

Disciplina:

BANCOS DE DADOS NoSQL

Professor: Augusto Zadra



2.6 MongoDB frameworks

INTRODUÇÃO

- Verificamos algumas das principais funções de operação com o MongoDB;
- Contudo, é ideal e importante conhecermos operações avançadas onde iremos aprender algumas facilidades para extração de dados em bancos com muitas informações.
- Continuaremos com o mongo para facilitar a nossa linha de aprendizado.

Bancos de dados NoSQL

mongoDB

Arrays

- A consulta de elementos de um *array* foi projetada para se comportar da mesma forma que a consulta de escalares.
- Como identificar *arrays*, vejam a diferença de representação para que possamos comparar e alterar os formatos para procedermos com as consultas.
- Conjuntos de dados menores, geralmente são armazenados nos formatos de *array*.
- ***Quais comando então podemos utilizar para trabalhar com estas estruturas?***

Bancos de dados

NoSQL

mongoDB

Arrays

```
_id: ObjectId("5bd761dcae323e45a93ccfe8")
saleDate: 2015-03-23T21:06:49.506+00:00
✓ items: Array
  > 0: Object
  > 1: Object
  > 2: Object
  > 3: Object
  > 4: Object
  > 5: Object
  > 6: Object
  > 7: Object
storeLocation: "Denver"
✓ customer: Object
  gender: "M"
  age: 42
  email: "cauho@witwuta.sv"
  satisfaction: 4
  couponUsed: true
  purchaseMethod: "Online"
```

```
{
  "_id" : ObjectId("5bd761dcae323e45a93ccfe8"),
  "saleDate" : ISODate("2015-03-23T21:06:49.506Z"),
  "items" : [
    {
      "name" : "printer paper",
      "tags" : [
        "office",
        "stationary"
      ],
      "price" : NumberDecimal("40.01"),
      "quantity" : 2
    },
    {
      "name" : "notepad",
      "tags" : [
        "office",
        "writing",
        "school"
      ],
      "price" : NumberDecimal("35.29"),
      "quantity" : 2
    },
    {
      "name" : "pens",
      "tags" : [
        "writing",
        "office",
        "school",
        "stationary"
      ],
      "price" : NumberDecimal("56.12"),
      "quantity" : 5
    }
  ]
}
```

Bancos de dados NoSQL

mongoDB

Consulta a subdocumentos

- Existem duas maneiras de consultar um documento incorporado: consultar todo o documento ou consultar seus pares chave / valor individuais.
- A consulta de um documento incorporado inteiro funciona de forma idêntica a uma consulta normal, no entanto, uma consulta para um subdocumento completo deve corresponder exatamente ao subdocumento.
- Este tipo de consulta também é sensível à ordem.

Bancos de dados NoSQL

mongoDB

Consulta a subdocumentos

- Existem duas maneiras de consultar um documento incorporado: consultar todo o documento ou consultar seus pares chave / valor individuais.
- A consulta de um documento incorporado inteiro funciona de forma idêntica a uma consulta normal, no entanto, uma consulta para um subdocumento completo deve corresponder exatamente ao subdocumento.
- Este tipo de consulta também é sensível à ordem.

Bancos de dados NoSQL

mongoDB

Arrays

- Vamos buscar no *array* objetos que estão tagueados com a rubrica de escritório: “office” no campo tag:
- Como sabemos a estrutura fica fácil construir o comando:

```
db.sales.find({"items.tags": "office"});
```

```
db.sales.find({"items.tags": "office", "items.quantity": {$gt : 6}})
```


Bancos de dados NoSQL

mongoDB

Arrays

\$all

- **\$all** ajuda a combinar valores que possam estar presentes no *array* em uma lista única de elementos.

```
db.sales.find({"items.tags": { $all : [ "office", "school" ] },  
              "items.quantity": { $gt : 6}})
```

- Você também pode consultar por correspondência exata usando todo o *array* no entanto se os parâmetros da pesquisa não forem exatamente igual ao documento não retornará:

```
db.sales.find({"items.tags": [ "office", "school" ],  
              "items.quantity": { $gt : 6}})
```

Bancos de dados NoSQL

mongoDB

Arrays

\$size

- **\$size** é útil para consultar *arrays* de um determinado tamanho, ou seja, semelhante a uma contagem de itens.

```
db.sales.find({"items.tags": { $size : 3 },  
              "items.quantity": { $gt : 2 }})
```

- Porém não pode ser combinada com outros operadores como \$gt ou \$gte.

```
db.sales.find({"items.tags": { $size : { $gt : 3 } },  
              "items.quantity": { $gt : 2 }})
```

Bancos de dados NoSQL

mongoDB

Arrays

\$inc

- \$inc fará o incremento do campo em todos os documentos retornados no **(agora vamos evoluir a linguagem)** cursor com os critérios de busca associados:

```
use sample_training  
db.zips.updateMany({ "city": "HUDSON" }, { "$inc": { "pop": -10 } })
```

Bancos de dados NoSQL

mongoDB

Arrays

- **\$push** adiciona valores no array quando desejamos acrescentar novas características aos dados mitigando o risco de mudança de tipos.

```
db.grades.updateOne({ "student_id": 250, "class_id": 339 },  
                    { "$push": { "scores": { "type": "extra credit", "score": 100 } }  
                    })
```

\$push

Bancos de dados NoSQL

mongoDB

Arrays

\$slice

- **\$slice** retorna um subconjunto de elementos para uma chave de *array* e também pode retornar páginas no meio dos resultados, obtendo um deslocamento e o número de elementos a serem retornados.

```
db.sales.find({},{"items.tags": { $slice : -1 }}).pretty()  
db.sales.find({},{"items.tags": { $slice : 2 }}).pretty()  
db.sales.find({},{"items.tags": { $slice : 1 }}).pretty()
```

Bancos de dados NoSQL

mongoDB

explain

- Quando precisarmos verificar a performance do nosso comando e informações sobre a execução das pesquisas devemos utilizar o método **explain** que pode ser acessado pela linha de comando e através da interface do Compass:

```
db.zips.find ({"city" : "BRENT" }).explain()
```

Bancos de dados NoSQL

mongoDB

explain

```
{
  "explainVersion" : "1",
  "queryPlanner" : {
    "namespace" : "sample_training.zips",
    "indexFilterSet" : false,
    "parsedQuery" : {
      "city" : {
        "$eq" : "BRENT"
      }
    },
    "queryHash" : "0491F17A",
    "planCacheKey" : "62E7C1DA",
    "maxIndexedOrSolutionsReached" : false,
    "maxIndexedAndSolutionsReached" : false,
    "maxScansToExplodeReached" : false,
    "winningPlan" : {
      "stage" : "COLLSCAN",
      "filter" : {
        "city" : {
          "$eq" : "BRENT"
        }
      }
    },
    "direction" : "forward"
  },
  "rejectedPlans" : [ ]
},
"command" : {
  "find" : "zips",
  "filter" : {
    "city" : "BRENT"
  },
  "$db" : "sample_training"
},
"serverInfo" : {
  "host" : "DESKTOP-K8P9JQH",
  "port" : 27017,
  "version" : "5.0.4",
  "gitVersion" : "62a84ede3cc9a334e8bc82160714df71e7d3a29e"
},
}
```

explain()

Bancos de dados

NoSQL

mongoDB

explain

Compass

sample_airbnb.listingsAndReviews

	DOCUMENTS	TOTAL SIZE	AVG. SIZE	INDEXES	TOTAL SIZE	AVG. SIZE
	5.6k	94.4MB	17.0KB	4	512.0KB	128.0KB

Documents Aggregations Schema **Explain Plan** Indexes Validation

FILTER { name : { \$regex : /garden/i } } **OPTIONS** **EXPLAIN** **RESET** ↺ ...

VIEW DETAILS AS **VISUAL TREE** RAW JSON

Query Performance Summary

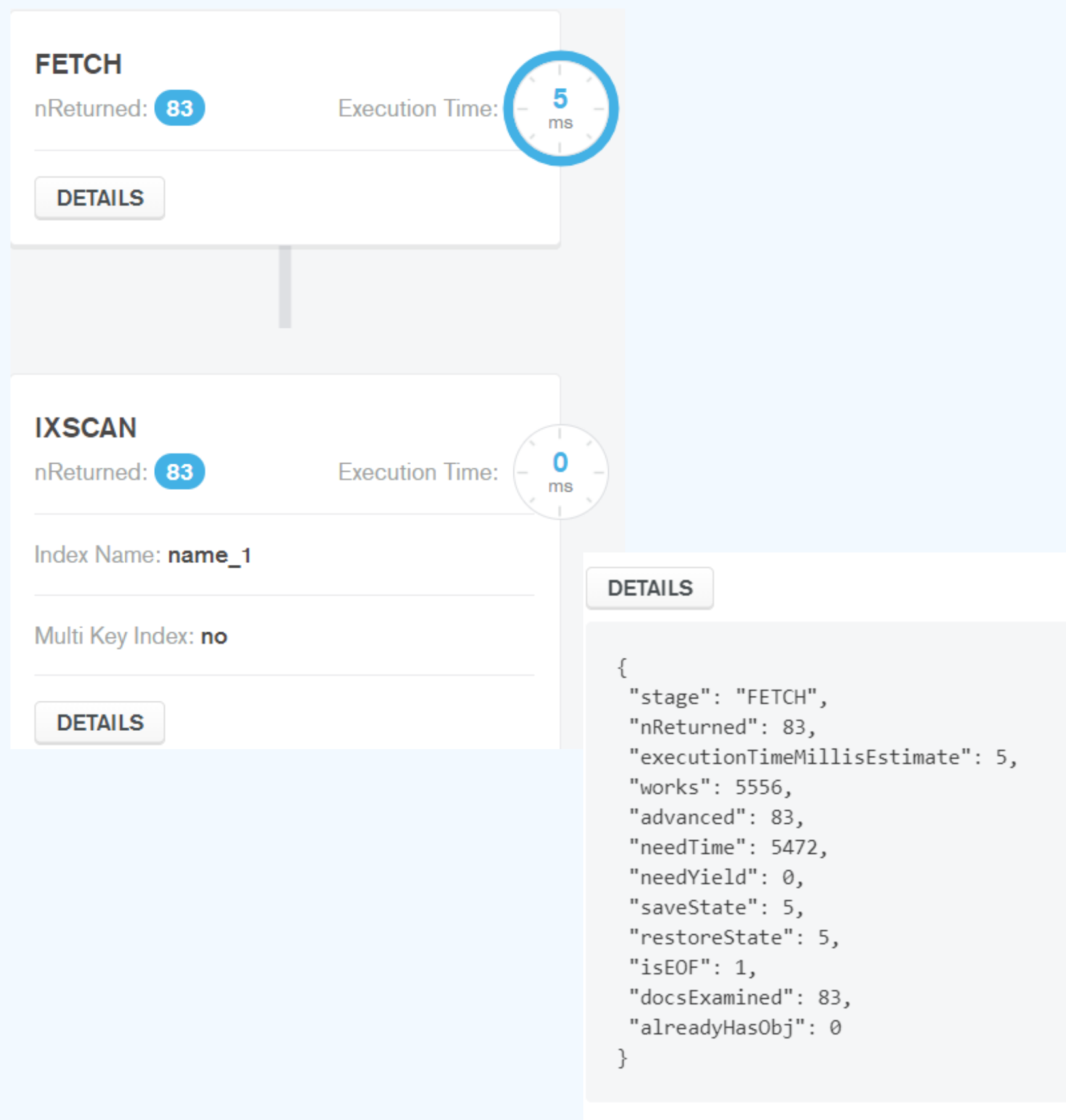
Documents Returned: 83	Actual Query Execution Time (ms): 14
Index Keys Examined: 5555	Sorted in Memory: no
Documents Examined: 83	Query used the following index: name ↑

Bancos de dados NoSQL

mongoDB

explain

Compass



The image displays the output of the MongoDB `explain()` command. It shows two stages: **FETCH** and **IXSCAN**. The **FETCH** stage has an execution time of 5 ms and returned 83 documents. The **IXSCAN** stage has an execution time of 0 ms and returned 83 documents, using index `name_1`. A **DETAILS** section provides a JSON representation of the execution plan.

FETCH
nReturned: 83 Execution Time: 5 ms
DETAILS

IXSCAN
nReturned: 83 Execution Time: 0 ms
Index Name: name_1
Multi Key Index: no
DETAILS

DETAILS

```
{
  "stage": "FETCH",
  "nReturned": 83,
  "executionTimeMillisEstimate": 5,
  "works": 5556,
  "advanced": 83,
  "needTime": 5472,
  "needYield": 0,
  "saveState": 5,
  "restoreState": 5,
  "isEOF": 1,
  "docsExamined": 83,
  "alreadyHasObj": 0
}
```

Bancos de dados NoSQL

mongoDB

Projeção

- Certo de que nos lembramos o conceito de projeção, este tipo de operação no MongoDB precisa de explicitação. Desta forma, a ordem dos parâmetros não faz diferença.
- Ponto de atenção nesta operação é que não é necessário informar quem não aparecerá, ou seja, será feita sua projeção no cursor.
- Só há uma exceção para esta situação que é a do campo `_id` que geralmente é gerado de forma automática.

Bancos de dados NoSQL

mongoDB

- A projeção alcança todos os níveis dos documentos.
- Pode-se realizar várias combinações para apresentarmos o melhor resultado.

Projeção

```
db.listingsAndReviews.find({"address.country": "Portugal"}, {"name": 1})

db.listingsAndReviews.find({"address.country": "Portugal"}, {"name": 1, "_id": 0})

db.listingsAndReviews.findOne({"address.country": "Portugal", "reviews.comments": {"$regex": "/melhor/i"}},
                               {"name": 1, "reviews.comments": 1, "_id": 0, "reviews.reviewer_name": 1})
```

Bancos de dados NoSQL

mongoDB

Arrays

\$elementMatch

- **\$elementMatch** é um operador que nos habilita a fazer consultas mais poderosas em campos que são matrizes de documentos (ou documentos embutidos, ou arrays).
- Ao se executar a consulta utilizando o **\$elementMatch** é necessário observarmos uma questão, qual a posição do parâmetro que está sendo inserida a condição.
- **Pode-se utiliza-lo tanto nos parâmetros de projeção quanto dos parâmetros de condição.**

```
db.grades.find({ "class_id": 431 },
               { "scores": { "$elemMatch": { "score": { "$gt": 85 } } } }
               ).pretty()

db.grades.find({ "scores": { "$elemMatch": { "type": "extra credit" } } },
               {"class_id":1, "_id": 0, "student_id" : 1}).pretty()
```

Bancos de dados NoSQL

mongoDB

\$and & \$or

Operadores Lógicos

- Já falamos algo sobre os operadores lógicos em nossa jornada, antes de avançarmos na combinação das expressões, vamos somente revisá-los de forma a organizar melhor o nosso código.
- Vamos lembrar que os operadores sempre iniciaram com o símbolo \$.
- **\$and** e **\$or** portanto são os principais e precisam ser lembrados para que não haja nenhuma inferência a erro.
- Eles precisam de parâmtros que são repassados através de colchetes e não chaves, **ok?**

Bancos de dados NoSQL

mongoDB

- Desta forma podemos associar e comparar os campos quando necessitarmos associá-los.
- Executamos comparações simples já utilizando recursos avançados como expressões regulares:

\$and & \$or

***Operadores
Lógicos***

```
db.companies.find({ $or : [{"category_code" : "social"},  
                             {"category_code" : "web"} ] } ).limit(3)  
  
db.companies.find({$and : [ {founded_year:{ $lte: 2006 } },  
                             {description : {$regex : /tech/i } }  
                             ] }).limit(5)
```

Bancos de dados NoSQL

mongoDB

\$and & \$or

Operadores Lógicos

- E comparações mais complexas associando os dois operadores.
- Reparem que é necessário bastante atenção na sintaxe para que não se perca no código.

```
db.companies.find({ $or: [{ $and : [
    {founded_year: 2004},
    { $or : [
        {"category_code" : "social"},
        {"category_code" : "web"}
    ] }
  ] },
  { $and : [ {founded_month: 10},
    { $or : [
        {"category_code" : "social"},
        {"category_code" : "web"}
    ] }
  ] }
] } )
```

Bancos de dados NoSQL

mongoDB

\$expr

Operador Expressivo

- Já vimos algo sobre as consultas "\$where" não devem ser usadas a menos que seja estritamente necessário: elas são muito mais lentas do que as consultas regulares.
- Cada documento deve ser convertido de BSON para um objeto *JavaScript* e, em seguida, executado por meio da expressão "\$where" no código.
- Os índices também não podem ser usados neste tipo de consulta e desta forma recomenda-se utilizar este tipo de consulta apenas quando não há outra alternativa.
- Há outra possibilidade que é a utilização do operador *\$expr*.

Bancos de dados NoSQL

mongoDB

\$expr

Operador Expressivo

- **\$expr** é um operador nas versões mais novas do MongoDB, a partir do 3.6, que não utiliza JavaScript e portanto, é mais rápido.
- Com ele é possível fazer diversas combinações de consulta aproveitando os índices e tornando as consultas mais performáticas.
- Permite o uso de expressões de agregação dentro da linguagem de consulta, variáveis e declarações condicionais.

Bancos de dados NoSQL

mongoDB

\$expr

Operador Expressivo

- **\$expr** permite comparar os campos dentro do mesmo documento entre si.
- Para utilizá-lo você precisa incluir o **\$** também nos nomes dos campos do documento e significa que você está olhando para o valor desse campo, em vez de apenas para o nome.

```
use sample_training

db.trips.find({ "$expr": { "$eq": [ "$end station id", "$start station id" ] } }).count()

db.trips.find({ "$expr": { "$and": [ { "$gt": [ "$tripduration", 1200 ] },
                                     { "$eq": [ "$end station id", "$start station id" ] } ] } }).count()
```

Bancos de dados NoSQL

mongoDB

count()

- O método **count()** nos exhibe a quantidade de registro que atendo ao filtro especificado.
- Lembrando que métodos são sempre aninhados ao final do nosso comando de consulta find().

```
db.trips.find({ "$expr": { "$eq": [ "$end station id", "$start station id" ] } }).count()

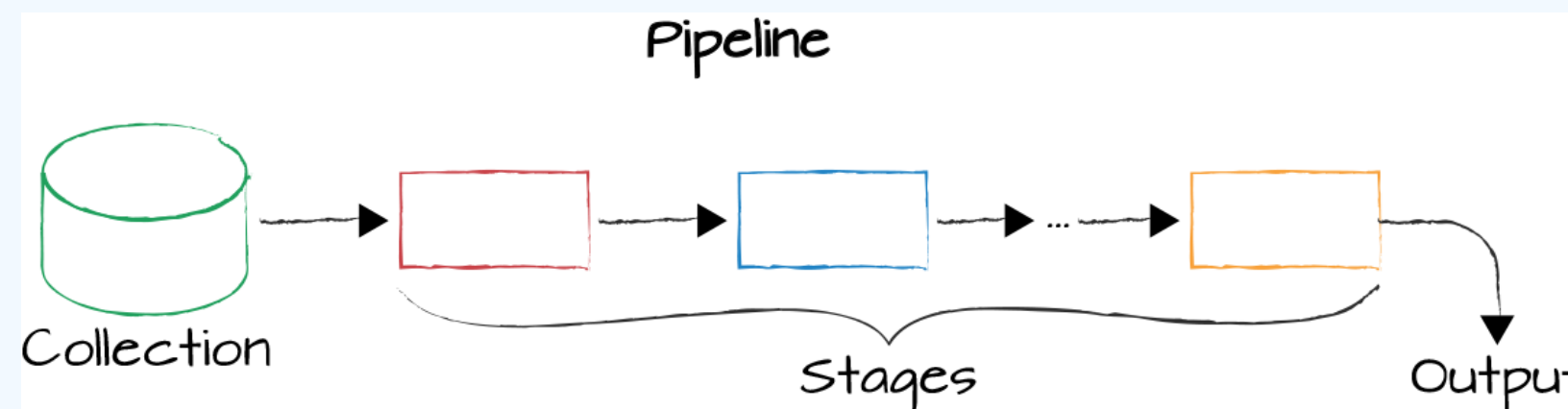
db.companies.find({ $or: [{ $and : [
    {founded_year: 2004},
    { $or : [
        {"category_code" : "social"},
        {"category_code" : "web"}
    ] }
  ] },
  { $and : [ {founded_month: 10},
    { $or : [
        {"category_code" : "social"},
        {"category_code" : "web"}
    ] }
  ] }
] }
} ).count()
```

Bancos de dados NoSQL

mongoDB

Aggregation framework

- A estrutura de agregação funciona como um **pipeline** (segmentação das instruções), onde a ordem das ações no **pipeline** é importante.
- Cada ação é executada na ordem em que a listamos e isto significa que fornecemos nossos dados ao pipeline e então descrevemos como esse pipeline os tratará usando estágios de agregação.
- Finalmente os dados transformados emergem no final do pipeline.

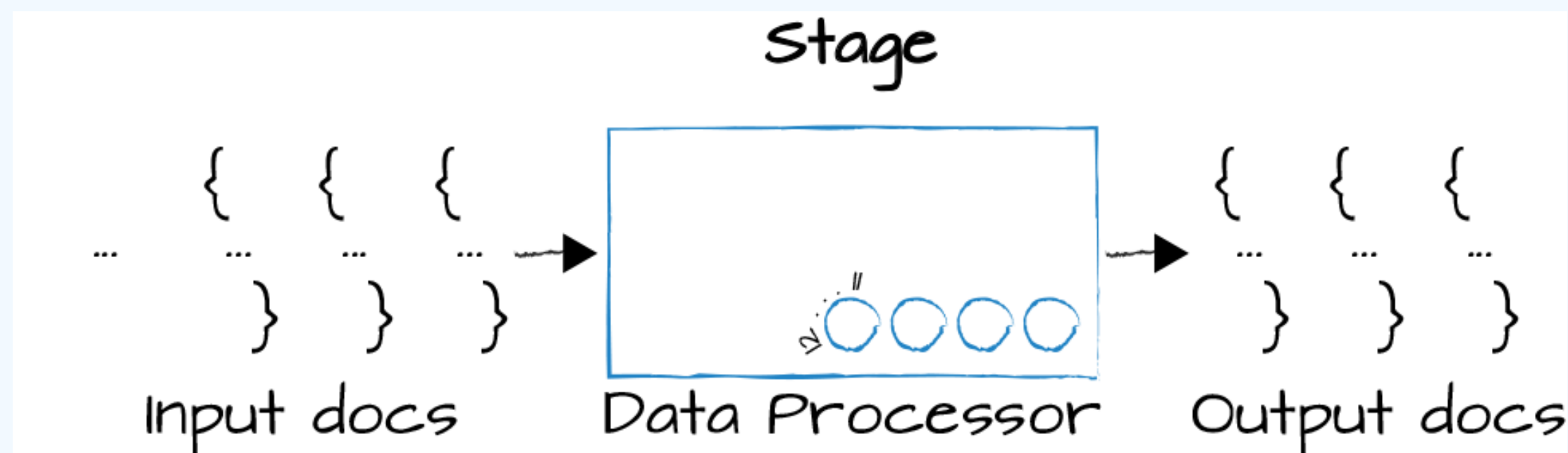


Bancos de dados NoSQL

mongoDB

Aggregation framework

- Aprofundando um pouco mais no assunto e considerar os estágios individuais.
- Um estágio individual de um pipeline de agregação é uma unidade de processamento de dados que recebe um fluxo de documentos de entrada um por vez, os processa e produz um fluxo de saída de documentos um por vez.



Bancos de dados NoSQL

mongoDB

Aggregation framework

- Cada estágio fornece um conjunto de configurações que podemos parametrizar para realizar qualquer tarefa parametrizado-o, para a coleção particular com a qual estamos trabalhando.
- Assim eles assumem a forma de operadores que podemos fornecer que modificarão campos, realizarão operações aritméticas, remodelarão documentos ou realizarão algum tipo de tarefa de acumulação ou uma variedade de outras coisas.

Bancos de dados NoSQL

mongoDB

Aggregation framework

- A estrutura de agregação, em sua forma mais simples, é apenas outra maneira de consultar dados no MongoDB.
- Tudo o que sabemos fazer usando a linguagem de consulta MongoDB (**MQL**) também pode ser feito usando a estrutura de agregação.
- É uma estrutura (método) diferente do **find()** para utilizar a estrutura de agregação deve-se utilizar o método **aggregate()**
- Vejam a comparação:

```
db.listingsAndReviews.find({ "amenities": "Wifi" },  
                           { "price": 1, "address": 1, "_id": 0 }).pretty()
```

Bancos de dados NoSQL

mongoDB

Aggregation framework

```
db.listingsAndReviews.aggregate([
  { $match : { "amenities": "Wifi" } },
  { $project: { "price": 1, "address" : 1, "_id": 0 } }
]).pretty();
```

- Os resultados são os mesmos porém, temos mais na minha opinião o código fica bem mais entendível não acham?
- E também temos mais recursos que podem ser combinados:

```
db.listingsAndReviews.aggregate([ { $project: { "address": 1, "_id" : 0 } },
  { $group: { _id : "$address.country",
    "count": { "$sum": 1 } }
  } ])
```

```
db.companies.aggregate([ { $match: { founded_year : 2004 } },
  { $limit : 5 },
  { $project : { _id: 0,
    name : 1 } } ])
```


Bancos de dados NoSQL

mongoDB

- Há uma infinidade de recursos para trabalharmos no universo do método de agregação, a ideia aqui é apenas demonstração inicial e vocês devem se aprofundar mais nas pesquisas.

Aggregation framework

```
db.companies.aggregate( [
    { $match: { "relationships.person": { $ne: null } } },
    { $project: { relationships: 1, _id: 0 } },
    { $unwind: "$relationships" },
    { $group: {
        _id: "$relationships.person",
        count: { $sum: 1 }
    } },
    { $sort: { count: -1 } }
]).pretty()
```

Bancos de dados NoSQL

mongoDB

Aggregation framework

```
db.companies.aggregate([
  { $match: { funding_rounds: { $exists: true, $ne: [ ] } } },
  { $unwind: "$funding_rounds" },
  { $sort: { "funding_rounds.funded_year": 1,
    "funding_rounds.funded_month": 1,
    "funding_rounds.funded_day": 1 } },
  { $group: {
    _id: { company: "$name" },
    first_round: { $first: "$funding_rounds" },
    last_round: { $last: "$funding_rounds" },
    num_rounds: { $sum: 1 },
    total_raised: { $sum: "$funding_rounds.raised_amount" }
  } },
  { $project: {
    _id: 0,
    company: "$_id.company",
    first_round: {
      amount: "$first_round.raised_amount",
      article: "$first_round.source_url",
      year: "$first_round.funded_year"
    },
    last_round: {
      amount: "$last_round.raised_amount",
      article: "$last_round.source_url",
      year: "$last_round.funded_year"
    },
    num_rounds: 1,
    total_raised: 1,
  } },
  { $sort: { total_raised: -1 } }
] ).pretty()
```

Bancos de dados NoSQL

mongoDB

Transactions

- Relembrando o conceito de transações, elas são grupos lógicos de processamento em um banco de dados.
- Cada grupo ou transação pode conter uma ou mais operações, como leituras e/ou gravações em vários documentos.
- O MongoDB oferece suporte a transações onde, existem situações em que seu aplicativo exige leituras e gravações em vários documentos (em uma ou mais coleções) como parte desta unidade lógica de processamento.
- O aspecto mais importante de uma transação é que ela nunca é parcialmente concluída - ela é bem-sucedida ou falha.

Bancos de dados NoSQL

mongoDB

Transactions

- No MongoDB as transações são disponibilizadas através de API com duas ofertas: core e a callback API.
- **API core:** não fornece lógica de nova tentativa para a maioria dos erros e exige que o desenvolvedor codifique a lógica para as operações, a função de confirmação da transação e qualquer tentativa e lógica de erro necessária.
- **API Callback:** fornece uma única função que envolve um grande grau de funcionalidade quando comparada à API principal, incluindo iniciar uma transação associada a uma sessão lógica especificada, executar uma função fornecida como a função de retorno de chamada e, em seguida, confirmar a transação (ou anular em erro).

Bancos de dados NoSQL

mongoDB

Transactions

- Em ambas as APIs, o desenvolvedor é responsável por iniciar a sessão lógica que será utilizada pela transação e exigem que as operações em uma transação sejam associadas a uma sessão lógica específica (ou seja, passar a sessão para cada operação).
- Uma sessão de cliente é iniciada por um aplicativo e usada para interagir com uma sessão de servidor.

TRANSACTIONS COMPARAÇÃO DE APIs

Core API	Callback API
Requer uma chamada explícita para iniciar a transação e confirmá-la.	Inicia uma transação, executa as operações especificadas e confirma (ou aborta em caso de erro).
Não incorpora lógica de tratamento de erros para <code>TransientTransactionError</code> e <code>UnknownTransactionCommitResult</code> e, em vez disso, fornece a flexibilidade de incorporar tratamento de erros personalizado para esses erros.	Incorpora automaticamente a lógica de tratamento de erros para <code>TransientTransactionError</code> e <code>UnknownTransactionCommitResult</code> .
Requer que uma sessão lógica explícita seja passada à API para a transação específica.	Requer que uma sessão lógica explícita seja passada à API para a transação específica.

Bancos de dados NoSQL

mongoDB

Compass

- Compass é a ferramenta gráfica UI para se acessar o mongoDB.
- **Vamos ao demo rápido do Compass!**
- Com o advento das soluções em nuvem há a oferta do Atlas que é a solução mongo em nuvem com oferta de acesso gratuita.
- Além disto há vários cursos que são ofertados de forma gratuita e que envolvem conteúdos do básico ao avançado.
- Cadastre-se e aproveite:
- <https://university.mongodb.com/>

***Now, Let's go to pratice!
On MongoDB!***

REFERÊNCIAS BIBLIOGRÁFICAS

SANTOS, Hudson Leandro, 2020. Vamos falar um pouco do WiredTiger?. Disponível em: <<https://hudsondosantos.com/2020/08/18/vamos-falar-um-pouco-do-wiredtiger/>>. Acesso em: 25 Nov. 2021.

PÚBLIO, Angelo. CRUD: o que é este conceito no Desenvolvimento de SistemasDisponível em: <<https://angelopublico.com.br/blog/crud> > Acesso em: 06 Dez. 2021.

LÓSCIO, B. F.;; **OLIVEIRA**, H. R. de;; **PONTES**, J. C. de S. NoSQL no desenvolvimento de aplicações web colaborativas. In: Simpósio Brasileiro de Sistemas Colaborativos – SBSC, 8., 2011, Paraty (RJ). Anais... Paraty: SBC, 2011.

VIEIRA, M. R. et al. Bancos de Dados NoSQL: conceitos, ferramentas, linguagens e estudos de casos no contexto de Big Data. In: Simpósio Brasileiro de Bancos de Dados, 27., 2012, São Paulo. Anais... São Paulo: SBC, 2012.