MongoDB®

Fundamentos, Benefícios e Aplicações Práticas

Ricardo Mello

Senior Software Engineer, MongoDB Community Creator, Oracle Certified Java Developer

SCAN ME

Agenda

- Relational vs. NoSQL
- Introduction to MongoDB
- Data Structure
- Data Modeling
- Basic Operations
- Aggregation Operations
- Components and Tools
  - Compass
  - Atlas
- Hands-On
- What's Next?

Relational vs. NoSQL

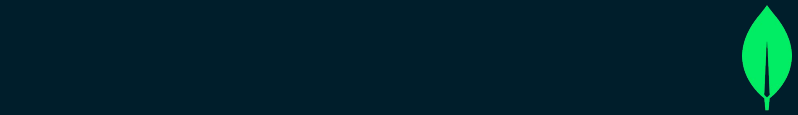
- SQL - Structured Query Language
- Modelo relacional
- Pouco recurso em disco

Relational vs. NoSQL

id	firstName	lastName	email	phone
1	Ricardo	Mello	ricardohsmello@gmail.com	16992..
2	Maria	Silva	maria@gmail.com	15887..

Relational vs. NoSQL

id	firstName	lastName	email	email	phone	phone
1	Ricardo	Mello	ricardohsmello@gmail.com	ricas@hotmail.com	16992..	
2	Maria	Silva	maria@gmail.com		15887..	163429..



Relational vs. NoSQL

id	firstName	lastName	email	email	phone	phone
1	Ricardo	Mello	ricardohsmello@gmail.com	ricas@hotmail.com	16992..	
2	Maria	Silva	maria@gmail.com		15887..	163429..
3	Henrique	Marques				

Relational vs. NoSQL

Relational vs. NoSQL

	SQL Databases	MongoDB
Data storage model		
Schemas		
Scaling		
Joins		

Relational vs. NoSQL

	SQL Databases	MongoDB
Data storage model	Tables, columns	
Schemas		
Scaling		
Joins		

Relational vs. NoSQL

	SQL Databases	MongoDB
Data storage model	Tables, columns	JSON
Schemas		
Scaling		
Joins		

Relational vs. NoSQL

	SQL Databases	MongoDB
Data storage model	Tables, columns	JSON
Schemas	Rigid	
Scaling		
Joins		

Relational vs. NoSQL

	SQL Databases	MongoDB
Data storage model	Tables, columns	JSON
Schemas	Rigid	Flexible
Scaling		
Joins		

Relational vs. NoSQL

	SQL Databases	MongoDB
Data storage model	Tables, columns	JSON
Schemas	Rigid	Flexible
Scaling	Vertical (scale-up with a larger server)	
Joins		

Relational vs. NoSQL

	SQL Databases	MongoDB
Data storage model	Tables, columns	JSON
Schemas	Rigid	Flexible
Scaling	Vertical (scale-up with a larger server)	Horizontal (scale-out across commodity servers)
Joins		

Relational vs. NoSQL

	SQL Databases	MongoDB
Data storage model	Tables, columns	JSON
Schemas	Rigid	Flexible
Scaling	Vertical (scale-up with a larger server)	Horizontal (scale-out across commodity servers)
Joins	Typically required	

Relational vs. NoSQL

	SQL Databases	MongoDB
Data storage model	Tables, columns	JSON
Schemas	Rigid	Flexible
Scaling	Vertical (scale-up with a larger server)	Horizontal (scale-out across commodity servers)
Joins	Typically required	Typically not required

Relational vs. NoSQL

Relacional	MongoDB

Relational vs. NoSQL

Relacional	MongoDB
Table	

Relational vs. NoSQL

Relacional	MongoDB
Table	Collection

Relational vs. NoSQL

Relacional	MongoDB
Table	Collection
Row	

Relational vs. NoSQL

Relacional	MongoDB
Table	Collection
Row	Document

Relational vs. NoSQL

Relacional	MongoDB
Table	Collection
Row	Document
Column	

Relational vs. NoSQL

Relacional	MongoDB
Table	Collection
Row	Document
Column	Field

Relational vs. NoSQL

Relacional	MongoDB
Table	Collection
Row	Document
Column	Field
Primary Key	

Relational vs. NoSQL

Relacional	MongoDB
Table	Collection
Row	Document
Column	Field
Primary Key	_id

Introduction to MongoDB

MongoDB é um banco de dados NoSQL orientado a documentos, desenvolvido pela empresa MongoDB Inc.

Introduction to MongoDB

1. ***Schema Flexível***: Permite que documentos dentro de uma coleção tenham diferentes estruturas, facilitando a evolução do modelo de dados.

Introduction to MongoDB

1. ***Schema Flexível***: Permite que documentos dentro de uma coleção tenham diferentes estruturas, facilitando a evolução do modelo de dados.

2. ***Alta Escalabilidade***: Suporta sharding (distribuição horizontal de dados), o que permite o gerenciamento eficiente de grandes volumes de dados.

Introduction to MongoDB

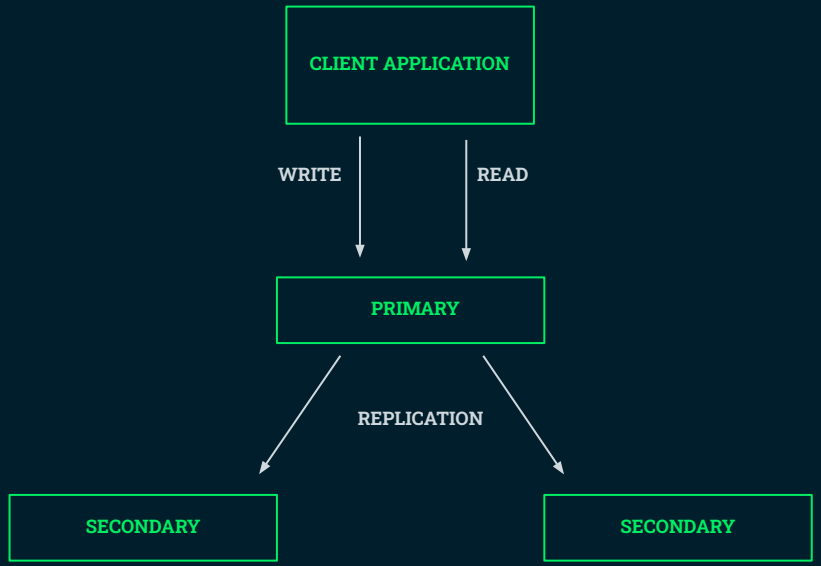
1. ***Schema Flexível***: Permite que documentos dentro de uma coleção tenham diferentes estruturas, facilitando a evolução do modelo de dados.

2. ***Alta Escalabilidade***: Suporta sharding (distribuição horizontal de dados), o que permite o gerenciamento eficiente de grandes volumes de dados.

3. ***Replicação***: Oferece replicação de dados automática para alta disponibilidade e recuperação de desastres.

Introduction to MongoDB

Replica Set



Data Structure

1. ***Definição:*** A unidade básica de dados no MongoDB é o documento, que é um objeto JSON-like (internamente armazenado em BSON - Binary JSON)***.***

2. ***Estrutura:*** Um documento contém um conjunto de pares chave-valor. Cada chave é uma string e cada valor pode ser um tipo de dado básico (como string, número, data) ou um documento aninhado (subdocumento) ou um array.

3. ***Flexibilidade:*** Os documentos dentro de uma coleção não precisam ter a mesma estrutura ou campos, permitindo que dados heterogêneos coexistem na mesma coleção.

Data Structure

```
{
  "_id": ObjectId("507f191e810c19729de860ea"),
  "name": "Alice",
  "age": 29,
  "address": {
    "street": "123 Main St",
    "city": "Springfield",
    "state": "IL"
  },
  "interests": ["reading", "traveling", "cooking"]
}
```

Data Modeling

To develop your schema, instead of thinking about our database, think about the application instead

Data Modeling

To develop your schema, instead of thinking about our database, think about the application instead

- O que meu aplicativo faz?
- Que dados irei armazenar?
- Como os os usuários irão acessar esses dados?
- Quais dados serão mais valiosos para mim?

Data Modeling

To develop your schema, instead of thinking about our database, think about the application instead

- O que meu aplicativo faz? Qual é a aparência dos meus dados
- Que dados irei armazenar? As relações entre os dados
- Como os os usuários irão acessar esses dados? as ferramentas que você planeja ter
- Quais dados serão mais valiosos para mim? Os padrões de acesso que podem surgir

Data Modeling

- Modelagem de dados é o processo de definir como os dados são armazenados.

- Definição dos relacionamentos que existem entre diferentes entidades nos seus dados.

Data Modeling

Tipos de relacionamentos de dados

- ***1:1*** one-to-one
- ***1:Many*** one-to-many
- ***Many:Many*** many-to-many

As duas formas principais de modelar relacionamentos de dados

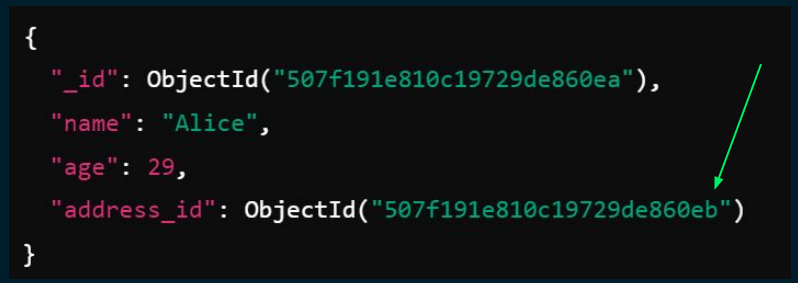
- **Embedding**
- **Referencing**

Data Modeling

Referencing

```
{
  "_id": ObjectId("507f191e810c19729de860eb"),
  "street": "123 Main St",
  "city": "Springfield",
  "state": "IL"
}
```

Address



```
{
  "_id": ObjectId("507f191e810c19729de860ea"),
  "name": "Alice",
  "age": 29,
  "address_id": ObjectId("507f191e810c19729de860eb")
}
```

Person

Data Modeling

Embedding

```
{
  "_id": ObjectId("507f191e810c19729de860ea"),
  "name": "Alice",
  "age": 29,
  "address": {
    "street": "123 Main St",
    "city": "Springfield",
    "state": "IL"
  }
}
```

Person

Data Modeling

Referencing

Vantagens	Desvantagens

Data Modeling

Referencing

Vantagens	Desvantagens
Documentos menores	

Data Modeling

Referencing

Vantagens	Desvantagens
Documentos menores	
Evita duplicação	

Data Modeling

Referencing

Vantagens	Desvantagens
Documentos menores	Performance durante leitura
Evita duplicação	

Data Modeling

Referencing

Vantagens	Desvantagens
Documentos menores	Performance durante leitura
Evita duplicação	Consultas adicionais

Data Modeling

Embedding

Vantagens	Desvantagens

Data Modeling

Embedding

Vantagens	Desvantagens
Desempenho em leituras	

Data Modeling

Embedding

Vantagens	Desvantagens
Desempenho em leituras	
Consistência de dados	

Data Modeling

Embedding

Vantagens	Desvantagens
Desempenho em leituras	Redundância de dados
Consistência de dados	

Data Modeling

Embedding

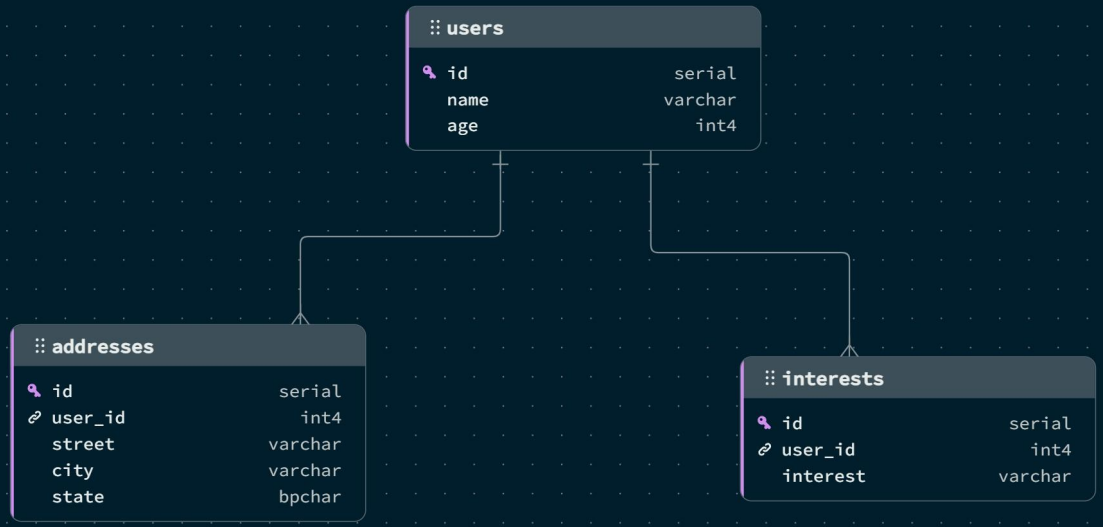
Vantagens	Desvantagens
Desempenho em leituras	Redundância de dados
Fácil manipulação	Tamanho do documento

Data Modeling

Referencing	Embedding
lado “n” muito grande	Integridade das operações
Uma parte usada com frequência e a outra não	Dados excluídos ou arquivados juntos
Restrição de memória	dados que são acessados juntos devem ser armazenados juntos

Data Modeling

- ***Relacional***



Data Modeling

- ***MongoDB***

Basic Operations

- insertOne
- insertMany
- updateOne
- updateMany
- deleteOne
- deleteMany
- replaceOne
- findOne
- find
- count

Basic Operations

- insertOne

```
db.getCollection('movies').insertOne(
    {
        "name": "novo filme",
        "year": 2024
    }
);
```

Basic Operations

- insertMany

```
db.getCollection('movies').insertMany(
    [
        {
            "name": "novo filme 2026",
            "year": 2026
        },
        {
            "name": "novo filme 2027",
            "year": 2027
        }
    ]
);
```

Basic Operations

- updateOne

```
db.getCollection('movies').updateOne(
    { "year": 2027 },
    {
        $set: { year: 2028 }
    }
)
```

Basic Operations

- updateMany

```
db.getCollection('movies').updateMany(
    { "year": 2010 },
    {
        $set: { active: false }
    }
)
```

Basic Operations

- deleteOne

```
db.getCollection('movies').deleteOne({ "year": 2029 })
```

Basic Operations

- deleteMany

```
db.getCollection('movies').deleteMany({"year": 2029})
```

Basic Operations

- replaceOne

```
db.getCollection('movies').replaceOne(
    { "year" : 2029 },
    { "year" : 2029, "title" : "novo titulo"},
    { upsert: true }
 );
```

Basic Operations

- findOne

```
db.getCollection('movies').findOne()
```

Basic Operations

- find

```
db.getCollection('movies').find({"year": 2029})
```

Basic Operations

- count

```
db.getCollection('movies').count()
```

Aggregation Operations

Operações de agregação processam vários documentos e retornam resultados computados. Você pode usar operações de agregação para:

- Agrupar valores de vários documentos juntos.
- Realizar operações nos dados agrupados para retornar um único resultado.
- Analisar mudanças nos dados ao longo do tempo.

Aggregation Operations

```
db.orders.aggregate( [

    // Stage 1: Filter pizza order documents by pizza size
    {
       $match: { size: "medium" }
    },

    // Stage 2: Group remaining documents by pizza name and calculate total quantity
    {
       $group: { _id: "$name", totalQuantity: { $sum: "$quantity" } }
    }

 ] )
```

Aggregation Operations

- $match
- $group
- $project
- $sort
- $limit
- $skip
- $sum
- $avg
- $min
- $max
- $gt
- $lt
- $lookup

Basic Operations

- $match

```
db.users.aggregate(
    [
        {
            $match: { age: { $gt: 25 } }
        }
    ]
);
```

Basic Operations

- $group

```
db.sales.aggregate([
    {
      $group: {
        _id: "$item",
        totalQuantity: { $sum: "$quantity" }
      }
    }
])
```

Basic Operations

- $project

```
db.sales.aggregate([
    {
      $project: {
        _id: 0,
        item: 1,
        quantity: 1,
        price: 0
      }
    }
  ]);
```

Basic Operations

- $sort

```
db.sales.aggregate([
    {
      $sort: { price: -1 } // Ordena pelo campo price em ordem decrescente
    }
  ]);
```

Basic Operations

- $limit

```
db.sales.aggregate([
    { $limit: 2 }
  ]);
```

Basic Operations

- $skip

```
db.sales.aggregate([   // Pula 2 registros
    { $skip: 2 }
  ]);
```

Basic Operations

- $sum

```
db.sales.aggregate([
    {
        $group: {
          _id: "$item",
          totalQuantity: { $sum: "$quantity" }
        }
      }
  ]);
```

Basic Operations

- $avg

```
db.sales.aggregate([
    {
      $group: {
        _id: "$item", // Agrupa por item
        averageQuantity: { $avg: "$quantity" } // Calcula a média dos valores de quantity
      }
    }
  ]);
```

Basic Operations

- $min

```
db.sales.aggregate([
    {
      $group: {
        _id: "$item", // Agrupa por item
        minQuantity: { $min: "$quantity" } // Encontra o valor mínimo de quantity
      }
    }
  ]);
```

Basic Operations

- $max

```
db.sales.aggregate([
    {
      $group: {
        _id: "$item", // Agrupa por item
        minQuantity: { $max: "$quantity" } // Encontra o valor máximo de quantity
      }
    }
  ]);
```

Basic Operations

- $lt

```
db.users.aggregate(
    [
        {
            $match: { age: { $lt: 25 } }
        }
    ]
);
```

Basic Operations

- $gt

```
db.users.aggregate(
    [
        {
            $match: { age: { $gt: 25 } }
        }
    ]
);
```

Basic Operations

- $lookup

```
db.customers.insertMany(
    [
        { "customer_id": 101, "name": "Alice", "city": "New York" },
        { "customer_id": 102, "name": "Bob", "city": "Los Angeles" }
    ]
)
```

```
db.orders.insertMany(
    [
        { "_id": 1, "order_id": "A001", "customer_id": 101, "amount": 500 },
        { "_id": 2, "order_id": "A002", "customer_id": 102, "amount": 200 },
        { "_id": 3, "order_id": "A003", "customer_id": 101, "amount": 700 }
    ]
)
```

Basic Operations

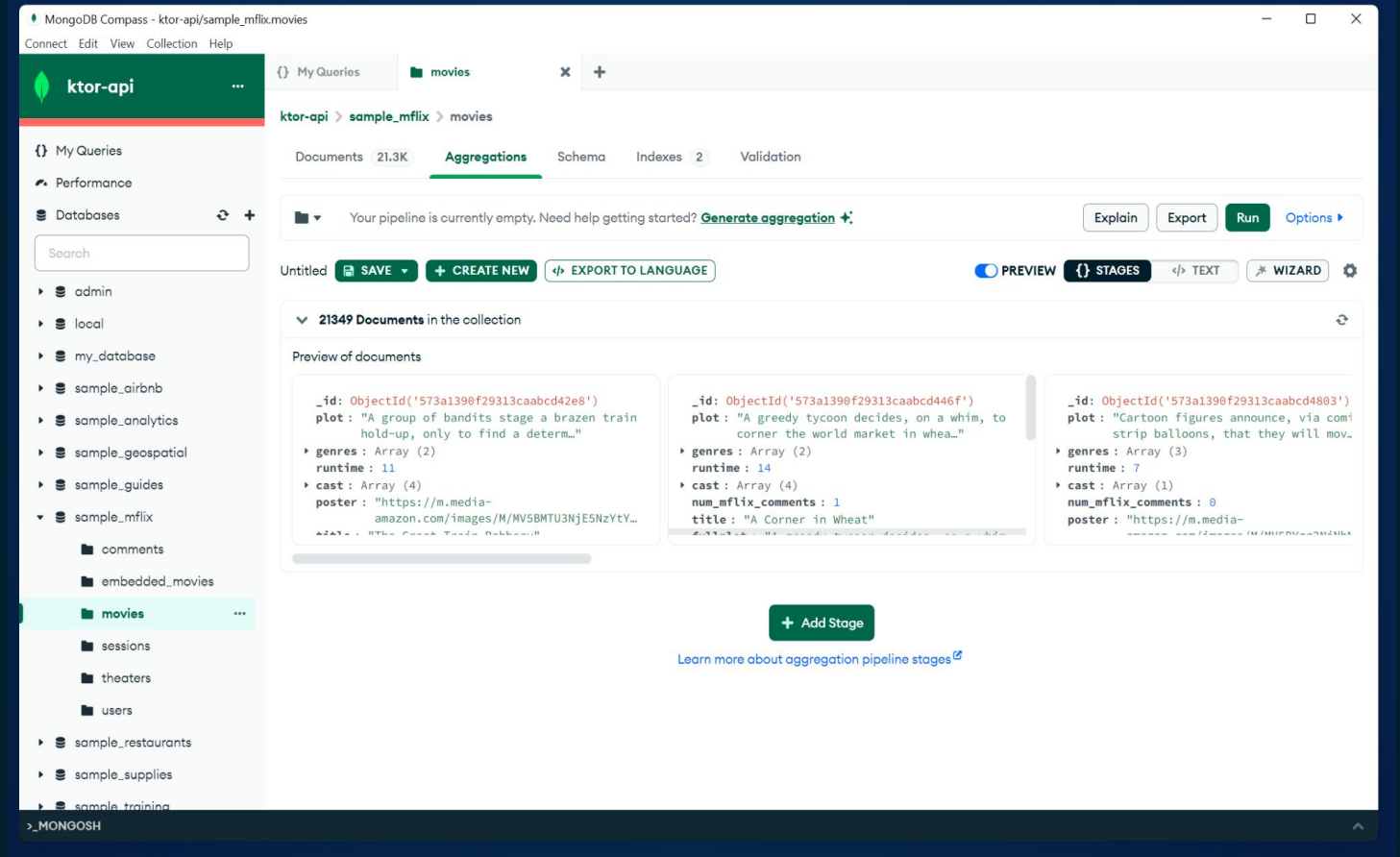
- $lookup

```
db.orders.aggregate([
    {
        $lookup: {
            from: "customers",  // Coleção para realizar a junção
            localField: "customer_id", // Campo na coleção orders
            foreignField: "customer_id", // Campo na coleção customers
            as: "customerResult" // Nome do campo de saída para os documentos combinados
        }
    }
])
```

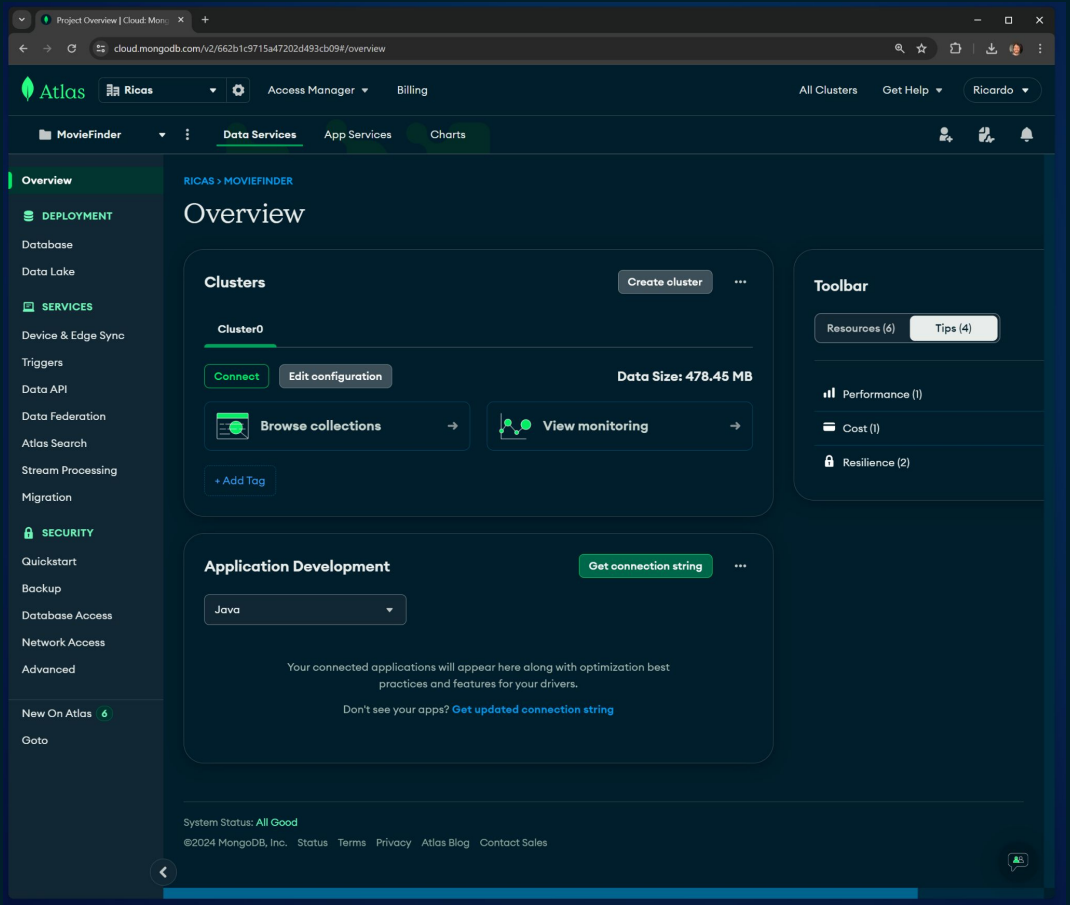
Components and Tools

- Compass - The GUI for MongoDB
- Atlas - Cloud database

Components and Tools



Components and Tools



Agenda

- Hands-On
  - insert
  - update
  - delete
- Scripts VS Code
- Atlas
  - Metrics
  - Charts
- Indexes