

Banco de Dados Relacional e Não Relacional

Aula: NoSQL Documentos - MongoDB

Ciência de Dados e Big Data

Prof. Anderson Theobaldo



Nesta Aula

- JSON
- Características do NoSQL tipo documento
- Demonstração

NoSQL Tipo Documento

- JSON (JavaScript Object Notation) é um formato de texto legível para intercâmbio de dados que define atributos e valores em um documento.
- Mais compacto do que XML, o que torna mais rápida a análise do documento.
- É completamente independente de linguagem, mas usa convenções que são familiares às linguagens C e familiares, incluindo C++, C#, Java, JavaScript, Perl, Python e muitas outras.
- Ideal para troca de dados.

- Sintaxe:
- Chaves (“{ }”) para iniciar nossos objetos ou conjuntos de dados.
- Colchetes (“[]”) servem para indicar um array.
- Separamos a chave de seus valores correspondentes por dois pontos (“:”).
- A vírgula (”,”) usamos para separar cada elemento contido em nosso objeto (ou array, por exemplo).

```
{  
  "id": 125,  
  "curso": "Engenheiro de Dados",  
  "disciplinas": [  
    {  
      "id": 101,  
      "nome": "Banco de Dados Relacional"  
    },  
    {  
      "id": 105,  
      "nome": "Banco de Dados NoSQL"  
    }  
  ]  
}
```

Tipos de Dados

String	{ "name" : "Jones" }
Número	{ "number_1" : 210 }
Booleano	{ "ativo" : false }
Nulo	{ "observacao" : null }
Objeto	{ "aluno" : { "nome" : "Jaxon" , "idade" : "42" , "cidade" , "São Paulo" } }
Matriz	{ "alunos": [{ "nome" : "Ana" , "idade" : "42" , "cidade" , "São Paulo" }, { "nome" : "Theo" , "idade" : "35" , "cidade" , "Belo Horizonte" }] }

NoSQL tipo documento: características

- Os bancos de dados documentos são conceitualmente semelhantes aos bancos de dados Chave-Valor.
- Armazena dados no formato de documentos com marcadores (tags) em pares de chave-valor.
- O documento pode estar em qualquer formato codificado, tais com XML e JSON.
- O banco de dados documento pode compreender o conteúdo armazenado no componente de valor.
- Apesar do uso de tags nos documentos, os bancos de dados de documentos são considerados sem esquema, ou seja, não impõem uma estrutura predefinida aos dados armazenados.

NoSQL tipo documento: características

- Cada documento pode ter sua própria estrutura.
- As tags dentro do documento são acessíveis ao SGBD, o que torna possível uma consulta mais elaborada.
- Agrupam documentos em grupos lógicos chamados coleções (collections).

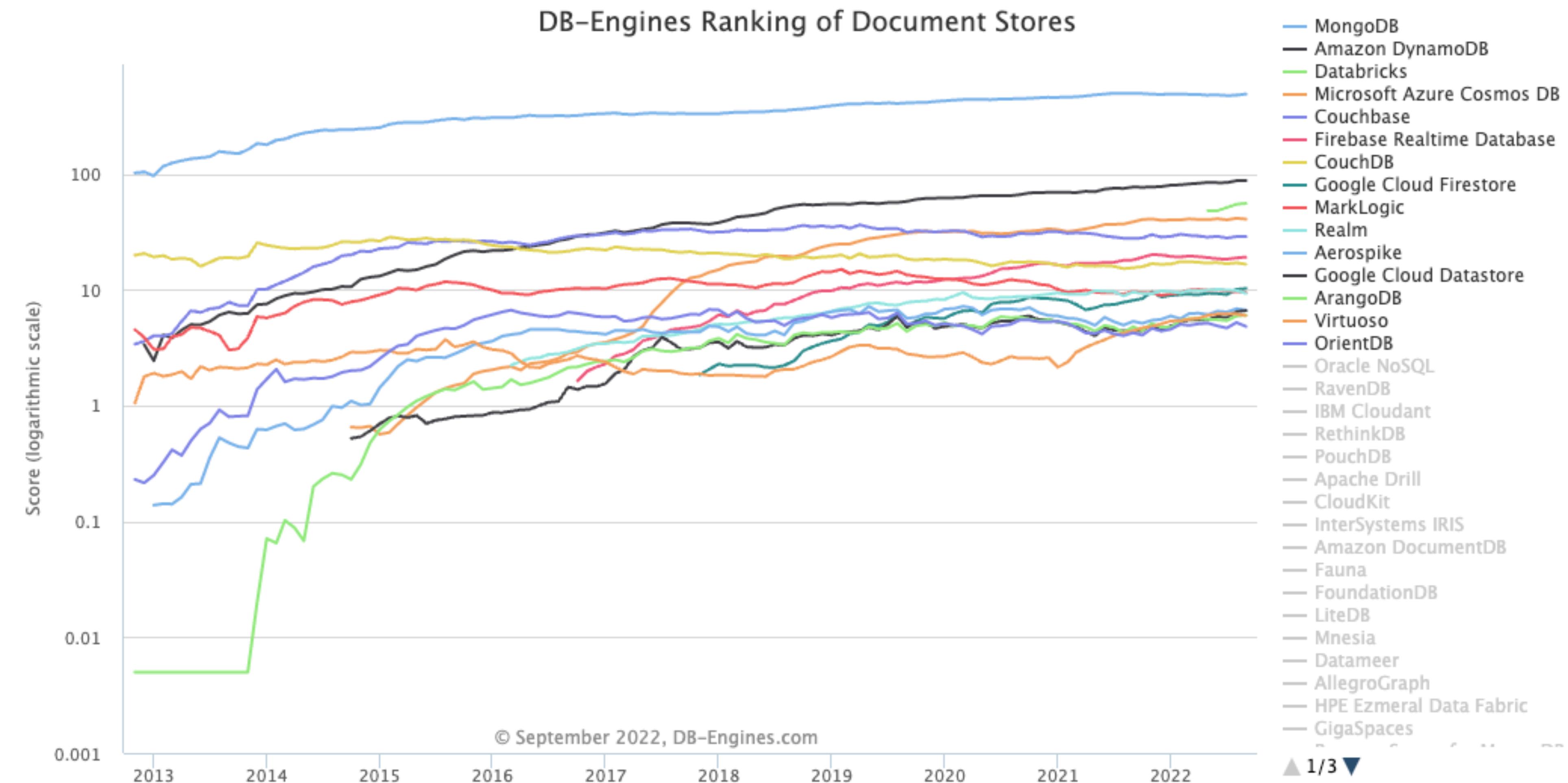
Collection: orders

Key	Document
1001	<pre>{ "CustomerID": 99, "OrderItems": [{ "ProductID": 2010, "Quantity": 2, "Cost": 520 }, { "ProductID": 4365, "Quantity": 1, "Cost": 18 }], "OrderDate": "04/01/2017" }</pre>
1002	<pre>{ "CustomerID": 220, "OrderItems": [{ "ProductID": 1285, "Quantity": 1, "Cost": 120 }], "OrderDate": "05/08/2017" }</pre>

NoSQL tipo documento: características

- Embora um documento possa ser recuperado especificando a coleção e a chave, também é possível consultar com base no conteúdo das tags.
- Os bancos de dados documentos até suportam algumas funções de agregadas, como somar ou calcular média dos saldos nas colunas.
- Diferentemente do relacional, nesse tipo de banco os dados tendem a ser agrupados na mesma coleção. Por exemplo, cada documento de pedido em uma coleção de Pedidos conteria dados sobre o cliente, o próprio pedido e os produtos adquiridos naquele pedido, tudo como um único documento independente.
- Os bancos de dados de documentos não armazenam relacionamentos conforme percebidos no modelo relacional e geralmente não tem suporte para operações de junção.

Bancos mais utilizados





MongoDB

- MongoDB é um software de banco de dados orientado a documentos de código aberto e multiplataforma, escrito na linguagem C++.
- Diferente dos bancos chave-valor, o valor é um documento estruturado e indexado, com metadados.
- Usa documentos semelhantes a JSON(BSON) com esquemas.
- Não requer esquemas predefinidos que permitem adicionar ou remover campos de documentos mais rapidamente.



MongoDB database



NoSQL documentDB



Stored in collections

Características

- Escalabilidade Horizontal: O MongoDB é altamente escalável horizontalmente. Isso significa que você pode adicionar mais servidores ao seu cluster MongoDB para lidar com cargas de trabalho crescentes.
- Sharding: oferece suporte ao sharding, que é uma técnica de distribuição de dados em vários servidores para melhorar o desempenho e a capacidade de armazenamento.
- Replicação: suporta replicação, permitindo que você crie cópias dos seus dados em vários servidores para fins de alta disponibilidade e recuperação de desastres.
- Open Source: é um projeto de código aberto, o que significa que você pode usá-lo gratuitamente e também contribuir para o seu desenvolvimento.

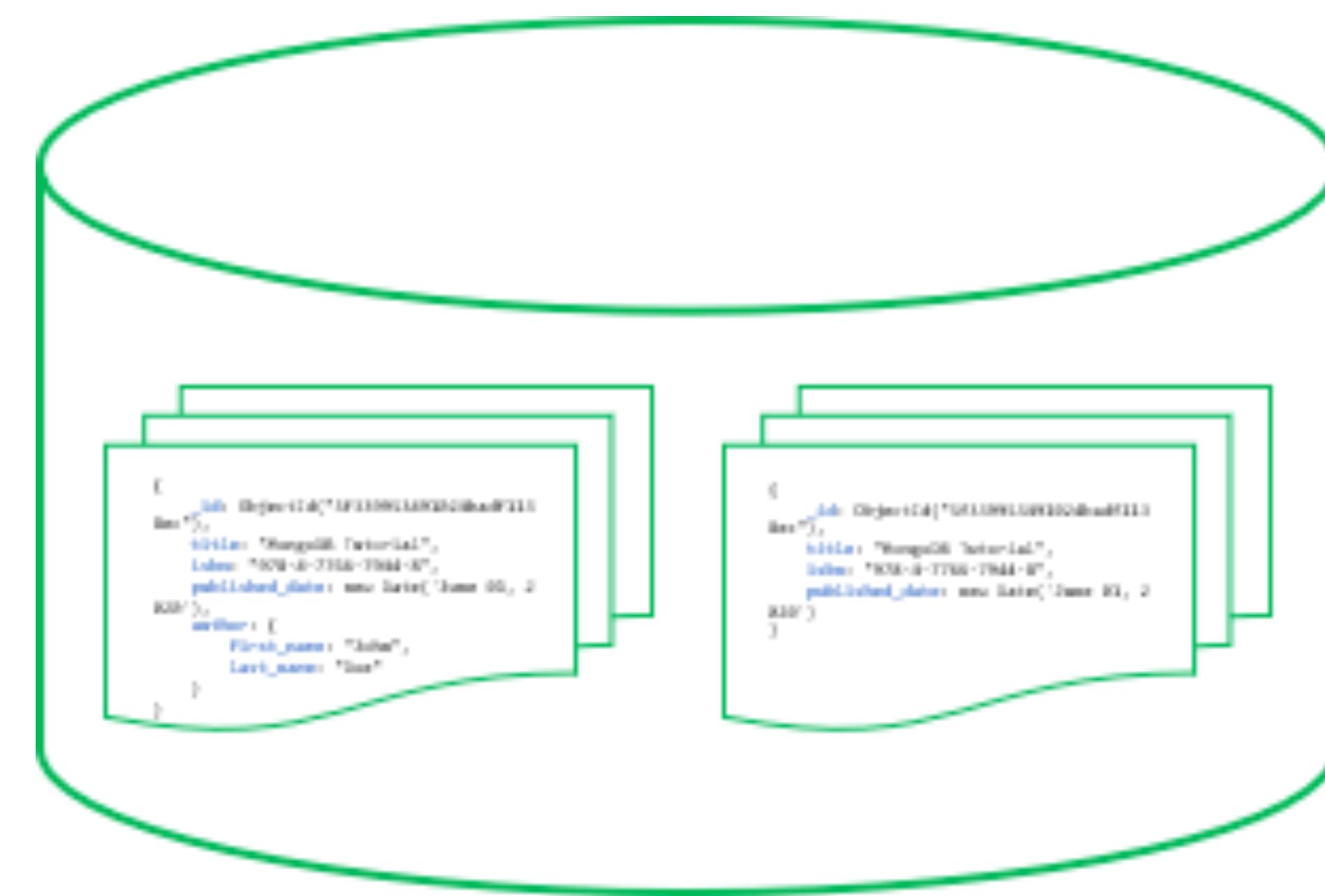
Coleção

- O MongoDB armazena documentos em uma coleção.
- Uma coleção é um grupo de documentos.
- É análoga a uma tabela em um RDBMS.
- Ao contrário de uma tabela que possui um esquema fixo, uma coleção possui um esquema dinâmico.
- Uma coleção pode ser acessada através de um namespace que concatena o nome do banco de dados e do nome da coleção (database_name.collection_name) para qualificar totalmente a coleção.

```
{  
  _id: ObjectId("5f339953491024badf1138ec"),  
  title: "MongoDB Tutorial",  
  isbn: "978-4-7766-7944-8",  
  published_date: new Date("June 01, 2020"),  
  author: {  
    first_name: "John",  
    last_name: "Doe"  
  }  
}
```

Banco de dados

- Armazena coleções em um banco de dados.
- Uma única instância do MongoDB pode hospedar vários bancos de dados.



- BSON significa Binary JSON, que é uma serialização codificada em binário de documentos semelhantes a JSON.

```
{  
  _id: ObjectId("5f339953491024badf1138ec"),  
  title: "MongoDB Tutorial",  
  isbn: "978-4-7766-7944-8",  
  published_date: new Date('June 01, 2020'),  
  author: {  
    first_name: "John",  
    last_name: "Doe"  
  }  
}
```

Termos SQL x MongoDB

Termos SQL	Termos MongoDB
Database	Database
Table	Collection
Row	JSON / BSON Document
Column	Field
Index	Index
Join	Documentos incorporados

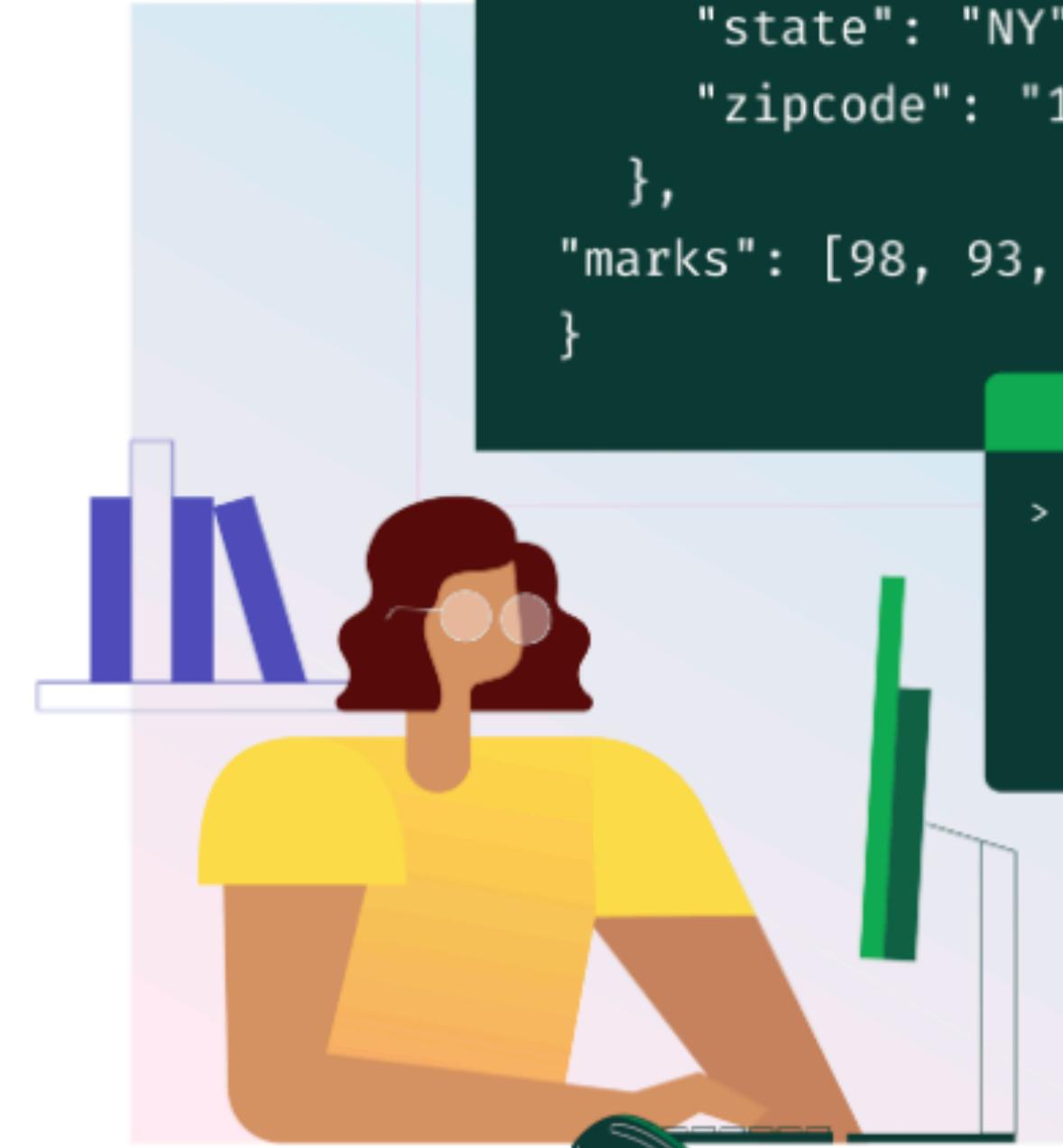
MongoDB x SQL

MongoDB

```
{
  "_id": 1,
  "student_name": "Jasmin Scott",
  "school": {
    "school_id": 226,
    "name": "Tech Secondary",
    "address": "100 Broadway St",
    "city": "New York",
    "state": "NY",
    "zipcode": "10001"
  },
  "marks": [98, 93, 95, 88, 100]
}
```

mongo

```
> db.students.find({"student_name": "Jasmin Scott"})
```


sql

```
SELECT s.name, m.mark, d.name as "school name", d.city
FROM students s
INNER JOIN marks m ON s.id = m.student_id
INNER JOIN school_details d ON s.school_id = d.id
WHERE s.name = "Jasmin Scott";
```

SQL

```

graph LR
    students[id] ---> marks[id]
    students ---> school_details[id]
    marks ---> school_details

```

students		
id	name	school_id
1	Jasmin Scott	226
...

marks		
id	student_id	mark
10	1	98
...

school_details					
id	name	address	city	state	zipcode
226	Tech Secondary	100 Broadway St	New York	NY	10001
...

Results

name	mark	school_name	city
Jasmin Scott	98	Tech Secondary	New York
...

Desenvolvimento de Esquema de Banco de Dados

- Para desenvolver o esquema de um banco de dados no MongoDB, devemos pensar na aplicação ao invés do banco de dados.
- Devemos considerar os seguintes requisitos:
 - O que o aplicativo faz?
 - Quais dados devem ser armazenados?
 - Como os usuários acessarão os dados?
 - Quais dados são mais valiosos?

Estrutura Flexível

- MongoDB implementa um modelo de dados de documentos flexíveis.
- Isso significa que as coleções não impõem nenhuma estrutura de documentos, ou seja, documentos na mesma coleção podem ter estruturas diferentes.

Document 1:

```
{  
    name: "John",  
    major: "EE",  
    course: "CS101"  
    amount: 1000,  
    paid: "Yes"  
}
```

Document 2:

```
{  
    name: "Anne",  
    course: "BI0101"  
    term: "Fall",  
    year: 2022  
}
```

Relacionamentos

- Em MongoDB, o conceito de relacionamento entre documentos é tratado de forma diferente em comparação com bancos de dados relacionais tradicionais.
- No MongoDB, os relacionamentos podem ser tratados de três maneiras principais:
 - Embedding (Incorporação)
 - Referencing (Referenciamento)
 - Modelagem Híbrida

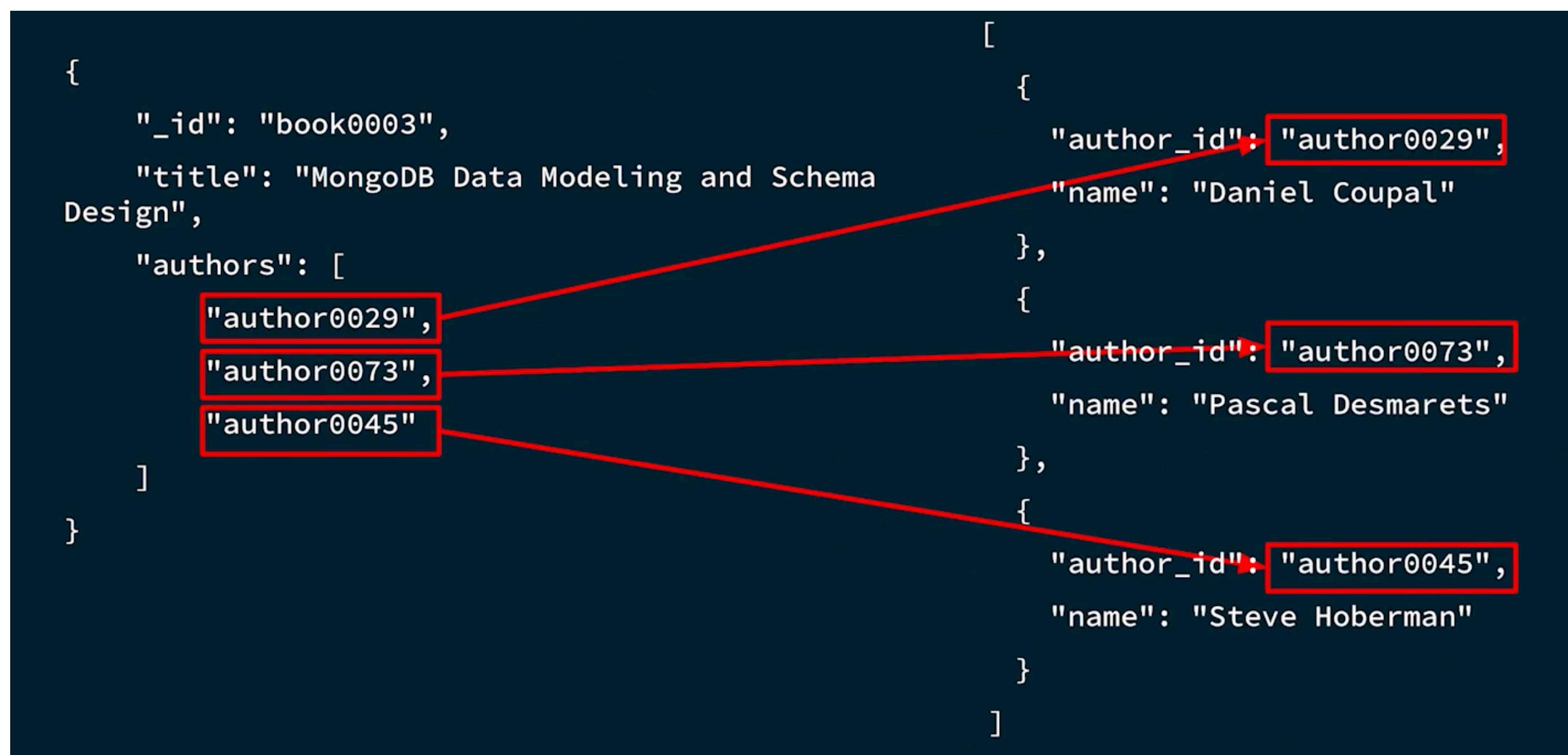
Relacionamento Embedding (Incorporação)

- É uma técnica onde um documento contém outros documentos aninhados.
- Isso significa que um documento pode conter dados diretamente em seus campos, sem a necessidade de referenciar documentos de outras coleções.
- A incorporação é útil quando os dados aninhados são acessados juntos e não precisam ser consultados de forma independente.
- No entanto, devemos estar ciente de que a incorporação pode levar a uma duplicação de dados e requer uma atualização em cascata se os dados incorporados forem alterados.

```
{  
  "_id": "book0003",  
  "title": "MongoDB Data Modeling and Schema  
Design",  
  "authors": [  
    {  
      "author_id": "author0029",  
      "name": "Daniel Coupal"  
    },  
    {  
      "author_id": "author0073",  
      "name": "Pascal Desmarets"  
    },  
    {  
      "author_id": "author0045",  
      "name": "Steve Hoberman"  
    }  
  ]  
}
```

Relacionamento Referencing (Referenciamento)

- É uma técnica na qual você armazena uma referência a documentos de uma coleção em documentos de outra coleção.
- Isso é semelhante ao conceito de chaves estrangeiras em bancos de dados relacionais.
- O referenciamento é útil quando você precisa referenciar dados de forma independente ou quando há muitos relacionamentos entre coleções.
- Pode ajudar a reduzir a duplicação de dados e simplificar a atualização de informações relacionadas.
- No entanto, as consultas podem exigir operações adicionais de junção (join) para recuperar os dados relacionados.



Embedding ou Referencing?

Diretriz	Pergunta	Embedding	Referencing
Simplicidade	Manter as informações juntas levaria a um modelo de dados e código mais simples?	Sim	Não
Juntos	As informações têm uma relação "tem-um", "contém" ou similar?	Sim	Não
Atomicidade da consulta	A aplicação consulta as informações juntas?	Sim	Não
Complexidade de atualização	As informações são atualizadas juntas?	Sim	Não
Armazenamento	As informações devem ser salvas ao mesmo tempo?	Sim	Não
Cardinalidade	Há uma alta cardinalidade (atual ou crescente) no lado filho do relacionamento?	Não	Sim
Duplicação de Dados	A duplicação de dados seria muito complicada de gerenciar e indesejada?	Não	Sim
Tamanho do Documento	O tamanho total das informações ocuparia muita memória ou largura de banda de transferência?	Não	Sim
Crescimento do documento	A parte incorporada crescerá indefinidamente?	Não	Sim
Carga	Os dados são inseridos em momentos distintos em uma carga de trabalho com muitas operações de escrita?	Não	Sim
Individualidade	Para o lado filho do relacionamento, os componentes podem existir por si mesmos sem um pai?	Não	Sim

Relacionamento 1:1

Publisher

```
{  
  "_id": "publisher00023",  
  "name": "MongoDB Press",  
  "hq_id": "B72398845"  
}
```

Headquarters

```
{  
  "_id": "B72398845",  
  "street": "1234 Ave",  
  "city": "New York",  
  "state": "New York",  
  "country": "US",  
  "zip": "00000"  
}
```

```
graph LR; subgraph Publisher ["Publisher"]; P["{_id: 'publisher00023', name: 'MongoDB Press', hq_id: 'B72398845'}"]; end; subgraph Headquarters ["Headquarters"]; HQ["{_id: 'B72398845', street: '1234 Ave', city: 'New York', state: 'New York', country: 'US', zip: '00000'}"]; end; P --> HQ
```

Publisher

```
{  
  "_id": "publisher00023",  
  "name": "MongoDB Press"  
}
```

Headquarters

```
{  
  "_id": "B72398845",  
  "publisher_id": "publisher00023",  
  "street": "1234 Ave",  
  "city": "New York",  
  "state": "New York",  
  "country": "US",  
  "zip": "00000"  
}
```

```
graph LR; subgraph Publisher ["Publisher"]; P["{_id: 'publisher00023', name: 'MongoDB Press'}"]; end; subgraph Headquarters ["Headquarters"]; HQ["{_id: 'B72398845', publisher_id: 'publisher00023', street: '1234 Ave', city: 'New York', state: 'New York', country: 'US', zip: '00000'}"]; end; P --> HQ; HQ --> P
```

Publisher

```
{  
  "_id": "publisher00023",  
  "name": "MongoDB Press",  
  "hq_id": "B72398845"  
}
```

Headquarters

```
{  
  "_id": "B72398845",  
  "publisher_id": "publisher00023",  
  "street": "1234 Ave",  
  "city": "New York",  
  "state": "New York",  
  "country": "US",  
  "zip": "00000"  
}
```

```
graph LR; subgraph Publisher ["Publisher"]; P["{_id: 'publisher00023', name: 'MongoDB Press', hq_id: 'B72398845'}"]; end; subgraph Headquarters ["Headquarters"]; HQ["{_id: 'B72398845', publisher_id: 'publisher00023', street: '1234 Ave', city: 'New York', state: 'New York', country: 'US', zip: '00000'}"]; end; P --> HQ; HQ --> P
```

Relacionamento 1:N (Embedding)

- Temos duas opções principais para o relacionamento de um para muitos:

- 1) Podemos utilizar uma matriz de subdocumentos (Embedding)

Opção preferida, entretanto, se a matriz é ilimitada ou muito grande pode degradar o desempenho.

- 2) Ou podemos usar um único subdocumento com vários pares de valor chave onde cada valor é um subdocumento e a chave é simplesmente um valor exclusivo.

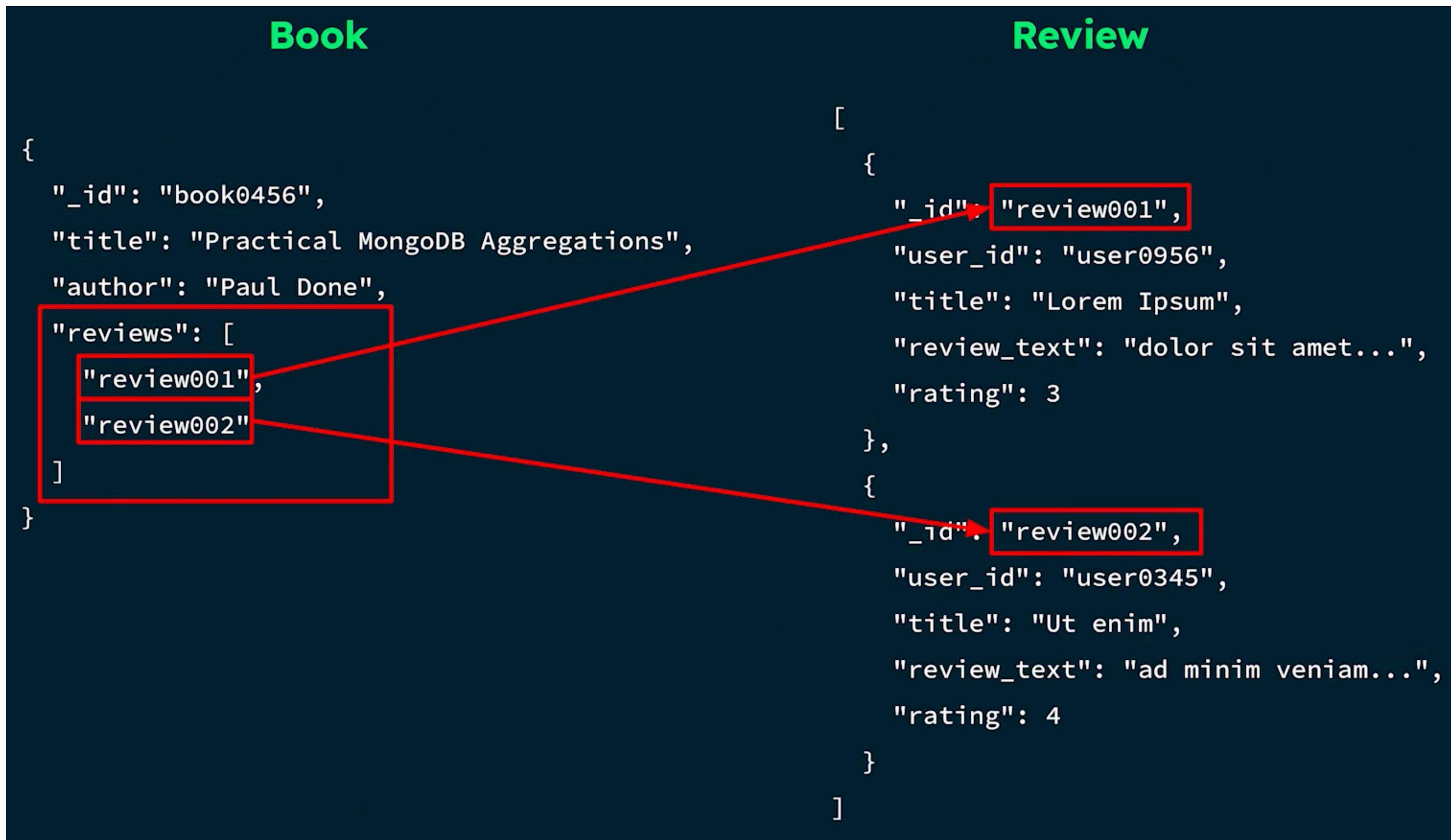
Book

```
{  
    "_id": "book0456",  
    "title": "Practical MongoDB Aggregations",  
    "author": "Paul Done",  
    "reviews": [  
        {"user0956": {  
            "title": "Lorem Ipsum",  
            "review_text": "dolor sit amet...",  
            "rating": 3  
        },  
        {"user0345": {  
            "title": "Ut enim",  
            "review_text": "ad minim veniam...",  
            "rating": 4  
        }}  
    ]  
}
```

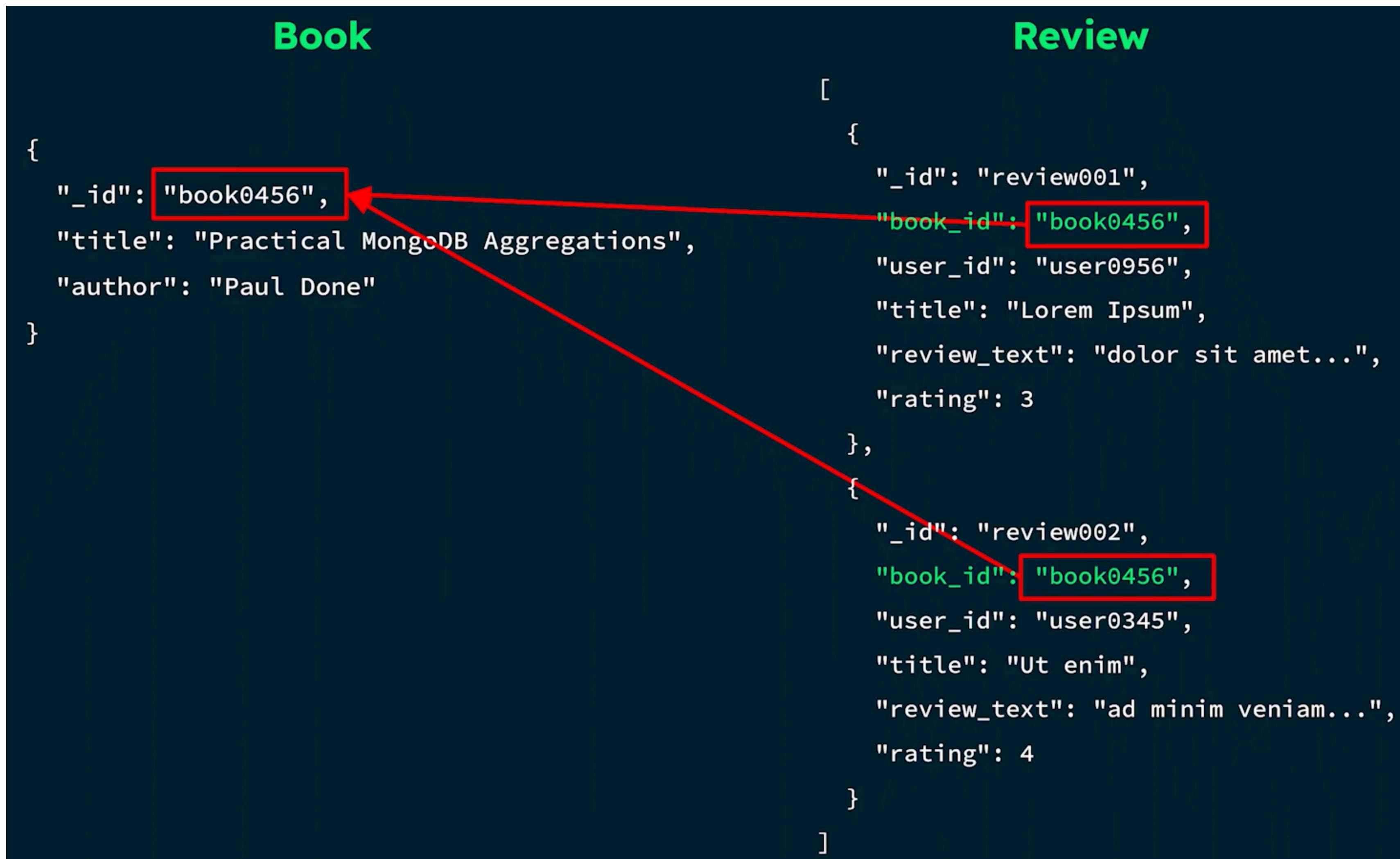
Book

```
{  
    "_id": "book0456",  
    "title": "Practical MongoDB Aggregations",  
    "author": "Paul Done",  
    "reviews": [  
        {"user0956": {  
            "title": "Lorem Ipsum",  
            "review_text": "dolor sit amet...",  
            "rating": 3  
        },  
        {"user0345": {  
            "title": "Ut enim",  
            "review_text": "ad minim veniam...",  
            "rating": 4  
        }}  
    ]  
}
```

Relacionamento 1:N (Referencing)



Relacionamento 1:N (Referencing)



Relacionamento N:N (Embedding)

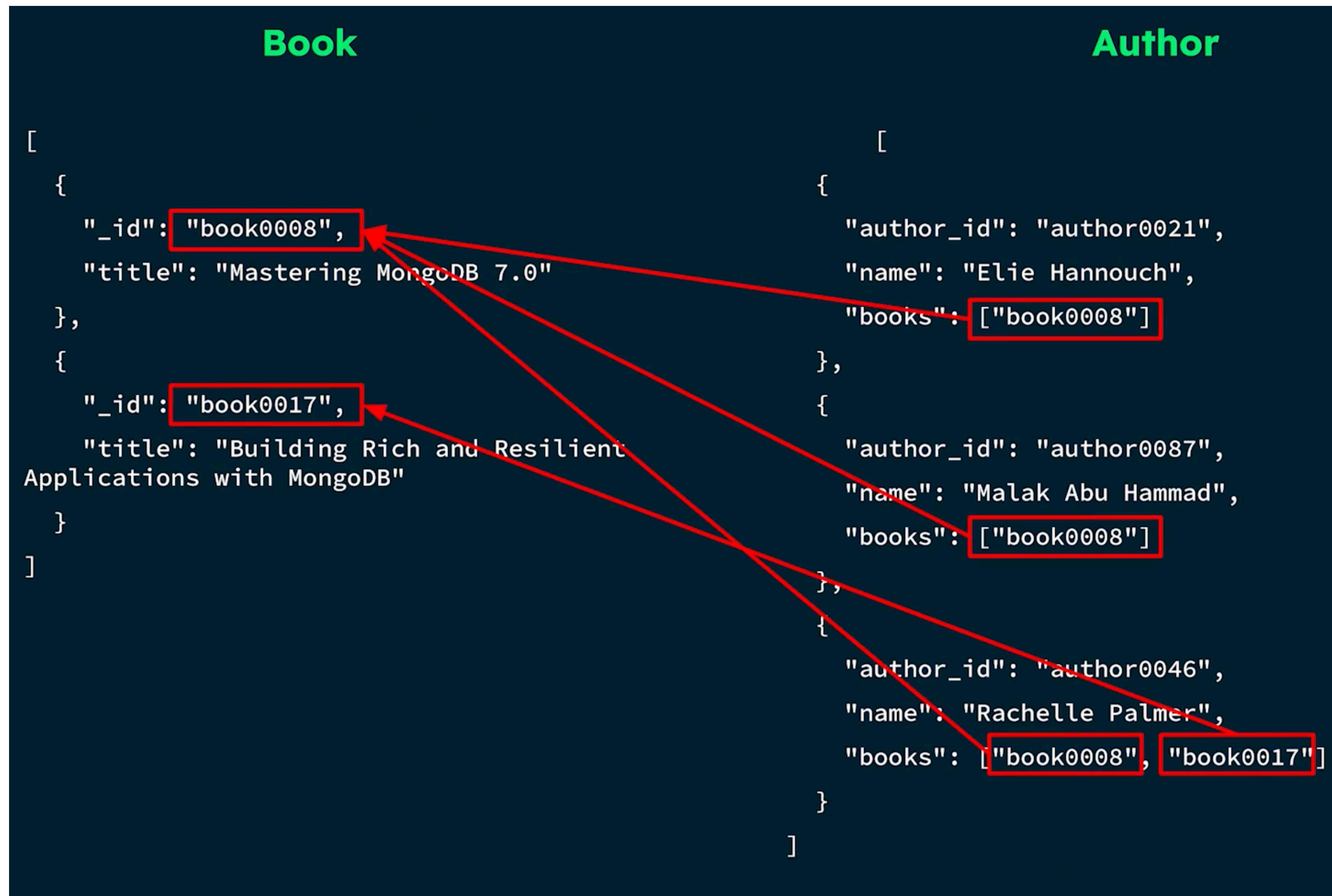
```
Book
{
  "_id": "book0008",
  "title": "Mastering MongoDB 7.0",
  "authors": [
    {
      "author_id": "author0021",
      "name": "Elie Hannouch"
    },
    {
      "author_id": "author0087",
      "name": "Malak Abu Hammad"
    },
    {
      "author_id": "author0046",
      "name": "Rachelle Palmer"
    },
    ...
  ]
}
```

```
Book
{
  "_id": "book0008",
  "title": "Mastering MongoDB 7.0",
  "authors": {
    "author0021": {
      "name": "Elie Hannouch"
    },
    "author0087": {
      "name": "Malak Abu Hammad"
    },
    "author0046": {
      "name": "Rachelle Palmer"
    },
    ...
  }
}
```

Relacionamento N:N (Referencing)

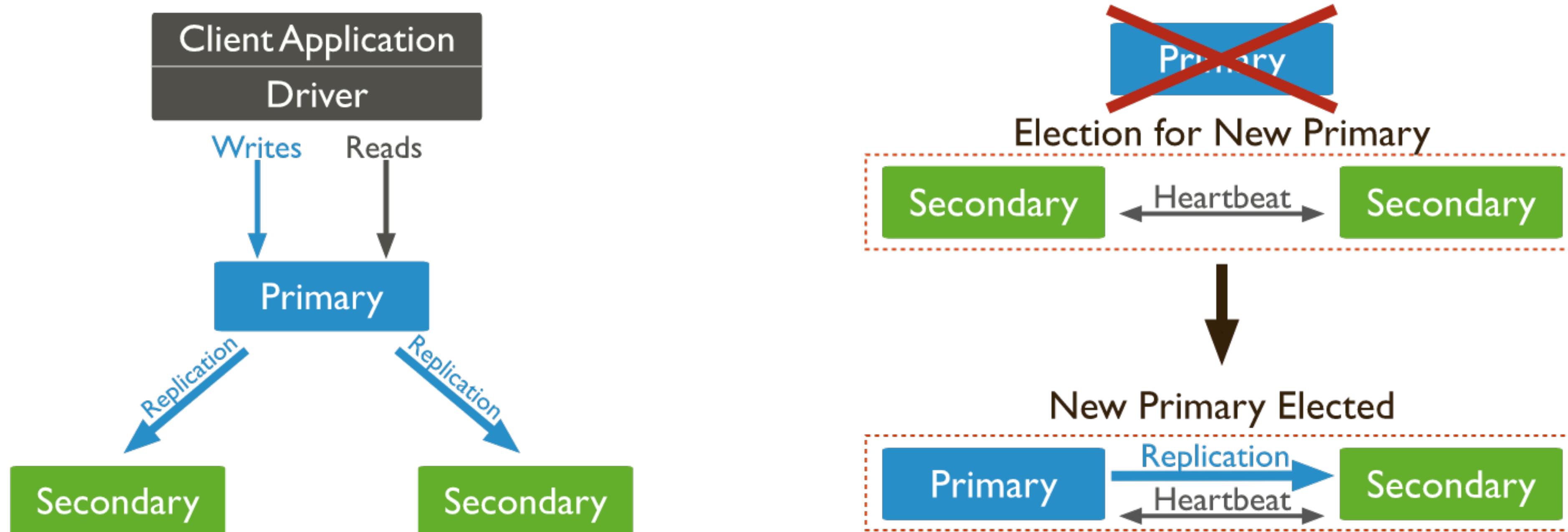


Relacionamento N:N (Referencing)

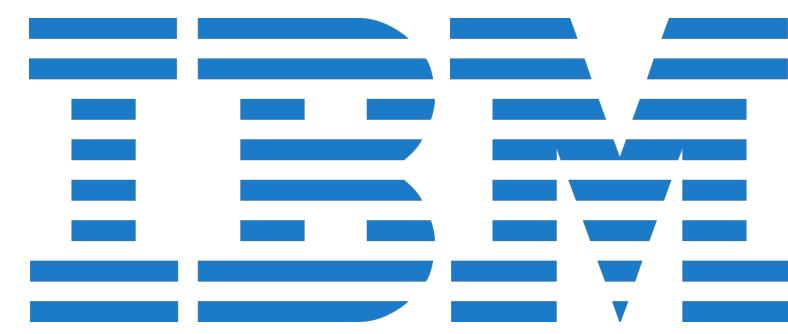


Alta Disponibilidade

O recurso de replicação do MongoDB, chamado conjunto de réplicas, fornece failover automático e redundância de dados.



Quem Está Usando



Uber



ebay



Edições do MongoDB

- Community Edition:
 - É gratuito e está disponível no Windows, Linux e macOS.
- Enterprise Server
 - É uma edição comercial do MongoDB como parte da assinatura do MongoDB Enterprise Advanced.
- Atlas
 - É um serviço global de banco de dados em nuvem. Está disponível em plataformas de nuvem comuns, como AWS, Azure e GCP.

Instalação do MongoDB Community Server

- Instale o MongoDB
<https://www.mongodb.com/try/download/community>
- Instale o MongoDB Shell
<https://www.mongodb.com/try/download/shell>
- MongoDB as a service
<https://www.mongodb.com/try>

Importar e Exportar Dados

```
mongodump --uri "mongodb+srv://<your username>:<your password>@<your cluster>.mongodb.net/sample_supplies"
```

```
mongoexport --uri="mongodb+srv://<your username>:<your password>@<your cluster>.mongodb.net/sample_supplies" --collection=sales --out=sales.json
```

```
mongorestore --uri "mongodb+srv://<your username>:<your password>@<your cluster>.mongodb.net/sample_supplies" --drop dump
```

```
mongoimport --uri="mongodb+srv://<your username>:<your password>@<your cluster>.mongodb.net/sample_supplies" --drop sales.json
```

Comando Básicos

- Para exibir os bancos de dados existentes:
 - ▶ `show dbs`
- Para alterar o database informe:
 - ▶ `use <nome-do-banco>`
- Para exibir as collections do banco de dados corrente:
 - ▶ `show collections`
- Para criar uma collection:
 - ▶ `db.createCollection('<nome-da-collection>')`
- Para exibir excluir uma collection:
 - ▶ `db.<nome-da-collection>.drop()`

Criando Um Banco de Dados

- Para criar um novo banco de dados, basta executar o comando:

`use <nome_do_banco>`

- Exemplo:

`use libraria` (se não existir, será criado)

Criando Uma Coleção

- Podemos criar uma nova coleção usando o método `createCollection` ou inserindo um documento nela.
- Por exemplo, para criar uma coleção chamada "livros", executamos o seguinte comando:

```
db.createCollection("minha_colecao")
```

- Ou, alternativamente, podemos inserir um documento vazio na nova coleção, usando o comando:

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

Inserir um único documento

- db.collection.insertOne() insere um único documento em uma coleção.
- Cada documento armazenado em uma coleção requer um campo `_id` que funciona como chave primária. Se um documento inserido omitir o campo `_id`, o driver MongoDB irá gerar um automaticamente.

```
db.users.insertOne(← collection  
{  
    name: "sue", ← field: value  
    age: 26, ← field: value  
    status: "pending" ← field: value } document  
}  
)
```

Inserir vários documentos

- db.collection.insertMany() inseri vários documentos em uma coleção.

Inclusão de vários documentos:

```
db.products.insertMany( [  
    { item: "card", qty: 15 },  
    { item: "envelope", qty: 20 },  
    { item: "stamps" , qty: 30 }  
] );
```

A operação retorna o seguinte documento:

```
{  
    "acknowledged" : true,  
    "insertedIds" : [  
        ObjectId("562a94d381cb9f1cd6eb0e1a"),  
        ObjectId("562a94d381cb9f1cd6eb0e1b"),  
        ObjectId("562a94d381cb9f1cd6eb0e1c")  
    ]  
}
```

Consultar documento único

- db.collection.findOne(query, projection) - Retorna um documento que atende aos critérios de consulta especificados na coleção.

```
db.bios.findOne(  
  { },  
  { name: 1, contribs: 1 }  
)
```

```
db.bios.findOne(  
  { contribs: 'OOP' },  
  { _id: 0, 'name.first': 0, birth: 0 }  
)
```

Operadores de Comparação

\$eq	Igualdade
\$gt	Maior que
\$gte	Maior ou igual
\$in	Contido
\$lt	Menor que
\$lte	Menor ou igual
\$ne	Diferente
\$nin	Não contido

Operadores Lógicos

\$and	retorna todos os documentos que correspondem às condições de ambas as cláusulas
\$or	retorna todos os documentos que correspondem às condições de qualquer uma das cláusulas.
\$not	Inverte o efeito de uma expressão de consulta e retorna documentos que não correspondem à expressão de consulta.
\$nor	retorna todos os documentos que não correspondem a nenhuma das cláusulas.

Consultar vários documentos

- db.collection.find(query, projection) retorna documentos de uma coleção de acordo com critério(s) de filtro

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

← collection
← query criteria
← projection
← cursor modifier

Atualizar um único documento

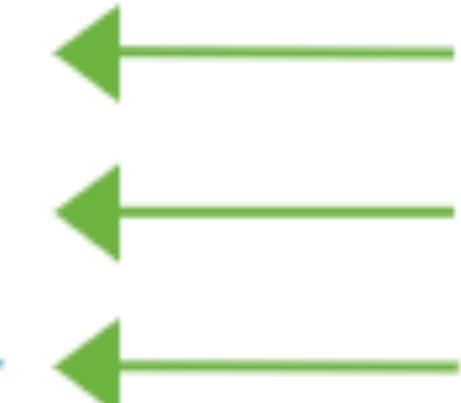
- db.collection.updateOne(filter, update, options) - atualiza um único documento na coleção de acordo com critério(s) de filtro

```
db.restaurant.updateOne(  
  { "name" : "Central Perk Cafe" },  
  { $set: { "violations" : 3 } }  
) ;
```

Atualizar vários documentos

- db.collection.updateMany(filter, update, options) - atualiza vários documentos na coleção de acordo com critério(s) de filtro

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



collection
update filter
update action

Remover um único documento

- db.collection.deleteOne(filter) - exclui o primeiro documento que corresponda ao filtro

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



The diagram illustrates the components of the MongoDB command. Two green arrows point from the labels 'collection' and 'delete filter' to their respective parts in the code. The arrow labeled 'collection' points to the 'users' part of the command. The arrow labeled 'delete filter' points to the '{ status: "reject" }' part of the command.

Remover vários documento

- db.collection.deleteMany(filter) - remove vários documentos na coleção de acordo com critério(s) de filtro

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



The diagram consists of two green arrows. One arrow points from the word 'collection' to the 'users' part of the command. Another arrow points from the word 'delete filter' to the filter condition '{ status: "reject" }'.

Referências

- MongoDB University. Disponível em: <https://www.mongodb.com/docs/manual/>
- MongoDB University. Disponível em: <https://university.mongodb.com/courses/catalog>