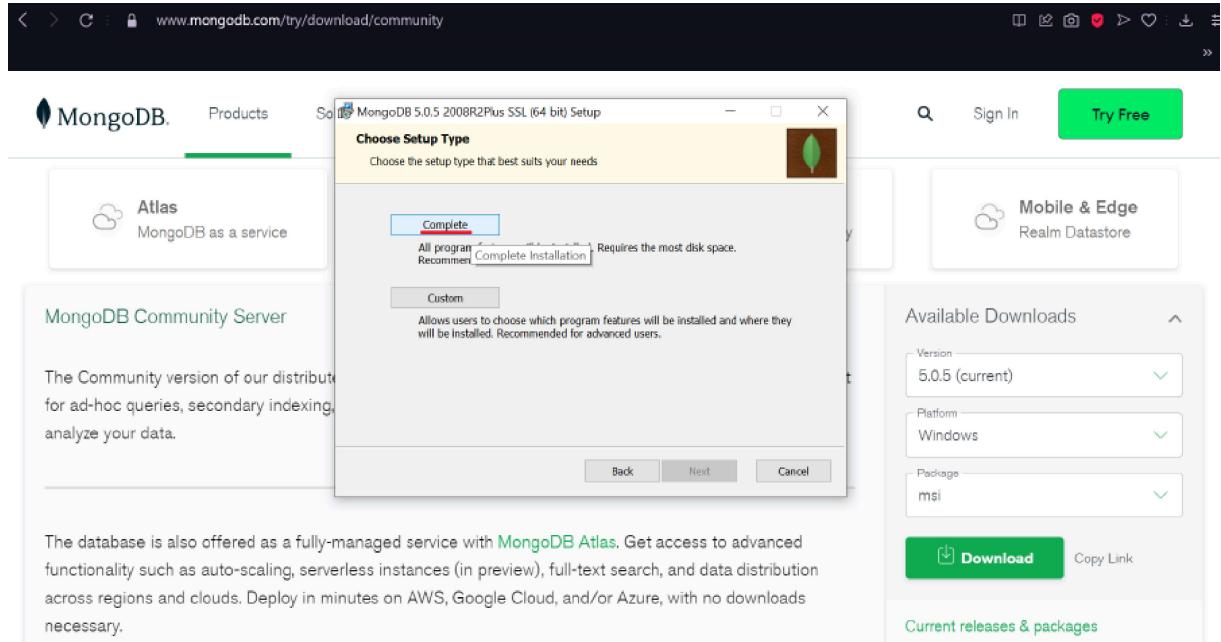
**Figura 5 – Tela inicial de instalação do MongoDB**

Fonte: <[www.mongodb.com/](http://www.mongodb.com/)>. Acesso em: 29 mar. 2022.



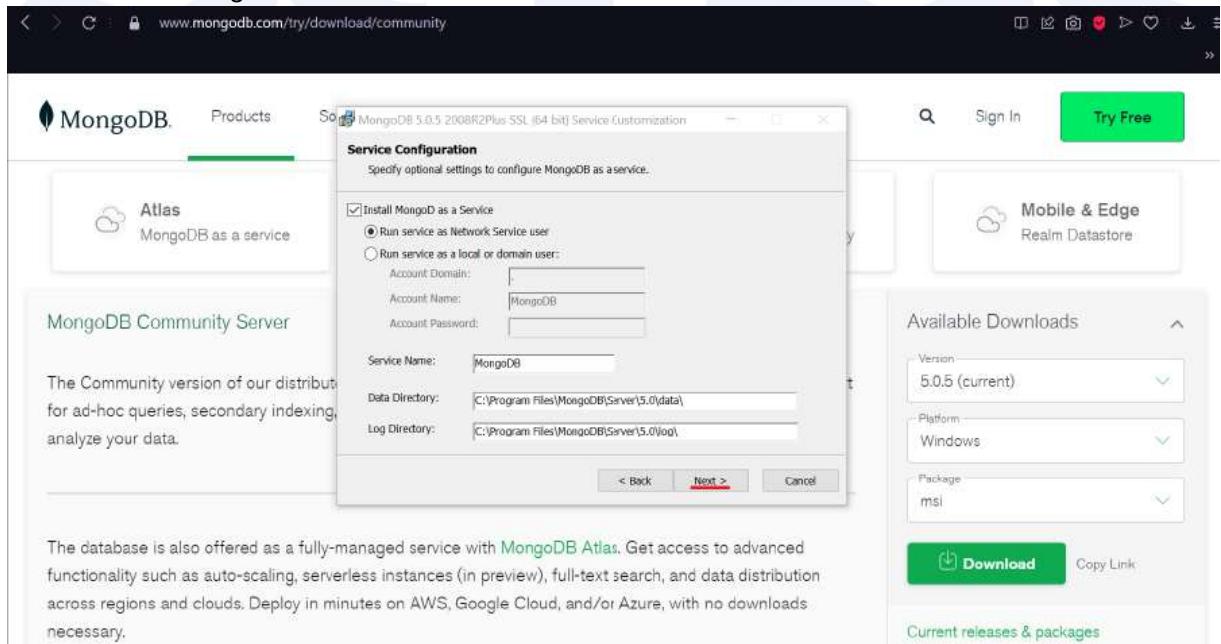
**Figura 6 – Tela seguinte de instalação do MongoDB**

Fonte: <[www.mongodb.com/](http://www.mongodb.com/)>. Acesso em: 29 mar. 2022.



**Figura 7 – Tela com duas opções de instalação do sistema**

Fonte: <[www.mongodb.com/](http://www.mongodb.com/)>. Acesso em: 29 mar. 2022.



**Figura 8 – Tela seguinte da instalação** Fonte: <[www.mongodb.com/](http://www.mongodb.com/)>. Acesso em: 29 mar. 2022.

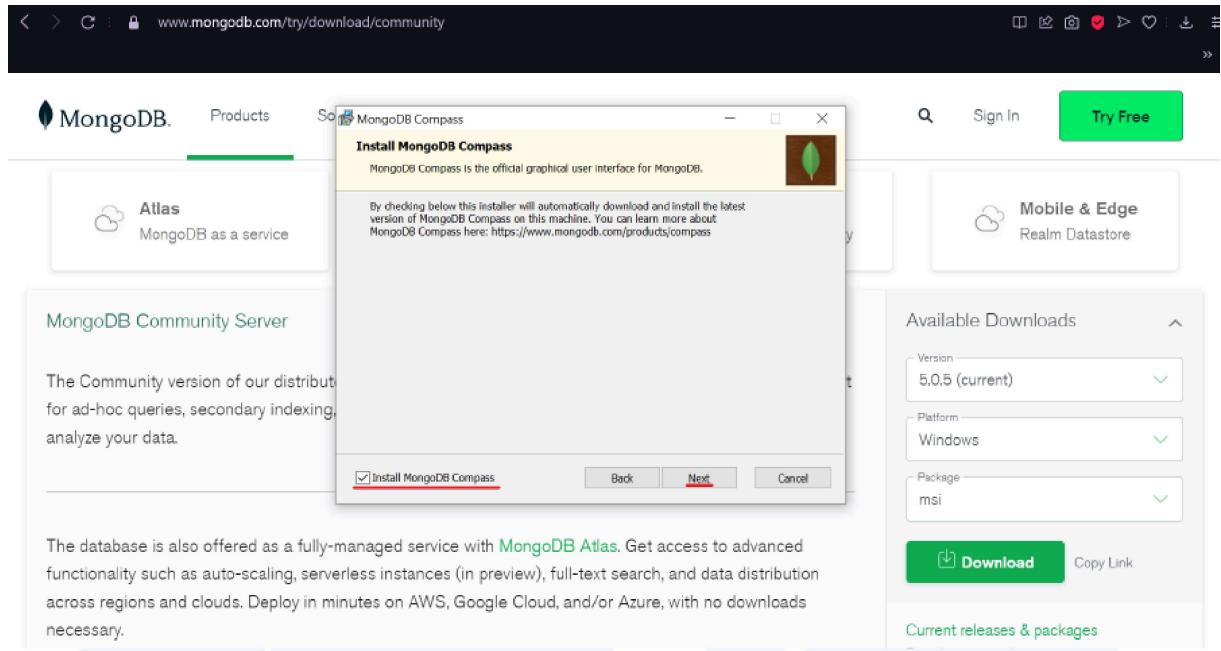


Figura 9 – Tela seguinte de instalação do MongoDB Fonte: <[www.mongodb.com/](http://www.mongodb.com/)>. Acesso em: 29 mar. 2022.

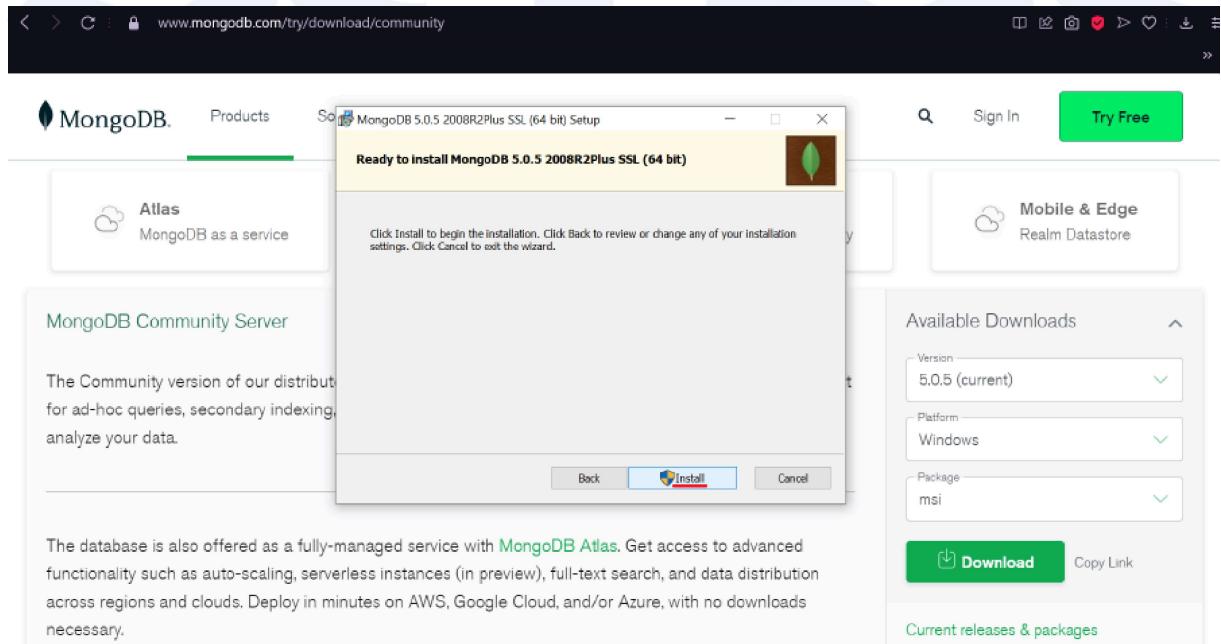


Figura 10 – Tela seguinte de instalação Fonte: <[www.mongodb.com/](http://www.mongodb.com/)>. Acesso em: 29 mar. 2022.

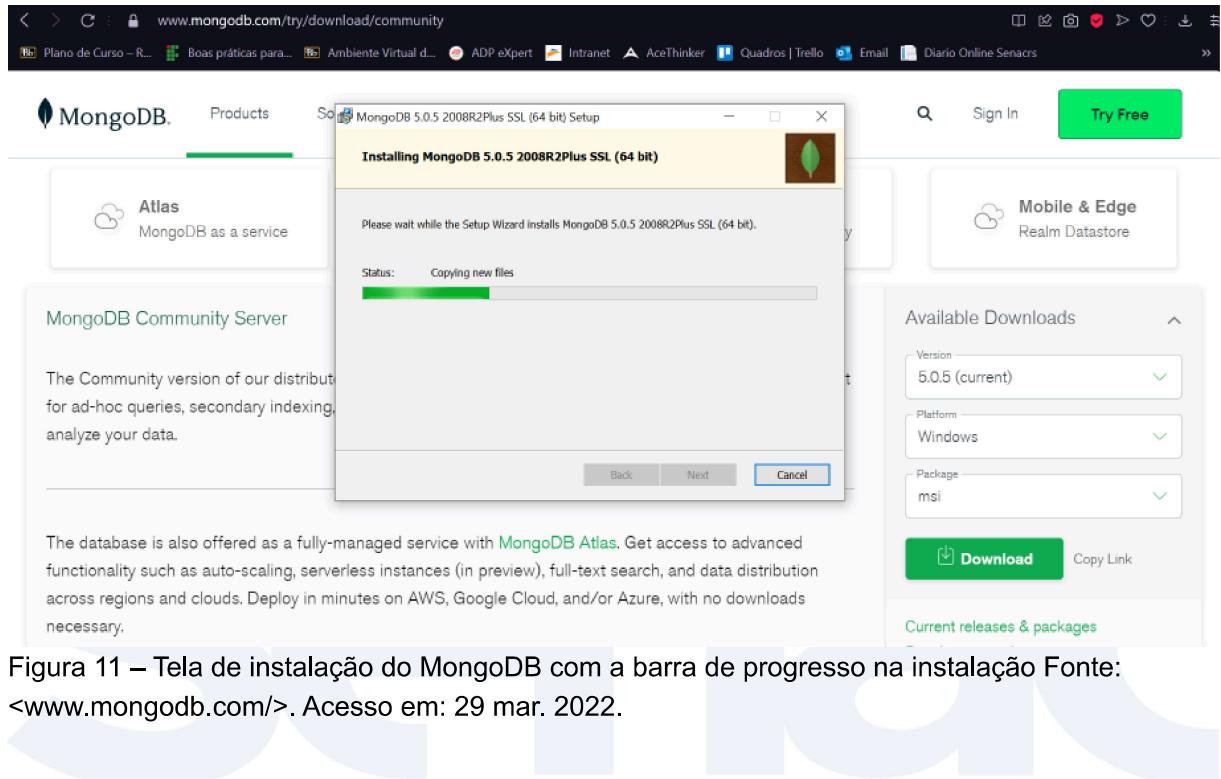


Figura 11 – Tela de instalação do MongoDB com a barra de progresso na instalação Fonte: <[www.mongodb.com/](http://www.mongodb.com/)>. Acesso em: 29 mar. 2022.

Após concluir a instalação e abrir o MongoDB Compass, você será direcionado à página inicial de conexão com sua base de dados.

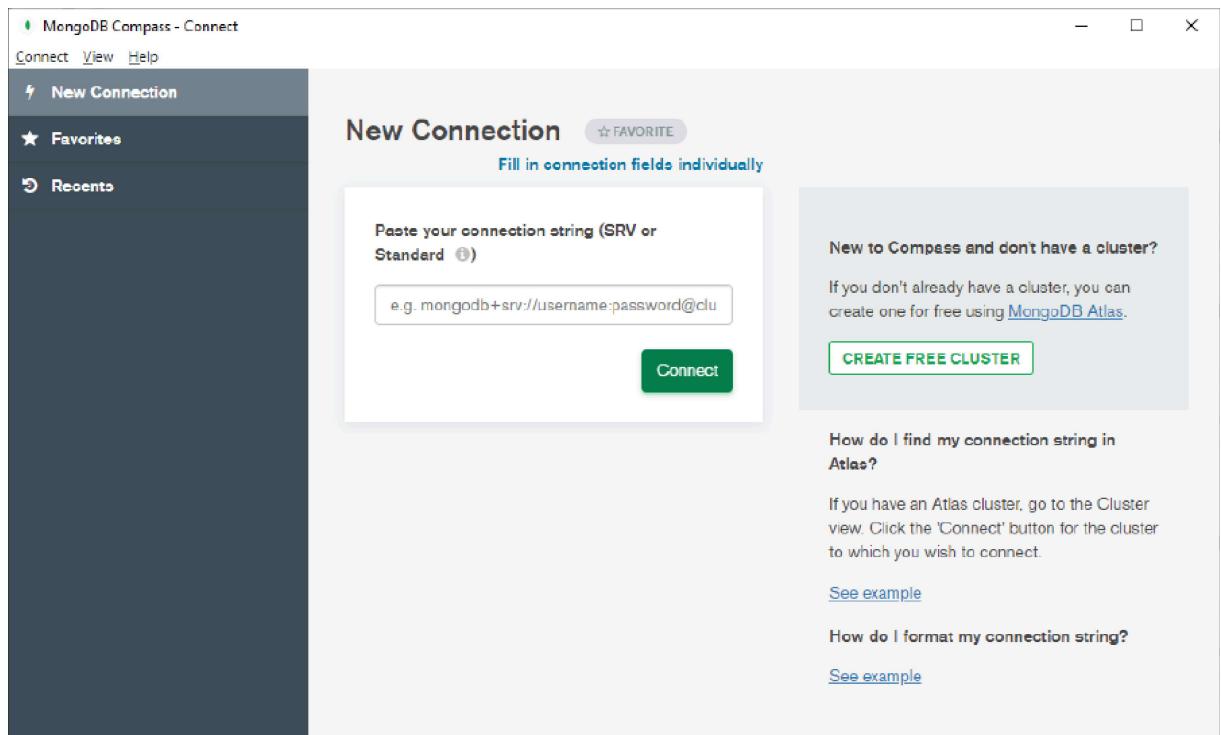


Figura 12 – Tela inicial do MongoDB Compass Fonte: <[www.mongodb.com/](http://www.mongodb.com/)>. Acesso em: 29 mar. 2022.

Caso você não tenha alterado nenhuma configuração padrão, clique no botão verde **Connect** para se conectar. Agora, você está apto a iniciar seus testes utilizando a estrutura do MongoDB.

Se você não quiser instalar o sistema para desenvolver seus códigos, poderá utilizar o **MongoDB Web Shell**, buscando por esse termo em seu navegador. Esse site simula um terminal para elaboração dos registros e permite que você execute todos os comandos, como faria em um terminal instalado em seu computador.

## Conceitos básicos

Como dito anteriormente, bancos de dados SQL contêm algumas diferenças em relação a um banco de dados NoSQL. Entre elas, a que mais se destaca é a estrutura de desenvolvimento da manipulação das informações.

Embora os conceitos sejam diferentes, para uma melhor compreensão dos termos utilizados, observe a seguir algumas comparações:

Termos SQL	Termos MongoDB
Base de dados ( <i>database</i> )	Base de dados ( <i>database</i> )
Tabelas ( <i>tables</i> )	Coleções ( <i>collection</i> )
Linhas ( <i>row</i> )	Documento ( <i>documents</i> )
Colunas ( <i>columns</i> )	Campos ( <i>fields</i> )

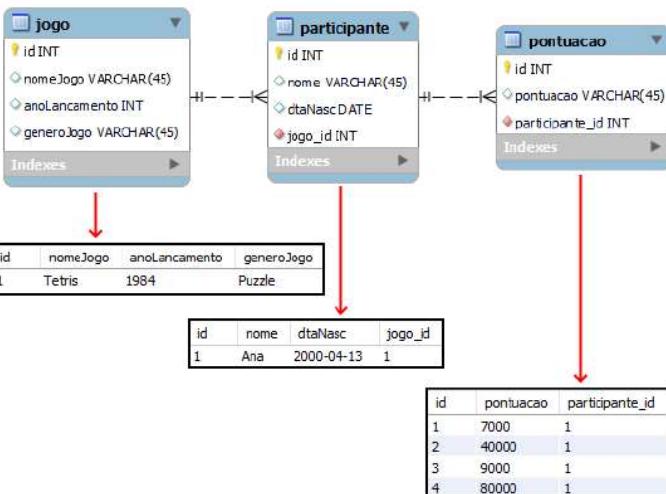
Quadro 1 – Quadro comparativo dos termos utilizados no SQL em relação aos termos do MongoDB

Ao longo deste conteúdo, haverá outras comparações para facilitar a compreensão dos termos utilizados.

Considere o seguinte contexto: Ana gosta muito de jogos! Recentemente, ela descobriu que estava acontecendo uma série de torneios de jogos em uma loja de sua cidade e logo foi participar do concurso. Sendo uma jogadora de estilo retrô, decidiu entrar na competição de Tetris. Quem diria que bloquinhos se encaixando poderiam se tornar uma competição, não é mesmo? Cada participante tinha direito a realizar cinco tentativas.

Com base nessas informações, você consegue imaginar como desenvolveria a base de dados em SQL? Você consegue imaginar como faria a consulta para obter o nome da participante, o jogo que está disputando e todas as pontuações dela?

Veja a seguir um quadro comparativo sobre como seria o desenvolvimento utilizando tabelas SQL e utilizando o MongoDB.

SQL	MongoDB (NoSQL)
 <p>Figura 13 – Entidade-relacionamento em SQL Fonte: Workbench (2022)</p> <pre> SELECT      p.nome,      pn.pontuacao,             j.nomeJogo FROM participante p JOIN pontuacao pn ON p.id = pn.participante_id JOIN jogo j ON p.jogo_id = j.id WHERE p.nome = 'Ana';     </pre>	<pre> db.participante.insertOne({   nomeParticipante: "Ana",   dtaNasc: 2000/04/13,   jogo: {     nomeJogo: "Tetris",     anoLancamento: 1984,     generoJogo: "Puzzle"   },   pontuacao: [7000, 40000, 9000, 80000] })     </pre>
	<pre> db.participante.find({nomeP "Ana"})     </pre>

nome	pontuacao	nomeJogo
Ana	7000	Tetris
Ana	40000	Tetris
Ana	9000	Tetris
Ana	80000	Tetris

```
{  
  ObjectId("622a1d99e4175f1a8cff"),  
  nomeParticipante: 'Ana',  
  dtaNasc: 38.46153846153846,  
  jogo: { nomeJogo: 'Tetris', anoLancamento: 1984, generoJogo: 'Puzzle' },  
  pontuacao: [ 7000, 40000, 9000, 80000 ]  
}
```

Quadro 2 – Quadro comparativo entre a estrutura de um banco de dados SQL e uma NoSQL

Para algumas situações e alguns contextos, um banco de dados não relacional pode apresentar soluções muito mais simples.

Observe em seguida um passo a passo para você elaborar o exemplo anterior e depois deletar ou modificar informações de que não precisa mais.

## Introdução ao uso do MongoDB e comandos iniciais de reconhecimento

Ao abrir o MongoDB Compass, confira no canto esquerdo da tela que já existem os bancos de dados que são criados por padrão: “admin”, “config” e “local”.

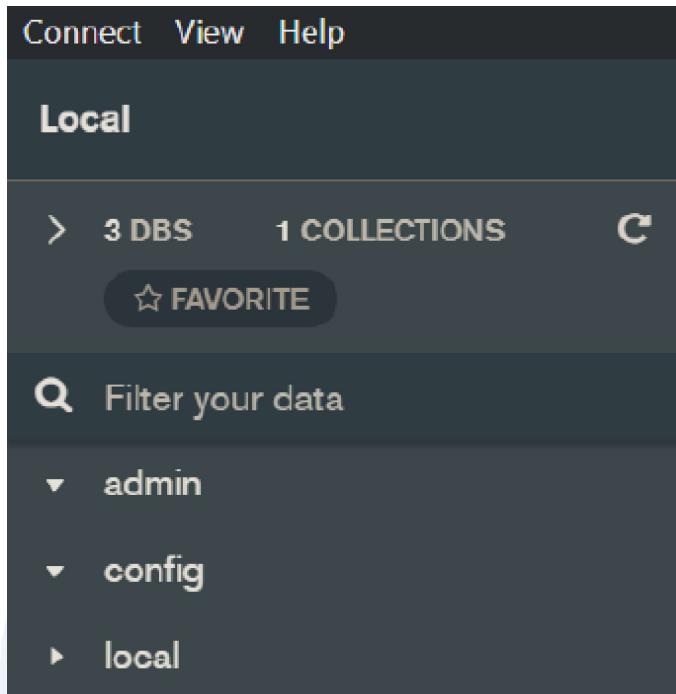


Figura 14 – Coleções iniciais padrão

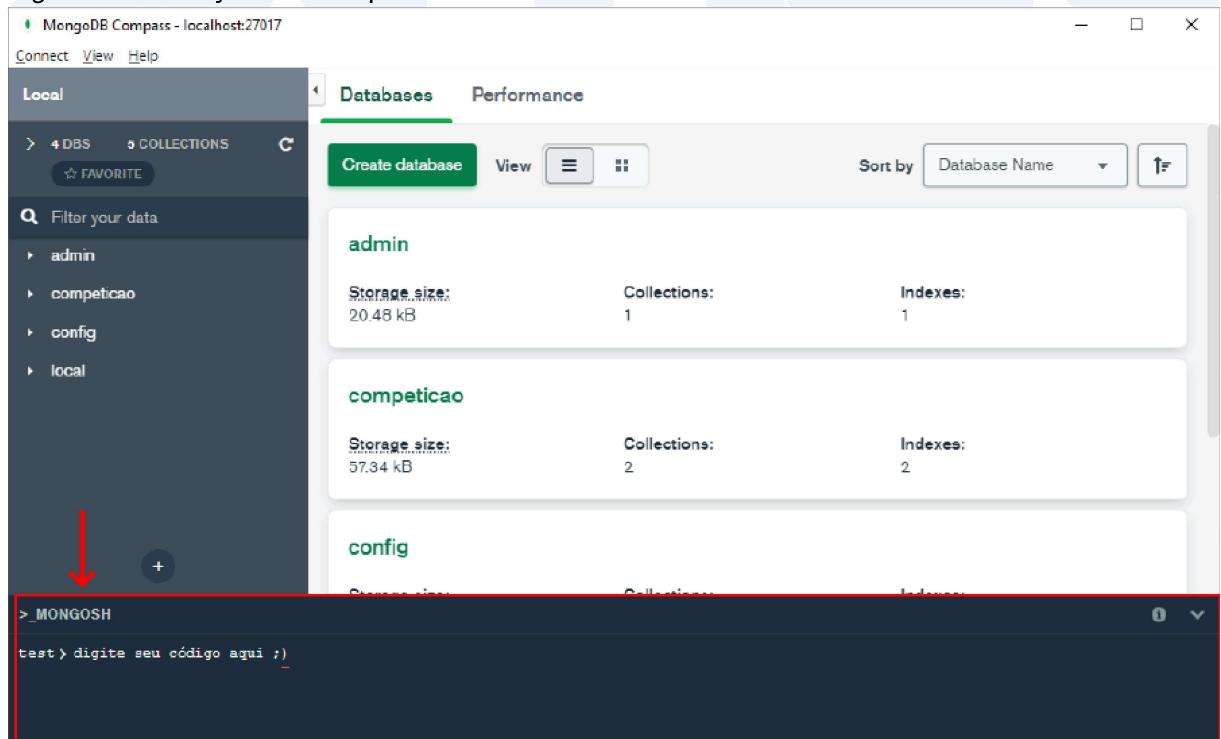


Figura 15 – Terminal \_MONGOSH

Por meio do terminal \_MONGOSH, serão inseridos os comandos de manipulação de dados. Caso você esteja realizando as consultas somente pelo terminal do sistema ou pelo terminal MongoDB Web Shell (destacado anteriormente), utilize o seguinte comando:

```
show dbs
```

Esse comando apresentará em sua tela todas as bases de dados criadas em seu sistema. O retorno será similar ao seguinte:

```
> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
```

Como padrão, os comandos estão vinculados à base “test”. Para que se possa manipular um banco de dados disponível, é preciso utilizar este comando:

```
use <nome_banco_dados>
```

Tente acessar a sua base de dados padrão “admin”.

```
> use admin
'switched to db admin'
```

A partir de agora, as operações realizadas ocorrerão no banco de dados selecionado, o banco “admin”.

Até aqui, está tudo certo. Mas, e se você quiser criar um novo banco de dados? Se você quiser manipular operações dentro da base selecionada? Acompanhe em seguida exemplos do CRUD (*create, read, update, delete*) no MongoDB.

## CRUD no MongoDB

Agora que você já explorou alguns detalhes iniciais do seu sistema, está na hora de começar a criar sua própria base de dados. Para o desenvolvimento dos códigos a seguir, será utilizada como exemplo a base de dados observada em “Conceitos básicos”, na qual se tem uma base para registrar informações de uma competição, inserindo os jogos, os participantes e suas respectivas pontuações.

## Criando uma database e realizando a primeira inserção

Como primeiro passo, é importante destacar que o processo de criar uma nova tabela utilizando o MongoDB é implícito, ou seja, não existe um comando específico para realizar a criação de um novo banco de dados. Para criar um novo banco de dados, insira o comando **use <seuNovoBancoDeDados>**, empregando o nome da base de dados que você deseja criar. Após esse comando, você já estará acessando seu banco de dados, entretanto, ele não aparecerá nos registros ainda se você utilizar o comando **show dbs**. Isso ocorre porque seu banco de dados será somente inicializado de fato quando você realizar alguma operação em sua base.

Conforme o contexto de competição de jogos apresentado anteriormente, pode-se observar o registro de um participante vinculado em um jogo e em uma pontuação. Caso você quisesse criar somente um novo banco de dados para a competição de jogos, você precisaria executar o seguinte comando:

```
use competicao
```

Com a base de dados selecionada, realize uma inserção. A sintaxe para inserir um documento é

```
db.<nome_colecao>.insertOne({ campo1: valor1, campo2: 'valor2' })
```

Lembre-se de que a coleção corresponde à tabela no banco de dados relacional. Perceba que não é necessário definir o tipo de campo.

É muito importante observar a escrita da sintaxe, pois o terminal é *case sensitive*, ou seja, ele considerará erro se houver uma letra maiúscula ou minúscula que não faça parte do comando original.

O tipo de campo é automaticamente reconhecido conforme você insere a informação, entretanto, é necessário colocar aspas simples em campos que deverão ser textos.

Observe o seguinte exemplo:

```
> db.jogo.insertOne({  
  nomeJogo: "Tetris",  
  anoLancamento: 1984,  
  generoJogo: "Puzzle"  
})
```

Esse trecho de inserção equivaleria ao seguinte código de inserção em SQL:

```
"INSERT INTO jogos (nomeJogo, anoLancamento, generoJogo) VALUES ('Tetris',  
1984, 'Puzzle')"
```

Ao digitar **Enter** no comando do Compass, a linha é executada. É possível digitar todo o *script* citado em uma linha só, mas, caso queira pular para a próxima linha, utilize **Shift + Enter**.

Após a execução desse comando, o sistema lhe retornará o seguinte trecho:

```
{ acknowledged: true,  
insertedId: ObjectId("61f1ac28148a2a6e867aee80") }
```

Esse retorno representa que o comando funcionou corretamente e lhe retornou a ID única que o próprio MongoDB gerou para o objeto inserido.

Note que, após esse comando, você estará criando não somente o banco de dados “competicao”, mas também a coleção “jogo”. Você poderá verificar isso com o comando **show dbs** ou na tela principal do Compass, clicando no ícone de atualizar e expandindo o novo banco de dados.

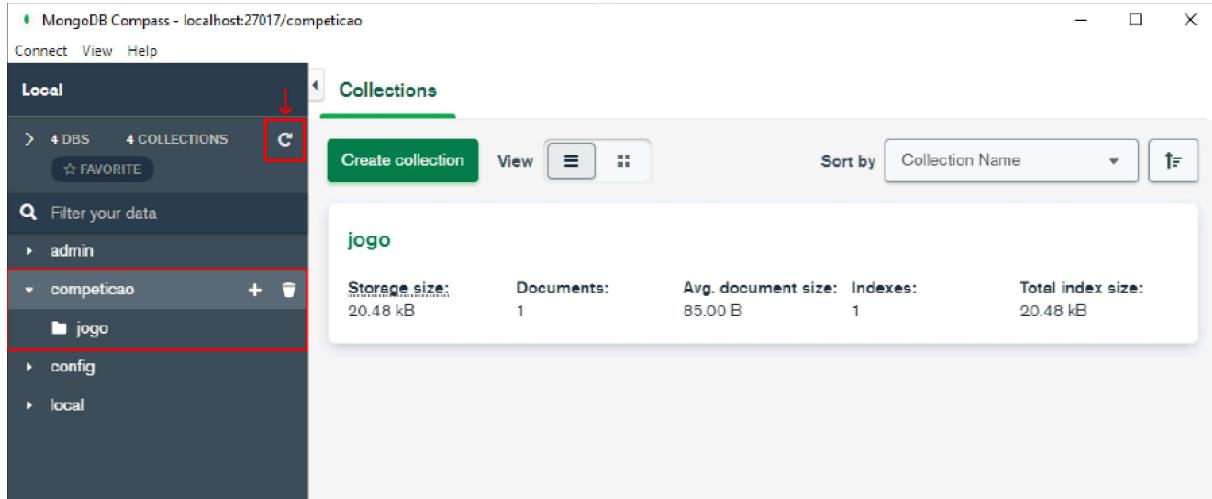


Figura 16 – Tela do MongoDB Compass mostrando o novo banco de dados

Também é possível adicionar mais de uma informação em um mesmo comando por meio da sintaxe:

```
db.insertMany( [ { x: 1, y: 2}, { x: 3, y: 4}, { x: 5, y: 6} ] )
```

Observe que, nesse modo de inserção, a informação dos dados é iniciada com um colchete e, no final da inserção, o colchete é fechado. Os documentos inicializados e finalizados com chaves são separados com vírgula para que o sistema consiga compreender que está ocorrendo a inserção de mais de um objeto. Por meio dessa inserção, o sistema retornará o número de IDs geradas correspondentes.

Uma das possibilidades disponibilizadas no processo de inserção de dados em uma coleção é a inserção de mais de um documento em um mesmo registro, ou seja, no lugar de criar várias tabelas vinculadas entre si, você pode colocar todas as informações dentro de um documento só. Veja o exemplo a seguir:

```
db.participante.insertOne({  
    nomeParticipante: "Ana",  
    dtaNasc: 2000/04/13,  
    jogo: {  
        nomeJogo: "Tetris",  
        anoLancamento: 1984,  
        generoJogo: "Puzzle"  
    },  
    pontuacao: [7000, 40000, 9000, 80000]  
})
```

Nesse trecho de código, perceba que foi possível englobar todas as informações necessárias sobre a competição em relação a um participante.

Não foi preciso criar três tabelas para realizar a relação entre informações, pois a informação de escolha de jogo da participante foi inserida como um documento.

Note também que não foram necessários quatro registros diferentes para inserir as pontuações que Ana obteve. No lugar de vários registros diferentes, bastou somente inserir um vetor (*array*) com as pontuações obtidas. Muito simples!

Observe que, apesar de incluir “jogo” no registro de “participante”, isso não significa que a coleção “jogo” do banco de dados será automaticamente povoada com esse valor informado. Não há aqui uma relação forte como a imposta por chaves estrangeiras nos bancos relacionais.

## Leitura e visualização de dados

Inicialmente, será utilizado o MongoDB Compass para uma análise visual das inserções.

Para verificar, então, as informações inseridas, na tela principal do programa, clique no nome de sua base de dados para listar as coleções. Ao selecionar uma das coleções, é possível ver os dados (documentos) que foram inseridos nela. Ao clicar no botão [>] de um dos documentos, pode-se expandir os dados.

The screenshot shows the MongoDB Compass application interface. On the left, the sidebar lists databases (Local, admin, config, local) and collections (jogo, participante). The 'participante' collection is selected and highlighted with a red box. The main pane displays the 'competicao.participante' collection details: 1 document, 4.1KB storage size, 1 index, and 4.1KB total size. Below this, the 'Documents' tab is active, showing a single document. A red box highlights the expand icon (a triangle) next to the document ID. The document details are shown in a tree view:

```

_id: ObjectId("612a1c99e4175f1a8cf898ta")
nomeParticipante: "Ana"
dataNasc: 38.46153846153846
jogo:
  nomeJogo: "Tetris"
  anoLancamento: 1984
  generoJogo: "Puzile"
pontuacao:
  0: 7000
  1: 40000
  2: 9000
  3: 60000
  
```

Figura 17 – Interface gráfica do MongoDB apresentando dados

Se houvesse mais registros correspondentes a essa coleção, eles apareceriam logo abaixo. Caso você não esteja utilizando o MongoDB Compass e queira visualizar ou simplesmente captar as informações que foram inseridas pelo terminal, utilize o seguinte comando:

```
db.<nome_colecao>.find()
```

Ao utilizar esse comando, todos os registros de sua coleção serão apresentados, pois não há nenhum filtro sendo aplicado.

Inclua agora mais alguns registros de participantes para a coleção:

```
db.participante.insertMany(  
  [  
    {  
      nomeParticipante: "Joao",  
      dtaNasc: 1995/05/05,  
      jogo: {  
        nomeJogo: "Pac-Man",  
        anoLancamento: 1982,  
        generoJogo: "Puzzle"  
      },  
      pontuacao: [100, 200]  
    },  
    {  
      nomeParticipante: "Maria",  
      dtaNasc: 1997/10/11,  
      jogo: {  
        nomeJogo: "Tetris",  
        anoLancamento: 1984,  
        generoJogo: "Puzzle"  
      },  
      pontuacao: [6000, 500, 1000]  
    }  
  ]  
)
```

Utilizando o comando **db.participante.find()**, o sistema retornará os três participantes registrados. Porém, e se você quiser filtrar a busca para que apareçam somente usuários com o nome de João? Semelhantemente ao SQL, também podem-se aplicar filtros em suas seleções de informações.

Para aplicar o filtro pelo nome dos participantes, utilize o seguinte comando:

```
db.participante.find({"nomeParticipante": "João"})
```

Por meio do objeto inserido dentro dos parênteses, o código filtrará a busca para que apareçam somente participantes com o nome de João. É possível usar mais de um campo como filtro, como a sintaxe a seguir:

```
db.participante.find({campo1:valor1, campo2:valor2})
```

Além disso, pode-se ainda aplicar filtros por campos de objetos aninhados. No exemplo de “participante”, é possível aplicar filtros por todos os jogadores que competiram no jogo Tetris – note que o campo “jogo” é um objeto com propriedades próprias dentro de “participante”. Para isso, utilize a seguinte sintaxe:

```
db.<nome_colecao>.find("<Documento>.<Campo>": "ValorPesquisado")
```

Para o exemplo em questão, seria possível consultar o seguinte:

```
db.participante.find({"jogo.nomeJogo": "Tetris"})
```

O resultado será todos os competidores de Tetris (Ana e Maria, no caso). É importante destacar que, nesta consulta, é necessário manter as aspas duplas no <Documento>.<Campo>, caso contrário, o código não será reconhecido.

O **find()** é análogo às operações de **SELECT** no banco de dados relacional. Nesse último exemplo, você utilizaria **JOIN** na consulta para alcançar o jogo no qual o participante competiu.

Existem outras formas para aplicar filtros de consulta em seus comandos de busca e especificar melhor as comparações entre tabelas. Por tal razão, sugere-se a leitura da documentação no [site oficial do MongoDB](#) para obter mais detalhes das opções disponíveis.

## Atualização de dados

Considere agora que houve um erro na hora de cadastrar o participante João. O nome dele era João Pedro, porém faltou o Pedro na hora da inserção e sua data de nascimento foi digitada incorretamente. É preciso corrigir esse erro imediatamente. Para aplicar uma atualização no sistema, você pode seguir este comando:

```
db.<nome_colecao>.updateOne("<filtro>,<Campo>")
```

No comando, especifique qual informação você está buscando no filtro e em seguida o campo a ser atualizado. Com base no contexto, o comando de atualização seria o seguinte:

```
db.participante.updateOne(  
  { "nomeParticipante" : "Joao" },  
  { $set: { "nomeParticipante" : "João Pedro", "dtaNasc" : "1999/01/27" } }  
)
```

Se você executar agora o comando **db.competicao.find()**, perceberá que a informação sobre o nome e a data de nascimento do participante foi ajustada. Em caso de sucesso, a mensagem retornada será a seguinte:

```
{ acknowledged: true,  
insertedId: null,  
matchedCount: 1,  
modifiedCount: 1,  
upsertedCount: 0 }
```

Note pelas informações **matchedCount: 1** e **modifiedCount: 1** que houve um documento encontrado e alterado com base no filtro (`nomeParticipante = "João"`). Caso houvesse mais participantes com o nome de João, poderiam ser realizadas mais especificações no filtro para tornar mais seletiva a busca pelo usuário.

O código mencionado equivale ao código SQL “UPDATE `participantes` SET `nomeParticipante = 'João Pedro'`, `dtaNasc = '1999/01/27'` WHERE `nomeParticipante = 'Joao'`”;

Da mesma forma que a inserção, é possível realizar apenas um *update* por vez, entretanto, existe também a escolha de realizar vários *updates* em um só comando:

```
db.<nome_colecao>.updateMany("<filtro>,<Campo>")
```

Também é possível adicionar novos campos no documento. No momento do *update*, basta atualizar um campo que ainda não existe e atribuir um valor padrão. Desta forma, se o campo não existia, passará a existir.

## Exclusão de documento e de *database*

No decorrer do campeonato, observou-se que Maria estava utilizando trapaças para ganhar dos outros participantes. Assim, ela foi desclassificada. Agora, você precisa excluir de sua base de dados o documento correspondente a essa participante. O comando para deletar um documento em específico é este:

```
db.<nome_colecao>.deleteOne("<filtro>")
```

Com esse código, somente um usuário será deletado com base no filtro. O usuário deletado corresponderá ao primeiro registro encontrado com o nome “Maria”, portanto, em situações reais, é válido realizar a exclusão pela ID ou por outras informações únicas, como CPF e RG. Aplique o seguinte comando neste contexto:

```
> db.participante.deleteOne({nomeParticipante:"Maria"})
```

O retorno deverá ser o seguinte:

```
{ acknowledged: true, deletedCount: 1 }
```

O participante Mario foi excluído dos registros e o retorno **deletedCount** está confirmado a exclusão de um usuário. Semelhantemente ao **INSERT** e ao **UPDATE**, também é possível excluir vários usuários de uma vez por meio do código:

```
db.<seuBancoDeDados>.deleteMany("<filtro>")
```

Com esse código, todos os documentos que contiverem as características definidas no filtro serão excluídos da base de dados. Caso não seja inserido nenhum filtro, todos os usuários serão excluídos, portanto, tenha cuidado ao utilizar o código

dessa forma.

Chegou-se então ao final do campeonato, no qual a Ana foi a vencedora do concurso de Tetris. Agora é a hora de apagar esse banco de dados, pois ele não será mais utilizado. Para isso, tem-se o seguinte comando:

```
> db.dropDatabase()  
{ ok: 1, dropped: 'competicao' }
```

Agora, sua base de dados foi apagada, sendo isso confirmado pelo retorno apresentado pelo sistema.

## Conclusão

Neste conteúdo, foi possível comparar comandos, desenvolver o CRUD, explorar e aprender um pouco mais sobre o NoSQL. Complementarmente ao estudo de uma nova linguagem de programação, é sempre interessante acompanhar a documentação oficial do que se está utilizando. Não deixe de visitar a documentação no site oficial do MongoDB para aprender mais detalhes dessa tecnologia.