mongoDB®

Boas práticas para que as coisas não saiam do controle

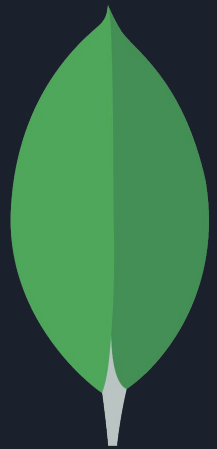
Um (não muito) breve guia

Escrito por Lucas Damaceno | Revisado por Leticia Gerola

~~let tableOfContents~~

let documentOfContents = {

```
    intro: {
        dbms: "Database Management Systems",
        acid: "ACID e non-ACID DBMS",
        mongodb: "MongoDB data management system"
    },
    collectionsAndDocuments: {
        collections: "O que são collections e do que se alimentam",
        documents: "O alimento das collections",
        cardinality: "Um pra quantos?",
        indexing: {
            indexes: "O que são índices?",
         ...
```



```
...
        selectivity: "Quanto seus índices te ajudam a filtrar todo \
                      conteúdo?",
        creatingIndexes: "Quando e como criar índices"
},
goodPractices: {
        lookups: "Um mal necessário",
        denormalizingAndEmbeddingData: "Manter sempre a \
                                        simplicidade, por favor",
        usefulDesignPatterns: [
                "Preallocation Pattern",
                "Bucket Pattern",
...
```

```
                ...
                    "Schema Versioning Pattern",
                    "Attribute Pattern",
                    "Computed Pattern",
                    "Extended Reference Pattern"
                ],
            }
        }
    }
```

> console.log(documentOfContents.intro.dbms)
Database Management Systems
undefined

- "Banco de dados" são só dados reunidos e estruturados;
- Os *DBMSs* fazem toda a mágica de leitura, agregação e estruturação de todos esses dados.

Source: https://en.wikipedia.org/wiki/Database#Database_management_system

> console.log(documentOfContents.intro.dbms)

ACID e non-ACID DBMS

undefined

- ACID é um conjunto de propriedades de um DBMS que garante a **validade dos dados** frente ao vários possíveis problemas.
- *ACID = Atomicity, Conformity, Isolation, Durability*;
  - *Atomicity:* transações são entidades singulares, independente do número de operações na mesma;
  - *Consistency:* qualquer transação deve ser completamente concluída e o banco de dados deve ir, sem intermediários, de um estado à outro;
  - *Isolation:* operações concorrentes devem ter o mesmo resultado, caso fossem executadas sequencialmente;
  - *Durability:* uma vez que qualquer transação é concluída, ela deve manter-se armazenada em qualquer evento de desligamento da máquina.

Source: https://en.wikipedia.org/wiki/ACID

> console.log(documentOfContents.intro.mongodb)
MongoDB data management system
undefined

- *Non-ACID* (a versão 4.0 introduziu ACID em várias operações que, a princípio, não eram ACID), *NoSQL*, orientado à documentos (*BSON*, um *superset* do *JSON*);
- *Single View Application-oriented*;
- Alta disponibilidade, escalável, alta performance;
- *Query language* expressiva e completa para filtragem e exibição dos dados;
- *Database as a Service (DbaaS)*, o MongoDB Atlas, como solução completa de *storage* e monitoramento;
- Os dados mais acessados ficam guardados em memória RAM, o restante em disco;
- Toda linguagem de consulta e escrita é baseada em Javascript.

Source: https://www.mongodb.com/scale/data-management-systems
https://www.mongodb.com/blog/post/multi-document-transactions-in-mongodb
https://stackoverflow.com/questions/8331099/what-is-the-javascript-engine-that-runs-mongodb-shell

> console.log(documentOfContents.collectionsAndDocuments.collections)
O que são collections e do que se alimentam
undefined

- Collections são análogas à Tabelas em bancos de dados não-relacionais;
- As collections se alimentam de Documentos, análogos às Linhas de uma tabela;
- Collections ficam dentro de um Database, um conjunto de Collections;
- Todas as operações de busca e escrita nos Documentos das Collections são conhecidas como Collection Level Operations;
- Caso a collection não tenha documentos indexados, Collection Level Operations resultam em um Collection Scan;
- Collection Scans tem Tempo Computacional de *O(n)*, sendo n = número de Documentos.

Source: https://docs.mongodb.com/manual/reference/glossary/#term-collection
https://docs.mongodb.com/manual/tutorial/analyze-query-plan/

```
> console.log(documentOfContents.collectionsAndDocuments.documents)
O alimento das collections
undefined
```

- Documentos são entidades unitárias dentro das collections;
- Documentos no MongoDB são representações binárias compactas e intertransacionais de um JSON - chamados de BSON;
- Conjunto das unidades de dados primitivas compatíveis com JSON (e algumas feitas especialmente para o BSON, como o ObjectId())

Source: https://docs.mongodb.com/manual/core/document/

> console.log(documentOfContents.collectionsAndDocuments.cardinality)
Um pra quantos?
undefined

- Assim como em banco de dados SQL, os documentos e collections podem ser considerados entidades;
- Cardinalidade é a relação de multiplicidade entre entidades;
- Por exemplo, uma entidade composta de várias entidades de outro tipo tem uma cardinalidade de uma-para-várias;
- Todavia, cardinalidade também engloba o quanto "várias" significa.

Source:
https://www.mongodb.com/blog/post/6-rules-of-thumb-for-mongodb-schema-design-part-1

> console.log(documentOfContents.collectionsAndDocuments.indexing.indexes)
O que são índices?
undefined

- Índices são estruturas de dados relacionados fortemente às Collections;
- São porções parciais de dados pré-organizados de acordo com sua criação;
- Todas as collections precisam de pelo menos um Índice, sendo o _id o "índice primário", único e impossível de apagar;
- Uma consulta (query) que tenha um campo indexado resultará em um Index Scan;
- Index Scans são sempre priorizados em uma query;
- Index Scans também possuem Tempo Computacional de *O(n)*, porém, nesse caso, é desejável que *n* seja menor que *n* de um Collection Scan;
- Diferentemente de alguns DBMS, o MongoDB reorganiza seus índices sempre que uma operação de escrita é efetuada.

Source: https://docs.mongodb.com/manual/indexes

> console.log(documentOfContents.collectionsAndDocuments.indexing.selectivity)
Quanto seus índices te ajudam a filtrar todo conteúdo?
undefined

- Seletividade significa a habilidade que os índices têm de filtrar o conteúdo de acordo com uma query;
- Um índice muito seletivo significa que ele é suficiente para filtrar todos os resultados em um Index Scan, sem necessitar de um Collection Scan subsequente.

Source: https://docs.mongodb.com/manual/tutorial/create-queries-that-ensure-selectivity/

> console.log(documentOfContents.collectionsAndDocuments.indexing.creatingIndexes)
Quando e como criar índices
undefined

- Antes de criar um índice, deve-se avaliar o quão frequente um certo campo é filtrado em uma query;
- Uma boa estratégia é tentar imaginar o quão seletivo é um Collection Scan para filtrar os documentos em uma query. Caso se avalie baixa seletividade em um Collection Scan, deve-se considerar a criação de índices baseados nos campos mais buscados;
- Outra boa estratégia é sempre usar índices para ajudar em suas queries, organizando pelos campos mais procurados pelas queries na aplicação.

Source: https://docs.mongodb.com/manual/applications/indexes/

> console.log(documentOfContents.collectionsAndDocuments.indexing.creatingIndexes)
Quando e como criar índices
undefined

Criando índices:

```
db.collection.createIndex( { name: -1 } )
```

Neste caso, *"name"* é o campo indexado e *-1* é a ordem que será organizado (1 ordenará crescentemente e -1 decrescentemente).

Source: https://docs.mongodb.com/manual/indexes/

> console.log(documentOfContents.collectionsAndDocuments.indexing.creatingIndexes)
Quando e como criar índices
undefined

É possível indexar vários campos de uma só vez. Caso um campo seja filtrado por mais de um campo, indexar esses campos adequadamente irá aumentar a performance das queries e economizar Index Scans:

```
db.products.createIndex( { "item": 1, "stock": 1 } )
```

```
db.events.createIndex( { "username" : 1, "date" : -1 } )
```

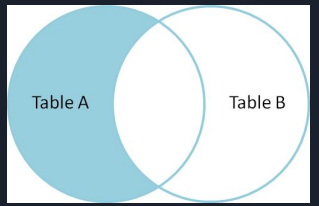
A ordem dos campos na criação de um índice é **extremamente importante** pois essa instrução diz ao MongoDB a ordem dos dados a serem indexados.

Source: https://docs.mongodb.com/manual/indexes/
https://docs.mongodb.com/manual/core/index-compound/

> console.log(documentOfContents.collectionsAndDocuments
.goodPractices.lookups)

Um mal necessário
undefined

- Um *lookup* performa um LEFT OUTER JOIN baseado numa referência local a uma collection externa;
- É um mecanismo que permite a circunvenção de um alto crescimento no tamanho de uma collection;
- Por mais que pareça uma boa ideia manter 100% de conformidade e reduzir a complexidade de operações de atualização, *lookups* tem um forte impacto na performance de uma query.
- Deve-se avaliar se o impacto na performance de Collection Scan ou Index Scan adicional é comparável à maior complexidade na manutenção da conformidade entre os documentos;
- Contudo, deve-se sempre considerar que lookups são **ruins** e devem ser evitados**.**



Source: https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/
https://www.mongodb.com/blog/post/building-with-patterns-a-summary

```
> console.log(documentOfContents.collectionsAndDocuments
                                    .goodPractices.denormalizingAndEmbeddingData)
Manter sempre a simplicidade, por favor
undefined
```

- *Normalizar* dados de uma entidade significa não repeti-los em outras que contenham sua referência. *Desnormalizar* é fazer o oposto;
- Incorporar (ou *embeddar)* informações desnormalizadas de outras entidades são uma das formas de evitar lookups;
- Alguns padrões de projetos reforçam que os dados com pouca taxa de escrita que fazem parte de uma ou várias entidades devem ser desnormalizados, mesmo que as entidades compostas possuam sua referência;
- Caso não seja possível eliminar lookups, deve-se reduzir ao máximo.

Source: https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/
https://www.mongodb.com/blog/post/building-with-patterns-a-summary

```
> documentOfContents \
...   .collectionsAndDocuments \
...   .goodPractices \
...   .usefulDesignPatterns === Infinity
True
```

```
> console.log(documentOfContents.collectionsAndDocuments
                                        .goodPractices.usefulDesignPatterns[0])
Preallocation Pattern
undefined
```

- Consiste em pré-alocar um documento padrão na collection e ir preenchendo com os dados caso haja necessidade;
- Em nível de aplicação, torna a organização de algumas entidades mais simples ao custo da performance (afinal, um *update* são duas operações em uma transação).

Source: https://www.mongodb.com/blog/post/building-with-patterns-a-summary
https://www.mongodb.com/blog/post/building-with-patterns-the-preallocation-pattern

> console.log(documentOfContents.collectionsAndDocuments
.goodPractices.usefulDesignPatterns[0])

Preallocation Pattern

undefined

```
{
    _id: ObjectId(),
    theater: "Cinemark SP",
    room: 3,
    available_seats: {
        lane_a: [1, 2, 3, 4, 5, 6],
        lane_b: [1, 2, 3, 4, 5, 6],
        lane_c: [1, 2, 3, 4, 5, 6]
    }
}
```

```
{
    _id: ObjectId(),
    theater: "Cinemark SP",
    room: 3,
    available_seats: {
        lane_a: [1, 2, 3, 4, 6],
        lane_b: [1, 2, 3, 4, 5, 6],
        lane_c: [1, 2, 3, 4, 5, 6]
    }
}
```

Source: Acervo pessoal.

```
> console.log(documentOfContents.collectionsAndDocuments
                                                .goodPractices.usefulDesignPatterns[1])
Bucket Pattern
undefined
```

- Em uma collection onde a escrita de documentos é constante (como um log de telemetria de algum dispositivo, por exemplo), é vantajoso criar uma só collection com um *"bucket"* incluindo todos os documentos que seriam separados (e também incluindo algumas métricas que só podem ser obtidas por Collection Level Operations);
- Isso aumenta a seletividade dos Index Scans e reduz Collection Scans, dando um grande aumento na performance, porém, uma filtragem mais profunda nas medições requer algumas operações mais complexas.

Source: https://www.mongodb.com/blog/post/building-with-patterns-a-summary
https://www.mongodb.com/blog/post/building-with-patterns-the-preallocation-pattern

> console.log(documentOfContents.collectionsAndDocuments
.goodPractices.usefulDesignPatterns[1])

Bucket Pattern

undefined

```
{
   sensor_id: 12345,
   timestamp: ISODate("2019-01-31T10:00:00.000Z"),
   temperature: 40
}

{
   sensor_id: 12345,
   timestamp: ISODate("2019-01-31T10:01:00.000Z"),
   temperature: 40
}

{
   sensor_id: 12345,
   timestamp: ISODate("2019-01-31T10:02:00.000Z"),
   temperature: 41
}
```

```
{
    sensor_id: 12345,
    start_date: ISODate("2019-01-31T10:00:00.000Z"),
    end_date: ISODate("2019-01-31T10:59:59.000Z"),
    measurements: [
       {
       timestamp: ISODate("2019-01-31T10:00:00.000Z"),
       temperature: 40
       },
       {
       timestamp: ISODate("2019-01-31T10:01:00.000Z"),
       temperature: 40
       },
       …
       {
       timestamp: ISODate("2019-01-31T10:42:00.000Z"),
       temperature: 42
       }
    ],
   transaction_count: 42,
   sum_temperature: 2413
}
```

Source: https://www.mongodb.com/blog/post/building-with-patterns-a-summary
https://www.mongodb.com/blog/post/building-with-patterns-the-bucket-pattern

```
> console.log(documentOfContents.collectionsAndDocuments
                                    .goodPractices.usefulDesignPatterns[2])
Schema Versioning Pattern
undefined
```

- Em ambientes em que as mudanças nas definições de algum *schema* são frequentes, documentos antigos podem ficar *outdated* e sair do padrão vigente.
- Uma das soluções é apontar a versão do *schema* e reinserir os documentos antigos com a versão vigente sem apagá-los (em caso de alguma dependência anterior) e re-indexando a versão nova para reduzir Collection Scans;
- Isso reforça a durabilidade dos dados pois só serão perdidos se realmente apagados. Também padroniza todos os documentos para a versão nova;
- Em contrapartida, queries devem sempre buscar a nova versão - mudanças no Schema podem ser demoradas e custosas, levando em conta que **todos** os documentos devem ser atualizados para nova versão.

Source: https://www.mongodb.com/blog/post/building-with-patterns-a-summary
https://www.mongodb.com/blog/post/building-with-patterns-the-bucket-pattern

> console.log(documentOfContents.collectionsAndDocuments
.goodPractices.usefulDesignPatterns[2])

Schema Versioning Pattern

undefined

```
{
    "_id": "<ObjectId>",
    "name": "Anakin Skywalker",
    "home": "503-555-0000",
    "work": "503-555-0010"
}
```

```
{
    "_id": "<ObjectId>",
    "schema_version": "2",
    "name": "Anakin Skywalker (Retired)",
    "contact_method": [
        { "work": "503-555-0210" },
        { "mobile": "503-555-0220" },
        { "twitter": "@anakinskywalker" },
        { "skype": "AlwaysWithYou" }
    ]
}
```

Source: https://www.mongodb.com/blog/post/building-with-patterns-a-summary
https://www.mongodb.com/blog/post/building-with-patterns-the-bucket-pattern

```
> console.log(documentOfContents.collectionsAndDocuments
                                                .goodPractices.usefulDesignPatterns[3])
Attribute Pattern
undefined
```

- Documentos com muitos atributos distintos se tornam difíceis de consultar e seus índices ficam menos seletivos a cada documento;
- Caso esses atributos resumam o mesmo tipo de elemento com diferenciação em alguma característica, pode-se colocar uma collection esquematizada com a tal característica junto ao valor;
- Isso torna a indexação mais eficaz e reduz a complexidade das queries.

Source: https://www.mongodb.com/blog/post/building-with-patterns-a-summary
https://www.mongodb.com/blog/post/building-with-patterns-the-attribute-pattern

> console.log(documentOfContents.collectionsAndDocuments
.goodPractices.usefulDesignPatterns[3])

Attribute Pattern

undefined

```
{
    title: "Star Wars",
    director: "George Lucas",
    ...
    release_US: ISODate("1977-05-20T01:00:00+01:00"),
    release_France: ISODate("1977-10-19T01:00:00+01:00"),
    release_Italy: ISODate("1977-10-20T01:00:00+01:00"),
    release_UK: ISODate("1977-12-27T01:00:00+01:00"),
    ...
}
```

```
{
    title: "Star Wars",
    director: "George Lucas",
    …
    releases: [
        {
        location: "USA",
        date: ISODate("1977-05-20T01:00:00+01:00")
        },
        {
        location: "France",
        date: ISODate("1977-10-19T01:00:00+01:00")
        },
        {
        location: "Italy",
        date: ISODate("1977-10-20T01:00:00+01:00")
        },
        {
        location: "UK",
        date: ISODate("1977-12-27T01:00:00+01:00")
        },
        …
    ],
    …
}
```

Source: https://www.mongodb.com/blog/post/building-with-patterns-a-summary
https://www.mongodb.com/blog/post/building-with-patterns-the-attribute-pattern

```
> console.log(documentOfContents.collectionsAndDocuments
                                                .goodPractices.usefulDesignPatterns[4])
Computed Pattern
undefined
```

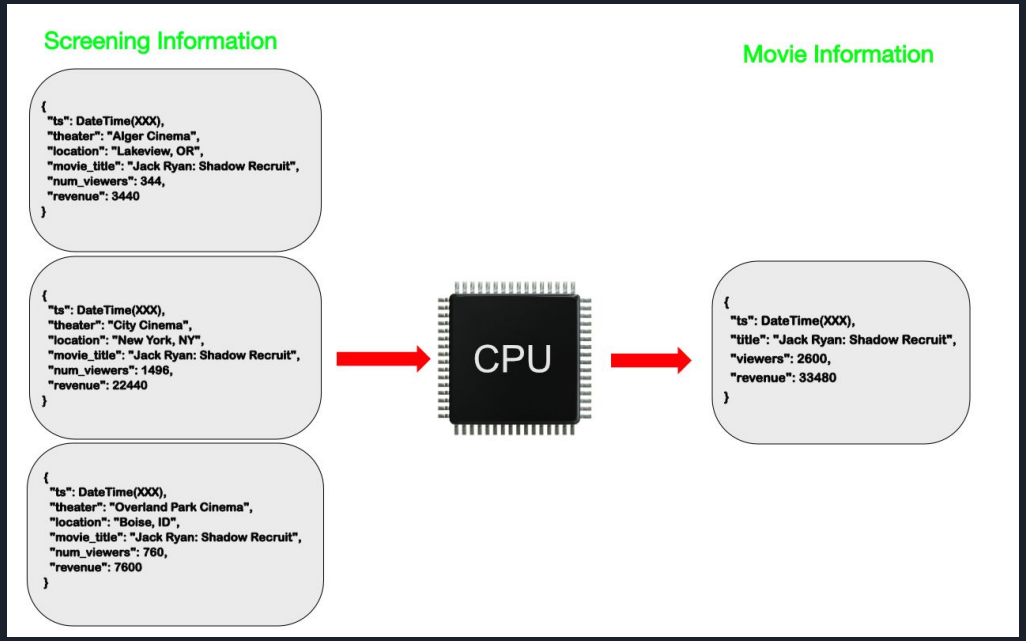
- Collections com padrão de acesso muito intenso na parte da leitura podem estar usando excessivamente o CPU da máquina caso as leituras envolvam cálculos;
- Fazer uma Collection contendo os valores calculados e atualizar a cada escrita pode ajudar a reduzir a carga na CPU;
- Em uma collection com uma relação de 1000/1 de leituras/escritas, usar esse padrão reduz-se a carga da CPU em um fator de 1000.

Source: https://www.mongodb.com/blog/post/building-with-patterns-a-summary
https://www.mongodb.com/blog/post/building-with-patterns-the-computed-pattern

```
> console.log(documentOfContents.collectionsAndDocuments
                                    .goodPractices.usefulDesignPatterns[4])
```

Computed Pattern

undefined



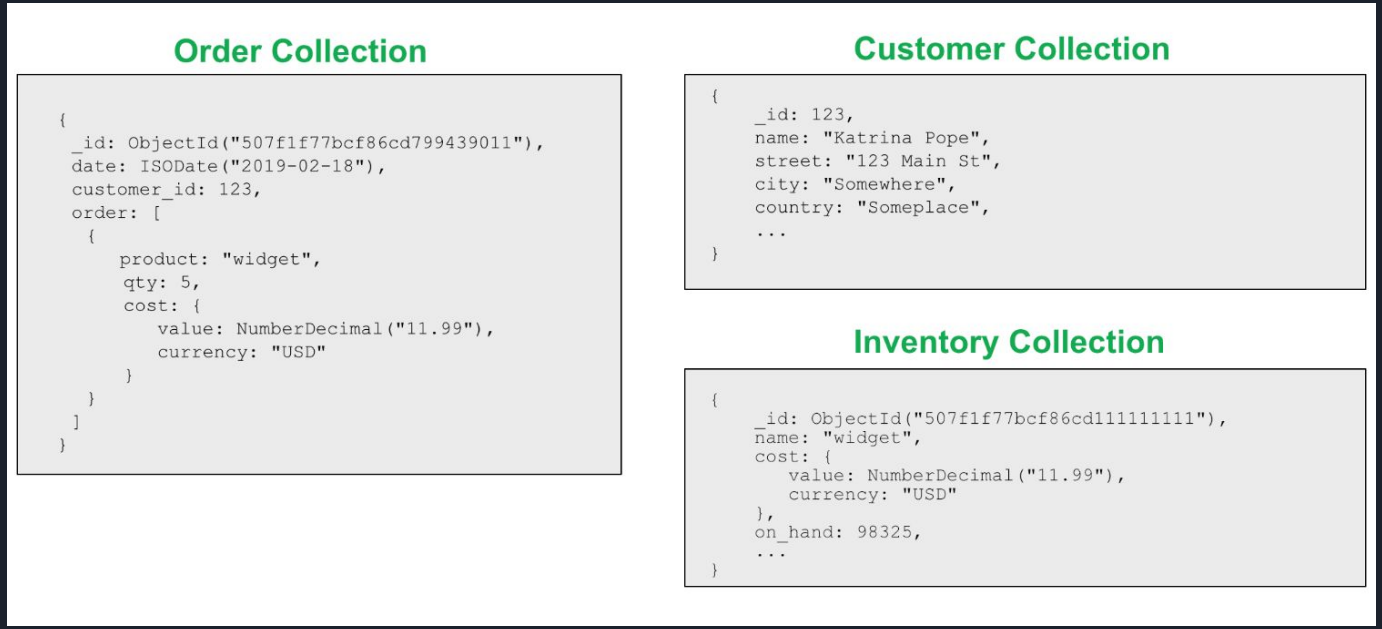
Source: https://www.mongodb.com/blog/post/building-with-patterns-a-summary
https://www.mongodb.com/blog/post/building-with-patterns-the-computed-pattern

```
> console.log(documentOfContents.collectionsAndDocuments
                                                .goodPractices.usefulDesignPatterns[5])
Extended Reference Pattern
undefined
```

- Para reduzir *lookups*, Collection Scans, e o número de collections (que reduz a complexidade das consultas *system-wide*) você pode desnormalizar a parte útil dos dados de uma entidade que é referenciada em outra;
- Isso aumenta a complexidade de possíveis updates. Porém, há uma considerável redução em Collection Scans, o que gera um incremento de performance em queries;
- O tamanho do documento de resultado da consulta também é reduzido, juntamente com a complexidade das queries.

Source: https://www.mongodb.com/blog/post/building-with-patterns-a-summary
https://www.mongodb.com/blog/post/building-with-patterns-the-extended-reference-pattern

```
> console.log(documentOfContents.collectionsAndDocuments
                                  .goodPractices.usefulDesignPatterns[5])
Extended Reference Pattern
undefined
```



Source: https://www.mongodb.com/blog/post/building-with-patterns-a-summary
https://www.mongodb.com/blog/post/building-with-patterns-the-extended-reference-pattern

> console.log(documentOfContents.collectionsAndDocuments
.goodPractices.usefulDesignPatterns[5])

Extended Reference Pattern

undefined

Source: https://www.mongodb.com/blog/post/building-with-patterns-a-summary
https://www.mongodb.com/blog/post/building-with-patterns-the-extended-reference-pattern

```
> presentation.on("end", presentation => {
...   presentation.closing_comments()
... })
```