## Embedding

Ao contrário do SQL, o MongoDB segue o princípio de **“dados que são acessados juntos devem ser armazenados juntos”**.

Pensando assim, embedding significa incorporar dados que são relacionados e inseri-los em um documento. É usado para simplificar as consultas (queries) aos dados e melhorar a performance geral das ferramentas nas consultas.

Veja abaixo um exemplo de documento autor com um array de livros incorporado ao restante dos dados. Arrays de objetos são uma forma comum de embedding de dados.

{

"\_id": ObjectId("579a7256f245878acabac457c"),

"nome": "JRR Tolkien",

"livros": [

{"titulo": "O Senhor dos Anéis", "paginas": 500},

{"titulo": "O Hobbit", "paginas": 200},

{"titulo": "O Silmarillion", "paginas": 400}

]

}

Porém, incorporar dados em um único documento pode criar documentos muito grandes, o que pode acabar prejudicando a performance da aplicação, pois um documento deve ser carregado em memória por inteiro. Além disso, pode também fazer com que novos dados sejam incorporados indefinidamente a um único objeto e aumentando o tamanho em bytes além do limite de 16 mb por documento de um [objeto BSON](https://cursos.alura.com.br/extra/alura-mais/json-bson-entendendo-os-documentos-do-mongodb-c84).

## Referencing

Referencing significa fazer referência a documentos em outra coleção. Nesse caso, os dados são guardados em coleções separadas, mas ainda é importante que mantenham vínculo entre eles. A referência é feita através de um campo específico do documento, normalmente o campo id ou equivalente.

A agregação de dados via reference visa evitar duplicação de dados (um aspecto muito importante no SQL, também chamada de “normalização de dados”) e também gerenciar o tamanho dos documentos para evitar a criação de documentos muito grandes, que prejudicariam a performance do sistema.

Por outro lado, a junção de dados via referência faz com que uma consulta se transforme em duas ou várias. Por exemplo, uma consulta aos dados de um livro resultaria em consultas tanto na coleção livros quanto na coleção autores, o que consome mais recursos e pode impactar a performance de leitura.

{

"\_id": ObjectId("9gPOwsIJaf5knkzpvYNlk9flz"),

"nome": "JRR Tolkien",

"livros": [

ObjectId("ctstNHEEKCLwTN7gN7KgNprYb"),

ObjectId("qdQwKNumukFzuSYh58WKLN3TV"),

ObjectId("KNUeheO0UbtpFYwLuJpdwbD5P")

]

}

Abaixo segue um resumo dos prós e contras de cada um dos métodos.

| | \*\*Embedding\*\* |

|:---:|:-------------------------------------------------:|

| PRO | Dados retornados em uma única consulta |

| PRO | atualização e exclusão de dados em única operação |

| CON | duplicação de dados |

| CON | documentos maiores |

| | \*\*Referencing\*\* |

| PRO | Não há duplicação de dados |

| PRO | documentos menores |

| CON | necessário "unir" dados de múltiplos documentos |

## Quando utilizar um ou outro?

A resposta é **depende**, pois isso passa pela prática de modelagem de dados e também pelas características de cada tipo de dado e coleção. Neste curso não vamos avançar nesse tópico, pois modelagem é um assunto para seu próprio curso.

Tanto o MongoDB quanto os gerenciadores SQL (MySQL, PostgreSQL, entre outros) têm suas próprias formações nas quais você pode aprender com mais profundidade sobre o funcionamento de cada um deles. Confira abaixo:

* [Formação MongoDB](https://cursos.alura.com.br/formacao-mongodb);
* [Formação SQL com PostgreSQL](https://cursos.alura.com.br/formacao-postgresql);
* [Formação SQL com MySQL](https://cursos.alura.com.br/formacao-oracle-mysql).

Apesar de serem parte importante do desenvolvimento back-end, bancos de dados são assuntos muito vastos por si, então, não se preocupe neste momento em aprender todos os detalhes de cada um deles!