

Banco de Dados - Parte 04

NoSQL with  mongoDB.

1

Estrutura da Aula

- Introdução a NoSQL
- Introdução a MongoDB
- Linguagem de consulta usando MongoDB Shell
- Framework de Agregação
- MongoDB Java Driver
- Mongo Charts
- Performance
- Exercícios

2

Vídeos Extras sobre Performance

- Usando o EXPLAIN
- Índices Geoespaciais e de Texto
- Uso Eficiente de Índices
- Ferramentas de Monitoramento
- Replicação
- Aspectos de Escrita
- Transações
- Fragmentação

3

Introdução a NoSQL

4

NoSQL

- Termo usado para Bancos de Dados NÃO RELACIONAIS de alto desempenho
 - Não normalizados
 - Relações aninhadas
 - Multi-valoração
 - Sem formato
 - Sem esquemas

5

NoSQL

- Usam diversos modelos de dados, baseados em:
 - Documentos
 - Grafos
 - Chave-valor
 - Colunas
- Nesta aula
 - Baseados em documentos

6

Vantagens NoSQL

- Fácil desenvolvimento
- Desempenho escalável
- Alta disponibilidade
- Alta resiliência

7

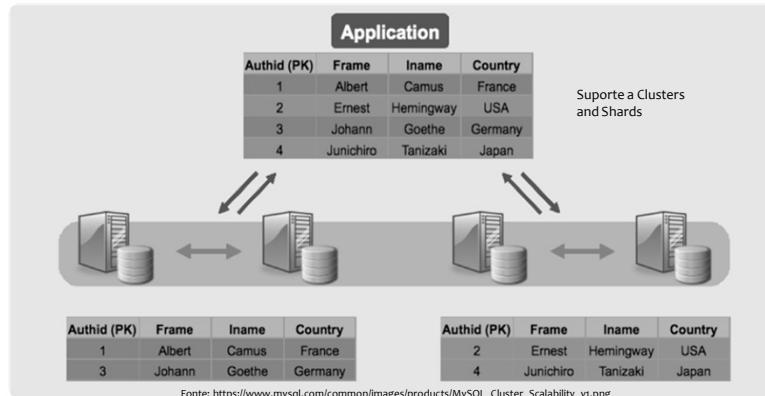
BD Relacional vs. BD NoSQL

Característica	BD Relacional	NoSQL
Validade de Dados	Alta garantia	Baixa garantia
Linguagem de Consulta	Linguagem de consulta estruturada (SQL)	Linguagem de consulta não declarativa
Tipo de Dados	Dados relacionais e relações armazenadas em tabelas separadas	Dados não estruturados e imprevisíveis
Armazenamento de Dados	Modelo relacional com tabelas (linhas e colunas)	Modelos de dados diferentes, baseados em: documentos, grafos, chave-valor, colunas
Esquemas e Flexibilidade	Fixo	Dinâmico
Escalabilidade	Vertical only? No	Horizontal
Conformidade	A maioria dos BD relacionais estão em conformidade com todas as funções de SGBD	Remoção de algumas funções de SGBD para obter performance e escalabilidade

Fonte: A mechanism to evaluate context-free queries inspired in LR(1) parsers over graph databases. Dissertação de Mestrado, 2018. Fred de Castro Santos.

8

BD Relacional vs. BD NoSQL



9

BD Relacional vs. BD NoSQL

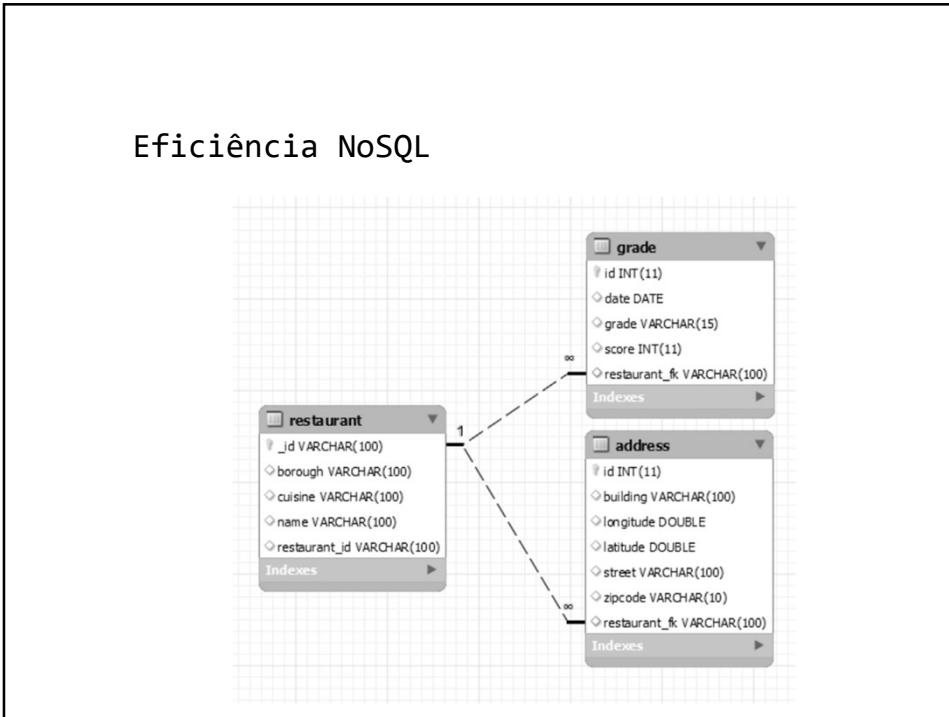
Característica	BD Relacional	NoSQL
Validade de Dados	Alta garantia	Baixa garantia
Linguagem de Consulta	Linguagem de consulta estruturada (SQL)	Linguagem de consulta não declarativa
Tipo de Dados	Dados relacionais e relações armazenadas em tabelas separadas	Dados não estruturados e imprevisíveis
Armazenamento de Dados	Modelo relacional com tabelas (linhas e colunas)	Modelos de dados diferentes, baseados em: documentos, grafos, chave-valor, colunas
Esquemas e Flexibilidade	Fixo	Dinâmico
Escalabilidade	Vertical only? No	Horizontal
Conformidade	A maioria dos BD relacionais estão em conformidade com todas as funções de SGBD	Remoção de algumas funções de SGBD para obter performance e escalabilidade

Fonte: A mechanism to evaluate context-free queries inspired in LR(1) parsers over graph databases. Dissertação de Mestrado, 2018. Fred de Castro Santos.

10



11



12

Eficiência NoSQL (SC Comp)

- Comparativo com Base de Dados de Restaurantes

```
MongoDB: Loaded 25359 restaurants in 853ms
MySQL: Inserting restaurants in MySQL...
25359 restaurantes inserted.
Inserting in MySQL: Inserted in 18234ms
MySQL Naive: Loading restaurants from MySQL...
MySQL Naive: Loaded 25359 restaurants in 8022ms
MySQL One Access: Loading restaurants from MySQL...
MySQL One Access: Loaded 25359 restaurants in 1711ms
Results: Comparing MongoDB data with MySQL One Access data...
OK! They are equal.
Results: Finished in 61ms
Results: Comparing MongoDB data with MySQL Naive data...
OK! They are equal.
Results: Finished in 23ms
```

13

14

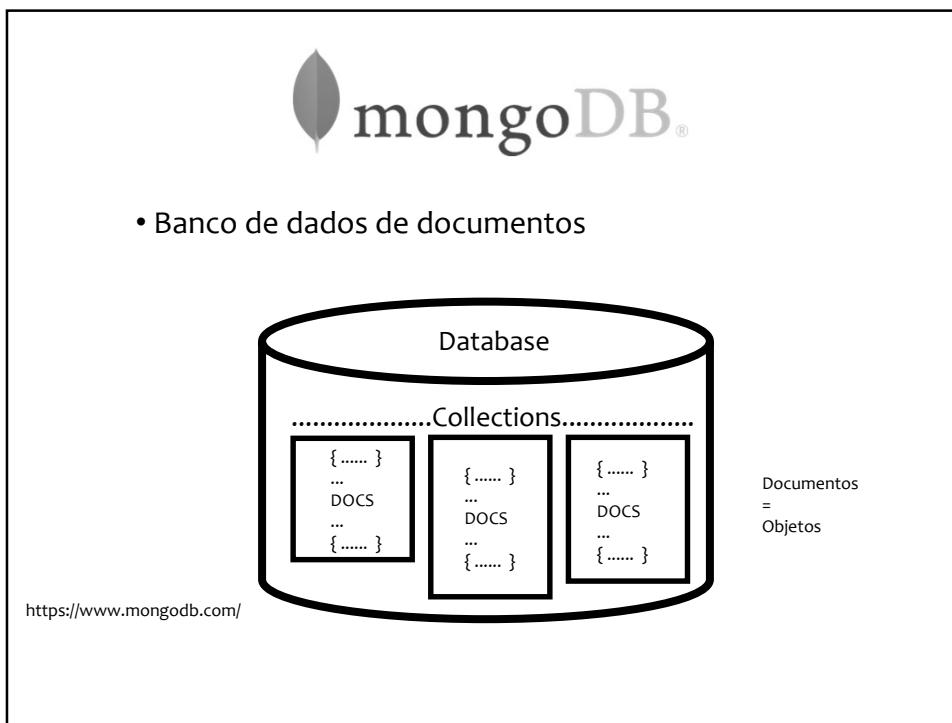
Estrutura da Aula

- Introdução a NoSQL
- Introdução a MongoDB
- Linguagem de consulta usando MongoDB Shell
- Framework de Agregação
- MongoDB Java Driver
- Mongo Charts
- Performance
- Exercícios

15

Introdução a  mongoDB®
Estrutura de Documentos
JSOB
Interfaces (Compass, Atlas, Shell)

16



17



18



- Esta hierarquia nos permite
 - Formar grupos de itens similares em coleções
 - Agrupar coleções para a mesma aplicação em um BD
- Políticas de segurança permitem estabelecer papéis diferentes e acessos diferentes em nível de base de dados e coleções
 - Nível de documentos ainda não é suportado

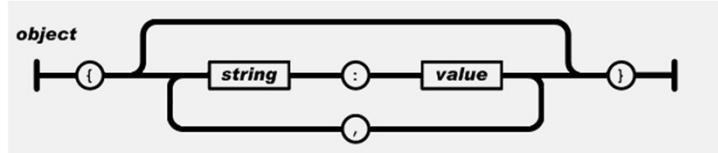
19

JSON

- JavaScript Object Notation (json.org)
- Notação usada para escrever os documentos
- Notação extremamente simples de representação de objetos

20

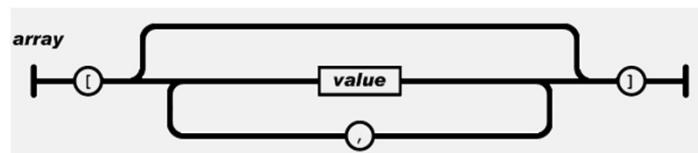
JSON



- `{"key":value, ..., "key":value}`
- value é um valor MongoDB

21

JSON



- `[value, value, ..., value]`

22

JSON

```
movie.json
1 {
2   "title": "Titanic",
3   "year": 1997,
4   "imdbId": "tt0120338",
5   "mpaaRating": "PG-13",
6   "genre": "Drama, Romance",
7   "viewerRating": 7.7,
8   "viewerVotes": 716392,
9   "runtime": 194,
10  "director": "James Cameron",
11  "cast": [
12    "Leonardo DiCaprio",
13    "Kate Winslet",
14    "Kathy Bates",
15    {
16      "name": "Billy Zane",
17      "birthName": "William George Zane, Jr."
18    }
19  ],
20  "plot": "A seventeen-year-old aristocrat falls in love with a kind",
21  "language": "English, French, German, Swedish, Italian, Russian",
22  "lastUpdated": "2015-09-13 00:41:42.117"
23 }
24 |
```

23

JSON

- In COMPASS

```
_id:ObjectID('595fd402a8c854210915ee15')
title:"Star Wars: Episode IV - A New Hope"
year:1977
rated:"PG"
runtime:121
countries:Array
0:"USA"
genres:Array
0:"Action"
1:"Adventure"
2:"Fantasy"
director:"George Lucas"
writers:Array
0:"George Lucas"
actors:Array
0:"Mark Hamill"
1:"Harrison Ford"
2:"Carrie Fisher"
3:"Peter Cushing"
plot:"Luke Skywalker joins forces with a Jedi Knight, a cocky pilot, a wookie..."
poster:"http://ia.media-imdb.com/images/M/MVSBHTU4NTcz0OkwM15Bhl58anBnXkFt2Tcw..."
imdb:Object
id:"tt0076759"
rating:8.7
votes:822849
tomato:Object
meter:94
image:"certified"
rating:8.5
reviews:99
fresh:93
consensus:"A legendarily expansive and ambitious start to the sci-fi saga, George..."
userMeter:96
userRating:4.1
userReviews:851120
metacritic:92
awards:Object
wins:38
nominations:27
text:"Won 6 Oscars. Another 38 wins & 27 nominations."
type:"movie"
```

24

BSON

- Binary JSON - <http://bsonspec.org/>
- MongoDB armazena dados usando arquivos BSON
- JSON tem apenas um tipo para números e não tem o tipo data (uso de strings ou objetos aninhados)
 - BSON estende JSON oferecendo inteiros, doubles, datas e dados binários

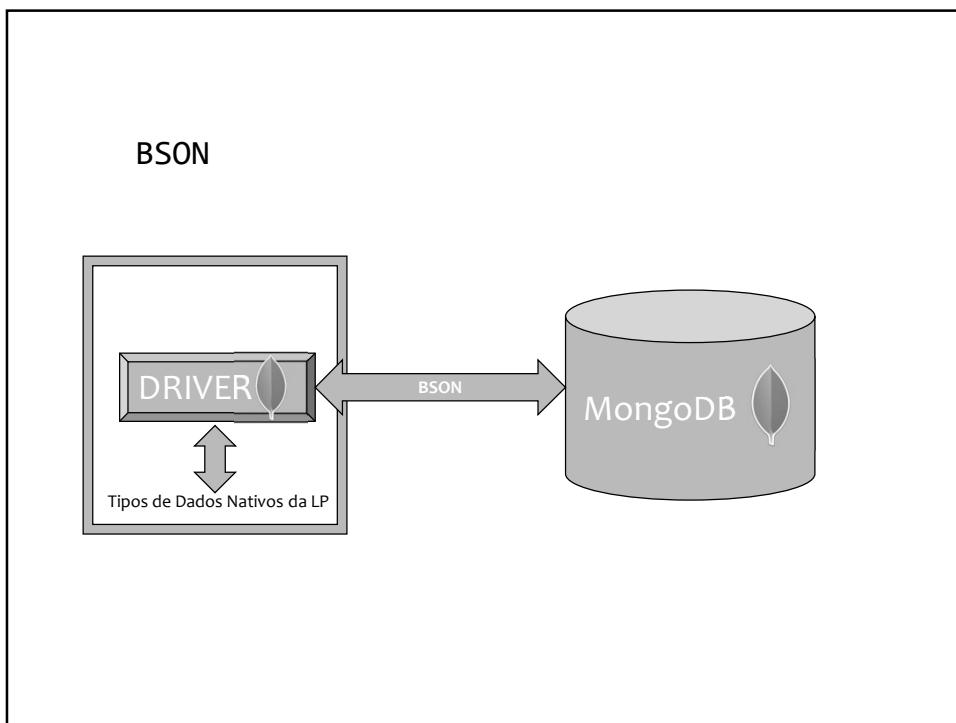
25

BSON

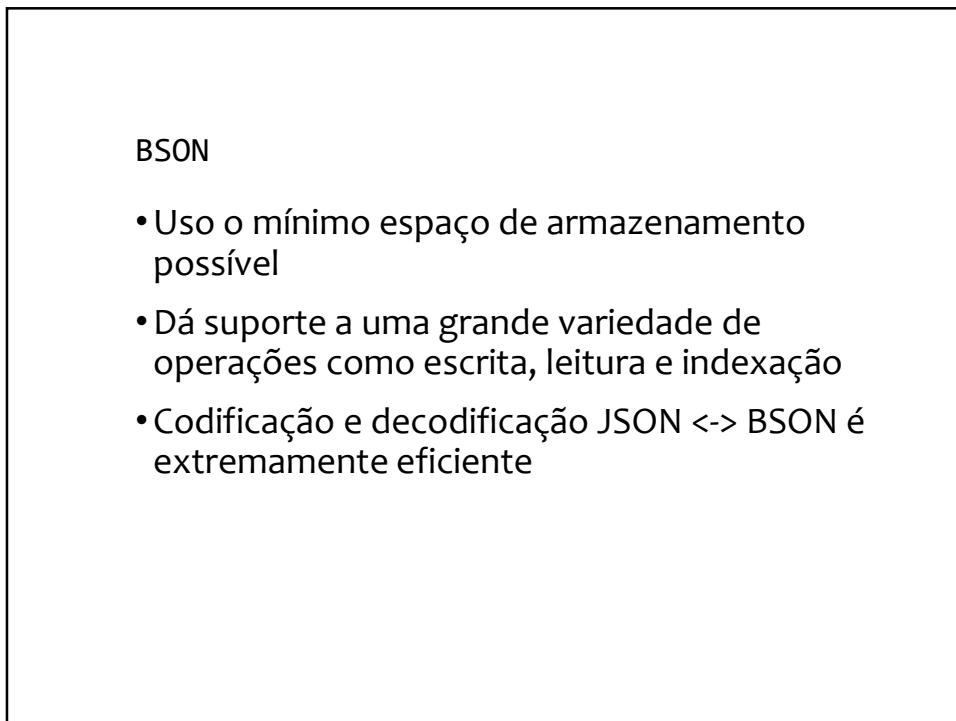
```
// JSON
{ "hello" : "world" }

// BSON
"\x16\x00\x00\x00\x02hello\x00
\x06\x00\x00\x00world\x00\x00"
```

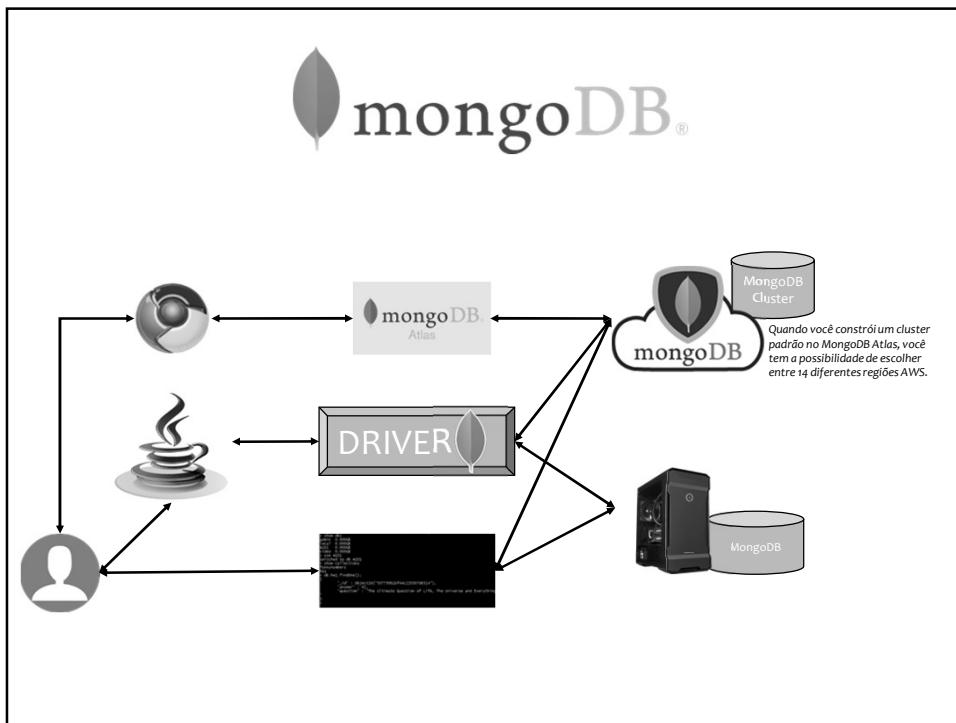
26



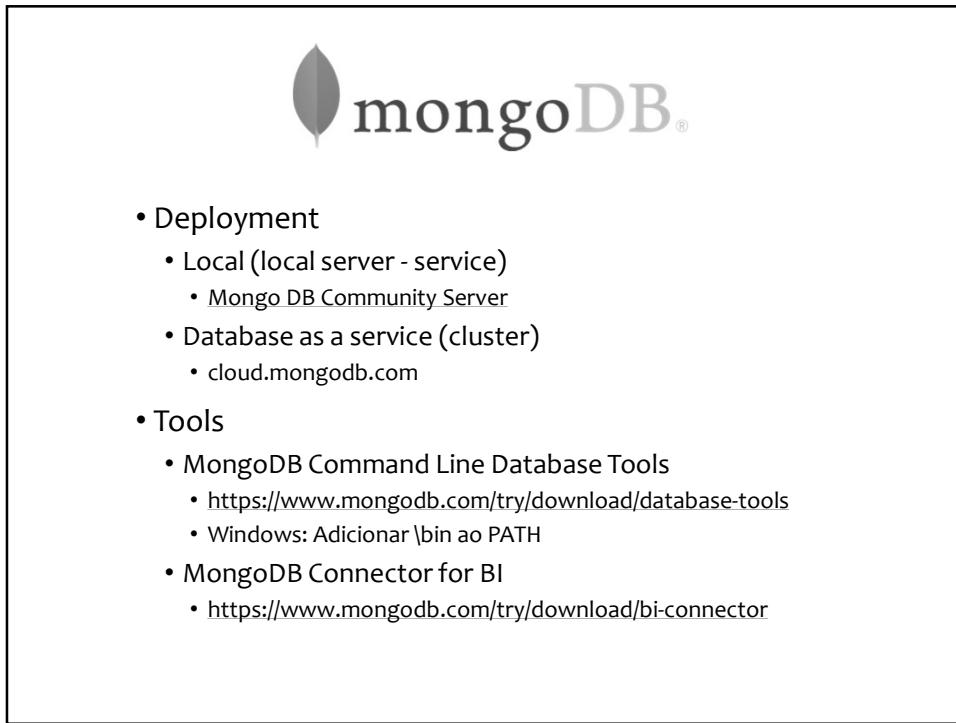
27



28



29



30



- Interfaces (SCo)
 - Atlas (Web client for cluster management)
 - <https://www.mongodb.com/cloud/atlas>
 - COMPASS (GUI Client)
 - <https://www.mongodb.com/products/compass>
 - Mongo Shell (Shell client)
 - Download from the atlas
 - Driver
- Deployment
 - Local (local server) or database as a service (cluster)

31

ATLAS

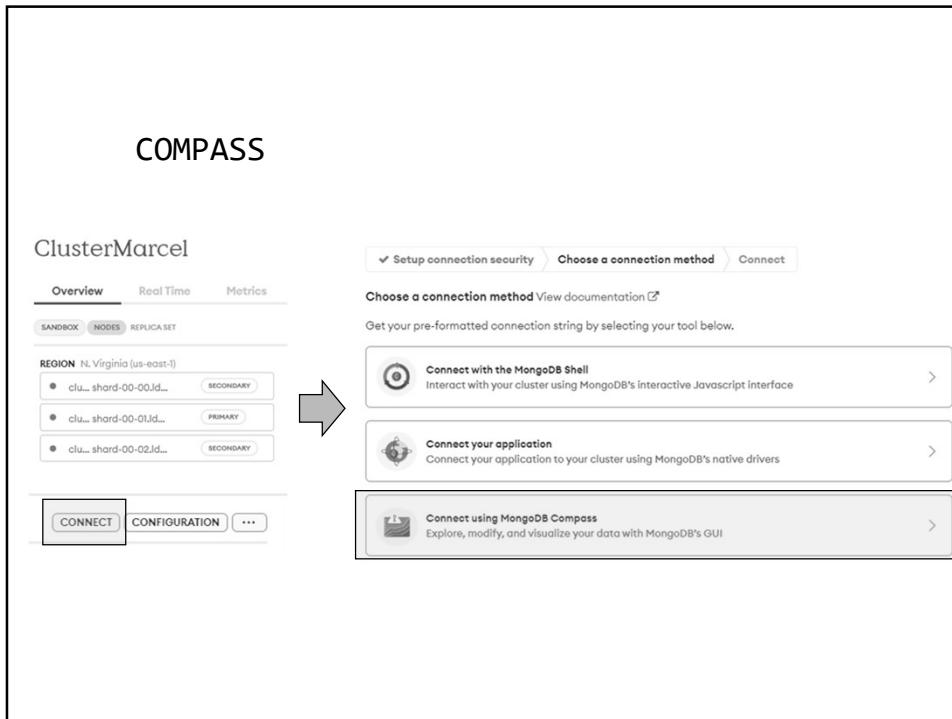
MARCELOLIVEIRA (ORGANIZATION) > MARCELOLIVEIRA

Database Deployments

+ Create

ClusterMarcel	Connect	View Monitoring	Browse Collections	...	FREE	SHARED
Enhance Your Experience For production throughput and richer metrics, upgrade to a dedicated cluster now!	R 0.01 0.03/s	W 0 Last 6 minutes	Connections 18.0 18.0	In 197.5 B/s 1.3 KB/s	Out 1.3 KB/s Last 6 minutes	Data Size 512.0 MB
VERSION: 5.0.9 REGION: AWS / N. Virginia (us-east-1) CLUSTER TIER: M0 Sandbox (General) TYPE: Replica Set - 3 nodes BACKUPS: Inactive LINKED APP SERVICES: None Linked ATLAS SEARCH: Create Index						

32



33

The screenshot shows the MongoDB Compass interface for the 'tourinfo.restaurants' collection. The left sidebar lists databases and collections, with 'tourinfo' expanded to show 'restaurants' and other sub-collections like 'agg', 'blog', 'config', 'course', 'local', 'school', 'video', and 'zipcode'. The main area displays the 'tourinfo.restaurants' collection with 25.4k documents and 1 index. The document list shows three entries:

```

{
  "_id": ObjectId("62d6ff32dc064643a6b997d54"),
  "address": Object,
  "borough": "Manhattan",
  "cuisine": "Irish",
  "grades": Array,
  "name": "O' Reynolds Pub And Restaurant",
  "restaurant_id": "30191941"
}

{
  "_id": ObjectId("62d6ff32dc064643a6b997d55"),
  "address": Object,
  "borough": "Bronx",
  "cuisine": "American",
  "grades": Array,
  "name": "Mild Asia",
  "restaurant_id": "40357217"
}

{
  "_id": ObjectId("62d6ff32dc064643a6b997d56"),
  "address": Object,
  "borough": "Brooklyn",
  "cuisine": "Delicatessen",
  "grades": Array,
  "name": "Wilken's Fine Food",
  "restaurant_id": "40356483"
}

```

Below the document list are buttons for ADD DATA, VIEW, and various search/filter options. The status bar at the bottom indicates 'Displaying documents 1 - 20 of 25359'.

34

The screenshot shows the MongoDB Atlas ClusterMarcel dashboard. On the left, there's a cluster overview with nodes listed under the 'N. Virginia (us-east-1)' region. One node is marked as 'PRIMARY' and another as 'SECONDARY'. Below this are buttons for 'CONNECT', 'CONFIGURATION', and '...'. On the right, a large arrow points from the 'CONNECT' button towards a section titled 'Choose a connection method'. This section contains three options: 'Connect with the MongoDB Shell' (selected), 'Connect your application', and 'Connect using MongoDB Compass'.

35

The screenshot shows the MongoDB shell (mongosh) running on Windows. It displays the following startup log:

```
C:\Users\marcelo>mongosh
Current Mongosh Log ID: 61eb33f121ad20a4201a1163
Connecting to: mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1
.1.9
Using MongoDB: 4.4.2
Using Mongosh: 1.1.9

For mongosh info see: https://docs.mongodb.com/mongodb-shell/
-----
The server generated these startup warnings when booting:
2022-01-18T15:56:16.260-03:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----
test> show databases
admin      65.5 kB
agg        279 kB
blog       8.99 MB
config     111 kB
course     246 kB
local      106 kB
school     768 MB
tourinfo    4.27 MB
video      1.57 MB
zipcode    1.65 MB
```

36

MongoDB Compass

- (SC1) Conectando COMPASS ao Cluster

The screenshot shows the MongoDB Compass interface. On the left, there's a cluster overview for 'ClusterMarcel' with tabs for Overview, Real Time, and Metrics. Under Overview, it shows a 'Sandbox' section with three nodes: 'clu... shard-00-00Id...' (Secondary), 'clu... shard-00-01Id...' (Primary), and 'clu... shard-00-02Id...' (Secondary). Below this is a 'CONNECT' button. On the right, there's a 'Choose a connection method' section with three options: 'Connect with the MongoDB Shell', 'Connect your application', and 'Connect using MongoDB Compass'. The 'CONNECT using MongoDB Compass' option is highlighted with a large grey arrow pointing from the 'CONNECT' button on the left.

37

MongoDB Compass

- (SC1) Conectando COMPASS ao Cluster

② Copy the connection string, then open MongoDB Compass.

The screenshot shows the 'New Connection' dialog in MongoDB Compass. It has a 'URI' field containing the connection string 'mongodb+srv://MarcelOliveira:<password>@clustermarcel.ldm9o.mongodb.net/test'. There are 'Save' and 'Save & Connect' buttons at the bottom. To the right of the URI field is a 'Edit Connection String' toggle and a 'FAVORITE' button with a star icon. A red box highlights the 'Save & Connect' button.

38

MongoDB Compass

- (SC2) Explorando o COMPASS

39

The screenshot shows the MongoDB Compass interface. On the left, there's a sidebar with connection details: 'localhost:27017', 'HOST localhost', 'CLUSTER Standalone', and 'EDITION MongoDB 4.4.2 Community'. Below this is a search bar labeled 'Filter your data' with suggestions: 'admin', 'config', 'local', and 'tourinfo'. The main area has tabs for 'Databases' (selected) and 'Performance'. Under 'Databases', there's a 'CREATE DATABASE' button and a table with the following data:

Database Name	Storage Size	Collections	Indexes
admin	20.0KB	0	1
config	36.0KB	0	2
local	20.0KB	1	1
tourinfo	3.6MB	1	1

40

Visualização de Documentos Com CRUD

The screenshot shows the MongoDB Compass interface connected to the 'tourinfo' database and the 'restaurants' collection. The main pane displays several documents with fields like address, borough, cuisine, and grades. A modal window is open, showing the detailed schema of a document, including its ID, address object, borough, cuisine, and grades array.

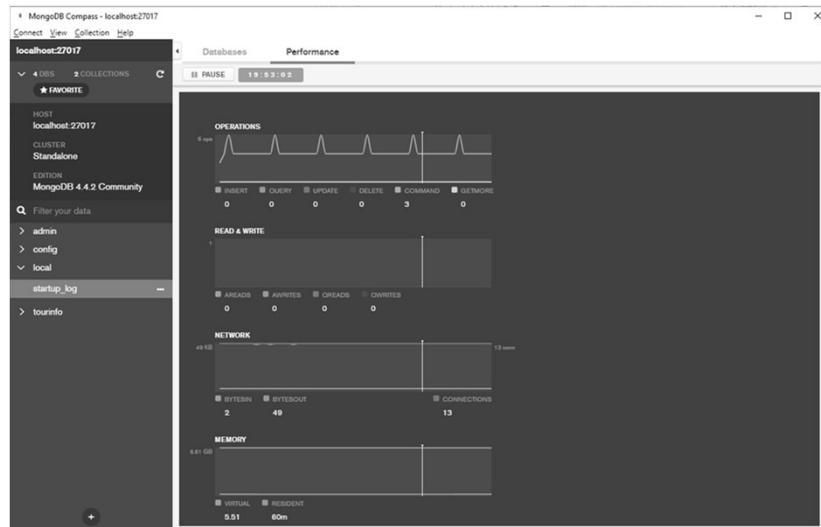
41

Visualização de Esquemas

The screenshot shows the MongoDB Compass interface with the 'Schema' tab selected for the 'tourinfo.restaurants' collection. It displays histograms for the 'address' field (document type), 'borough' field (string type), and 'cuisine' field (string type). The 'borough' histogram shows categories for Manhattan, Queens, Brooklyn, and Bronx.

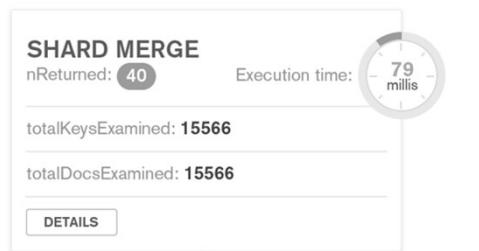
42

Status do Servidor e Performance de Consultas



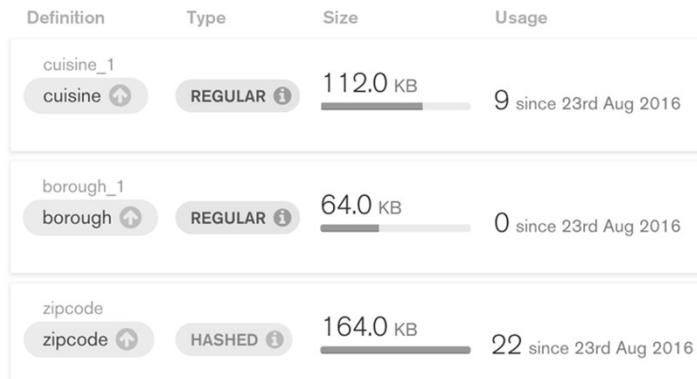
43

Análise de Performance



44

Gerenciamento de Índices



45

Validação de Dados

Field Name		Rule	Nullable
age	EXISTS		<input type="checkbox"/>
email	REGEX	(?^a-zA-Z0-9!#\\$%&+*/)	<input type="checkbox"/> OPTIONS
status	EXISTS		<input type="checkbox"/>

46

MongoDB Compass

- Extensível através de plug-ins
 - API para Compass Plugin Framework
- Tipos escalares
 - String
 - Integer (int32)
 - Floating point (double)
 - Date

47

MongoDB Compass

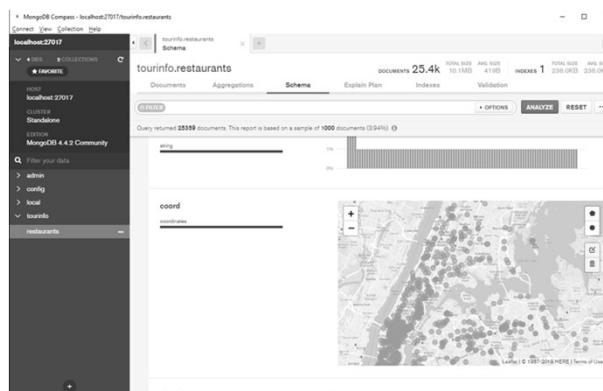
- Tipos de valores agregados
 - Documentos aninhados
 - Array
 - Visão esquema vs Visão documento

48

MongoDB Compass

- Dados geoespaciais

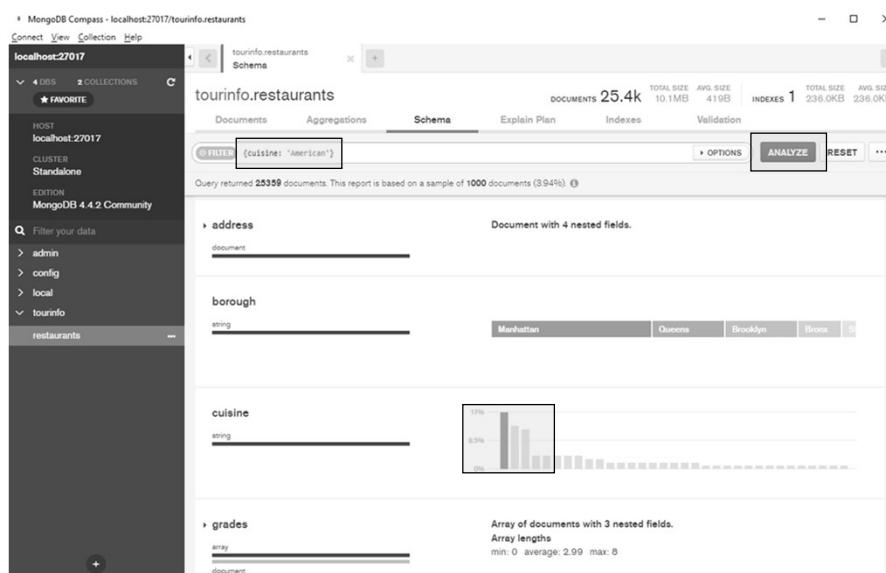
- Não é um tipo de dados
- Podemos manipular dados geoespaciais usando estruturas de documentos



49

MongoDB Compass

- (SC3) Filtrando coleções com consultas



50

MongoDB Compass - localhost:27017/tourinfo.restaurants

localhost:27017

4 DBS 2 COLLECTIONS

FAVORITE

HOST localhost:27017

CLUSTER Standalone

EDITION MongoDB 4.4.2 Community

Filter your data

admin config local tourinfo restaurants

tourinfo.restaurants

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER {cuisine: 'American'}

Query returned 6183 documents This report is based on a sample of 1000 documents (16.17%).

address Document with 4 nested fields.

document

borough

string Manhattan Brooklyn Queens Bronx

cuisine

string American

grades Array of documents with 3 nested fields.

Array lengths min: 0 average: 3.8 max: 8

array document

51

MongoDB Compass - localhost:27017/tourinfo.restaurants

localhost:27017

4 DBS 2 COLLECTIONS

FAVORITE

HOST localhost:27017

CLUSTER Standalone

EDITION MongoDB 4.4.2 Community

Filter your data

admin config local tourinfo restaurants

tourinfo.restaurants

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER {cuisine: 'American'}

ADD DATA FIND RESET

Displaying documents 1 - 20 of 6183

```
_id:ObjectId("5fd3ced15a3e97e69151fbbe")
address:Object
borough:"Manhattan"
cuisine:"American"
grades:Array
name:"Kiviera Caterer"
restaurant_id:"40356018"

_id:ObjectId("5fd3cd15a3e97e69151fbbe3")
address:Object
borough:"Queens"
cuisine:"American"
grades:Array
name:"Runos On The Boulevard"
restaurant_id:"40356151"

_id:ObjectId("5fd3cd15a3e97e69151fbbe5")
address:Object
borough:"Bronx"
cuisine:"American"
grades:Array
name:"Wild Asia"
restaurant_id:"40357213"

_id:ObjectId("5fd3cd15a3e97e69151fbbe6")
address:Object
borough:"Brooklyn"
cuisine:"American"
grades:Array
```

52

De volta ao COMPASS

- (SC4) Intervalo de valores

- Carregando a coleção movieDetails usando comando load()

53

Document	Runtime (min)	Title	Director
1	175	Once Upon a Time in the West	Sergio Leone
2	104	A Million Ways to Die in the West	Seth MacFarlane

54

MongoDB Compass - localhost:27017/video.movieDetails

localhost:27017

HOST Standalone EDITION MongoDB 4.4.2 Community

Filter your data

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER: {"runtime": {"\$gte": 100, "\$lt": 120}}

ADD DATA

Displaying documents 1 - 20 of 412

```
_id: ObjectId("5fd3d908ad174a840afcc4a03")
title: "A Million Ways to Die in the West"
year: 2014
rated: "R"
runtime: 105
countries: []
genres: Array
  0: "Action"
  1: "Comedy"
  2: "Romantic"
  3: "Science Fiction"
  4: "Thriller"
director: "Seth MacFarlane"
writers: Array
  0: "Seth MacFarlane"
  1: "Larry Charles"
  2: "Mike Judge"
  3: "Pete Docter"
  4: "Pete Gelpke"
  5: "Pete Molaro"
  6: "Pete Shinkman"
  7: "Tina Packer"
actors: Array
  0: "Anne Heche"
  1: "Chris Pratt"
  2: "Dakota Johnson"
  3: "Eiza González"
  4: "Fergie"
  5: "Hannibal Buress"
  6: "Jesse Plemons"
  7: "Lena Waithe"
  8: "Liam Neeson"
  9: "Lily Collins"
  10: "Luis Guzman"
  11: "Matt Damon"
  12: "Matthew McConaughey"
  13: "Michael Caine"
  14: "Sam Rockwell"
  15: "Tina Fey"
  16: "Will Arnett"
  17: "Zach Galifianakis"
plot: "A once-courteous farmer begins to fall for the mysterious new woman in town."
poster: "http://ia.media-imdb.com/images/MV5BMjQyMjg0MjIwNzAxMzIwMzIwXkZcZw... "
imdb: Object
tomato: Object
metacritic: 44
awards: Object
type: "movie"

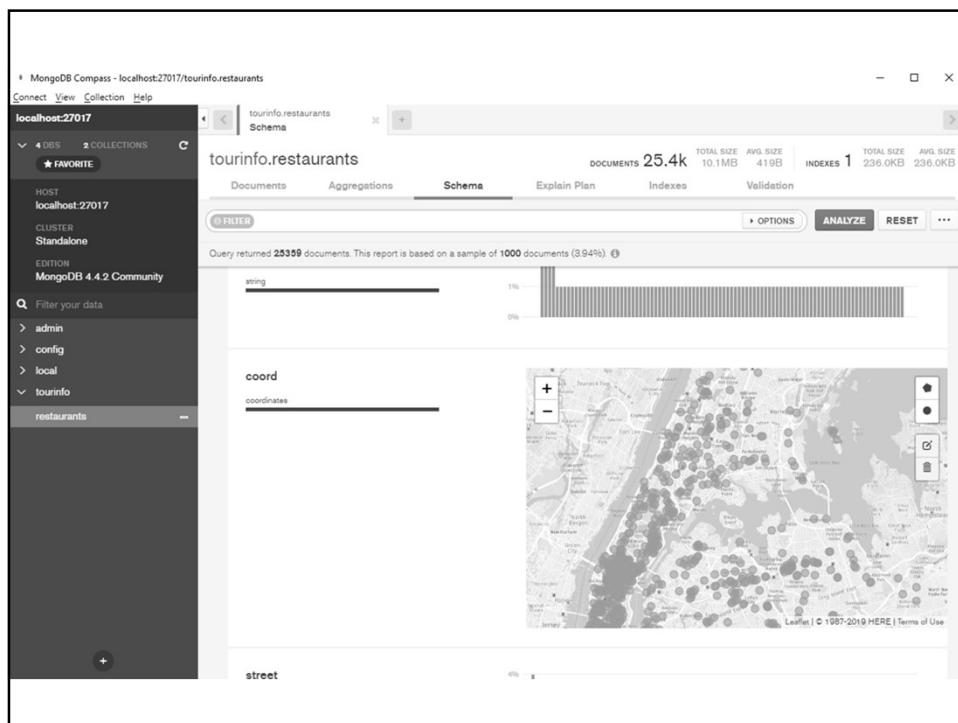
_id: ObjectId("5fd3d908ad174a840afcc4a04")
title: "Wild Wild West"
year: 1999
rated: "PG-13"
runtime: 106
countries: Array
  0: "United States"
director: "Barry Sonnenfeld"
writers: Array
  0: "Barry Sonnenfeld"
  1: "Jeffrey Klarik"
actors: Array
  0: "Keanu Reeves"
  1: "Mel Gibson"
  2: "Steve Buscemi"
  3: "Tommy Lee Jones"
  4: "Wendell Pierce"
  5: "William Fichtner"
  6: "Yaphet Kotto"
  7: "Zack Ward"
plot: "The two best hired guns in the West must save President Grant from the... "
poster: "http://ia.media-imdb.com/images/MV5BNjQzMjg0MjIwNzAxMzIwMzIwXkZcZw... "
imdb: Object
tomato: Object
metacritic: 38
```

55

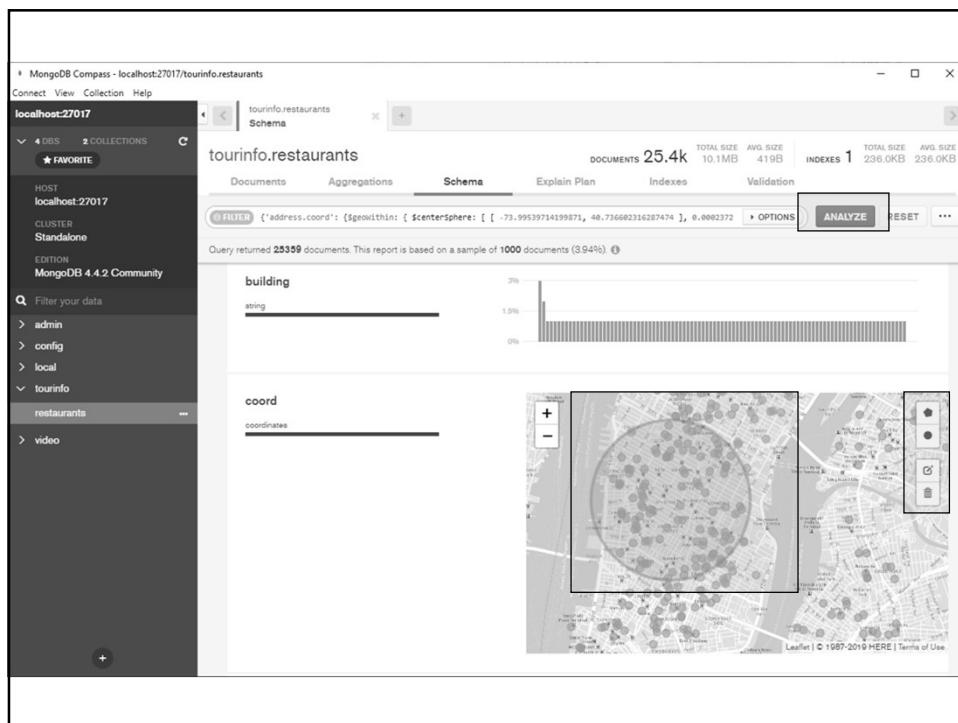
COMPASS (SC5)

- Consultas geoespaciais

56

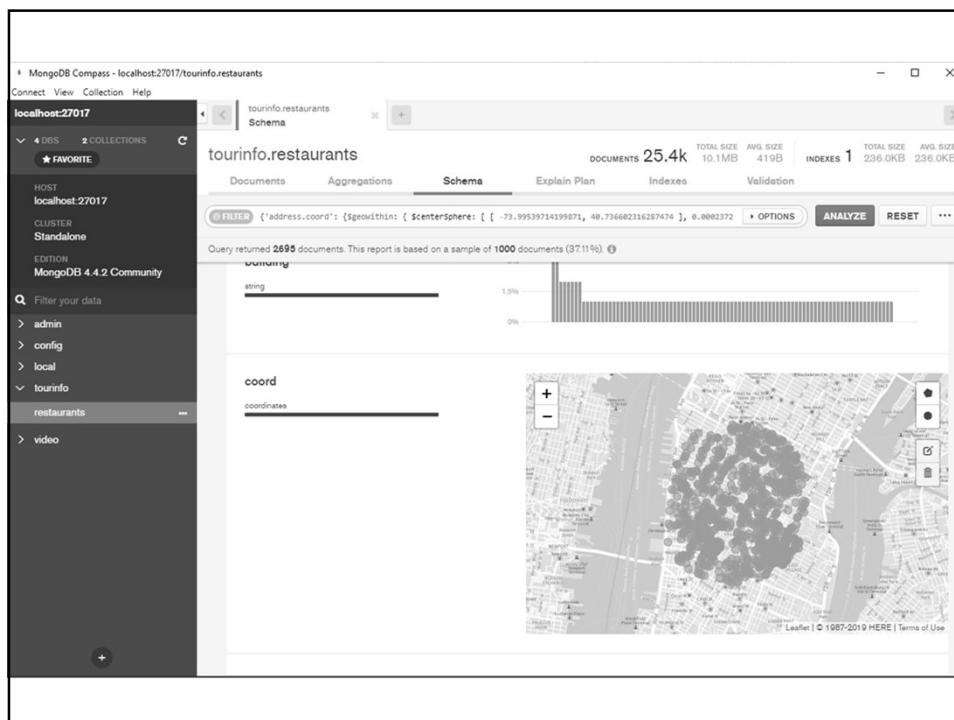


57

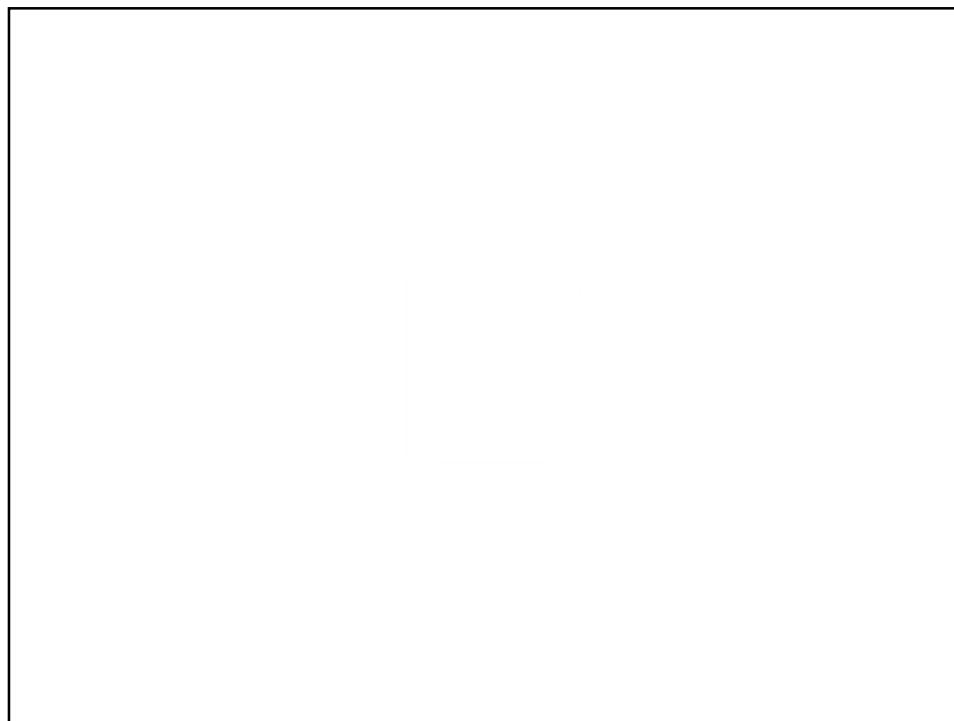


58

29



59



60

Estrutura da Aula

- Introdução a NoSQL
- Introdução a MongoDB
- Linguagem de consulta usando MongoDB Shell
- Framework de Agregação
- MongoDB Java Driver
- Mongo Charts
- Performance
- Exercícios

61

Linguagem de Consulta

CRUD

Operadores de Comparação

Operadores de Tipos

Operadores de Expressões Regulares

Operadores de Arrays

<https://www.mongodb.com/docs/v5.0/reference/operator/>

62

Linguagem de Consulta MongoDB

- Permite fazer operações CRUD
 - Create
 - Read
 - Update
 - Delete

63

Criação de Documentos (SC6)

- db.collection.find()
- db.collection.find().pretty()
- db.collection.insertOne({...})
- db.collection.insertMany(
 - [{...}, ..., {...}])
- db.collection.drop()

64

Criação de Documentos

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

65

Criação de Documentos

- crud/create/insertOne.js
 - crud/create/insertMany-ordered.js
- db.moviesScratch.drop()**
- crud/create/insertMany-ordered-imdb-ids-with-duplicate.js
 - Segundo documento é repetido
 - Apenas uma entrada é inserida

66

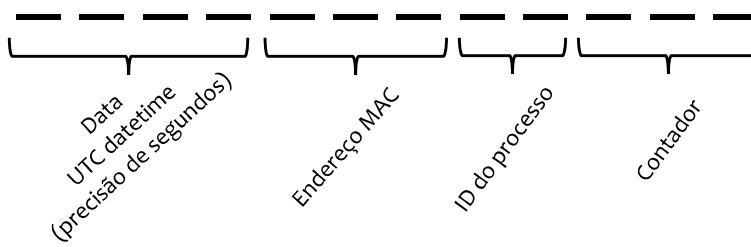
Criação de Documentos

```
db.moviesScratch.drop()  
• crud/create/insertMany-ordered-imdb-ids.js  
  
db.moviesScratch.drop()  
• crud/create/insertMany-unordered-with-  
  duplicate.js
```

67

ObjectId

String hexadecimal de 12 bytes



68

Leitura de Documentos

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



69

Leitura de Documentos

• Comparações de Igualdade

- Valores escalares

- crud/reading_marcel.js - PART I

- Documentos embutidos

- Arrays

- Baseados em todo o array

- Baseados em qualquer elemento

- Baseados em um elemento específico

- Operadores mais complexos

- crud/reading_marcel.js - PART II

70

Cursores e Projeção

- O método `find()` retorna um cursor com 20 elementos
 - Para iterar, use `it`
 - É possível o uso de variáveis e funções javascript
 - `crud/reading_marcel.js` - PART III e IV
- Projeção
 - Exibição de apenas alguns campos do documento
 - `crud/reading_marcel.js` - PART V

71

Operadores de Comparaçāo

- `$eq`
- `$ne`
- `$gt`
- `$gte`
- `$lt`
- `$lte`
- `$in`
- `$nin`
- `crud/comparisonOperators.js`

72

Operadores de Tipo

- Os modelos de dados em MongoDB são flexíveis
 - Documentos podem ou não ter um determinado campo
 - O mesmo campo pode ter tipos diferentes
- \$exists
- \$type
- crud/elementOperators.js

73

Operadores de Expressões Regulares

- \$regex
- Expressões regulares Perl (PCRE) versão 8.39 com suporte UTF-8
- crud/regexOperator.js

74

Operadores de Array

- \$all
- \$elemMatch
- \$size
- crud/arrayOperators.js

75

Atualizando documentos

- db.collection.updateOne()
- db.collection.updateMany()
- db.collection.replaceOne()

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

← collection
← update filter
← update action

76

Atualizando documentos

- `db.collection.updateOne(filterDocument, updateOperator)`
- `$set:document`
 - Usa um documento para atualizar o documento que satisfaz o documento filtro. Se o documento original tem o mesmo campo, ele altera o valor deste campo
- `$unset`
 - Remove o campo do documento que satisfaz o critério de busca
- `db.collection.updateMany(filterDocument, updateOperator)`

77

Atualizando documentos

- Operadores para arrays
- UPSERTS
 - Se nenhum documento satisfazendo o filtro for encontrado, INSERIR um novo documento
- `crud/updatingDocument.js`

78

Removendo documentos

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



The diagram shows a code snippet for MongoDB's `deleteMany` method. Two arrows point from the words 'collection' and 'delete filter' to the corresponding parts of the code: 'users' and the filter condition '{ status: "reject" }'.

79

80

Estrutura da Aula

- Introdução a NoSQL
- Introdução a MongoDB
- Linguagem de consulta usando MongoDB Shell
- Framework de Agregação
- MongoDB Java Driver
- Mongo Charts
- Performance
- Exercícios

81

Framework de Agregação

Pipeline
Estágios de Pipeline
Expressões de Agrupamento
Comparação com SQL

82

Framework de Agregação

- Utilizado para consultas como o GROUP BY de SQL

NAME	CATEGORY	MANUFACTURER	PRICE
Ipad	Tablet	Apple	499
Nexus 1	Cellphone	Samsung	350
Iphone	Cellphone	Apple	699
J7	Cellphone	Samsung	499
Galaxy Tab	Tablet	Samsung	799

```
SELECT manufacturer, count(*) FROM products GROUP BY manufacturer;
```

MANUFACTURER	COUNT(*)
Apple	2
Samsung	3

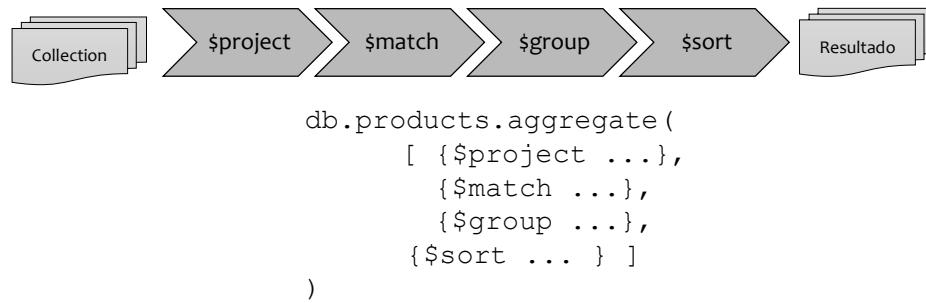
83

Framework de Agregação

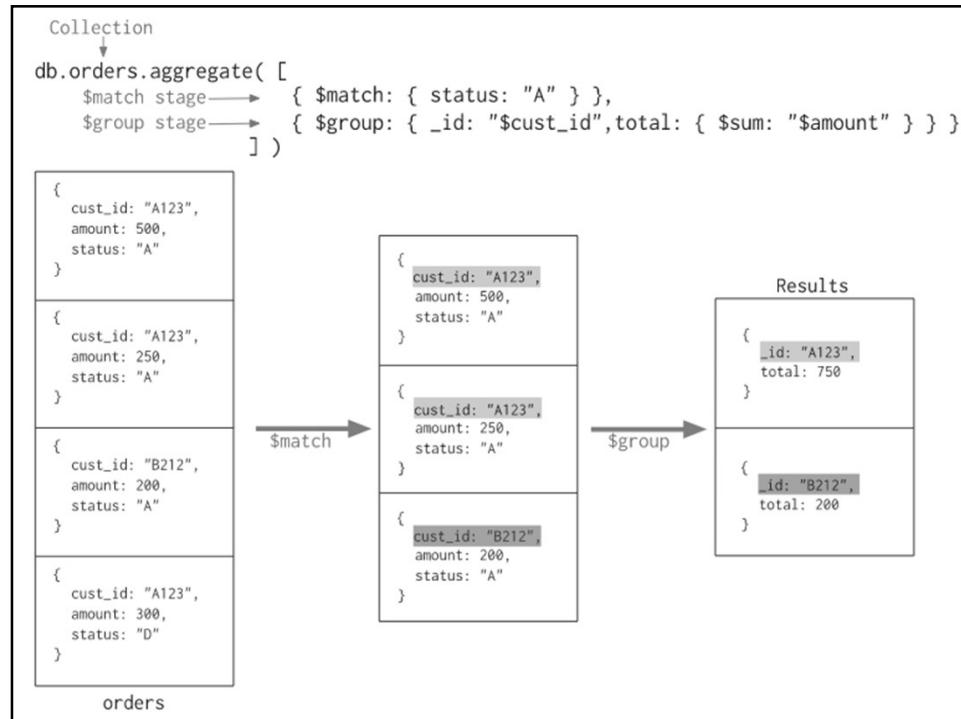
- Como fazer este tipo de consulta em MongoDB?
- Vamos ver um exemplo simples (SC7)

84

O Pipeline de Agregação



85



86

Estágios do Pipeline

- `$project`
 - Restrutura o documento selecionando alguns campos do mesmo
 - 1 para 1: todo documento que entra sai
- `$match`
 - Passo de filtro
 - Potencialmente reduz a quantidade de documentos
- `$group`
 - Permite fazermos efetivamente a agregação
 - Dentro dele, podemos usar `$sum`, `$count`, e outros
 - Potencialmente reduz a quantidade de documentos

87

Estágios do Pipeline

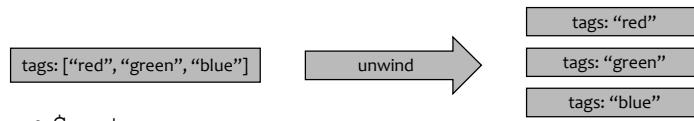
- `$sort`
 - Ordena os documentos
 - 1 para 1: todo documento que entra sai
- `$skip`
 - Ignora um certo número de documentos
 - Potencialmente reduz a quantidade de documentos
- `$limit`
 - Limita a quantidade de documentos que sai deste estágio
 - Potencialmente reduz a quantidade de documentos

88

Estágios do Pipeline

- `$unwind`

- Normaliza os dados de arrays
- Potencialmente aumenta a quantidade de documentos
- Exemplo



- `$out`

- Pega a saída de um estágio e o coloca em uma coleção
- 1 para 1: todo documento que entra sai

89

Exemplo Simples Expandido

```

db.products.aggregate([
  {$group: {
    _id: "$manufacturer",
    num_products: {$sum: 1} } }])
  
```

- Para todo documento, UPSERT da seguinte maneira

- Existe algum documento com mesmo `_id`?
 - Não: Crie um novo documento no resultado com `num_products = 1`
 - Sim: Atualize o seu `num_products` incrementando em 1
- `_id` pode ser um documento, apenas precisa ser único

90

Agrupamento Composto

- E se quisermos usar várias chaves de agrupamento?

```
SELECT manufacturer, count(*)  
FROM products  
GROUP BY manufacturer, category;
```

- Podemos usar várias chaves na agregação
 - aggregation/compound1.js
 - aggregation/simple_example1.js

91

Expressões de Agrupamento

- Podem ser usados na etapa de agrupamento (\$group)
 - \$sum
 - Adiciona o valor dado para cada documento do grupo
 - use 1 para contar
 - \$avg
 - Retorna a média do grupo
 - \$min
 - Retorna o valor mínimo do grupo
 - \$max
 - Retorna o valor máximo do grupo

92

Expressões de Agrupamento

- Podem ser usados na etapa de agrupamento (`$group`)
 - `$push`
 - Coloca o valor em um array (com repetições)
 - `$addToSet`
 - Coloca o valor em um array (sem repetições)
 - `$first`
 - Retorna o primeiro valor no grupo (geralmente usado após o `$sort`)
 - `$last`
 - Retorna o último valor no grupo (geralmente usado após o `$sort`)

93

Usando `$sum`

- Criando a base de `zipcodes` (SC8)
- Calculando a população de cada estado
 - `aggregation/sum_by_state.js`

94

Usando `$avg`

- Calculando a população média por `zipcode` de cada estado
 - `aggregation/avg_by_state.js`

95

Usando `$addToSet`

- Sem correspondência direta em SQL
- Retornando a lista de zipcodes de cada cidade
 - `aggregation/addToSet_by_city.js`

96

Usando \$push

- Similar a \$addToSet, mas não garante unicidade dos elementos inseridos no array
- Resultado similar mas com repetição

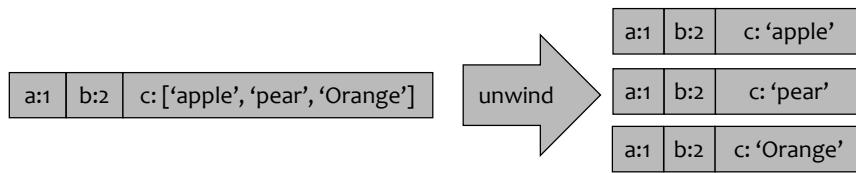
97

Usando \$max e \$min

- Retornando a população do zipcode de cada estado com a maior população
 - aggregation/using_max.js

98

Usando \$unwind



- Risco de explosão de dados
- aggregation/unwind.js

99

Repetindo Estágios de Agregação

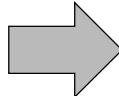
- Podemos repetir estágios de agregação mais de uma vez no mesmo pipeline
- Criando uma pequena base de notas (SC9)
- Como calcular a nota média de cada classe?
 - Precisamos calcular a média de cada estudante na classe
 - Depois calculamos a média dessas médias individuais (por classe)
 - aggregation/double_group.js

100

Usando \$project

- Remove, Adiciona, Renomeia chaves
- Alterando a coleção de zipcodes

```
{
  "city" : "ACMAR",
  "loc" : [
    -86.51557,
    33.584132
  ],
  "pop" : 6055,
  "state" : "AL",
  "_id" : "35004"
}
```



```
{
  "city" : "acmar",
  "pop" : 6055,
  "state" : "AL",
  "zip" : "35004"
}
```

- aggregation/using_project.js

101

Usando \$match

- Filtra os documentos
- Como agregar apenas documentos do estado de NY
 - aggregation/match.js
- Agora, como agrupar por cidades de NY?
 - aggregation/match_and_group.js
- Finalmente, como projetar campos deste resultado?
 - aggregation/match_group_and_project.js

102

Usando \$sort

- Ordenação
- Pode ser feita em qualquer momento do pipeline
- Ordenar a coleção zipcode crescentemente por estado e cidade
 - aggregation/sort_intro.js
- Retornar o nome e a poluição das cidades de NY ordenadas decrescentemente
 - aggregation/sort.js

103

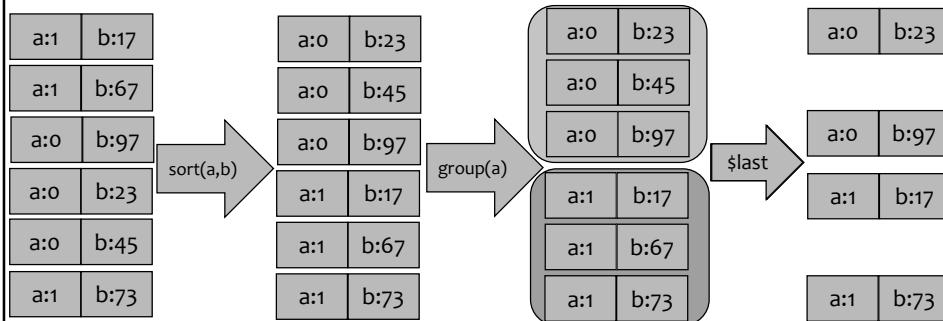
Usando \$skip e \$limit

- Geralmente usado com o \$sort
- Geralmente usado na ordem \$group-\$limit
 - \$skip(n) : descartar os n primeiros elementos da resposta
 - \$limit(n) : retornar apenas os n primeiros elementos da resposta
- Descarte os 10 primeiros elementos e retorne os próximos 5 elementos da consulta do slide anterior
 - aggregation/limit.js

104

Revisitando \$first e \$last

- Geralmente usado com depois de um \$sort



105

Revisitando \$first e \$last

- Como encontrar a cidade com maior população de cada estado?

 1. Calcule a população de cada cidade de cada estado
 - aggregation/first_phase1.js
 2. Ordene por estado e população (decrescente)
 - aggregation/first_phase2.js
 3. Agrupe por estado e pegue o primeiro item de cada grupo
 - aggregation/first_phase3.js
 4. [opcional] Ordene por estado novamente
 - aggregation/first.js

106

Exemplo com \$unwind e mais operadores (SC10)

- Exibindo os tags de um blog com mais likes (apenas os 10 primeiros)
 - aggregation/blog_tags.js

107

\$unwind Duplo

- aggregation/double_unwind.js
- Curiosidade: revertendo unwind
 - aggregation/reversing_double_unwind.js
 - aggregation/reversing_double_unwind2.js

108

Limitações do Framework de Agregação

- Limite de 100MB para estágios do pipeline
 - Default
 - Possibilidade de usar a opção `allowDiskUse`
- O retorno em um documento tem um limite de 16MB
 - Você pode trabalhar com um cursor e vários documentos

109

Limitações do Framework de Agregação

- Sistemas fragmentados (*sharded*)
 - `$project` e `$match` podem acontecer em paralelo
 - Se a agregação usar `$sort` ou `$group` e consulta todos os dados, os resultados podem ser trazidos de volta para o primeiro fragmento (*shard*)
 - Baixa performance

110

Comparações com SQL

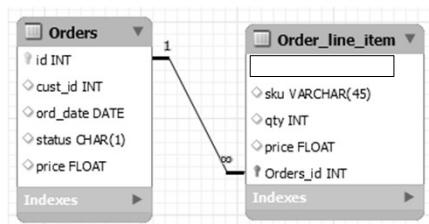
SQL Terms, Functions, and Concepts MongoDB Aggregation Operators

WHERE	\$match
GROUP BY	\$group
HAVING	\$match
SELECT	\$project
ORDER BY	\$sort
LIMIT	\$limit
SUM()	\$sum
COUNT()	\$sum
join	\$lookup

111

Comparações com SQL

Relacional



MongoDB

```
{
  cust_id: "abc123",
  ord_date: ISODate("2012-11-02T17:04:11.102Z"),
  status: 'A',
  price: 50,
  items: [ { sku: "xxx", qty: 25, price: 1 },
            { sku: "yyy", qty: 25, price: 1 } ]
}
```

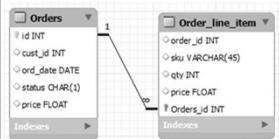
<https://docs.mongodb.com/manual/reference/sql-aggregation-comparison/>

112

Comparações com SQL

Relacional

```
SELECT COUNT(*) AS count
FROM orders
```



```
db.orders.aggregate( [
  {
    $group: {
      _id: null,
      count: { $sum: 1 }
    }
  }
])
```

MongoDB

Count all
records from
orders

```
{
  cust_id: "abc123",
  ord_date: ISODate("2012-11-02T17:04:11.102Z"),
  status: 'A',
  price: 50,
  items: [ { sku: "xxx", qty: 25, price: 1 },
            { sku: "yyy", qty: 25, price: 1 } ]
}
```

<https://docs.mongodb.com/manual/reference/sql-aggregation-comparison/>

113

Comparações com SQL

Relacional

```
SELECT SUM(price) AS total
FROM orders
```



```
db.orders.aggregate( [
  {
    $group: {
      _id: null,
      total: { $sum: "$price" }
    }
  }
])
```

MongoDB

Sum the
price field
from orders

```
{
  cust_id: "abc123",
  ord_date: ISODate("2012-11-02T17:04:11.102Z"),
  status: 'A',
  price: 50,
  items: [ { sku: "xxx", qty: 25, price: 1 },
            { sku: "yyy", qty: 25, price: 1 } ]
}
```

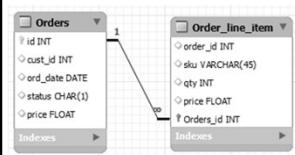
<https://docs.mongodb.com/manual/reference/sql-aggregation-comparison/>

114

Comparações com SQL

Relacional

```
SELECT cust_id,
       SUM(price) AS total
  FROM orders
 GROUP BY cust_id
```



MongoDB

```
db.orders.aggregate( [
  {
    $group: {
      _id: "$cust_id",
      total: { $sum: "$price" }
    }
  }
] )
```

```
{
  cust_id: "abc123",
  ord_date: ISODate("2012-11-02T17:04:11.102Z"),
  status: 'A',
  price: 50,
  items: [ { sku: "xxx", qty: 25, price: 1 },
            { sku: "yyy", qty: 25, price: 1 } ]
}
```

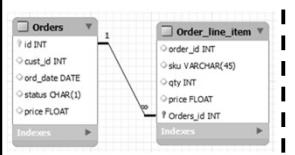
<https://docs.mongodb.com/manual/reference/sql-aggregation-comparison/>

115

Comparações com SQL

Relacional

```
SELECT cust_id,
       SUM(price) AS total
  FROM orders
 GROUP BY cust_id
 ORDER BY total
```



MongoDB

```
db.orders.aggregate( [
  {
    $group: {
      _id: "$cust_id",
      total: { $sum: "$price" }
    }
  },
  { $sort: { total: 1 } }
] )
```

```
{
  cust_id: "abc123",
  ord_date: ISODate("2012-11-02T17:04:11.102Z"),
  status: 'A',
  price: 50,
  items: [ { sku: "xxx", qty: 25, price: 1 },
            { sku: "yyy", qty: 25, price: 1 } ]
}
```

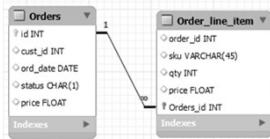
<https://docs.mongodb.com/manual/reference/sql-aggregation-comparison/>

116

Comparações com SQL

Relacional

```
SELECT cust_id,
       ord_date,
       SUM(price) AS total
  FROM orders
 GROUP BY cust_id,
          ord_date
```



MongoDB

```
db.orders.aggregate( [
  {
    $group: {
      _id: {
        cust_id: "$cust_id",
        ord_date: {
          month: { $month: "$ord_date" },
          day: { $dayOfMonth: "$ord_date" },
          year: { $year: "$ord_date" }
        }
      },
      total: { $sum: "$price" }
    }
  }
])
```

For each unique **cust_id**, **ord_date** grouping, sum the **price** field. Excludes the time portion of the date.

```
{
  cust_id: "abc123",
  ord_date: ISODate("2012-11-02T17:04:11.102Z"),
  status: 'A',
  price: 50,
  items: [ { sku: "xxx", qty: 25, price: 1 },
            { sku: "yyy", qty: 25, price: 1 } ]
}
```

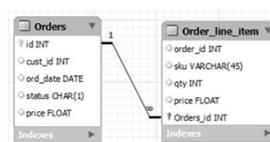
<https://docs.mongodb.com/manual/reference/sql-aggregation-comparison/>

117

Comparações com SQL

Relacional

```
SELECT cust_id,
       count(*)
  FROM orders
 GROUP BY cust_id
 HAVING count(*) > 1
```



MongoDB

```
db.orders.aggregate( [
  {
    $group: {
      _id: "$cust_id",
      count: { $sum: 1 }
    }
  },
  { $match: { count: { $gt: 1 } } }
])
```

For **cust_id** with multiple records, return the **cust_id** and the corresponding record count.

```
{
  cust_id: "abc123",
  ord_date: ISODate("2012-11-02T17:04:11.102Z"),
  status: 'A',
  price: 50,
  items: [ { sku: "xxx", qty: 25, price: 1 },
            { sku: "yyy", qty: 25, price: 1 } ]
}
```

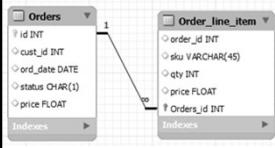
<https://docs.mongodb.com/manual/reference/sql-aggregation-comparison/>

118

Comparações com SQL

Relacional

```
SELECT cust_id,
       ord_date,
       SUM(price) AS total
  FROM orders
 GROUP BY cust_id,
          ord_date
 HAVING total > 250
```



MongoDB

```
db.orders.aggregate([
  {
    $group: {
      _id: {
        cust_id: "$cust_id",
        ord_date: {
          month: { $month: "$ord_date" },
          day: { $dayOfMonth: "$ord_date" },
          year: { $year: "$ord_date" }
        }
      },
      total: { $sum: "$price" }
    }
  },
  { $match: { total: { $gt: 250 } } }
])
```

For each unique **cust_id**,
grouping, sum the **price** where the sum is greater than 250. Excludes the time portion of the date.

```
{
  cust_id: "abc123",
  ord_date: ISODate("2012-11-02T17:04:11.102Z"),
  status: 'A',
  price: 50,
  items: [ { sku: "xxx", qty: 25, price: 1 },
            { sku: "yyy", qty: 25, price: 1 } ]
}
```

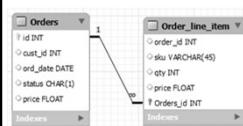
<https://docs.mongodb.com/manual/reference/sql-aggregation-comparison/>

119

Comparações com SQL

Relacional

```
SELECT cust_id,
       SUM(price) as total
  FROM orders
 WHERE status = 'A'
 GROUP BY cust_id
```



MongoDB

```
db.orders.aggregate(
  { $match: { status: 'A' } },
  {
    $group: {
      _id: "$cust_id",
      total: { $sum: "$price" }
    }
  }
)
```

For each unique **cust_id** with status A, sum the **price** field.

```
{
  cust_id: "abc123",
  ord_date: ISODate("2012-11-02T17:04:11.102Z"),
  status: 'A',
  price: 50,
  items: [ { sku: "xxx", qty: 25, price: 1 },
            { sku: "yyy", qty: 25, price: 1 } ]
}
```

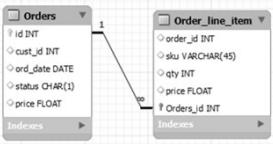
<https://docs.mongodb.com/manual/reference/sql-aggregation-comparison/>

120

Comparações com SQL

Relacional

```
SELECT cust_id,
       SUM(price) as total
  FROM orders
 WHERE status = 'A'
 GROUP BY cust_id
 HAVING total > 250
```



MongoDB

```
db.orders.aggregate([
  { $match: { status: 'A' } },
  {
    $group: {
      _id: "$cust_id",
      total: { $sum: "$price" }
    }
  },
  { $match: { total: { $gt: 250 } } }
])
```

For each unique **cust_id** with status A, sum the **price** field and return only where the sum is greater than 250.

```
{
  cust_id: "abc123",
  ord_date: ISODate("2012-11-02T17:04:11.102Z"),
  status: 'A',
  price: 50,
  items: [ { sku: "xxx", qty: 25, price: 1 },
            { sku: "yyy", qty: 25, price: 1 } ]
}
```

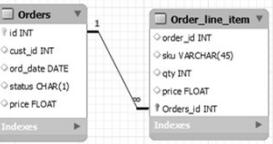
<https://docs.mongodb.com/manual/reference/sql-aggregation-comparison/>

121

Comparações com SQL

Relacional

```
SELECT cust_id,
       SUM(li.qty) as qty
  FROM orders o,
       order_lineitem li
 WHERE li.order_id = o.id
 GROUP BY cust_id
```



MongoDB

```
db.orders.aggregate([
  { $unwind: "$items" },
  {
    $group: {
      _id: "$cust_id",
      qty: { $sum: "$items.qty" }
    }
  }
])
```

For each unique **cust_id**, sum the corresponding line item **qty** fields associated with the orders.

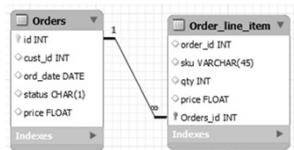
```
{
  cust_id: "abc123",
  ord_date: ISODate("2012-11-02T17:04:11.102Z"),
  status: 'A',
  price: 50,
  items: [ { sku: "xxx", qty: 25, price: 1 },
            { sku: "yyy", qty: 25, price: 1 } ]
}
```

<https://docs.mongodb.com/manual/reference/sql-aggregation-comparison/>

122

Comparações com SQL Relacional

```
SELECT COUNT(*)
FROM (SELECT cust_id,
            ord_date
     FROM orders
    GROUP BY cust_id,
            ord_date)
      as DerivedTable
```



```

db.orders.aggregate( [
  {
    $group: {
      _id: {
        cust_id: "$cust_id",
        ord_date: {
          month: { $month: "$ord_date" },
          day: { $dayOfMonth: "$ord_date" },
          year: { $year: "$ord_date" }
        }
      }
    }
  },
  {
    $group: {
      _id: null,
      count: { $sum: 1 }
    }
  }
] )

```

Count the number of distinct `cust_id`, `ord_date` groupings. Excludes the time portion of the date.

<https://docs.mongodb.com/manual/reference/sql-aggregation-comparison/>

123

Estrutura da Aula

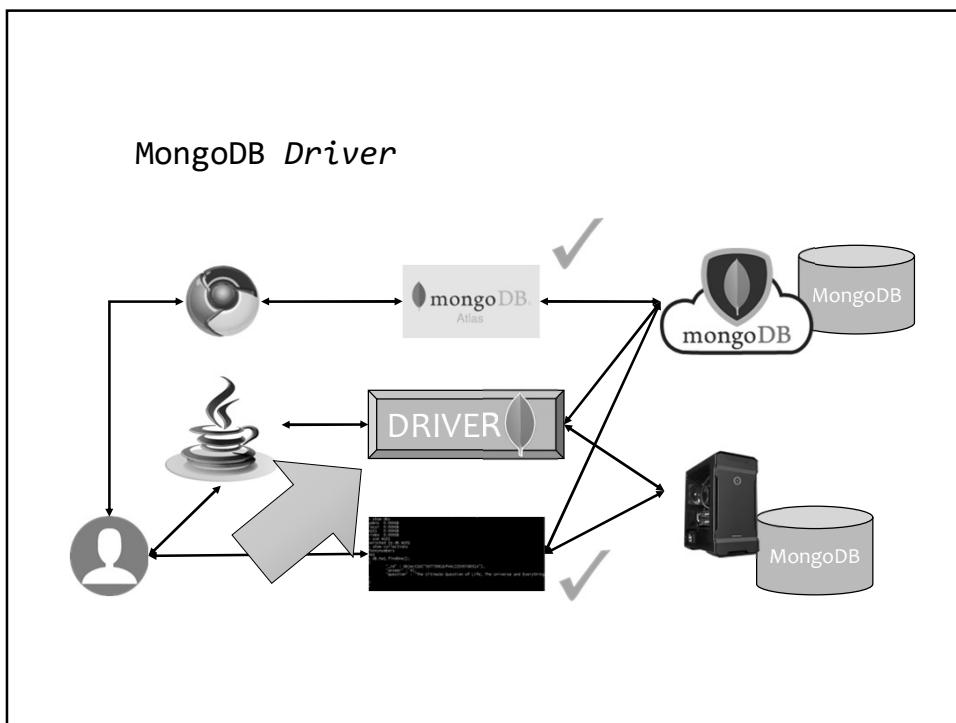
- Introdução a NoSQL
- Introdução a MongoDB
- Linguagem de consulta usando MongoDB Shell
- Framework de Agregação
- MongoDB Java Driver
- Mongo Charts
- Performance
- Exercícios

125

MongoDB Java Driver

CRUD
Framework de Agregação

126



127

MongoDB Java Driver

- Adicione dependências no projeto Maven
- MongoClient é um objeto “pesado” que representa um pool de conexões ao servidor
 - Faça apenas a quantidade necessárias de conexões
 - Use variáveis estáticas ou *singletons*
 - com.mongodb.crud.IntroToMongoDBDriver

128

Documentos

- com.mongodb.crud.DocumentTest

129

CRUD

- Inserção
 - com.mongodb.crud.InsertTest
- Consulta com count
 - com.mongodb.crud.FindTest
- Consulta com filtro
 - com.mongodb.crud.FindWithFilterTest
- Consulta com projeção
 - com.mongodb.crud.FindWithProjectionTest

130

CRUD

- Consulta com sort, skip e limit
 - com.mongodb.crud.FindWithSortSkipLimitTest
- Atualização
 - com.mongodb.crud.UpdateTest
- Remoção
 - com.mongodb.crud.DeleteTest

131

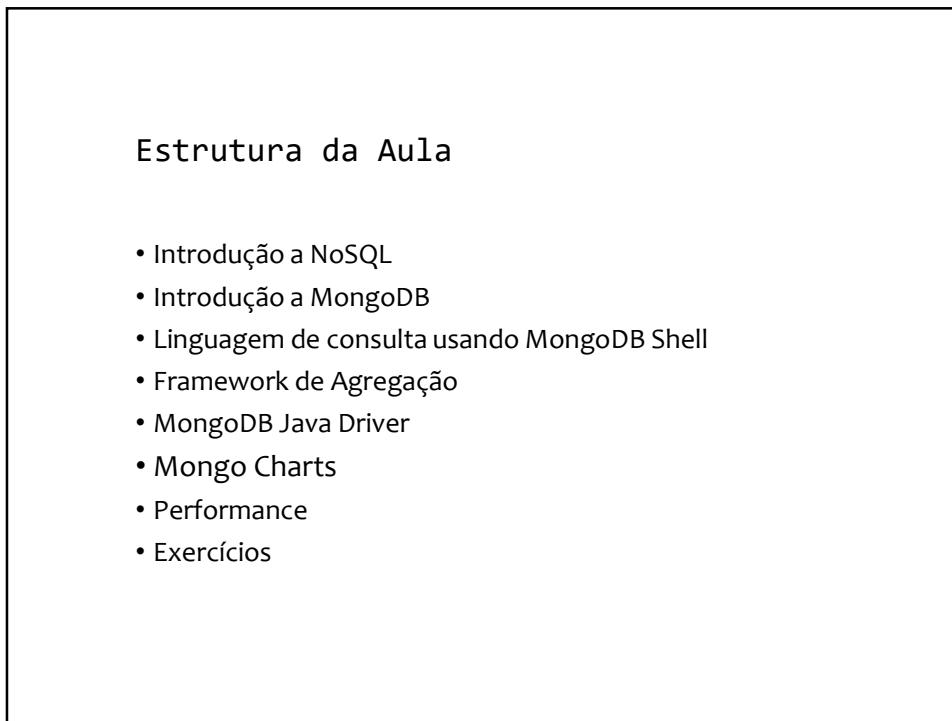
Framework de Agregação

- Usando a base de dados de zipcodes
- Exemplos
 - com.mongodb.aggregation.ZipCodeAggregationTest
 - com.mongodb.aggregation.ZipCodeAggregationWithBuildersTest
 - com.mongodb.aggregation.ZipCodeAggregationUsingParserTest

132



133



134

Estrutura da Aula

- Introdução a NoSQL
- Introdução a MongoDB
- Linguagem de consulta usando MongoDB Shell
- Framework de Agregação
- MongoDB Java Driver
- Mongo Charts
- Performance
- Exercícios

Mongo Charts

- Ferramenta de visualização de dados integrada à plataforma de dados em nuvem MongoDB.
- Rápida, fácil e em tempo real
- Permite a criação de diferentes dashboards de diferentes fontes de dados
- Não necessita extração de dados do BD
 - Gráficos são projetados para a riqueza dos dados formatados em JSON
 - A única configuração necessária é selecionar a quais clusters conectar os gráficos e o “carregamento” dos dados pode ser feito com apenas alguns cliques.

<https://www.mongodb.com/products/charts>

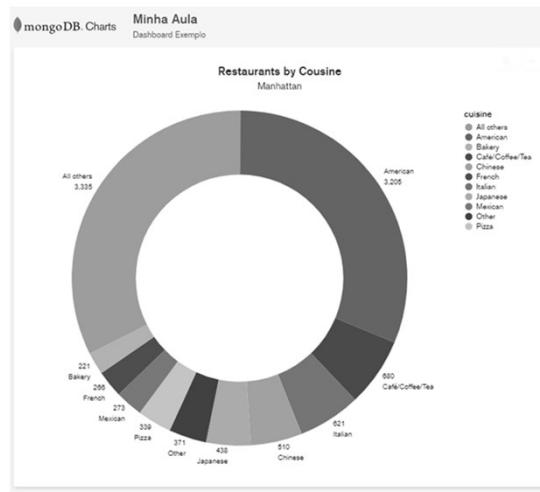
135

Usando o Mongo Charts (SC12)

```
mongoimport --host clustermarcel-shard-00-00-1dm9o.mongodb.net:27017,clustermarcel-shard-00-01-1dm9o.mongodb.net:27017,clustermarcel-shard-00-02-1dm9o.mongodb.net:27017 --authenticationDatabase admin --ssl --username MarceloOliveira --drop --file restaurants.json --db tourinfo --collection restaurants --password [REDACTED]
2021-07-08T14:30:06.362-0300 connected to: mongodb://clustermarcel-shard-00-00-1dm9o.mongodb.net:27017,clustermarcel-shard-00-01-1dm9o.mongodb.net:27017,clustermarcel-shard-00-02-1dm9o.mongodb.net:27017/
2021-07-08T14:30:06.491-0300 dropping: tourinfo.restaurants
2021-07-08T14:30:09.375-0300 [###.....] tourinfo.restaurants 2.87MB/11.3MB (18.3%)
2021-07-08T14:30:12.378-0300 [#####..] tourinfo.restaurants 3.62MB/11.3MB (32.0%)
2021-07-08T14:30:15.363-0300 [######.....] tourinfo.restaurants 5.16MB/11.3MB (45.6%)
2021-07-08T14:30:18.374-0300 [#######.....] tourinfo.restaurants 6.71MB/11.3MB (59.2%)
2021-07-08T14:30:21.373-0300 [########.....] tourinfo.restaurants 8.19MB/11.3MB (72.3%)
2021-07-08T14:30:24.368-0300 [#########.....] tourinfo.restaurants 9.42MB/11.3MB (83.1%)
2021-07-08T14:30:27.375-0300 [##########.....] tourinfo.restaurants 10.4MB/11.3MB (91.9%)
2021-07-08T14:30:30.363-0300 [##########.....] tourinfo.restaurants 11.3MB/11.3MB (99.4%)
2021-07-08T14:30:32.161-0300 [##########.....] tourinfo.restaurants 11.3MB/11.3MB (100.0%)
2021-07-08T14:30:32.161-0300 25359 document(s) imported successfully. 0 document(s) failed to import.
```

136

Usando o Mongo Charts (SC12)



137

Estrutura da Aula

- Introdução a NoSQL
- Introdução a MongoDB
- Linguagem de consulta usando MongoDB Shell
- Framework de Agregação
- MongoDB Java Driver
- Mongo Charts
- Performance
- Exercícios

138

Performance

Mecanismos de Armazenamento (*Storage Engines*)
Índices

139

Mecanismos de Armazenamento

- Interface entre o disco e o servidor do banco de dados
- Utiliza a memória rápida e decide o que manter em memória e o que escrever em disco para obter alta eficiência
- Podemos alterar o mecanismo de armazenamento de acordo com a nossa necessidade

140

Mecanismos de Armazenamento

- Não afetam a comunicação entre clusters
- Não afetam as APIs usadas pelos programadores
- Afetam
 - Formato dos arquivos de dados
 - Formato dos índices

141

Principais Mecanismos de Armazenamento

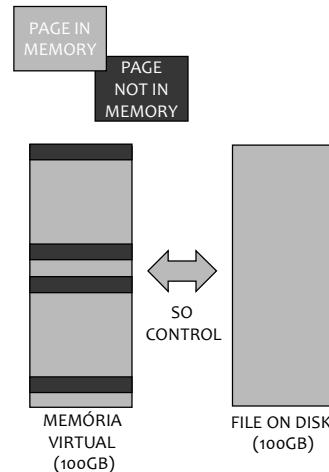
- MMAPv1 (**Deprecated**)
- In-memory Storage Engine (**Enterprise Only**)
- WiredTiger (**Padrão**)

<https://www.vembu.com/blog/mongodb-mmap-vs-wiredtiger-comprehensive-comparison/>

142

MMAPv1

- Usa a chamada de sistema MMAP para armazenar dados
 - Se não estiver na memória, traga do disco



143

MMAPv1

- Em MongoDB, oferece:
 - Bloqueio a nível de coleção
 - Se operações de escrita à mesma coleção são executadas concurrentemente, uma deve esperar a conclusão da outra
 - Operações de escrita à coleções diferentes podem ocorrer concurrentemente

144

MMAPv1

- Em MongoDB, oferece:

- Atualização *in-place*
 - Tenta atualizar o documento no seu local da memória
 - Caso não consiga, move todo o documento para outro lugar na memória
 - Utiliza documentos *power-of-two-sized* para minimizar mudanças de lugar

145

WiredTiger

- Concorrência a nível de documento

- Sem bloqueio

- Visão otimista
- Permite concorrência livre
- Se duas operações escrevem no mesmo documento, uma delas será descartada e terá que ser refeita

146

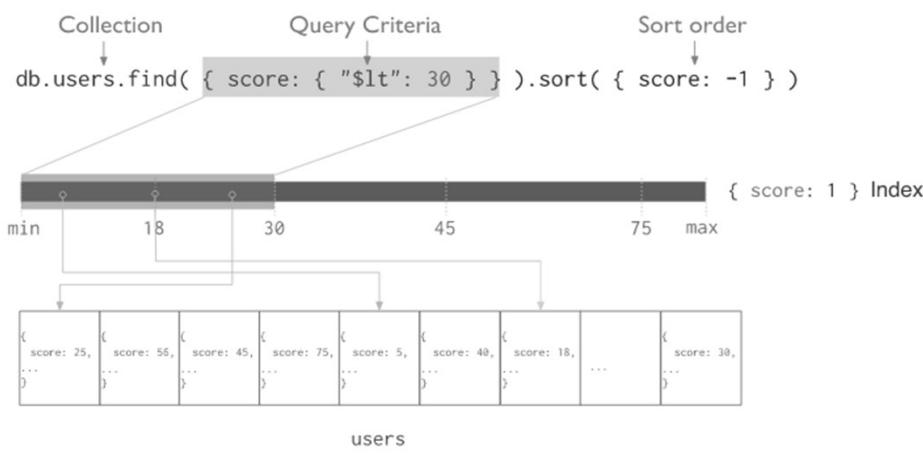
WiredTiger

- Compressão de documentos e índices
 - MongoDB gerencia a compressão ao escrever/recuperar do disco
- Sem atualização *in-place*
 - Atualizações de documentos escrevem os novos documentos em outro lugar em memória e liberam o espaço do documento inicial
 - Causam grandes escritas para copiar todo o documento
 - Permite a concorrência a nível de documento

147

Índices

- Define a ordem dos documentos na coleção



148

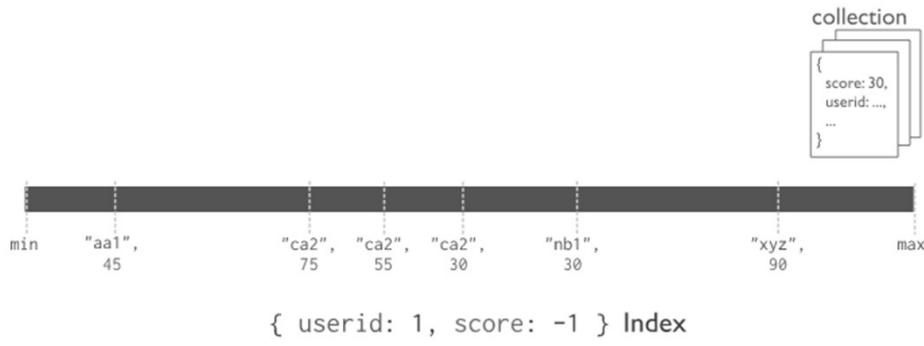
Índices

- MongoDB usa uma *BTree* para armazenar os índices
- Uma busca binária encontra os elementos eficientemente ($\log n$)
- Definimos índices em items que acreditamos que faremos as buscas

149

Índices

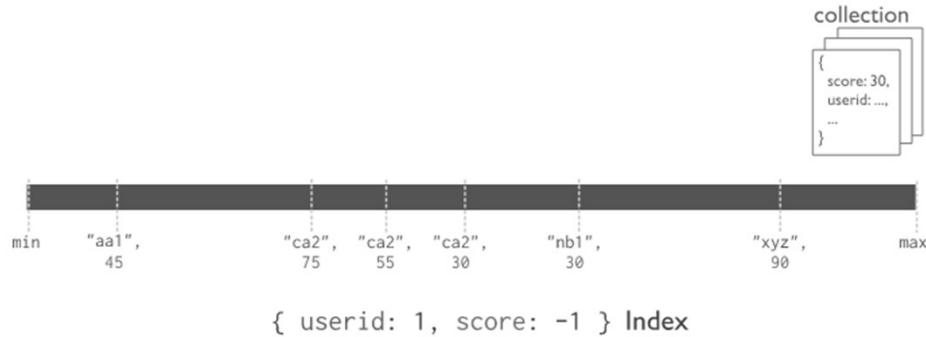
- E se quisermos fazer a busca por `userId` e por `score`?



150

Índices

- Note que neste exemplo, se fizermos a busca por score não utilizamos os índices



151

Índices

- Buscas eficientes devem seguir a ordem dos índices

Índice (a, b, c)

Efficiency →

a ✓
 a, b ✓
 a, b, c ✓
 c ✗
 c, b ✗

152

Custo de um Índice

- “*There is no such thing as a free lunch*”
- Leitura (usando índice) serão mais rápidas, mas...
- Alterações em um documento que afetam o índice causam uma reordenação dos índices na memória
 - Escritas geralmente são mais lentas
 - Se você está apenas escrevendo, você não precisa de índices
 - Quando for inserir muitos dados, remova os índices, insira os dados, e só depois crie os índices
 - Provavelmente não desejamos índices em todas as chaves da coleção

153

Boas Práticas

- Ter um índice por consulta
- Ter uma consulta por índice

154

Exemplo

- Base de dados com 1.000.000 de alunos, cada um, assistindo 10 cursos (SC13)
 - 10.000.000 de documentos
- Consultando sem índices
 - index/test_index.js - Part I

```

    "executionStages" : [
        "stage" : "COLLSCAN",
        "filter" : {
            "student_id" : {
                "$seq" : 5
            }
        },
        "nReturned" : 10,
        "executionTimeMillisEstimate" : 3674,
        "works" : 10000002,
        "advanced" : 10,
        "needTime" : 9999991,
        "needYield" : 0,
        "saveState" : 78186,
        "restoreState" : 78186,
        "isEOF" : 1,
        "invalidates" : 0,
        "direction" : "forward",
        "docsExamined" : 10000000
    ],

```

155

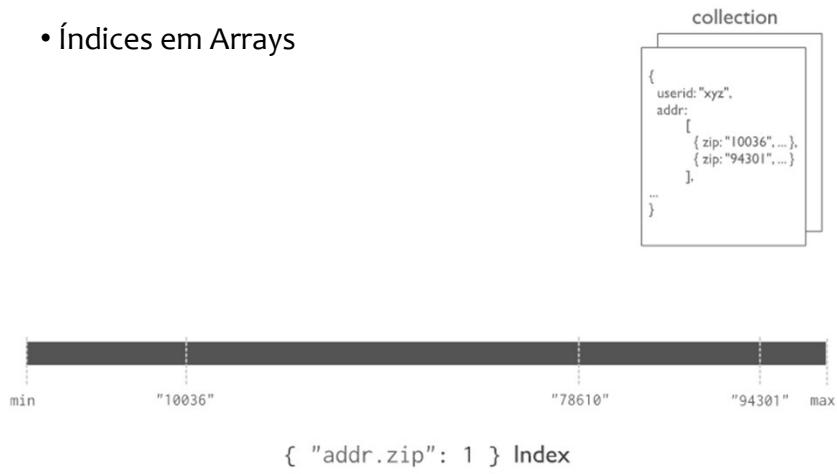
Exemplo

- Adicionando índices (Simples e Compostos)
 - index/test_index.js - Part II
- Descobrindo e removendo índices
 - index/test_index.js - Part III
 - _id: exists in all collections and cannot be deleted.
 - student_id: the one we have just created.

156

Índices Multikey

- Índices em Arrays



157

Índices Multikey

- Restrições

- Não são permitidos mais de um índice multikey em um mesmo documento
 - Índices em dois campos onde ambos sejam arrays
 - Explosão de pontos devido ao produto cartesiano

- `index/test_index.js` - Part IV

158

Índices e *Dot Notation*

- Criando e usando índices usando a notação dot
- index/test_index.js - Part V

159

Índices e *Dot Notation*

- Procurando por documentos com uma nota de exam maior do que 99.8

```
db.students.find(
  { $and: [
    { 'scores.type': 'exam' },
    { 'scores.score': { '$gt': 99.8 } }
  ] }
).count()
```

X

Retorna documentos que tem um score do tipo exam e um score acima de 99.8



160

Índices e *Dot Notation*

- Procurando por documentos com um nota de *exam* maior do que 99.8

```
db.students.find(
  { 'scores': {
    '$elemMatch': {
      'type': 'exam',
      'score': { '$gt': 99.8 } }
    }
  }).pretty()
```

Procure um elemento com tipo e score satisfazendo as restrições

- Faça a busca usando o índice em score
- Filtre pelo tipo

- index/test_index.js - Part VI

161

Opções de Criação de Índices (*unique*)

- Todos os índices criados até agora não foram únicos
- Podemos forçar os índices a serem únicos na coleção
- index/test_index.js - Part VII
- Note que o índice padrão em *_id* não explicitamente dito ser único, mas é!!!

162

Opções de Criação de Índices (sparse)

- Usado quando a chave do índice está faltando em mais de um documento

```
{a:1, b:2, c:5}  
{a:10, b:5, c:10}  
{a:13, b:17}      ➔ {a:13, b:17, c:null}  
{a:7, b:23}       ➔ {a:7, b:23, c:null}
```

```
db.foo.createIndex(  
  {c:1},  
  {unique:true, sparse:true})
```

163

Opções de Criação de Índices (sparse)

- Índices com esta opção não podem ser usados para ordenação se, de fato, existem documentos sem este índice

164

Opções de Criação de Índices (*background*)

- Por padrão, índices são criados no *foreground*
 - Rápido
 - Bloqueia escritores e leitores
 - `db.students.createIndex({'scores.score':1})`
- Podemos usar a opção *background*
 - Lento
 - Sem bloqueio
 - `db.students.createIndex({'scores.score':1}, {background:true})`
- `index/test_index.js` - Part VIII

165

Quando o índice é usado?

- Baseado no formato da consulta
 - Campos, ordenação, ...
- MongoDB encontra índices candidatos
- Cria um plano de consulta para cada índice
- Executa os planos em paralelo
- O plano vencedor tem todos os resultados de maneira mais eficiente

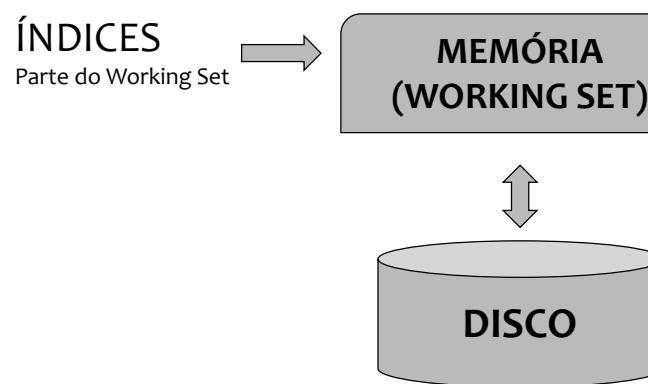
166

Quando o índice é usado?

- O plano vencedor é mantido em cache para ser usado em consultas de mesmo formato
- Ele sai da cache se:
 - O número limite de escritas é atingido
 - O índice foi reconstruído
 - Algum índice é inserido ou removido
 - MongoDB é reiniciado

167

Qual o tamanho do seu índice?



168

Qual o tamanho do seu índice?

- index/test_index.js - Part IX

```
'nindexes' : 4,
"totalIndexSize" : 351768576,
"indexSizes" : {
    "_id_" : 100962304,
    "student_id_1" : 63901696,
    "student_id_1_class_id_-1" : 119562240,
    "class_id_1" : 67342336
},
```

- Visualização também no COMPASS

169

Qual o tamanho do seu índice?

- *WiredTiger* também aplica compressão aos índices
- Em nosso exemplo, chega a cerca de 1/3 do tamanho
- Obviamente, isto custa CPU para usar os índices

170

Quantidade de Entradas de Índices

- **Regular**

- Para cada valor da chave, temos uma entrada
- Se algum documento não tem a chave, existirá uma entrada para *no entry*
- Praticamente uma entrada por documento

- **Sparse**

- Quando o documento não tem a chave indexada, ele não está no índice
- A quantidade de entradas é potencialmente menor que a quantidade de documentos

171

Quantidade de Entradas de Índices

- **Multikey**

- Cada documento pode ter várias entradas
- A quantidade de entradas é potencialmente muito maior que a quantidade de documentos

172

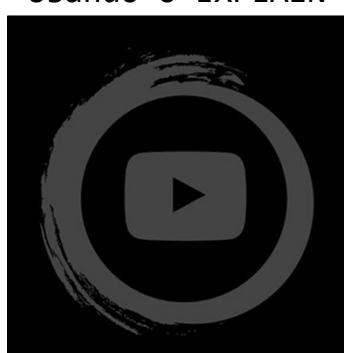
Vídeos Extras



- Usando o EXPLAIN: https://youtu.be/zd_oOrNvKQw
- Índices Geoespaciais e de Texto: <https://youtu.be/TWn-zlObj5g>
- Uso Eficiente de Índices: <https://youtu.be/WEALR8VETw>
- Ferramentas de Monitoramento: <https://youtu.be/haQjMqBKgAw>
- Replicação: <https://youtu.be/y-a4qAfZbsI>
- Aspectos de Escrita: <https://youtu.be/xJo-FEtLWuE>
- Transações: <https://youtu.be/DMU5jgRmE5c>
- Fragmentação: <https://youtu.be/WKA9VIxvcal>

173

Usando o EXPLAIN



- Temos usado este comando para detalhar a execução de nossas consultas
 - Como foi feita
 - Índices usados
 - Quantos documentos foram inspecionados
- Não retorna os dados pois não é uma consulta completa
- Pode ser usado também usando o driver

174

EXPLAIN

- `db.foo.explain().OPERAÇÃO`
 - `find()`
 - `update()`
 - `remove()`
 - `aggregate()`
 - `help()`
 - Não podemos usar o `insert()`

175

EXPLAIN

- `index/test_index.js` - Part X
- Modo padrão
 - `queryplanner`
 - `queryPlanner`
 - `winningPlan`
 - `rejectedPlans`

176

EXPLAIN

- Opções de sintaxe

```
db.students.find(  
    {student_id:0,class_id:485}  
).sort({class_id:-1})  
.explain();
```

Retorna um cursor
Aplicamos explain a este cursor

177

EXPLAIN

- Opções de sintaxe

```
db.students.find(  
    {student_id:0,class_id:485}  
).sort({class_id:-1})  
.count()  
.explain();
```



Não retorna um cursor

178

EXPLAIN

- Outros modos

- executionStats

- Inclui queryPlanner

- Nos mostra os resultados por termos usado o índice escolhido pelo otimizador de consulta

- allPlansExecution

- Inclui executionStats

- Nos mostra os resultados usando todos os índices

- index/test_index.js - Part XI

179

Consultas Cobertas

- Consultas totalmente satisfeitas por um índice

- Exemplo

- index/test_index.js - Part XII

- totalDocsExamined = 10

180

Outros Tipos de Índices

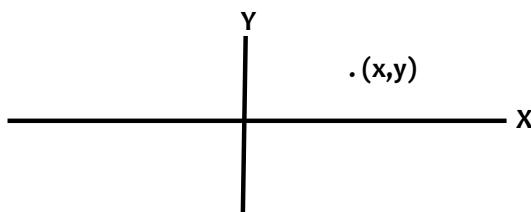


- Índices Geo-Espaciais
 - Ajudam na busca com dados de localização
 - Dois modelos: 2D e 3D

181

Índices Geo-Espaciais (2D)

- Como encontrar o ponto mais próximo ao ponto P?



182

Índices Geo-Espaciais (2D)

- O documento precisa de um campo com as coordenadas
 - {..., location: [x,y], ... }
- Indicar ao MongoDB que o campo é um índice geo-espacial
 - db.collection.ensureIndex({location:'2d'})
- A busca abaixo retorna as localizações mais próximas em ordem

```
db.collection
    .find({location:{$near:[x,y]}})
    .limit(20)
```

183

Índices Geo-Espaciais (3D)

- O documento precisa de um campo com longitude e latitude (nesta ordem)
 - {..., location:[long,lat],... }
- Indicar ao MongoDB que o campo é um índice geo-espacial
 - db.collection.ensureIndex({location:'2dsphere'})

184

Índices Geo-Espaciais (3D)

- A busca abaixo retorna as localizações a uma distância máxima de 2000 metros ordenadas por distância (menor para maior)

```
db.collection.find({
  location: {
    $near: { $geometry: {
      type: "Point",
      coordinates: [mylong, mylat] },
    $maxDistance: 2000 } }
})
```

185

Índices de Texto

- Oferece uma busca eficiente textos grandes
 - {..., mytext = "...", ...}
- Como procurar palavras em mytext?
 - Busca por strings não funciona pois procurará a string toda
 - Dividir o texto em um array de palavras é ineficiente
- Use um índice de busca de texto completo - *full text search index*

186

Índices de Texto

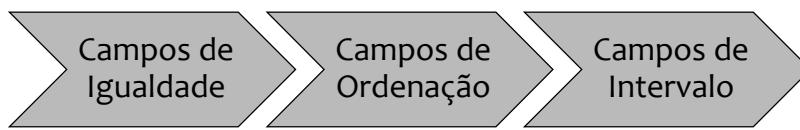
- Exemplo
 - Nomes dos restaurantes
 - Visualize no COMPASS
- index/test_index.js - Part XIII

187

Uso Eficiente de Índices



- index/test_index.js - Part XIV
- Quando criar índices compostos siga a seguinte ordem



188

Ferramentas de Monitoramento



- Registro de consultas Lentas
- mongotop
- mongostat

189

Registrando Consultas Lentas

- MongoDB registra consultas lentas automaticamente
 - index/test_index.js - Part XV

```
2017-10-06T18:41:05.798-0300 I COMMAND [conn15] command school.students appName: "MongoDB" command: find { find: "students", filter: { student_id: 10000.0 } } planSummary: COLLSCAN :0 docsExamined:10000000 cursorExhausted:1 numYields:78173 nreturned:10 reslen:2448 lockTimeMS:0 acquireCount: { r: 156348 } }, Database: { acquireCount: { r: 78174 } }, collection: { t: { r: 78174 } } } protocol:op_command 4444ms
```

190

Perfil de Registro

- MongoDB escreve documentos em `System.profile` para consultas que demorem mais do que um tempo especificado
- 3 níveis
 - 0 – desligado
 - 1 – registre apenas consultas lentas (valor de `slowms`)
 - 2 – registre todas as minhas consultas
- Podemos fazer consultas a este documento
- `index/test_index.js` – Part XVI

191

`mongotop`

- Visão alto nível de onde Mongo está gastando seu tempo
- Exemplo (SC14)
 - `mongotop 2`
 - `db.students.find({student_id:10000})`

	ns	total	read	write
	<code>school.students</code>	17426ms	17426ms	0ms
	<code>admin.system.roles</code>	0ms	0ms	0ms
	<code>admin.system.version</code>	0ms	0ms	0ms
	<code>agg.grades</code>	0ms	0ms	0ms
	<code>agg.inventory</code>	0ms	0ms	0ms
	<code>agg.items</code>	0ms	0ms	0ms
	<code>agg.products</code>	0ms	0ms	0ms
	<code>blog.posts</code>	0ms	0ms	0ms
	<code>blog.sessions</code>	0ms	0ms	0ms
	<code>blog.users</code>	0ms	0ms	0ms

192

mongostat

- Verifica a base de dados a cada segundo e apresenta um conjunto de informações
 - Inserções
 - Consultas
 - Atualizações
 - Remoções

193

mongostat

- Outras informações adicionais são dadas de acordo com o mecanismo de armazenamento usado
- <https://docs.mongodb.com/database-tools/mongostat/>
- Exemplo (SC15)

194

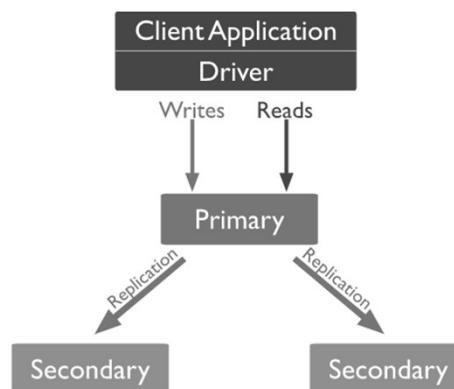
Replicação



195

Replicação

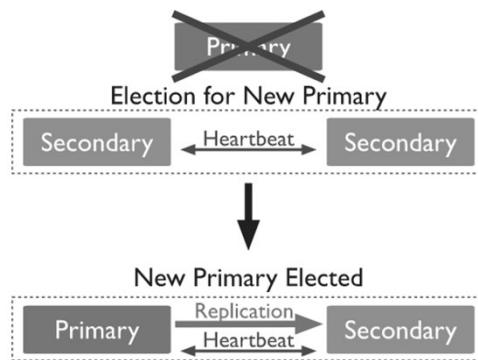
- Usado para oferecer disponibilidade e tolerância a falhas
- Aplicação “conversa” com o nó primário do conjunto de replicação (*replica set*)
- Ações são replicadas transparentemente para os nós secundários



196

Replicação

- Se o nó primário “cai”, os outros nós fazem uma eleição para eleger um novo nó primário
 - Todo nó tem um voto mas isto pode ser alterado
- A aplicação se conecta transparentemente ao novo nó primário
- Quando o antigo nó primário voltar, ele volta como secundário
- A menor quantidade de nós é 3



197

Replicação Tipos de nós na eleição

- Regular
 - um nó que possui dados e pode se tornar um nó primário
- Árbitro
 - usado para fins de eleição
- Atrasado (*Delayed*)
 - usado para recuperação de desastres
 - Está sempre atrasado (1hr, 2hr, ...) com relação ao nó primário
 - É usado na eleição, mas não pode ser primário
- Escondido (*Hidden*)
 - Usado para fazer análises
 - Nunca poderá ser um nó primário

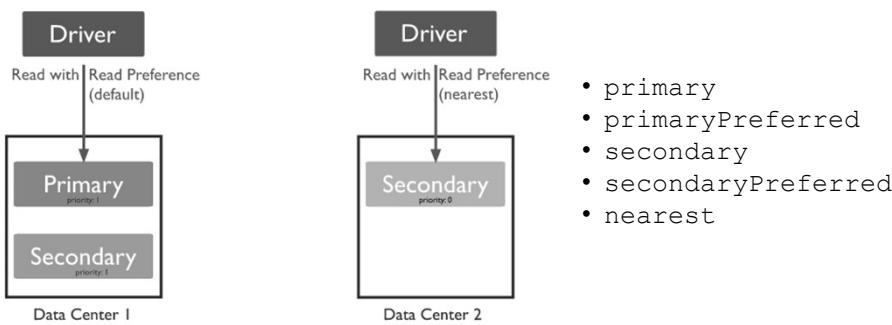
198

Replicação Consistência de Escrita

- Replicação é assíncrona
- Escritas podem ser feitas apenas no nó primário
- Consistência forte
 - A aplicação escreve e lê do nó primário (padrão)
- Consistência eventual
 - Aplicação pode ler de nós secundários
 - Pode ler dados antigos
 - O padrão é proibir esta leitura, mas pode ser alterado

199

Replicação Opções Preferência de Leitura (Read Preference)



200

Replicação com Java Driver

- Criando um conjunto de replicação (SC16)

- com.mongodb.replication.ReplicaSetTest
- com.mongodb.replication.ReplicaSetFailoverTest
- com.mongodb.replication.ReadPreferenceTest

201

Aspectos de Escrita (*Write Concern*)



202

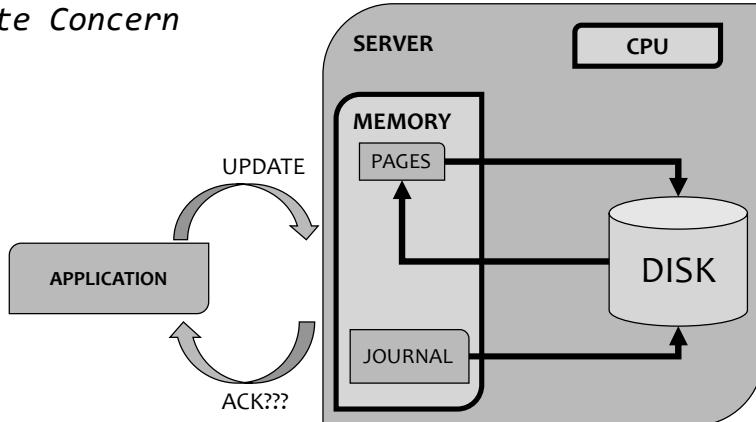
Aspectos de Escrita (*Write Concern*)

- Define o nível de confirmação solicitado pelo MongoDB para operações de gravação em:
 - Uma instância standalone do servidor MongoDB, ou
 - Um conjuntos de réplicas de servidores, ou
 - Clusters fragmentados

<https://docs.mongodb.com/manual/reference/write-concern/>

203

Write Concern



- Aplicação envia atualização para servidor
- Servidor atualiza dados nas páginas e no jornal
- Jornal é escrito imediatamente em disco (*write ahead logging*)
- Que confirmações aguardar???

204

Opções

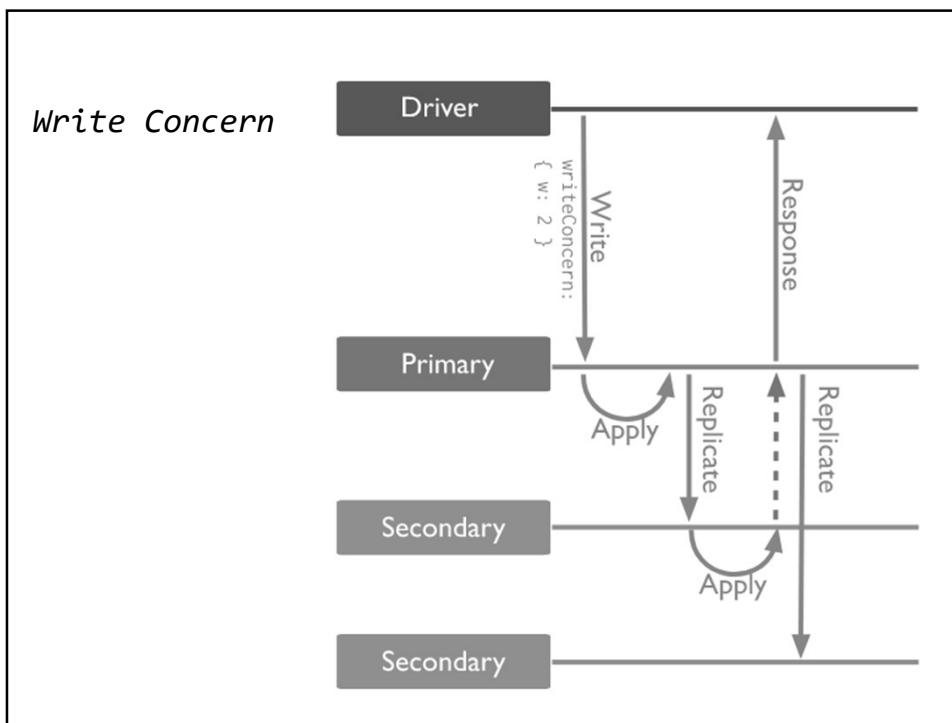
- w: solicita a confirmação de que a escrita foi propagada para um número especificado de instâncias mongod ou para instâncias mongod com tags especificadas.
 - "majority" (**default**)
 - n
 - **n=0:** Não solicita nenhuma confirmação de escrita.
 - **n=1:** Solicita confirmação de que a escrita foi propagada para a instância standalone do mongod ou o nó primário em um conjunto de réplicas.
 - **n>1:** Solicita confirmação de que a escrita foi propagada para a n instâncias do mongod (incluindo o nó primário em um conjunto de réplicas).
 - Membros específicos usando tags

205

Opções

- j: solicita confirmação do MongoDB de que a operação de gravação foi gravada no jornal em disco.
 - true: solicita confirmação de que as instâncias mongod, conforme especificado em w:<value>, foram gravadas no jornal em disco.
 - false (**default**)
- wtimeout:
 - Esta opção especifica um limite de tempo, em milissegundos, para as confirmações.
 - Só é aplicável para valores w maiores que 1.

206



207

Write Concern com Java Driver

- Criando um conjunto de replicação (SC16)
 - `com.mongodb.replication.WriteConcernTest`

208

Transações



209

Transações

- Operações em documento são atômicas por padrão
- Para atomicidade entre múltiplos documentos (em uma ou mais coleções) utilize transações
- Poder ser utilizado no *mongo shell* ou via API
 - Exemplo no *mongo shell* (SC17)

210

Transações e Performance

- Na maioria dos casos, transações de múltiplos documentos têm um alto custo de desempenho
- Para muitos cenários, o modelo de dados desnormalizado (arrays e documentos incorporados) continuará a ser o ideal
- **CONCLUSÃO: modelar seus dados de forma adequada minimizará a necessidade de transações com vários documentos.**

211

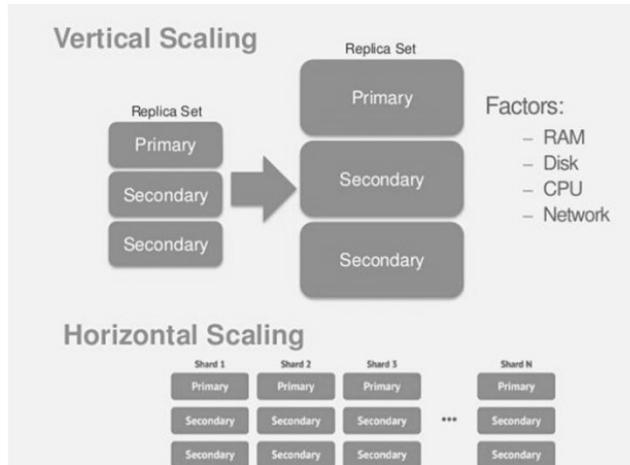
Fragmentação (*Sharding*)



212

Fragmentação (*Sharding*)

- Usado para escalar horizontalmente



213

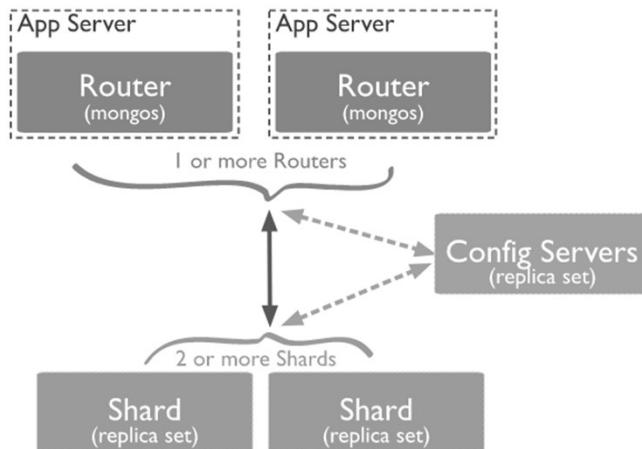
Fragmentação (*Sharding*)

- Divide os dados em pedaços (*chunks*) e os distribui em bancos de dados diferentes (fragmentos) de uma maneira transparente ao usuário
- Cada fragmento pode ser um conjunto de replicação
- O roteador de consultas é o `mongos`
 - Mantém uma tabela de mapeamento da chave do fragmento (*shard key*) para os pedaços de dados
 - A chave do fragmento é parte dos dados dos documentos

214

Fragmentação (*Sharding*)

- shard: Each shard contains a subset of the sharded data. Each shard can be deployed as a replica set.
- mongos: The **mongos** acts as a query router, providing an interface between client applications and the sharded cluster.
- config servers: Config servers store metadata and configuration settings for the cluster. As of MongoDB 3.4, config servers must be deployed as a replica set (CSRS).



215

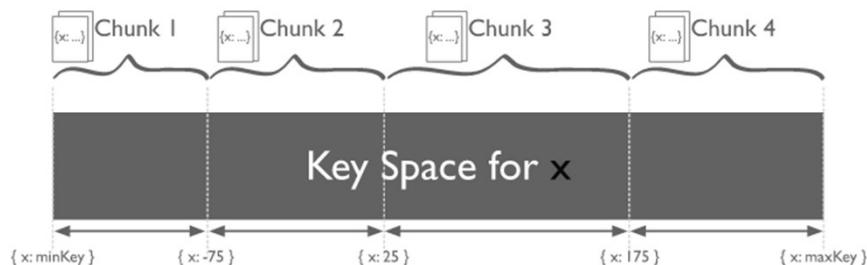
Fragmentação (*Sharding*)

- Existem duas maneira de fragmentação
 - Baseado em Intervalo
 - Baseado em tabela *hash*

216

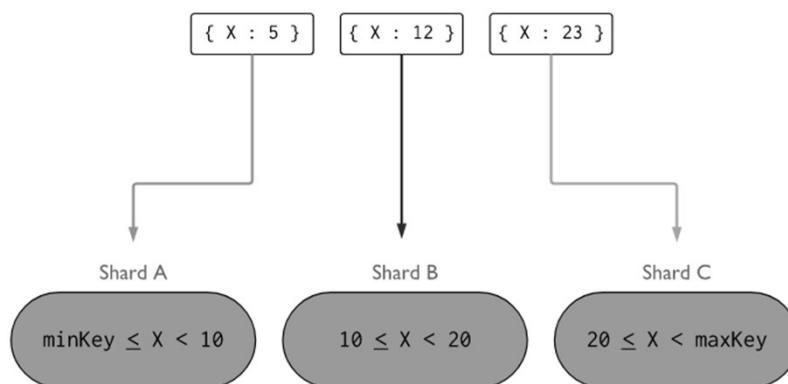
Fragmentação baseada em intervalo

- Recebe um intervalo de documentos e se baseia na chave do fragmento para dividir os pedaços
- Eficiente para consultas baseadas em intervalos



217

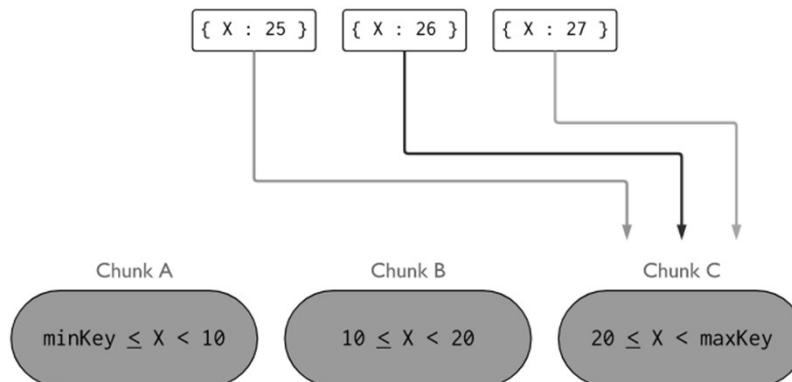
Fragmentação baseada em intervalo



218

Fragmentação baseada em intervalo

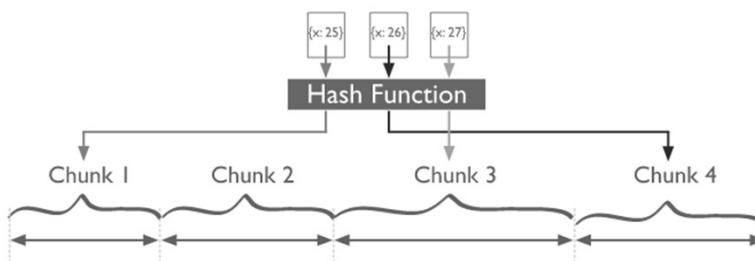
- Em caso de chave monotonicamente crescente



219

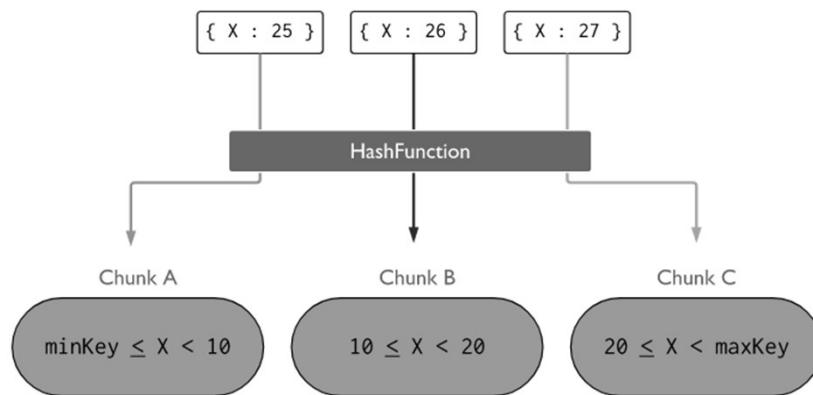
Fragmentação baseada em tabela hash

- Executa um algoritmo de criação de tabela hash em cima da chave do fragmento
- Todo documento deve possuir esta chave (não precisa ser uma chave primária)



220

Fragmentação baseada em tabela hash



221

Construindo um ambiente fragmentado (SC18)

- 3 fragmentos
 - Cada fragmento sendo um conjunto de replicação com 3 nós
- Os 3 servidores de configuração sabem como os pedaços estão distribuídos nos fragmentos
- Teremos no total 12 processos

222

Implicações de Fragmentação no Desenvolvimento

- A fragmentação é transparente ao usuário, mas existem algumas implicações
 - Todo documento deve ter a chave de fragmentação (*shard key*)
 - A chave de fragmentação é imutável
 - Você precisa de um índice que comece com a chave de fragmentação (não pode ser *multikey*)
 - A chave de fragmentação deve ser usada na atualização
 - Não usar a chave de fragmentação na consulta implica em consulta a todos os fragmentos
 - Você não pode ter chave única a não ser que ela faça parte da chave de fragmentação

223

Fragmentação & Replicação

- Quase sempre feitos juntos
 - Fragmentos normalmente são conjuntos de replicação
 - `mongos` faz o que `mongod` faz com relação à replicação
 - Uma falha no nó primário causa uma conexão com um nó secundário
 - *Write concern* acontece com o fragmento que está envolvido na consulta

224

Escolhendo uma chave de fragmentação

- A chave de fragmentação deve ter uma cardinalidade suficiente
- Evite chaves de fragmentação com valores monotonicamente crescente
 - Valor está sempre crescendo
 - Ex: timestamp
 - Causa hotspotting

225

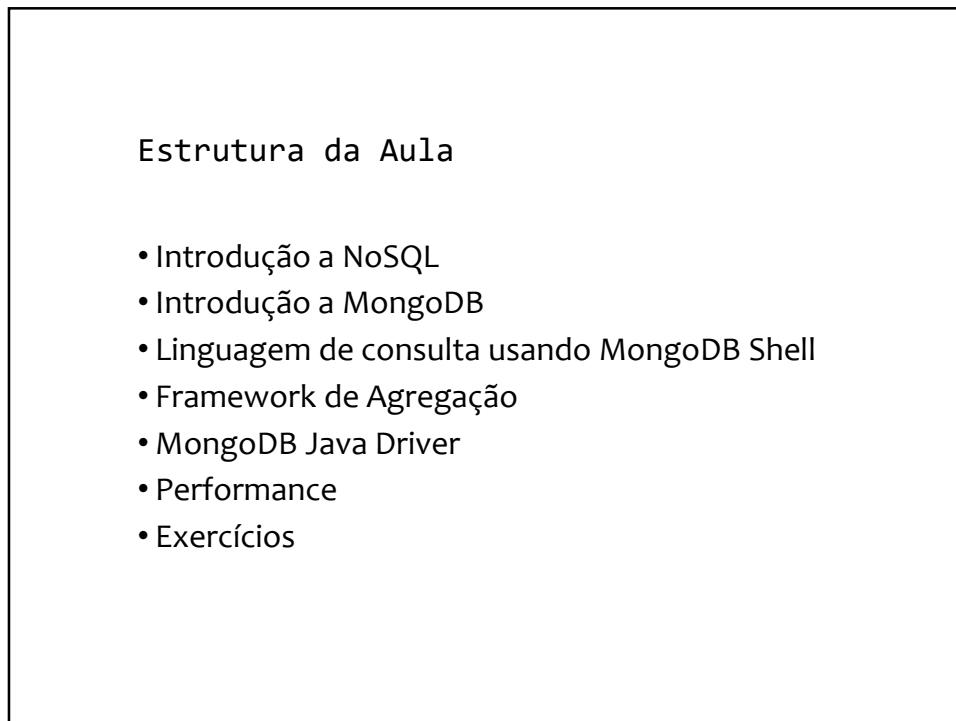
Muito mais material

- MongoDB Manual
<https://docs.mongodb.com/manual/>
- MongoDB University
<https://university.mongodb.com/>
 - MongoDB for Developers
 - Java, JavaScript, Python, .NET
 - MongoDB for DBAs
 - Cluster Administration, Authentication & Authorization, Atlas Security, Performance, Diagnostics and Debugging

226



227



228

Estrutura da Aula

- Introdução a NoSQL
- Introdução a MongoDB
- Linguagem de consulta usando MongoDB Shell
- Framework de Agregação
- MongoDB Java Driver
- Performance
- Exercícios

MÃO NA MASSA

- Exercícios no MongoDB