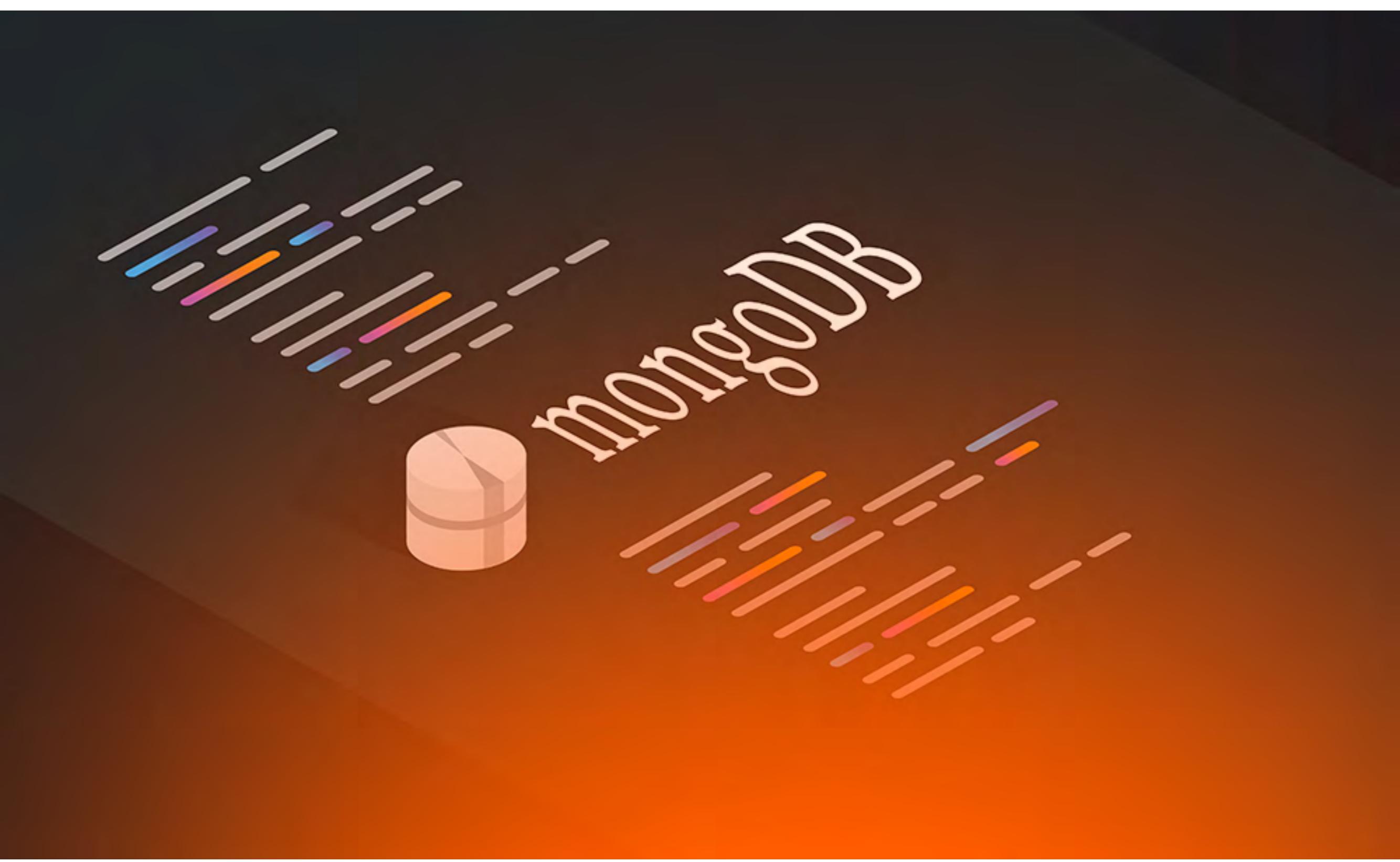


PROJETO DE DADOS EM MONGODB

Unidade 04



Sumário

03	Apresentação
04	Introdução
04	Objetivos de Aprendizagem
06	4.1 Modelagem de Dados
07	4.2 Modelo relacional de uma pequena aplicação de blog
12	4.3 Modelando documentos para uma pequena aplicação de blog
15	5. Quando um projeto MongoDB é mais vantajoso?
16	5.1 Explorando a modelagem da coleção SlotsGRU
20	6. Criando índices e analisando a execução dos acessos
21	6.1 createIndex()
22	6.2 createIndexes()
23	7. Usando o MongoDB Compass para a criação de índices
27	8. Analisando a execução dos acessos
29	9. Remoção de índices
32	10. Boas práticas de modelagem
32	10.1 Mantenha todos os dados de uma entidade em um único documento
32	10.2 Evite documentos muito grandes
33	10.3 Modele com base na forma como os dados serão consultados
34	Síntese
36	Fullture Insights

Olá, **Fullturist,**

Seja bem-vindo(a)!

Vamos acompanhá-lo(a) no desenvolvimento desta trilha de aprendizagem, a qual vai trazer informações a respeito da tecnologia e do ambiente de negócios em bancos de dados.

Pode ser surpreendente notar que esse é um tema muito importante em nossa vida atualmente. Praticamente todos os eventos de nossa vida passada, e até mesmo de nossas tendências futuras, estão registrados em um banco de dados, desde informações de nossa vida civil, documentos pessoais, registros de propriedades, nossa vida pessoal – por meio de redes sociais - e até mesmo nosso trajeto ao trabalho.

Como você pôde perceber, esse tema é vasto em nossa vida cotidiana, e a tecnologia oferecida para atender a essas diversas situações também é ampla. As opções tecnológicas são numerosas e esta trilha fornecerá a você todos os conhecimentos para entender o ambiente tecnológico e de negócios disponíveis para a solução da maioria das necessidades. Conheceremos, ainda, o que há por trás de toda essa tecnologia e como todas as informações são armazenadas e recuperadas! Para tanto, utilizaremos um dos bancos de dados mais promissores do mercado atualmente: o MongoDB.

Vamos aprender a instalar, inserir, recuperar e remover dados, bem como organizar e executar um projeto completo nesse banco de dados.

Bom estudo!

Unidade 04

Projeto de dados em MongoDB

Olá, Fullturist!

Esta é a Unidade 4 da trilha de aprendizagem Banco de Dados. Aqui, você terá a oportunidade de conhecer e/ou se aprofundar em como projetar, ou seja, como modelar o armazenamento de dados em um documento MongoDB.

A decisão de como os dados serão armazenados é intimamente ligada com a forma como eles serão recuperados pelo banco de dados. Como a modelagem de dados é uma das primeiras etapas na construção de uma aplicação, o cuidado nessa atividade é fundamental. A partir da modelagem pronta - e ainda tendo em mente como esses dados serão recuperados -, o desenvolvedor deve ter especial atenção à criação e ao monitoramento dos índices que podem aumentar o desempenho das consultas projetadas. Assim, além de conhecer como modelar, criar e monitorar os índices de uma coleção, esta unidade ainda apresentará algumas das melhores e mais utilizadas pelo mercado na modelagem de documentos.

Vamos lá?

Objetivos de aprendizagem

Ao final do estudo desta unidade, você será capaz de:

- saber como os dados são modelados em um banco de dados relacional e em um documento, identificando as diferenças entre as modelagens e por que em documentos elas são mais simples;

- criar índices simples e compostos para uma coleção, utilizando tanto o MongoDB Shell como o MongoDB Compass;
- entender que o excesso de índices em uma coleção pode impactar tanto o desempenho de consultas quanto de operações de escrita no banco de dados;
- compreender como utilizar o MongoDB Compass para analisar se o índice utilizado pelo MongoDB é a melhor opção no processamento de uma consulta;
- saber como utilizar o MongoDB Compass para avaliar índices subutilizados e como utilizar tanto o MongoDB Shell como o MongoDB Compass para removê-los;
- assimilar as melhores práticas utilizadas pelo mercado na modelagem de documentos MongoDB.

4.1. Flux e Redux

Na construção de uma aplicação, a modelagem de dados é a etapa em que o desenvolvedor avalia como os dados devem ser armazenados de forma que ao processá-los seja possível associá-los para responder às necessidades da aplicação.

Na atividade de modelagem de dados, o desenvolvedor trabalha com três informações:

- Entidades: são objetos do mundo real e são fáceis de reconhecer porque possuem uma identificação bastante clara. Um carro, um produto, um cliente e uma nota fiscal são exemplos de uma entidade.
- Relacionamento: é a quantidade de ligações que um objeto de uma entidade tem em relação aos objetos das demais entidades.
 - Relacionamento 1..1 (um para um): cada entidade se corresponde com, no máximo, um objeto da outra entidade, por exemplo, um aluno possui apenas um endereço.
 - Relacionamento 1..n ou 1..* (um para muitos): uma entidade se relaciona com vários objetos da outra entidade. Por exemplo, um professor pode lecionar várias disciplinas, porém cada disciplina possui apenas um professor.
 - Relacionamento m..n (muitos para muitos): como você já pôde deduzir, entre duas entidades, os objetos possuem múltiplos relacionamentos entre si. Evoluindo nosso exemplo, uma disciplina tem muitos alunos e, para cada aluno, há muitas disciplinas. Observe que nem os alunos são ligados a todas as disciplinas, nem todos as disciplinas são ligadas a todos os alunos.
- Atributos: são dos dados de cada entidade que interessam à aplicação, como nome, telefone endereço etc.

Refletindo com cuidado sobre os relacionamentos 1..n (um para muitos) e m..n (muitos para muitos), percebemos que “muitos” significa que existem dados que são compartilhados por mais de uma entidade. Nesses casos, ao determinar como os dados serão armazenados, o desenvolvedor deve avaliar que estratégia atende melhor a aplicação: se o dado compartilhado deve ser armazenado em uma única entidade ou se será armazenado de forma redundante no banco de dados. Essa resposta depende totalmente das características da aplicação.

Dados redundantes são péssimos para aplicações intensivas em I/O. Ao mesmo tempo, aplicações que lidam com análise de quantidades massivas de dados ou baixa latência podem se beneficiar muito da redundância controlada dos dados. Nessas situações, grandes quantidades de dados podem ser processadas sem a necessidade de relacionamento com outras entidades, o que incrementa muito o desempenho de certas aplicações.

Para que você visualize melhor esses conceitos, vamos utilizar como exemplo uma página de blog. Para essa aplicação, veremos uma modelagem relacional e NoSQL, utilizando armazenamento por documentos.

4.2 Modelo relacional de uma pequena aplicação de blog

Para a modelagem de dados, a primeira coisa a fazer é reconhecer as entidades da aplicação. Você poderia citar algumas dessas entidades para uma pequena aplicação de blog?

As Figuras 1, 2 e 3 apresentam um artigo publicado pela revista ComputerWorld e destacam algumas entidades.

Nessas telas, podemos identificar as entidades “Artigo”, “Seções do Blog”, “Autor”, “Conteúdo” e “Tags”.

The screenshot shows a news article from ComputerWorld. At the top, there's a navigation bar with categories like 'Carreira', 'Negócios', 'Segurança', etc. Below the navigation is a main title: '9 habilidades que valem a pena desenvolver para ser um programador melhor'. A subtext below the title says: 'São questões até conhecidas no mercado, mas que podem fazer bastante diferença para quem está em inicio de carreira e quer acelerar desenvolvimento'. On the left, there's a small box indicating the article was written by 'Da Redação' on '23/08/2020 às 12h01'. To the right of the article is a sidebar with an advertisement for hybrid infrastructure.

Fonte: ComputerWorld, 2021 (on-line).

1. **Lógica de programação:** Essa habilidade consiste em entender um problema e ordenar uma solução em uma sequência de comandos que seja possível de se computar.

2. **Estudo constante das linguagens:** Com o devido empenho no estudo da lógica, aprender as diversas linguagens de programação é fundamental para identificar e processar os algoritmos que resultarão no projeto. Entre as principais linguagens, estão: Javascript e Python.

3. **Dominar o inglês:** Apesar de não fazer parte da técnica de programação, saber inglês é importante para compreender os processos que envolvem a profissão, visto que a maioria dos termos estão em inglês. O profissional que buscar aprender o idioma, com certeza estará mais preparado.

4. **Bibliotecas e Frameworks:** Essas são as ferramentas de produtividade utilizadas pelos profissionais e que tem alta procura no mercado, como NodeJS, React, Redux e Flask.

5. **Bancos de dados:** armazenar e recuperar valores é parte essencial da rotina de qualquer produto ou aplicação, logo saber SQL é vital para um full-stack. Saber tanto NoSQL como MongoDB são diferenciais importantes também.

A vertical advertisement for IT services. It features a large purple 'IT' logo with small figures of people working around it. Below the logo, text reads: 'Baixe materiais sobre tecnologia gratuitos em um só lugar!'. At the bottom, there's a purple button with the text 'ACESSO JÁ!' and logos for 'itmidia' and 'itforum'.

Fonte: ComputerWorld, 2021 (on-line).

Veja também outros conteúdos de qualificação publicados pela Computerworld:

- [Conheça 6 soft skills que estão em alta dentro do mercado de tecnologia](#)
- [Google oferece curso gratuito para desenvolvimento de Kotlin para Android](#)
- [De Agile a DevOps: Mackenzie está com 5 cursos de tecnologia que são gratuitos, on-line e de curta duração](#)
- [5 cursos de tecnologia on-line e gratuitos oferecidos pelo LinkedIn](#)
- [11 habilidades que headhunters buscam em profissionais de TI](#)

Tags

[Línguagem de programação](#) [programador](#)



Mais lidas

Estas são as 50 melhores empresas para se trabalhar no Brasil, segundo os próprios funcionários

7 ferramentas gratuitas e eficazes para análise de dados

Como ativar o Wi-Fi de 5 GHz no laptop

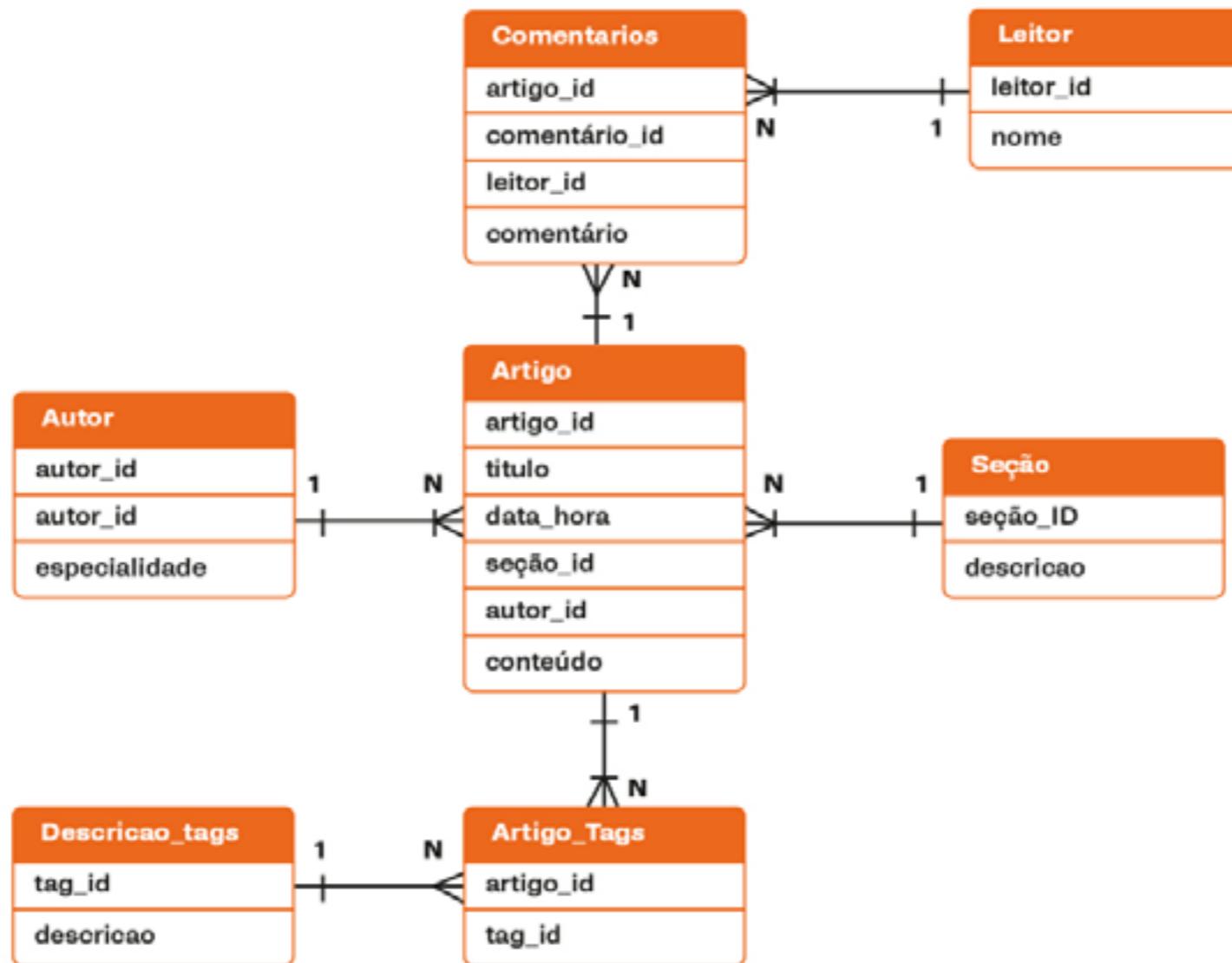
O que é a Kyndryl? A empresa spin-off de serviços de infraestrutura gerenciada da IBM explicada

Fonte: ComputerWorld, 2021 (on-line).

Além dessas entidades, podemos incluir ainda “Leitor” e “Comentários”, e assim teríamos as seguintes entidades:

- Artigo;
- Autor;
- Seção do artigo;
- Leitor;
- Comentário do leitor;
- Tags do artigo;
- Cadastro das Tags.

Se essa aplicação fosse modelada em um banco de dados relacional, uma possível modelagem pode ser aquela demonstrada pela Figura 4.



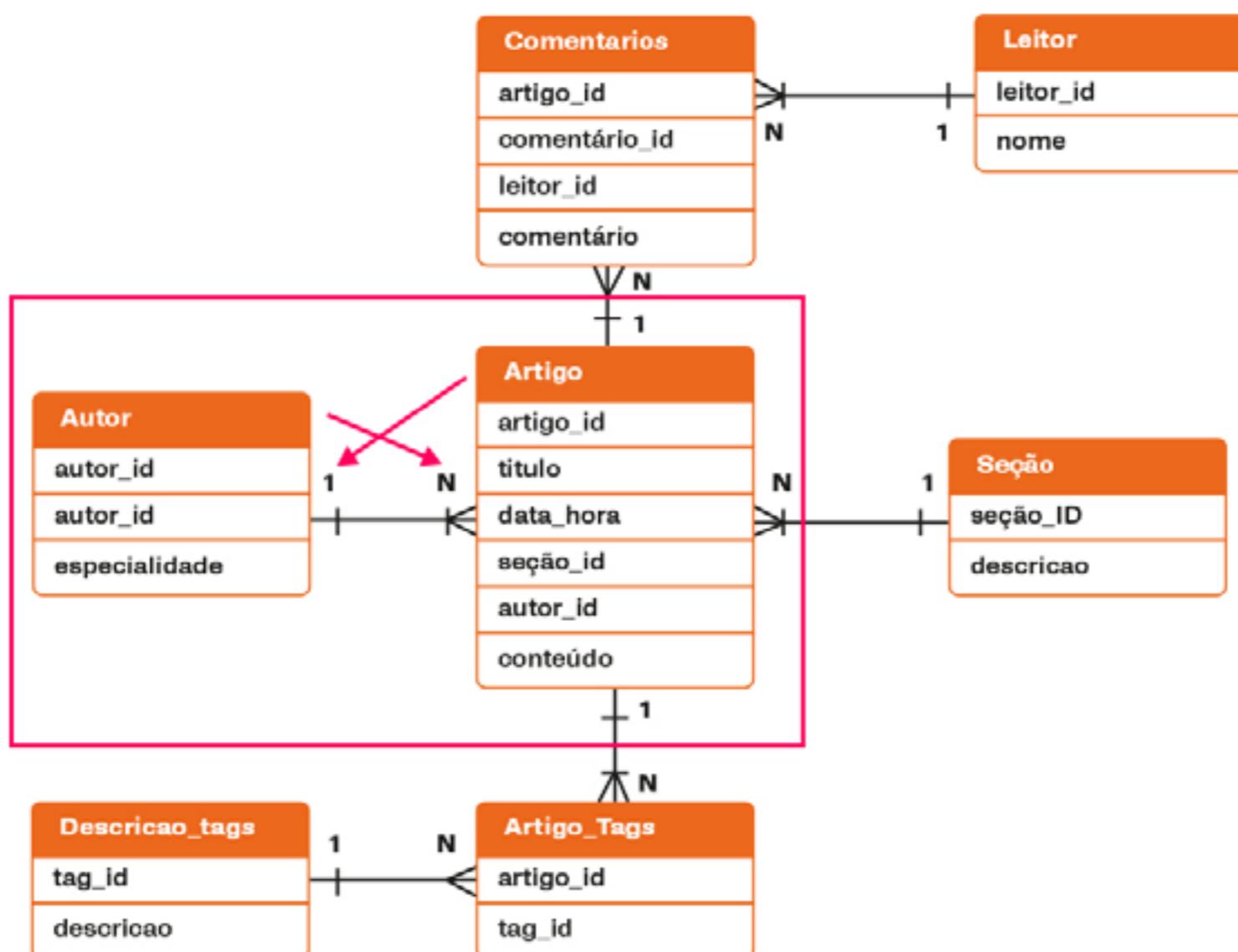
Fonte: Elaborada pelo autor (2021).

Vamos explorar esse modelo, aprendendo a ler os relacionamentos mapeados. A leitura deve ser feita sempre em pares de entidades.

EXEMPLO

Observe, na Figura 5, o relacionamento entre o Artigo e o Autor e teremos a seguinte leitura:

- 1 artigo possui 1 autor;
- 1 autor pode ter N artigos.



Fonte: Elaborada pelo autor (2021).

Agora, observe as entidades cuja ponta indica um relacionamento “para muitos”. Elas foram mapeadas para evitar a redundância dos dados, isto é, evitar a repetição de um conteúdo já gravado.

EXEMPLO

Nome do leitor;
Descrição da seção;
Descrição da Tag;
Especialidade do autor.

Em relação a esses atributos, podemos verificar que, se não fossem mapeados em entidades próprias para seu armazenamento, estariam gravados “N” vezes, de forma redundante no banco de dados. Como você já sabe, redundância em aplicações intensivas de I/O e com alta concorrência pelos dados é extremamente prejudicial ao desempenho da aplicação. Entretanto, uma aplicação de blog não é uma aplicação que se encaixe nesse perfil. Não há necessidade de se impor essa complexidade na

modelagem e no armazenamento dos dados se isso não trará um benefício concreto.

Com isso em mente, vamos então conhecer como a modelagem dessas entidades pode ser feita em um documento e como isso pode ser mais simples e ainda proporcionar maior desempenho.

4.3 Modelando documentos para uma pequena aplicação de blog

Modelagem de dados não é uma ciência exata, e para um mesmo objetivo pode-se modelar os dados de várias formas. Entretanto, diferentemente de bancos relacionais, que exigem um esquema fixo para armazenamento dos dados, e uma vez definido, qualquer alteração é trabalhosa, bancos NoSQL trabalham sem esquema, e poder modelar livremente os dados conforme necessidades específicas da aplicação é bastante produtivo.

Embora se possa trabalhar livremente, ainda podemos pensar em um processo de trabalho que embase o raciocínio de modelagem. Para modelar o armazenamento em documentos, podemos, primeiramente, avaliar as entidades envolvidas e determinar qual delas é o foco principal da aplicação.

O desenvolvedor pode tirar proveito da agregação de dados proporcionada por essa forma de armazenamento, projetando o documento para ter foco nessa entidade e ao mesmo tempo agregar no mesmo documento os dados das demais entidades.

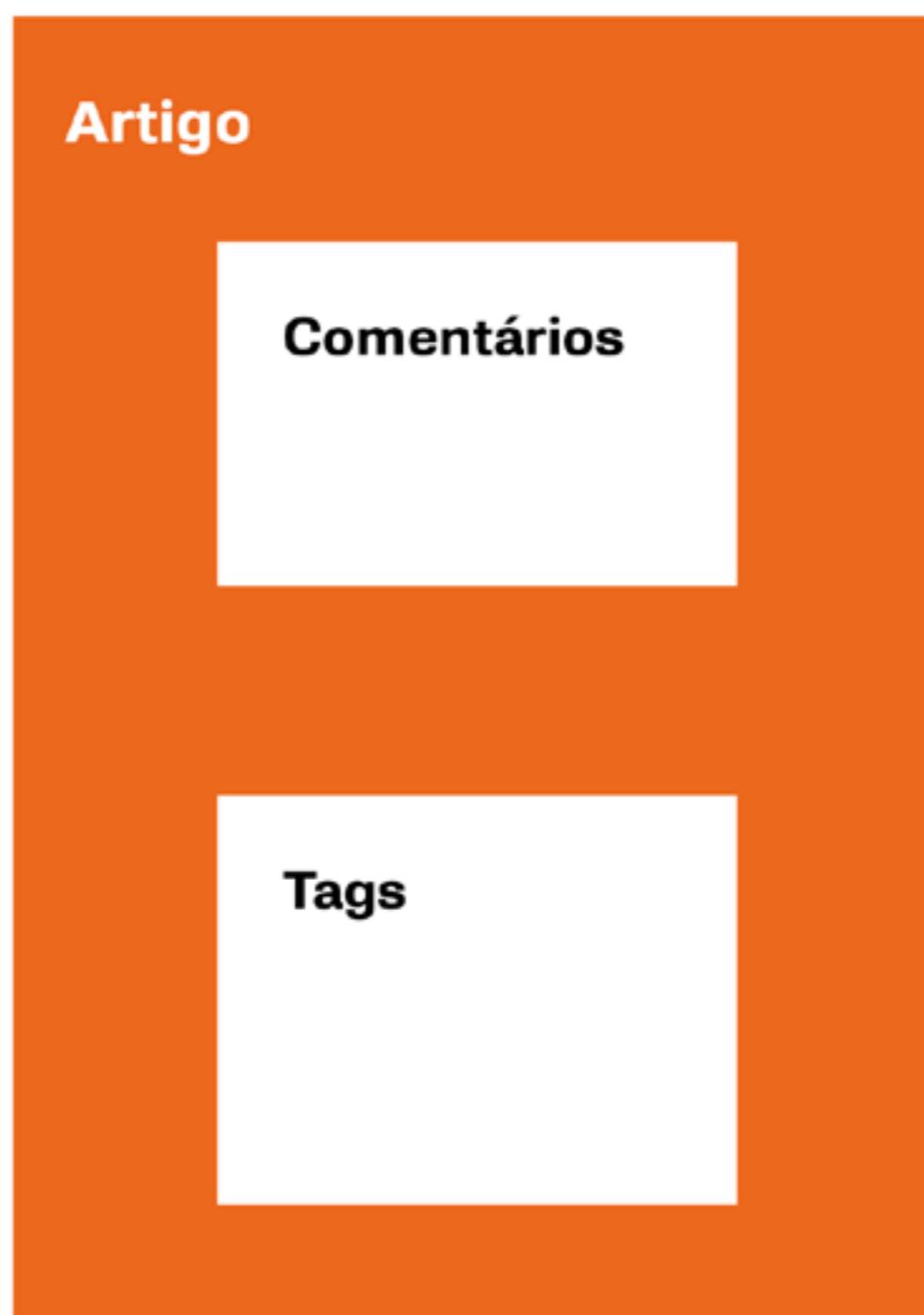
Por exemplo, em relação ao foco do documento e suas agregações:

com foco no Artigo: o documento pode agregar as informações dos comentários para responder a questionamentos como “quais artigos foram comentados mais positivamente”;

com foco Autor: o documento pode agregar as informações sobre as Tags e Comentários e responder a questionamentos como “os autores e os assuntos que geraram mais avaliações positivas ou negativas”;

com foco no Leitor: o documento pode agregar as informações sobre os comentários, respondendo a questionamentos como “quais os leitores mais ativos em determinados assuntos ou qual a qualidade de suas avaliações”.

Seguindo esse raciocínio, tendo como foco o Artigo e agregando as informações das demais entidades, teremos uma representação gráfica de um documento como apresentado pela Figura 6.



Fonte: Elaborada pela autora (2021).

Essa representação foi construída com base no raciocínio já exposto:

- O documento principal conterá as informações do Artigo e os dados de seus relacionamentos 1..1, no caso, os dados da Seção e do Autor.
- Além disso, agregará as relações 1..n em documentos ou arrays aninhados dentro do próprio documento Artigo, no caso, os documentos Comentários e Tags.
- Cada documento aninhado conterá suas relações 1..1. O documento Comentários conterá as informações do Leitor, e o documento Tags conterá a descrição de cada Tag associada ao Artigo.

Usando esses passos para construir efetivamente o documento, teríamos a modelagem da Figura 7. É interessante que você faça uma comparação cuidadosa entre o modelo relacional, apresentado pela Figura 4 e o documento apresentado pela Figura 7. No modelo relacional, cada entidade modelada será uma tabela armazenada no banco de dados, contendo seus respectivos atributos e os relacionamentos terão que ser codificados pelo desenvolvedor. Percebe-se como a modelagem em um único documento é muito mais simples. Entretanto, você não deve esquecer que cada tipo de banco de dados foi desenvolvido para endereçar necessidades próprias e que saber avaliar e escolher a melhor forma de armazenamento para os dados da aplicação é passo fundamental para o sucesso do desenvolvimento.

```
{
  "artigo_id": "6679",
  "titulo": "9 habilidades que valem a pena desenvolver para ser um programador melhor",
  "data_hora": "23/08/2020 12h01",
  "conteúdo": "São questões até conhecidas no mercado.....",
  "secao": { "secao_id": "1", "descricao": "Carreira" },
  "autor": { "autor_id": "234", "nome": "da redação", "especialidade": "Publicações em TI" },
  "tags": [
    {
      "tag_id": "34", "descrição": "Linguagem de programação" },
      {"tag_id": "15", "descrição": "Programador" }
    ],
  "comentários": [
    {
      "comentário_id": "1", "leitor_id": "987", "nome": "Felipe Guerra" }
    ]
}
```

Observe que a especificação de arrays ou documentos aninhados assume que poderá haver redundância de dados. No documento da Figura 7, o “nome” do leitor será repetido em todos os comentários que ele fizer. Como o MongoDB é escalável e processa com bom desempenho altos volumes de informação, isso não será um problema. Entretanto, o desenvolvedor deverá ter atenção quando a aplicação permitir alterações em campos redundantes. Dependendo dos requisitos, obviamente, alterações em campos redundantes devem ser propagadas a todos os documentos que armazenam o mesmo conteúdo.

Conhecendo como modelar os dados, no próximo tópico, você conhecerá quais situações o MongoDB é vantajoso.

5. Quando um projeto MongoDB é mais vantajoso?

Você já sabe que duas características muito interessantes no MongoDB são a possibilidade de trabalhar sem um esquema rígido e também agregar em um único documento todas as informações de interesse. Pensando nessas características, e com o conhecimento que você já possui sobre a modelagem de dados em documentos, reflita sobre as situações em que o MongoDB é especialmente útil:

- na modelagem de entidades complexas cujos atributos evoluem rapidamente ou variam entre os objetos da entidade;
- dados em que a redundância controlada pode beneficiar muito o desempenho da aplicação;
- aplicações que precisam se adaptar rapidamente a mudanças;
- quando a escalabilidade ou a elasticidade no armazenamento é fator determinante para o sucesso da aplicação;
- aplicações com baixa latência, por exemplo, streaming;
- aplicações que necessitam de alta escalabilidade no processamento de volumes massivos de informação

Percebeu como a modelagem por documentos e a escalabilidade do MongoDB atendem essas situações?

Que tal voltarmos ao nosso database FlightSlots?

Nos próximos tópicos, analisaremos a modelagem dos dados da coleção SlotsGRU e veremos como esses dados são usados pelo site do aeroporto de Guarulhos.

5.1 Explorando a modelagem da coleção SlotsGRU

Neste tópico, vamos analisar a modelagem dos documentos da coleção SlotsGRU. Uma vez que você usou essa coleção em exercícios e testes na Unidade 3, aconselho fortemente que você exclua a coleção e importe novamente seus dados. Vamos utilizá-los neste e nos próximos tópicos desta unidade.

Primeiro, será necessária a remoção da coleção SlotsGRU.

Vamos conhecer como fazer isso:

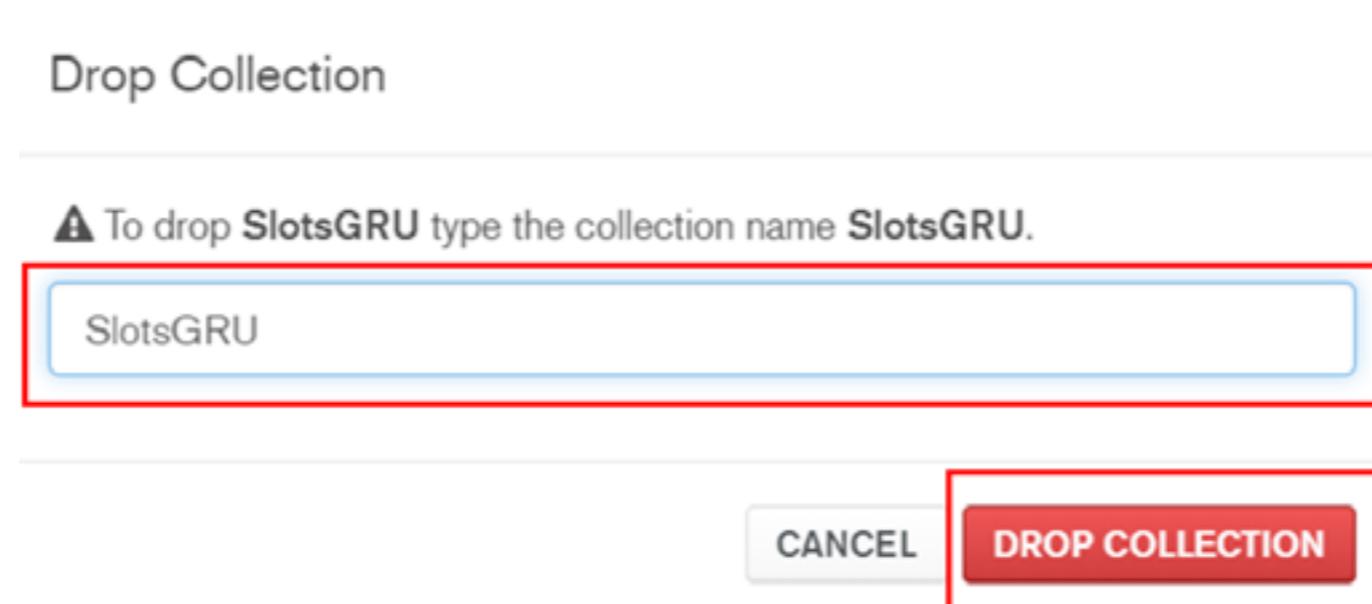
- acesse o MongoDB Compass;
- conecte com o MongoDB;
- selecione o database FlightSlots.

Ao selecionar o database, a tela da Figura 8 será apresentada. Observe que à direita da coleção SlotsGRU você verá um ícone para a exclusão dessa coleção.

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
SlotsGRU	54,119	388.5 B	20.1 MB	1	952.0 KB	

Fonte: Adaptado de MongoDB (2021).

Acione o ícone e confirme a exclusão da coleção. A Figura 9 apresenta essa operação.



Fonte: Adaptado de MongoDB (2021).

Uma vez que a coleção SlotsGRU é a única coleção no database, o database FlightSlots também será excluído. Você necessitará preparar novamente o ambiente de testes, da mesma maneira como foi feito na Unidade 3. Caso você precise relembrar esses passos, essas informações estão nos seguintes tópicos da Unidade 3:

■ Tópico 4 – Preparando uma base de testes

■ Recrie o database FlightSlots;

■ Recrie a coleção SlotsGRU.

■ Tópico 7 – Aumentando a base de testes

Acesse e faça download do arquivo “SlotsGRU–Completo.txt” a partir da área de materiais complementares desta unidade. Observe que, na Unidade 3, a totalidade dos documentos foi carregada em duas etapas (arquivos “SlotsGRU-insertMany.txt” e “SlotsGRU.txt”). Dessa vez, você vai trabalhar com o arquivo “SlotsGRU–Completo.txt” que já contém a totalidade desses documentos. Consulte [aqui](#) o arquivo “SlotsGRU–Completo.txt” que será usado ao longo desta unidade.

■ Importe o arquivo “SlotsGRU–Completo.txt”, populando novamente a coleção SlotsGRU.

Com os documentos carregados na coleção SlotsGRU, utilize o MongoDB Compass para visualizar os documentos dessa coleção. A Figura 10 apresenta os dados armazenados pelo documento.

The screenshot shows the MongoDB Compass interface with the database 'FlightSlots' selected. In the 'Documents' tab, a single document is displayed. The document has the following fields and values:

- `_id: ObjectId("613ba52f5a63de99ab94809d")`
- `TipodeOperacao: "A"`
- `CodEmpresaAerea: "2F"`
- `NumerodoVoo: "5600"`
- `HorariodoVoo: "02:30"`
- `DatadoVoo: "2021-03-29"`
- `DiadaSemana: "1"`
- `Equipamento: "208"`
- `Assento: "0"`
- `AeroportodeOrigemouDestino: "QDV"`
- `Aeroportodaetapaanteriorouse... : "QDV"`
- `Tipodeserviço: "F"`
- `Terminal: "CARGO-DOM"`
- `PaísdeOrigemouDestino: "BR"`
- `Paísdaetapaanteriorouseguinte: "BR"`

Fonte: Adaptado de MongoDB (2021).

Percebemos que essa modelagem é muito simples e não possui redundância de dados, ou seja, não há as descrições dos aeroportos ou o nome da empresa aérea, apenas suas identificações. Percebe-se que essa modelagem não se destina à consolidação de valores ou aplicações analíticas. Elas podem ser informações para utilização, por exemplo, em consultas simples. Modelada dessa forma, a aplicação encontra todas as informações no mesmo documento. Não é necessário codificar relacionamentos para a demonstração da informação de interesse.

Que tal fazer uma consulta às partidas e decolagens de Guarulhos no próprio site do aeroporto e ver o que encontramos? A Figura 11 apresenta esta página do site.

HORÁRIO	DESTINO	LIA	VOO	TERMINAL	ESTIMADO	STATUS
013:50	Maceió	LATAM	4526	Terminal 2	013:50	Decolado
013:55	Recife	LATAM	3148	Terminal 2	013:55	Partindo
014:00	Rio de Janeiro	LATAM	3187	Terminal 2	014:00	Partindo

Fonte: GRUAirport, 2021 (on-line).

Vemos, nesta tela, algumas informações do voo 3187 da Latam. Muito embora o arquivo “SlotsGRU—Completo.txt” contenha dados atualizados até data de 16 de março 2021 (essa informação está na tela em que o arquivo foi disponibilizado, no Portal Brasileiro de Dados Abertos), podemos consultar em nossa coleção alguns dados anteriores a essa data. Execute a consulta da Figura 12 no MongoDB Compass para o voo 3187.

```

{
    "_id": "60c18f1f0000000000000001",
    "TipoOperacao": "D",
    "CodEmpresaAerea": "31",
    "NumeroVoo": "3187",
    "HorarioLidoVoo": "13:45",
    "DataLidoVoo": "2021-06-01T13:45:00Z",
    "Itinerario": "2",
    "Equipamento": "318",
    "Assento": "144",
    "AeroportoDeOrigemVoo": "SOU",
    "AeroportoDeEtapasAnterior": "SOU",
    "TipoDesvio": "I",
    "Terminal": "001Q",
    "PaisDeOrigemDestino": "BR",
    "PaisDaEtapasAnteriores": "BR"
}

```

Fonte: Adaptado de MongoDB (2021).

É uma pena que o arquivo não contenha informações mais atualizadas. Seria muito interessante executar essas consultas com informações do momento. Entretanto, você pode verificar a modelagem do documento e associar com as informações apresentadas no site do aeroporto. Caso queira executar outras consultas na coleção SlotsGRU e verificar os voos no site do aeroporto, você deve usar um filtro no campo “Assento”. Os voos de passageiros possuem “Assento” diferente de zero (Assento:{\$ne:"0"}).

Outro assunto importante e que não poderia deixar de ser abordado é a criação e a utilização de índices adequados às consultas. Em processamentos de consultas com volumes massivos de dados a utilização de índices é fundamental. No próximo tópico, veremos como fazer isso.

6. Criando índices e analisando a execução dos acessos

Resumidamente, índice sem banco de dados são como índices em um livro. Amazonam duas informações: o conteúdo do campo indexado e o endereço físico do documento em que o campo está armazenado. Dessa forma, em vez de varrer todos os documentos em busca das informações solicitadas pela consulta, o MongoDB, primeiramente, acessa o índice e, de posse do endereço físico, busca diretamente o documento de interesse.

Além do índice único (`_id`), criado automaticamente pelo MongoDB na inclusão do documento, neste tópico você conhecerá que o MongoDB também possibilita ao desenvolvedor a criação de índices que possam aumentar o desempenho de consultas da aplicação. É possível criar índices em um único campo de interesse com o uso do método `createIndex()` e também criar índices compostos por vários campos com o método `createIndexes()`. Vamos começar pelo método `createIndex()`, o qual cria índice em um único campo.

6.1 createIndex()

A construção do método é bastante simples e a Figura 13 apresenta a sintaxe mais básica e também um exemplo de construção do método.

Sintaxe

```
db.collection.createIndex( < documento que define o índice > )
```

Exemplo

```
db.SlotsGRU.createIndex( { NumerodoVoo:1 } )
```

Fonte: Adaptado de MongoDB (2021).

Como você pôde verificar, a sintaxe segue o padrão dos métodos CRUD. Informa a coleção e o documento que define o índice. No exemplo, o campo a ser indexado é o NumerodoVoo e o valor “1” indica que o índice será criado de forma ascendente. Quando o campo a ser indexado é definido como “-1”, o ordenamento do índice será decrescente. A Figura 14 apresenta a execução desse comando no MongoDB Shell.

```
>_MONGOSH  
> use FlightSlots  
< 'switched to db FlightSlots'  
> db.SlotsGRU.createIndex( {NumerodoVoo:1} )  
< 'NumerodoVoo_1'  
FlightSlots >
```

Fonte: Adaptado de MongoDB (2021).

Observe que, ao especificar o método, não foi informado um nome para a criação do índice. Na tela da Figura 14 você observa que, ao executar o método, o MongoDB automaticamente cria um nome para o índice construído.

No link a seguir você encontra um guia de referência para o método `createIndex()` no MongoDB:

Há consultas em que um índice composto é mais apropriado. No caso da coleção SlotsGRU, em uma consulta que tenha como objetivo identificar todos os voos de uma companhia aérea, seria muito vantajoso um índice composto pelos campos CodEmpresaAerea e NumerodoVoo. Para isso, o método `createIndexes()` é mais apropriado. No próximo tópico, veremos como usar esse método.

6.2 `createIndexes()`

O método `createIndexes()` possibilita a criação de índices compostos por dois ou mais campos. A Figura 15 apresenta a sintaxe mais básica e um exemplo de utilização dela. Na Figura 16 você pode verificar a execução do método.

Sintaxe

```
db.collection.createIndexes( [ <Array de documentos definindo os campos do índice> ] )
```

Exemplo

```
db.SlotsGRU.createIndexes( [ { CodEmpresaAerea:1 }, { NumerodoVoo:1 } ] )
```

Fonte: Adaptado de MongoDB (2021).

```
>_MONGOSH  
  
> use FlightSlots  
< 'switched to db FlightSlots'  
> db.SlotsGRU.createIndexes([ {CodEmpresaAerea:1}, {NumerodoVoo:1} ])  
< [ 'CodEmpresaAerea_1', 'NumerodoVoo_1' ]  
FlightSlots >
```

Fonte: Adaptado de MongoDB (2021).

Da mesma forma que o método `createIndex()`, `createIndexes()` também nomeia automaticamente o índice definido pelo método.

No link a seguir você encontra um guia de referência para o método `createIndex()` no MongoDB:

A criação de índices também pode ser feita de maneira bastante intuitiva pelo MongoDB Compass. Vejamos como isso é feito.

7. Usando o MongoDB Compass para a criação de índices

O MongoDB Compass oferece uma funcionalidade bastante interativa para a criação e visualização dos índices de uma coleção. Observe a Figura 17.

The screenshot shows the MongoDB Compass interface. On the left, the sidebar displays the database structure: Local, HOST (localhost:27017), CLUSTER (Standalone), EDITION (MongoDB 5.0.2 Community), and collections (FlightSlots, SlotsGRU, Vendas, admin, config, local). The main window shows the 'FlightSlots.SlotsGRU' collection. At the top, there are tabs for Documents, Aggregations, Schema, Explain Plan, Indexes (which is highlighted with a red box), and Validation. Below these tabs, a green button labeled 'CREATE INDEX' is visible. The 'Indexes' section displays three existing indexes: 'CodEmpresaAerea_1' (Type: REGULAR, Size: 286.7 kB, Usage: 0 since Wed Sep 22 2021), 'NumroVoo_1' (Type: REGULAR, Size: 438.3 kB, Usage: 0 since Wed Sep 22 2021), and '_id' (Type: REGULAR, Size: 696.3 kB, Usage: 20 since Wed Sep 22 2021, Properties: UNIQUE).

Fonte: Adaptado de MongoDB (2021).

A figura acima apresenta as informações dos índices criados nos tópicos anteriores. Para visualizar esses índices, na tela em que se acessa os documentos da coleção, acione a aba “Indexes”. Nessa aba, você verá todos os índices criados para a coleção.

Caso você encontre dificuldade em visualizar os índices, acione o ícone “Open new tab” em destaque na Figura 18. As informações serão carregadas.

Name and Definition	Type	Size	Usage	Properties
CodeEmpresaria_1	REGULAR	286.7 kB	0	
NumerodoVoo_1	REGULAR	438.3 kB	0	
HorarioVoo_1	REGULAR	696.3 kB	20	UNIQUE

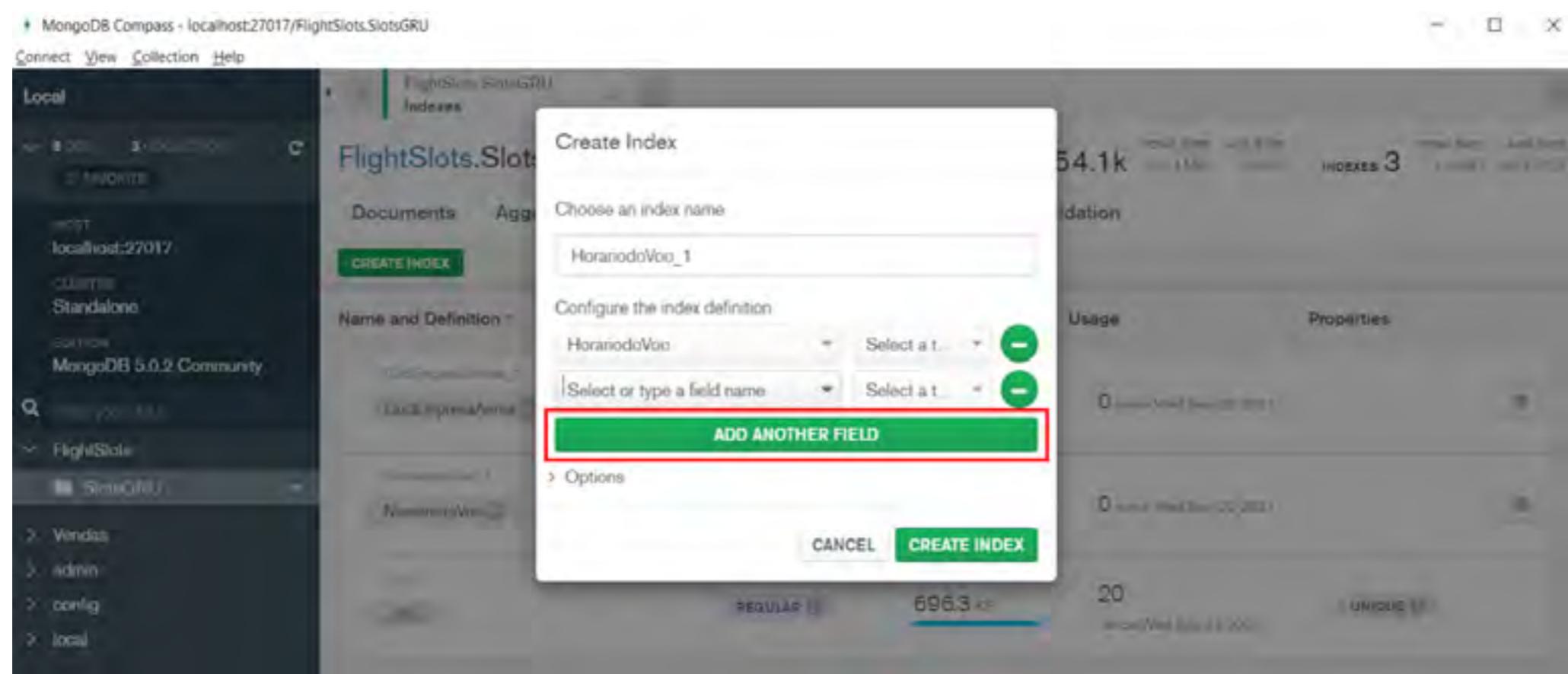
Fonte: Adaptado de MongoDB (2021).

Vamos pensar em uma consulta que demonstre todos os slots em cada horário. Para uma consulta como essa, poderíamos criar um índice composto por “Horario do Voo + NumerodoVoo + CodEmpresaAerea”. Acione o botão CREATE INDEX apresentado pela aba “Indexes” e a tela da Figura 19 será apresentada.

Fonte: Adaptado de MongoDB (2021).

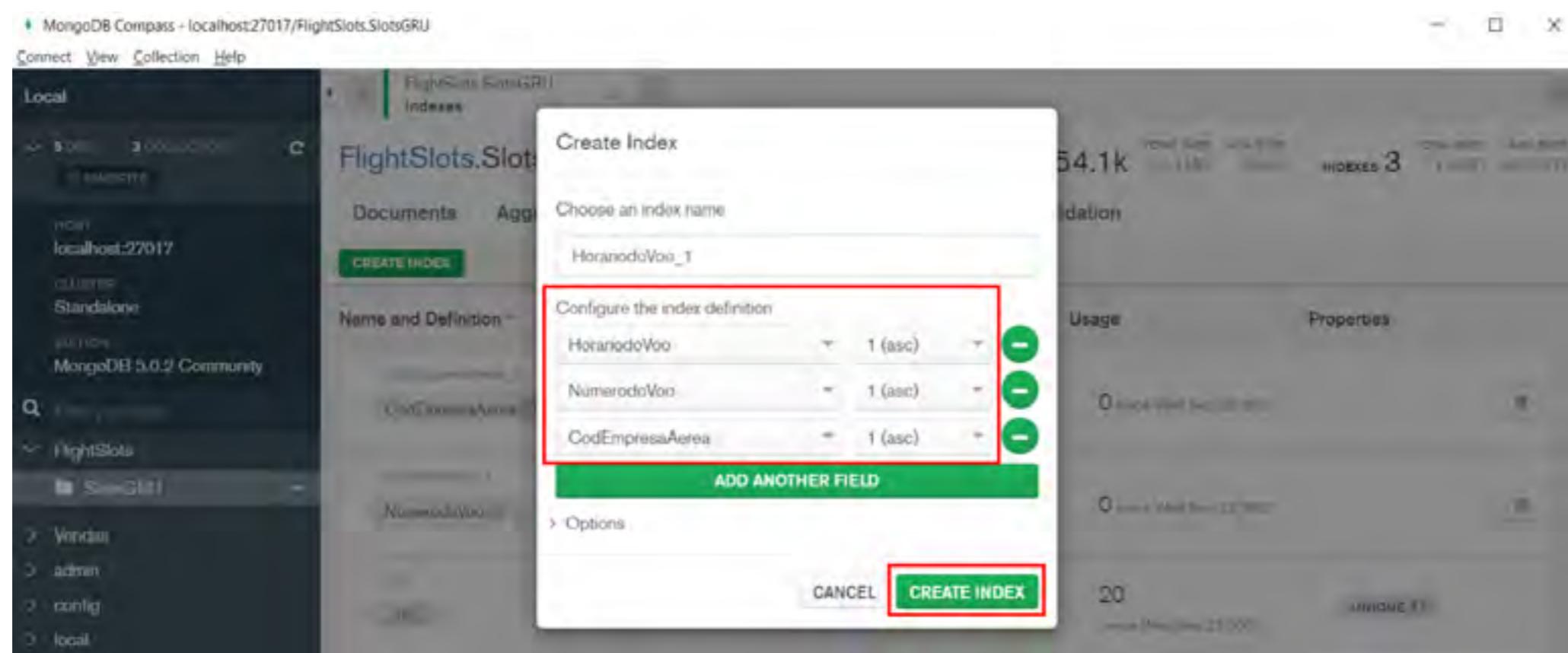
Nessa tela, você deve informar um nome e os campos que farão parte do índice. Observe os destaques em vermelho. Na janela mostrada na figura acima, já temos informado o primeiro campo – Horario do Voo - e também que esse campo deve ser indexado de forma crescente.

Observe, na Figura 20, a adição dos demais campos. Isso é feito acionando o botão “ADD ANOTHER FIELD”.



Fonte: Adaptado de MongoDB (2021).

Ao completar a informação de todos os campos, confirme a criação do índice. A Figura 21 apresenta todos os campos informados, aguardando a confirmação.



Fonte: Adaptado de MongoDB (2021).

Após confirmação da criação do índice, a tela da Figura 22 será apresentada, confirmando que o índice foi criado com sucesso. Observe ainda que o índice está qualificado como composto (compound).

Name and Definition	Type	Size	Usage	Properties
CodEmpresaAerea_1	REGULAR	286.7 KB	0 since Wed Sep 22 2021	
HorarioVoo_1	REGULAR	340.0 KB	0 since Wed Sep 22 2021	COMPOUND
NumerodoVoo_1	REGULAR	438.3 KB	0 since Wed Sep 22 2021	
_id	REGULAR	696.3 KB	21 since Wed Sep 22 2021	UNIQUE

Fonte: Adaptado de MongoDB (2021).

Embora na maioria das vezes os índices sejam muito úteis, ainda existem situações que o desenvolvedor sempre deve monitorar:

Quantidade de índices para o mesmo campo: cada operação de escrita (inclusão, alteração e remoção) demanda também a atualização de todos os índices que contenham os campos afetados. Portanto, a criação de índices deve ser criteriosa. Ao mesmo tempo que aceleram consultas, podem impactar o desempenho de escritas no banco de dados.

Índices não utilizados pelo banco de dados: uma vez que é possível a criação de vários índices, o banco de dados sempre executa uma avaliação para determinar o melhor plano de execução para a consulta. Assim, pode acontecer que um índice exista e seja pouco ou até mesmo nunca utilizado. Como já vimos, índices subutilizados podem impactar operações de escrita ou até mesmo atrapalhar a análise do plano de consulta pelo banco de dados.

Pelas razões citadas, o desenvolvedor deve sempre monitorar a utilização ou não dos índices e remover ou adequar os índices existentes, garantindo que índices subutilizados não atrapalhem a decisão do plano de consulta pelo MongoDB e, principalmente, não prejudique o desempenho em operações de escrita. Dessa forma, na próxima seção, você conecerá como identificar quais índices são utilizados em uma consulta e também saberá como monitorar se há algum índice subutilizado.

8. Analisando a execução dos acessos

Como você conheceu no tópico anterior, muito embora um índice exista, isso não garante que o MongoDB decida utilizá-lo. Por essa razão, o desenvolvedor sempre deve analisar qual índice do banco de dados está sendo utilizado no processamento da consulta. O MongoDB Compass também é muito útil nessa análise.

Lembre-se de que estamos trabalhando com a coleção SlotsGRU. Acionando a aba “Explain Plan” você pode informar uma consulta e, ao executá-la, verificar o plano utilizado para processá-la. Observe a tela da Figura 23.

The screenshot shows the MongoDB Compass interface for the 'FlightSlots.SlotsGRU' collection. The 'Explain Plan' tab is selected. A red box highlights the 'EXPLAIN' button at the top right of the query input area. Another red box highlights the 'Query used the following index: NumerodoVoo' message in the results section, which indicates that the query is using the 'NumerodoVoo' index.

Fonte: Adaptado de MongoDB (2021).

No exemplo apresentado, o único filtro usado foi NumerodoVoo:"5600". Esse campo é indexado por dois índices: o índice simples NumerodoVoo_1 e o índice composto HorariodoVoo_1. Os dois índices são organizados de forma ascendente. Entretanto, o índice composto é encabeçado pelo campo HorariodoVoo. A utilização desse índice seria muito mais custosa. Portanto, o índice mais adequado ao processamento da consulta é mesmo o índice NumerodoVoo_1.

Caso o plano de consulta tivesse decidido pela utilização de outro índice, o desenvolvedor teria que analisar todos os índices da coleção em busca de índices subutilizados ou não atualizados. A aba “Indexes”, mostrada na tela da Figura 24, apresenta as informações sobre a utilização dos índices da coleção SlotsGRU. Observe a quantidade de vezes em que cada índice foi utilizado:

Name and Definition	Type	Size	Usage	Properties
CodEmpresaAerea_1	REGULAR	286.7 KB	0	
HorarioVoo_1	REGULAR	340.0 KB	0	COMPOUND
NumerodoVoo_1	REGULAR	438.3 KB	2	
_id	REGULAR	696.3 KB	22	UNIQUE

Fonte: Adaptado de MongoDB (2021).

Para completar seu conhecimento sobre índices, no próximo tópico você conhecerá como é possível remover um índice de uma coleção.

9. Remoção de índices

Vamos conhecer duas formas de remoção de um índice: com o método `dropIndexes()` e com o MongoDB Compass. A sintaxe do método `dropIndexes()` é demonstrada pela Figura 25, e a Figura 26 apresenta a execução desse método no MongoDB Shell.

Sintaxe

```
db.collection.dropIndexes( < nome do índice > )
```

Exemplo

```
db.SlotsGRU.dropIndexes( "CodEmpresaAerea_1" )
```

Fonte: Adaptado de MongoDB (2021).

```
>_MONGOSH  
  
> use FlightSlots  
< 'switched to db FlightSlots'  
> db.SlotsGRU.dropIndexes("CodEmpresaAerea_1")  
< { nIndexesWas: 4, ok: 1 }  
FlightSlots>|
```

Fonte: Adaptado de MongoDB (2021).

Após a execução do método `dropIndexes()` é possível verificar que a coleção agora possui apenas os índices `NumerodoVooo_1` e `HorariodoVoo_1`. A Figura 27 demonstra isso.

The screenshot shows the MongoDB Compass interface. On the left, the sidebar displays the database 'FlightSlots' with its collections: 'SlotsGRU' (selected), 'Vendas', 'admin', 'config', and 'local'. A red box highlights the 'SlotsGRU' section and the 'Open in New Tab' button. The main window shows the 'FlightSlots.SlotsGRU' collection with 54.1k documents and 3 indexes. The 'Indexes' tab is selected. A second red box highlights the first index entry, which is a compound index named 'HorarioVoo_1'.

Name and Definition	Type	Size	Usage	Properties
<code>HorarioVoo_1</code>	REGULAR	340.0 kB	0 since Wed Sep 22 2021	COMPOUND
<code>NumerodoVoo_1</code>	REGULAR	438.3 kB	2 since Wed Sep 22 2021	
<code>id_1</code>	REGULAR	696.3 kB	32 since Wed Sep 22 2021	UNIQUE

Fonte: Adaptado de MongoDB (2021).

Agora vamos utilizar o MongoDB Compass para executar a remoção de um índice. Para isso, basta acionar o ícone de remoção do índice, conforme demonstrado pela Figura 28.

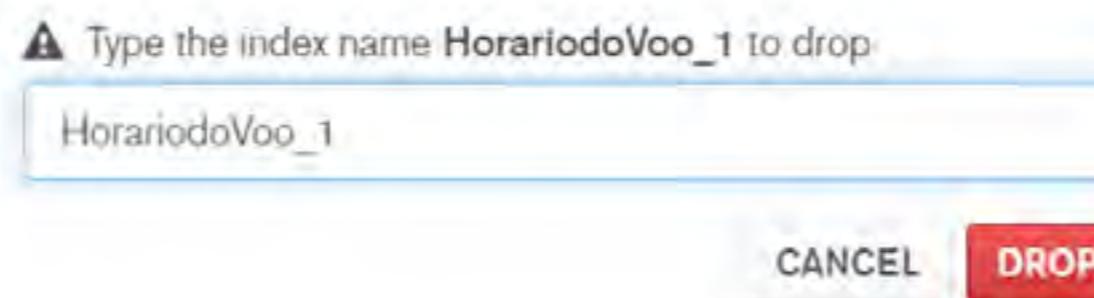
This screenshot is similar to the previous one, showing the 'Indexes' tab for the 'FlightSlots.SlotsGRU' collection. A red box highlights the 'HorarioVoo_1' index entry. A second red circle highlights the trash can icon in the 'Properties' column for this index entry.

Name and Definition	Type	Size	Usage	Properties
<code>HorarioVoo_1</code>	REGULAR	340.0 kB	0 since Wed Sep 22 2021	COMPOUND
<code>NumerodoVoo_1</code>	REGULAR	438.3 kB	2 since Wed Sep 22 2021	
<code>id_1</code>	REGULAR	696.3 kB	32 since Wed Sep 22 2021	UNIQUE

Fonte: Adaptado de MongoDB (2021).

Ao acionar o ícone, será aberta uma janela para a confirmação da exclusão, conforme apresentado pela Figura 29. Após informar o nome do índice, você deve confirmar a exclusão teclando em “DROP”.

Drop Index



Fonte: Adaptado de MongoDB (2021).

Com a confirmação, na próxima tela, apresentada pela Figura 30, é possível verificar a exclusão do índice.

A screenshot of the MongoDB Compass interface showing the 'Indexes' tab for the 'FlightSlots.SlotsGRU' collection. The table displays the following index details:

Name and Definition *	Type	Size	Usage	Properties
HorariodoVoo_1	REGULAR	438.3 KB	2 since Wed Sep 22 2021	
_id	REGULAR	696.3 KB	33 since Wed Sep 22 2021	UNIQUE

Fonte: Adaptado de MongoDB (2021).

No link a seguir você encontra um guia de referência para o
método `dropIndexes()` no MongoDB:

Você já tem todos os conhecimentos sobre a modelagem em MongoDB. No próximo tópico, você conhecerá as principais boas práticas que aumentarão a chance de uma boa modelagem de dados.

10. Boas práticas de modelagem

Você já percebeu que o sucesso da aplicação depende muito da modelagem, ou seja, de como os dados são armazenados. Como a modelagem dos dados é uma das primeiras etapas do desenvolvimento, as decisões tomadas nessa etapa podem acarretar retrabalhos em todas as demais etapas da construção da aplicação. Como já dito, modelagem de dados não é uma ciência exata, mas existem práticas que reduzem muito a possibilidade de decisões improdutivas. Conheça, a seguir, as mais importantes.

10.1 Mantenha todos os dados de uma entidade em um único documento

Manter todos os dados relacionados à uma entidade em um único documento aumenta consideravelmente o desempenho do processamento. Como já vimos, situações em que os dados associados à entidade variam consideravelmente, por exemplo, grandes varejistas como a Amazon, que trabalham com uma gama altamente variada de produtos, podem ser altamente complexas de se construir. Devemos tirar proveito da possibilidade de agregar livremente os dados. Por essa razão, a modelagem por documentos é altamente produtiva tanto para no desenvolvimento quanto no processamento dos dados.

10.2 Evite documentos muito grandes

Ao mesmo tempo que é uma boa prática modelar todos os dados relacionados a uma entidade em um único documento, temos que também ter em mente que, ao processar os acessos, o MongoDB deve acessar todo seu conteúdo. Assim, o desenvolvedor deve analisar criteriosamente as consultas da aplicação e evitar o armazenamento de campos desnecessários ou que são acessados com menor frequência. Somente documentos ajustados à necessidade real da aplicação podem trazer a melhor relação custo-benefício para ela.

Outra observação em relação ao tamanho do documento é o armazenamento de dados binários. O MongoDB possui limitação do tamanho de documento em 16 MB. Isso pode ser um problema caso você armazene binários, como vídeos e imagens, no mesmo documento.

10.3 Modele com base na forma como os dados serão consultados

Tendo em mente as boas práticas anteriores, o desenvolvedor deve analisar os objetivos e as informações que a aplicação deverá entregar ao usuário. Com essas informações, ao evoluir a modelagem, o desenvolvedor sempre deve validar o cumprimento das outras práticas já mencionadas em relação ao objetivo da aplicação. Se assim a modelagem for conduzida, a aplicação terá alta chance de extrair todo potencial do banco de dados.

No link a seguir você encontra um guia de referência para as melhores práticas em **modelagem de dados no MongoDB:**

Você já tem todos os conhecimentos para usar o MongoDB. Claro que essa tecnologia é muito mais ampla, mas você possui agora todo o embasamento para realizar outros testes e buscar outras fontes de conhecimento. Espero que você continue investindo e continue evoluindo seus conhecimentos nessa tecnologia!

Síntese

Nesta unidade, você se aprofundou em como projetar com eficiência o armazenamento de dados em um documento MongoDB. Você também pôde compreender o quanto a modelagem pode impactar outras etapas de desenvolvimento e viu que, ao mesmo tempo que o MongoDB facilita em muito esse processo, o cuidado do desenvolvedor nessa atividade é fundamental. Você aprendeu que a identificação e a criação de índices também integram a atividade de modelagem e que ao mesmo tempo que esses índices aumentam o desempenho, também devem ser frequentemente monitorados.

Confira a seguir os itens mais importantes:

- Identificar os relacionamentos das entidades envolvidas no armazenamento é fundamental. A modelagem deve levar em conta esses relacionamentos no projeto de objetos, documentos ou arrays aninhados no documento. Todos os dados de interesse da aplicação devem estar presentes.
- O MongoDB permite a criação tanto de índices simples, com um único campo, quanto de índices compostos por dois ou mais campos.
- A criação de índices pode ser feita usando o MongoDB Shell ou o MongoDB Compass.
- Podemos usar o MongoDB Shell na criação de um índice simples usando o método `createIndex()`.
- Podemos usar o MongoDB Shell na criação de um índice composto usando o método `createIndexes()`.
- O excesso de índices em uma coleção pode impactar tanto o desempenho de consultas quanto de operações de escrita no banco de dados. Portanto, o desenvolvedor deve ter muito critério na criação desses índices.
- O desempenho das consultas deve ser frequentemente analisado pelo desenvolvedor.

Síntese

- O MongoDB Compass oferece uma função bastante intuitiva para análise do plano de execução escolhido pelo MongoDB no processamento da consulta. Com as informações demonstradas, o desenvolvedor pode identificar os índices utilizados pela consulta.
- O MongoDB Compass também oferece informações que podem identificar índices subutilizados ou nunca utilizados.
- Podemos usar o MongoDB Shell para remover um índice utilizando o método `dropIndexes()`. Entretanto, o MongoDB Compass também oferece uma função mais intuitiva para essa remoção.
- As melhores práticas na modelagem de documentos em MongoDB são: manter todos os dados relacionados a uma entidade em um mesmo documento; evitar documentos muito grandes; e modelar pensando nas consultas.

Fullture Insights

- COMPUTERWORLD. Homepage. Disponível em: <https://computerworld.com.br/carreira/9-habilidades-que-vale-a-pena-desenvolver-para-ser-um-programador-melhor/>. Acesso em: 20 set. 2021.
- GRUAIRPORT. Chegadas e partidas. Disponível em: <https://www.gru.com.br/pt/passageiro/voos/chegadas-e-partidas>. Acesso em: 21 set. 2021.
- MONGODB. Documentação do banco de dados MongoDB. Disponível em: <https://docs.mongodb.com/guides/>. Acesso em: 21 set. 2021.

FU
L
T
U
RE

www.fullture.com